Number Theory

In Context and Interactive

Number Theory

In Context and Interactive

Karl-Dieter Crisman
Gordon College

January 24, 2017

About the Author Karl-Dieter Crisman has degrees in mathematics from Northwestern University and the University of Chicago. He has taught at a number of institutions, and has been a professor of mathematics at Gordon College in Massachusetts since 2005. His research is in the mathematics of voting and choice, and one of his teaching interests is (naturally) combining programming and mathematics using SageMath. He has given invited talks on both topics in various venues on three continents.

Other (mathematical) interests include fruitful connections between mathematics and music theory, the use of service-learning in courses at all levels, connections between faith and math, and editing. Non-mathematical interests he wishes he had more time for include playing keyboard instruments and exploring new (human and computer) languages. But playing strategy games and hiking with his family is most interesting of all.

Edition: 2017/1 Edition

Website: math.gordon.edu/ntic

To my students and the Sage community;
let's keep exploring together.

Acknowledgements

This text evolved over the course of teaching MAT 338 Number Theory for many years at Gordon College, and immense thanks are due to the students through five offerings of this course for bearing with using a text-in-progress. The Sage Math team and especially the Sage cell server have made an interactive book of this nature possible online, and the Mathbook XML and MathJax projects have contributed immensely to its final form, as should be clear.

In addition, no acknowledgement would be complete without recognizing the patience of my family with respect to the days and weeks of travel, from an hour away in New England to as far away as Cape Town and India, in order to learn more about Sage and teach using Sage in the classroom. It was always done with the goal in view of enriching others' lives and not just my own, and I hope I have lived up to that promise.

To Everyone

Welcome to Number Theory! This book is an introduction to the theory and practice of the integers, especially positive integers – the numbers. We focus on connecting it to many areas of mathematics and dynamic, computer-assisted interaction. Let's explore!

Carl Friedrich Gauss, a great mathematician of the nineteenth century, is said to have quipped that if mathematics is the queen of the sciences, then number theory is the queen of mathematics ([C.4.4]). If you don't yet know why that might be the case, you are in for a treat.

Number theory was (and is still occasionally) called 'the higher arithmetic', and that is truly where it starts. Even a small child understands that there is something interesting about adding numbers, and whether there is a biggest number, or how to put together fact families. Well before middle school many children will notice that some numbers don't show up in their multiplication tables much, or learn about factors and divisors. One need look no further than the excellent picture book *You Can Count on Monsters* [C.5.1] by Richard Evans Schwartz to see how compelling this can be.

Later on, perfect squares, basic geometric constructs, and even logarithms all can be considered part of arithmetic. Modern number theory is, at its heart, just the process of asking these same questions in more and more general situations, and more and more interesting situations.

They are situations with amazing depth. A sampling:

- The question of what integers are possible areas of a right triangle seems very simple. Who could have guessed it would lead to fundamental advances in computer representation of elliptic curves?
- There seems to be no nice formula for prime numbers, else we would have learned it in middle school. Yet who would have foreseen they are so very regular on average?
- Taking powers of whole numbers and remainders while dividing are elementary and tedious operations. So why should taking remainders of tons of powers of whole numbers make online purchases more secure?

This book is designed to explore that fascinating world of whole numbers. It covers all the 'standard' questions, and perhaps some not-quite-as-standard topics as well. Roughly, it covers the following broad categories of topics.

- Basic questions about integers
- Basic congruence arithmetic
- Units, primitive roots, and Euler's function (via groups)
- Basics of cryptography, primality testing, and factorization

- Integer and rational points on conic sections
- The theory and practice of quadratic residues
- Basics of arithmetic functions
- The prime counting function and related matters
- Connecting calculus to arithmetic functions

Finally, it won't take long to notice that the way in which this book is constructed emphasizes connections to other areas of math and encourages dynamic interaction. (See the note To the Instructor.) It is my hope that all readers will find this 'in context and interactive' approach enjoyable.

To the Student

Hi! Not too many students read this bit in textbooks, but I hope you do, and I hope you circle stuff you think is important. In pen.

Doing math without writing in the book (or on something, if you're only using an electronic version) is sort of like reading much literature (like Shakespeare or Homer) or many religious texts (like the Psalms or Vedas) without paying attention to the spoken aspect. It's possible, and we all may have done it (some successfully), but it's sort of missing the point.

So read this book and write in it. My students do. They even like it.

Here are three things that will lead to success with this book.

- You should like exploring numbers and playing with them. If you were the kind of kid who added

  $$1+2+3+4+5+6+7+8+9+10+\cdots$$

  on your calculator when you were bored to see if there would be an interesting pattern, and actually liked it, you will like number theory. If you then tried

  $$2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7\cdot 8\cdot 9\cdot 10\cdot\cdots$$

  you will really like it.

- I also hope you are open to using computers to explore math and check conjectures. As Picasso said, "[T]hey can only give you answers" – but oh what answers! We use the SageMath system, one that will grow with you and that will always be free to use (for several meanings of the word free). You don't have to know how to program to use this, though it's useful. Plus, you are using number theory under the hood anyway if you use the internet much, so why not?

- Finally, you should want to know why things are true. I assume a standard introduction to proof course as background, but different people are ready in different ways for this. If you are reasonably familiar with proofs by induction and contradiction, and have some basic experience with sets and relations, that is a good start. Some good free resources online include *A Gentle Introduction to the Art of Mathematics* [C.2.2] and *The Book of Proof* [C.2.1].

  Some of the proofs will be hairy, and some exercises challenging. (Not all!) Do not worry; by trying, you will get better at explaining why things are true that you are convinced of. And that is a very useful skill. (Provided you *are* convinced of them; if not, go back to the first bullet point and play with more examples!)

Remark 0.0.1. As a final note before you dig in, if you think that it is worth exploring the possible truth (see Section 25.3) of

$$1+2+3+4+5+6+7+8+9+10+\cdots=-\frac{1}{12}$$

or if as a kid you did

$$2^{3^{4^{5^{6^{7^{8^{9^{\cdots}}}}}}}}$$

to see what would happen, then maybe you should become a mathematician. In that case, click on all links in the text and find a cool problem that interests you!

To the Instructor

Assuming that the reader of this preface is an instructor of an actual course, may I first say thank you for introducing your students to number theory! Secondly of course I'm grateful for your at least briefly considering this text.

In that case, gentle reader, you may be asking yourself, "Why on earth yet another undergraduate number theory text?" Surely all of these topics have been covered in many excellent texts? (See the preface To Everyone for a brief topic list, and the Table of Contents for a more detailed one.) And surely there is online content, interactive content, and all the many topics here in other places? Why go to the trouble to write another book, and then to share it? These are excellent questions I have grappled with myself for the past decade.

There are two big reasons for this project. The first is reminiscent of Tertullian's old quote about Athens and Jerusalem; what has arithmetic to do with geometry? (Or calculus, or combinatorics, or anything?) At least in the United States, away from the most highly selective institutions (and in my own experience, there as well), undergraduate mathematics can come across as separate topics connected by some common *logical* threads, and being at least vaguely about 'number' or 'magnitude', but not necessarily part of a *unified whole*.

When I first taught this course, I was dismayed at how few texts really fully tackled the geometry, algebra, and analysis inherent in number theory. Many do one or two (especially algebra, since number theory might often be a second course in abstract algebra), but few attacked all connections. Still, there are some which do, and I even found *Elementary Number Theory* by Jones and Jones [C.1.1] which does a very good job of this, though at a slightly higher level of sophistication than I found my students ready for. Those familiar with it will find that my presentation of certain topics (e.g. arithmetic functions, the zeta function), of certain proofs, and some topic order is influenced by it. I try to point out the former cases, and I have substantively modified even those in ways more appropriate for typical US undergraduates, as well as with somewhat different emphases.

Given my first goal, I would have happily used this text with some extra details for my students, were it not for the magic and wonder of the internet. How could I not harness this to have my students do approximations to the size of computations that their browsers are constantly doing as they go shopping on the web? Having found Sage, I found it hard to avoid using it whenever I could, and encouraging students to do the same to explore things like Euler's ϕ function (as I encourage yours to do in Section 9.2 by hand).

Interactivity and visualization is becoming common currency in mathematics education. In calculus and lower-level courses this has been true for some time, but even in abstract algebra there are books like Nathan Carter's *Visual Group Theory* [C.5.2], specialized software projects like PascGalois, and many general applets (including ones from the Wolfram Demonstrations or Maple

Möbius projects). This has been coming into number theory too, naturally, beyond the programming projects many books have included. An early number theory text involving explicit programs (and a CD-ROM!) written for extensive course work was [C.3.7], and the first book invoking extensive use of Sage commands was probably the founder's own [C.1.3]. Very recently (in fact, after the unofficial release of this text) the book [C.1.10] (which has similar content and aims to the current work, though at a somewhat higher level) appeared in second edition with complete SageMath worksheets on its website, which could be used on a Sage or SageMathCloud server. Hence the time is more than right for a fully online resource.

So my second goal for this book is to bring online interactivity into a mainstream number theory text. It is wonderful to see students with an interest in the arts respond to the dynamic visualization in Sage **interacts**, while those with interests in computer science love to ask questions about how to view the source code or some of the details of representing large numbers. And all the students have access to computations from simple ones involving the aliquot parts function to the full Riemann formula for the prime number function.

Why should you *not* use this book? First, I make few claims to topical or mathematical originality[1]. The ordering is somewhat different than usual, I include a few topics I haven't seen addressed adequately very often in truly introductory texts (notably a beginning of the geometry of numbers and long-term averages of arithmetic functions), and I have created many visualization and exploration oriented applets.

At the low end of other reasons you might not use it, some topics of great importance which are perfect for beginners (especially partitions and continued fractions) are absent. You can't cover *everything* in a semester, after all, and I have shied away a bit from more purely combinatorial stuff, though I hope to return to it in future editions[2]. At the high end of preparation, I do not and cannot expect a course in abstract algebra or complex (or even real) analysis for my students, and so this book reflects that reality. Knowing about proofs by induction and contradiction, as well as basics of sets, integers, and relations, is what I can assume. In fact, I have great recommendations for you if you know all your students can do contour integration or are ready to define a number field – see References and Further Resources. Finally, I don't have a corporation behind me.

On the other hand, I think you *should* consider using it. This is class-tested material for standard topics (plenty for a semester-long course at most institutions), and not beholden to any interests beyond being a good resource for instructors in 'mainstream' undergraduate math programs in the United States. There are plenty of exercises (though not a surfeit, so feel free to supplement), fun links, and hopefully a quirky and engaging sense of wonder and exploration. The price is also right. Finally, I don't have a corporation behind me.

Should you choose to use this text, I have only a few recommendations for how to use it (see also my notes To the Student).

- Encourage in-class exploration. Put away books, turn off the computers, and just try stuff out. Create your own worksheet to explore (say) the Möbius function or solutions to linear Diophantine equations. In short, make sure your students see mathematics as a dynamic enterprise – particularly because so many of the theorems involved are highly abstract.

[1] I have tried hard to credit any non-standard proofs which are essentially in the form I found them, and appreciate forbearance (and notification) if I have missed any such citations.

[2] See [C.1.11] for a nice introduction in a more combinatorial vein, particularly to partition identities.

- Less is more. I will often pick *one* representative proof in a section, project it on the screen, and then really follow it through on an adjacent blackboard with specific numbers (such as $p = 13$, which is just big enough to be interesting but not so big as to be overwhelming).

- Use computer examples judiciously. Sage (or any other system) can just as easily become a Delphic oracle (pun intended) spewing forth cryptic utterances as a useful tool to help create and solve conjectures. You're possibly doing your students a disservice if you don't use it at all, but despite having written this text with Sage in mind throughout, I don't regard its use as completely essential. Number theory in this form has been around since Euclid, so the past thirty years of mass-market computation is a drop in the bucket of time. If you want a true inquiry-based approach, I like the text *Number Theory through Inquiry* [C.1.5] a lot.

- Note the Sage notes (full list at the List of Sage notes). Especially if you have more than just a few students who have a little programming experience, this is a perfect course to find projects to challenge them with, such as those in the venerable [C.1.4]. The Sage notes gently remind or give short introductions to some aspects of how to use Python and Sage. They are not formally structured or arranged, or comprehensive; if you are looking for this, you should supplement your course with a real basic programming text in Python, such as [C.2.6] or [C.2.7].)

- Use the exercises, and ones outside the book if you want. There are exercises for each chapter, of varying difficulty levels (in the grand tradition of upper-level math texts, I do *not* provide solutions). In general, assigning daily, collecting weekly seems to be a decent model – though be sure to give students ample warning as to *which* ones will be collected! The last few chapters' material is more advanced, and there are correspondingly fewer possible exercises. I find this to be a good time for a small project in the history of number theory; especially if you have students from several different cultural heritages, having them discover where it comes up in theirs (it nearly always does) has been a perennial favorite.

There are no sections marked as optional, or table of dependencies, though these should be pretty similar to most elementary texts. (I do pretty much everything in my own course, picking results or sections to skip on the fly if time or the students seem to require this.) Here are some minor suggestions, though.

- If you are teaching a shorter course or wish to spend more time on some topic, the chapters on Beyond Sums of Squares and More on Prime Numbers are certainly optional in this sense.

- The chapters concerning Points on Curves and Long-Term Function Behavior are not optional in my view of number theory, but may be viewed as 'selected topics'.

- The introductory (short) chapters 1 and 18 should *not* be considered optional, but may be emphasized or not to instructor taste. The point is just to motivate what we are doing before getting to formal definitions.

- If you don't like cryptography or believe (like Hardy) that there are no applications to number theory, you can certainly create a nearly application-free course by skipping the chapters on An Introduction to Cryptography and Some Theory Behind Cryptography.

- I don't consider the last several chapters on the prime counting function and other arithmetic functions connecting to calculus to be optional, but I have the luxury of having mostly juniors and seniors for a full semester. In a quarter course or one aimed more at sophomores (in the United States), one should still at the very least spend a couple days at the end of the course talking about these topics, perhaps discussing sections 21.2 and 21.3, and smatterings of Chapter 25.

As a final note, I hope you enjoy using the text as much as I've enjoyed teaching from it. Everyone should have that day where a student's jaw drops from a cool theorem displayed visually, or when the students are working so intently on an in-class project that they don't even notice the class period end. It's been my privilege to have that happen, and my hope is this text can bring you closer to that goal.

Contents

Chapter 1

Prologue

What is number theory? Briefly, it is the study of the integers and questions arising from them.

Definition 1.0.1. The set of **counting numbers** is denoted

$$\mathbb{N} = \{0, 1, 2, 3, 4, \cdots\}$$

Note that in this text, this set begins at zero. The **integers** is the set of positive *and* negative counting numbers:

$$\mathbb{Z} = \{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$$

This is a fairly dry definition, though. The best way to find out what that means is just try to answer some questions *about* integers!

1.1 A First Problem

Let's start! Suppose you have lots of left-over postage stamps[1] that are of just a few different denominations. It could be fun to see what amounts you could make from them.

To be concrete, let's assume first that all your stamps are numbered 2¢ and 3¢. Here are two questions we could ask. They are mathematically equivalent, but might take your exploration in two very different directions!

- Which denominations of postage can you get by combining just these kinds?
- Which denominations can you *not* get with just these two kinds?

Once you've thought about that, try the same problem with 2¢ and 4¢ stamps. What is the same, what is different?

Now let's get to a nontrivial case; what about with 3¢ and 4¢ stamps? In this case, after some experimentation, it looks like only 1, 2, and 5 are not possible, so anything six or above is possible. We call this number the **conductor** of the set $\{3, 4\}$.

What we are really asking, which might be clear by now, is which positive integers n are impossible (or possible) to write in the form $n = 3x+4y$, for some nonnegative integers x and y. This is also sometimes called the Frobenius[2] or coin problem.

[1] Perhaps because you only use email or texting now; too bad for you!

[2] For a very full discussion, see [C.6.20]. But not until after you have started the next chapter of this book!

Continue trying this with different small pairs of numbers (see also Exercise 1.4.5–Exercise 1.4.7). Pay attention to two things:

- What is the conductor of the pair? (You might want to ask *whether* there is such a number!)
- How *many* numbers lower than the conductor *cannot* be written in this way?

1.2 Review of Previous Ideas

Before going further, we need a bit of review. The following three topics may be considered prerequisites for the course.

1.2.1 Well-Ordering

The first principle is both simply and deep. It is an axiom of the positive integers, but we give it its usual name.

Axiom 1.2.1 (Well-Ordering Principle)**.** *Any nonempty set of positive integers has a least/smallest element.*

This principle actually holds with any subset of $\mathbb{Z}$ which is bounded below, such as $\mathbb{N}$.

Let's use it as an example to prove the following fact you probably didn't know required proof.

Fact 1.2.2 (Consecutive Integers)**.** *There are no integers between 0 and 1.*

Proof. This proof proceeds by contradiction. Assume there are some such integers, and let

$$S = \{x \in \mathbb{Z} \mid 0 < x < 1\} .$$

This set must then have a least element a, and $0 < a < 1$. Thus a^2 is another element such that $0 < a^2 < 1$, but we also know that $a^2 < a$ in this case, because of the usual properties of ordering positive numbers.

But then a^2 should be in S but is smaller than any element of S. That is a contradiction, so our original assumption was wrong, and there are no such integers (i.e. S is empty). □

To review, proofs by **contradiction** and **contrapositive** both start by assuming the negation of the conclusion. A proof by contrapositive uses that to prove the negation of the original assumption, while a proof by contradiction can negate any other true fact or lead to some other absurdity; in this case, you can't have two different smallest elements of a set.

1.2.2 Induction

Sometimes we need a way to prove a statement for all integers n after a certain point, for instance greater than or equal to $n = 1$. This is usually called **proof by induction**. There are (usually) two steps in a typical 'simple' induction.

- First we prove the "base case" (often $n = 1$ or $n = 0$).
- Then we prove the "induction step", that the case $n = k$ implies case $n = k + 1$.

These combine to prove a fact for *all* cases $n \geq 1$.

Example 1.2.3 (Archetype for Induction)**.** We shall show that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Solution. The base case is that $1 = \frac{1(1+1)}{2}$, which is easy.

The induction step begins with the assumption that

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

For this proof, to add just one more integer $k+1$ means

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1)$$

and we can just plug in the induction *assumption* to obtain

$$\frac{k(k+1)}{2} + (k+1) = (k+1)(\frac{k}{2}+1) = \frac{(k+1)(k+2)}{2}$$

which is exactly what is required to finish the induction step.

We could use well-ordering (Axiom 1.2.1) to prove that the induction proof technique works, but will not do so here.

1.2.3 Divisibility

Definition 1.2.4. If an integer n can be written as a product $kd = n$ of two integers k and d, then we say that d **divides** n, or that n is **divisible** by d, or that d is a **divisor** of n. We write $d \mid n$ for this concept.

Examples:

- For instance, n even is just the same thing as $2 \mid n$.
- The *divisors* of 8 are … $\pm 1, 2, 4, 8$! (Don't forget negative divisors.)
- Very often we can write this generically, so that $n \mid x+1$ means that $x+1$ can be written as the product of n and some other integer m.

There are lots of interesting things to say about divisibility. We can prove a somewhat unexpected statement using induction and just what we already know.

Example 1.2.5. Show that $4 \mid 5^n - 1$ for $n \geq 0$.

Solution. This proof will proceed by induction. This time the base case will be $n = 0$.

- Base step: If $n = 0$ this is just saying that 4 divides $5^0 - 1 = 1 - 1 = 0$, which is definitely true.
- Induction step:

- Suppose $4 \mid 5^k - 1$. Then, by Definition 1.2.4, $5^k - 1 = 4x$ for some integer x.
- Hence $5^k = 1 + 4x$.
- Our goal in this step is to show $4 \mid 5^{k+1} - 1$.
- Since we need something true about $5^{k+1} - 1$, let's start with just 5^{k+1}. Using the induction *assumption* we can write this as $5^k \cdot 5 = (1 + 4x)5$; this means that

$$5^{k+1} - 1 = 5(1 + 4x) - 1 = 20x + 4.$$

- Certainly $20x + 4$ is divisible by 4, so we are done with the proof.

There are lots of other propositions about divisibility you are probably familiar with from previous courses. Here is a sampler.

Proposition 1.2.6 (Divisibility Facts)**.**

1. *If $a \mid b$ and $b \mid c$ then $a \mid c$.*
2. *If $a \mid b$ then $ca \mid cb$.*
3. *If $c \mid a$ and $c \mid b$ then $c \mid au + bv$ for any integers u, v.*
4. *All divisors of n are less than or equal to n, unless $n = 0$.*

These are not hard to prove (see Exercise 1.4.1). For instance, the second one can be proved by simply noting $b = ka$ for some $k \in \mathbb{Z}$, so that $cb = c(ka) = c(ak) = (ca)k$. The others are similar, and are good practice with such manipulation.

1.3 Where are we going?

Before moving on from these preliminaries and our introductory Prologue, let's step back. What will we cover in this text?

- We have started by exploring basic integer questions, and will continue like this at first (Chapter 1–Chapter 3).
- We'll be essentially *forced* to move to the concepts of congruences and primes by the material (Chapter 4–Chapter 7).
- Next, we'll explore a more advanced point of view of integers and congruences, including groups, to attack cryptography efficiently (Chapter 8–Chapter 12).
- About halfway through, geometry and how it infiltrates number theory will be up (Chapter 13–Chapter 17).
- Finally, functions and limits will help us illuminate primes in depth, as well as show us how the ideas of calculus really do show up in number theory quite naturally (Chapter 18–Chapter 24), concluding with an introduction to the legendary Riemann Hypothesis in Chapter 25.

Let's get ready for an exciting exploration of number theory!

1.4 Exercises

1. Prove some or all of the facts in Proposition 1.2.6.

2. Find a counterexample to show that when $a|b$ and $c|d$, it is not necessarily true that $a + c|b + d$.

3. Prove that $2^n > n$ for all integers $n \geq 0$ by induction.

4. Prove, by induction, that if c divides integers a_i and we have other integers u_i, then $c \mid \sum_{i=1}^{n} a_i u_i$.

5. Write up a proof of the facts from the first discussion about the conductor idea (in Section 1.1) with the pairs $\{2, 3\}$, $\{2, 4\}$, and $\{3, 4\}$.

6. What is the conductor for $\{3, 5\}$ or $\{4, 5\}$? Prove these in the same manner as in the previous problem.

7. Try finding a pattern in the conductors. Can you prove something about it for at least certain pairs of numbers, even if not all pairs?

8. What is the largest number d which is a divisor of both 60 and 42?

9. Try to write the answer to the previous problem as $d = 60x + 42y$ for some integers x and y.

10. Get a Sage account somewhere, such as at the SageMath Cloud or a Sage notebook server on your campus, if you don't already have one.

1.5 How to Use Computation

This text is advertised as having interactive computation, but so far any computation has been your own. How does it fit in? We'll skip ahead slightly here to see how this will work.

In the interactive version of this text, the areas below are called **Sage cells**, or cells for short. Assuming you're connected to the internet, this very first cell will use SageMath (usually just called Sage) to check whether this number leaves a fraction when reduced, or whether it reduces to an integer. Click "Evaluate" to try it out.

```
38/19
```

Go ahead, try changing the numbers and clicking the evaluate button again.

As we go through the text, you'll see *lots* of opportunities to use Sage. Sometimes I'll give you the opportunity to learn a little bit about how to use it in Sage notes, such as the following one.

Sage note 1.5.1 (About Sage notes)**.** Sage notes will teach you useful things about basic programming, or more general facts about Sage and Python, the language Sage is based on.

Let's try another computational cell. We haven't defined prime numbers yet (see Chapter 6), but I figure you know what they are. Here you can check whether an integer is prime.

```
is_prime(3169)
```

Sage note 1.5.2 (Using commands in Sage cells)**.** Assuming you are using this online, you can put *any legitimate Sage command* in the cells above. (Try `integrate(x^3,x)` if you know some calculus.) Or you use these commands in your own Sage worksheet at your local Sage server or in the SageMathCloud, so that you can save your work!

If you are using an offline or hard copy version, I still highly recommend sifting through some of the code and commands; much of it will enlighten the reader. (Then try it out online or on your local computer!)

Finally, let's test some conductor ideas using technology. In the cell below, Sage will automatically list all the nonnegative numbers up to n that can be written as $n = ax + by$ for nonnegative integers x and y. The default values are the $a = 3, b = 4$ combination.

```
@interact
def _(a=(3,[2..10]),b=(4,[2..10]),n=(20,[10..50])):
   list_of_them=list(set([a*x+b*y for x in srange(n/a+1)
       for y in srange(n/b+1)]))
   list_of_them=[item for item in list_of_them if item <=
       n ];list_of_them.sort()
   pretty_print(html("The␣nonnegative␣integers␣up␣to␣
       $n=%s$␣which␣can␣be"%(str(n))))
   pretty_print(html("written␣as␣positive␣combinations␣of␣
       $a=%s$␣and␣$b=%s$␣are:"%(str(a),str(b))))
   print list_of_them
```

Notice that with the one tried above we definitely are getting the same answers. Also notice that the algorithm I used in the code is very naive – I just listed all possible combinations under a certain size. It would be interesting to use this to try to verify patterns you may have noticed about the precise size of the conductor, and when it exists.

Chapter 2

Basic Integer Division

In this chapter, we introduce some concepts of numbers which are familiar, but key for our further study. In particular, we try to understand *why* they work.

- The division algorithm (Section 2.1),
- The greatest common divisor (Section 2.2), and
- The Euclidean algorithm (Section 2.3).

Then we'll put them together with the *Bezout identity* (Section 2.4).

2.1 The Division Algorithm

2.1.1 Statement and examples

Let's start off with the division algorithm. This says that if you divide an integer a by a positive integer b, you will always get an integer remainder r that is nonnegative, but less than b. Equally important, there is *only one way* to write this – that is, there is a *unique* remainder under these circumstances.

Theorem 2.1.1 (Division Algorithm)**.** *For $a, b \in \mathbb{Z}$ and $b > 0$, we can always write $a = qb + r$ with $0 \leq r < b$ and q an integer. Moreover, there is only one way to do this.*

The proof is below in Subsection 2.1.2.

Using this is really easy to do for small examples like $a = 13, b = 3$ by division.

$$13 = 4 \cdot 3 + 1 \text{ so } q = 4 \text{ and } r = 1$$

For bigger values it's nice to have it implemented in Sage.

```
divmod(281376,29)
```

```
(9702, 18)
```

The first element is the quotient, the second the remainder. We can check that this algorithm works by multiplying and adding back together.

```
9702*29+18
```

```
281376
```

Sage note 2.1.2 (Counting begins at zero)**.** There are several things to note about this early computation. First, note that the answer to `divmod` came in parentheses, a so-called 'tuple' data type.

Second, there is another way to approach this computation, more programmatically so that it's easier to reuse. What do you think the `[0]` and `[1]` mean?

```
divmod(281376,29)[0] * 29 + divmod(281376,29)[1]
```

```
281376
```

To access the first and second parts of the answer (the quotient and remainder), we use square brackets, asking for the *0th and 1st* parts! In Python (the programming language behind Sage), as in many other languages, counting begins at zero.

The discussion in the previous note actually turns out to be an enduring argument in number theory, too. Do we only care about positive numbers, or nonnegative ones as well? We saw this in the stamps example, since one could send a package for free under certain circumstances (campus mail), but might not care about that case. Similarly, are we required to use at least one of each type of stamp, or is it okay (as in our problem) to not use one type?

2.1.2 Proof of the Division Algorithm

The neat thing about the division algorithm is that it is not hard to prove but still uses the Well-Ordering Principle; indeed, it depends on it. The key set is the set of all possible remainders of a when subtracting multiples of b, which we call

$$S = \{a - kb \,|\, k \in \mathbb{Z}\}\,.$$

Here is the proof:

- S must be nonempty (why?), and call the nonnegative piece $S' = S \cap \mathbb{N}$. (Why must this also be nonempty?)
- The well-ordering principle must apply to S'. Give the name r to the smallest element of S', which must be writeable as $r = a - bq$ (that's the definition of being in $S' \subset S$, after all).
- Now, if $r \geq b$, we could have subtracted another copy of b while keeping the remainder in S'; hence, we must have that $r < b$. (Note that r is the smallest nonnegative such number.)
- These elements r and q must be unique. First let's observe that the r is unique, which would follow if the least element of a well-ordered set is unique. But think about that! If r and r' are both least elements of S, then in particular they are less than (or equal to) each other – the definition of equality.
- Finally, we will need a factoring argument, the fact that (for any integers x and y) if $xy = 0$ then $x = 0$ or $y = 0$ (or both). Since we have shown $r = r'$, then $bq = bq'$. Hence $b(q - q') = 0$ so (since $b > 0$) we have that $q - q' = 0$, which means the quotient is unique.

It's worth actually trying out this proof with some a and b, say with $a = 26$ and $b = 3$.

As a scholium (see Exercise 2.5.1) note that if $b < 0$ there can still be a positive remainder, but here we would need $0 \leq r < |b|$ in the theorem.

2.1.3 Uses of the division algorithm

It's kind of fun to prove interesting things about powers with this fact, and likely you did in a previous course. Here is some pattern for remainders of perfect squares modulo 4, for instance.

```
for i in [0..10]:
   pretty_print(html("The␣remainder␣of␣%s␣squared␣with␣
       respect␣to␣4␣is␣%s"%(i,divmod(i^2,4)[1])))
```

Sage note 2.1.3 (Repeating commands for different input)**.** The syntax `for i in [0..10]:` just means we want to do the next command for integers from 0 to 10.

The rest of the command (all the percent symbols and so forth) is mostly for correct formatting. That includes the indentation in the second line – an essential part of Python and Sage.

This certainly provides strong numerical evidence for the following proposition. But better is a proof!

Proposition 2.1.4. *A perfect square always leaves remainder $r = 0$ or $r = 1$ when divided by 4.*

Proof. Using the division algorithm, take $n = 4q + r$.

- What happens if we square it, $(4q + r)^2$?
- Algebraically, we get $16q^2 + 8qr + r^2$. Clearly this is a multiple of 4 plus r^2. So the only possible remainders are the remainders of r^2, where r *is already known to be less than 4!*
- Now check these yourself to see that the only possibilities are the ones stated.

□

One cool thing about this proof is that if we just change the proof from using $n = (4q+r)^2$ to one using $n = (mq+r)^2$, we can essentially do the same thing for several divisions at once. If the number we divide by is m, then

$$(mq + r)^2 = m^2q^2 + 2mqr + r^2 = m(mq^2 + 2qr) + r^2\,,$$

hence all that matters for the final remainder is r^2, since the rest is already divisible by m.

But we know that there are only b possibilities for r, so it's easy to check all their *squares*. For $m = 6$:

```
for i in [0..5]:
   pretty_print(html("The␣remainder␣of␣%s␣squared␣with␣
       respect␣to␣6␣is␣%s"%(i,divmod(i^2,6)[1])))
```

This verifies that $r = 0, 1, 3, 4$ are the only possible remainders of perfect squares when you divide by six.

2.2 The Greatest Common Divisor

It seems intuitive that of all the numbers dividing a number (the divisors), one is biggest.

Definition 2.2.1 (Common Divisors)**.** If we consider the various divisors of two numbers a and b, we say that d is a **common divisor** of a and b if $d \mid a$ and $d \mid b$. If d is the biggest such common divisor, it is called the **greatest common divisor**, or *gcd*, written $d = \gcd(a, b)$.

What are all the common divisors of 6 and 10? What is the gcd?
We now come to a great *definition-theorem.*

Theorem 2.2.2. *The* greatest common divisor *(*GCD *or gcd) of two integers a and b (not both zero) is:*

- *The largest integer d such that $d|a$ and $d|b$. (See above.)*
- *The number achieved by applying the Euclidean algorithm (a repeated division algorithm) to a and b. (See Section 2.3.)*
- *The smallest positive number which can be written as $ax + by$ for some integers x and y. (See Section 2.4 and Subsection 2.4.2.)*

This is amazing, and the first real indication of the power of having multiple perspectives on a problem. It means that the very theoretical issue of when a gcd exists (*and* finding it) can be treated as a purely computational problem, characterized *completely independent* of actually finding divisors. And further, there is a definition purely in terms of addition and multiplication, not division or subtraction.

If you need to actually calculate a gcd, you use the algorithm. If you want to prove something about it that has to do with dividing, you use the original definition. And if you need to prove something about it where division is hard to use, you use the third characterization. This sort of idea will come up again and again in this book – that having multiple ways to define something *really helps.*

2.3 The Euclidean Algorithm

The Euclidean algorithm says that to find the gcd of a and b, you do the division algorithm until you get zero as the remainder, each time replacing the old division by the new remainder, and the old number by the old division number. The last non-zero remainder is the gcd.

We'll state and prove this momentarily (Algorithm 2.3.2). Let's try it with a reasonably sized problem, $a = 60$ and $b = 42$.

$$\begin{aligned} 60 &= 42 \cdot 1 + 18 \\ 42 &= 18 \cdot 2 + 6 \\ 18 &= 6 \cdot 3 + 0 \end{aligned}$$

So $\gcd(60, 42) = 6$.

This procedure is named after Euclid because of this proposition in Euclid's Elements. There is an amazing complete Java interactive implementation of all the propositions, by David Joyce; his version of this proposition includes some explanation of what Euclid assumes in doing this. In particular, Euclid basically assumes the Well-Ordering Principle, although of course he didn't think of it in such anachronistic terms.

Remark 2.3.1. Euclid, a mathematician in Alexandria during the Hellenistic era, appears to have written the Elements as a compendium of rigorous mathematical knowledge. In addition to being the main geometry textbook in the Western and Islamic worlds for two millennia (as late a teacher as Charles Dodgson a.k.a. Lewis Carroll was extolling its virtues in print), there are substantial number-theoretic portions as well. No one really knows how much of the Elements is original to Euclid, but the work as a whole is monumental and well-organized, despite some well-known criticisms.

Try the algorithm on your own by hand for the gcd of 280 and 126. Or, for even more practice, try it with gcd(2013, 1066) and then check your work with Sage.

```
gcd(2013,1066)
```

Algorithm 2.3.2 (Euclidean algorithm)**.** *To get the greatest common divisor of a and b, perform the division algorithm until you hit a remainder of zero, as below.*

$$\begin{aligned} a &= bq_1 + r_1 \\ b &= r_1q_2 + r_2 \\ r_1 &= r_2q_3 + r_3 \\ &\dots \\ r_{n-3} &= r_{n-2}q_{n-1} + r_{n-1} \\ r_{n-2} &= r_{n-1}q_n + 0 \end{aligned}$$

Then the previous remainder, r_{n-1}, is the greatest common divisor.

Proof. First let's see why this even does anything. The division algorithm says each r_i is less than the previous one, yet they may not be less than zero. So let's apply the Well-Ordering Principle to the set of remainders. (Think about that.) This set must have a least positive element, and will be the answer. Another way to think about it is that since b is finite, there won't be an infinite number of steps.

Of course, that just gives a number, with no guarantee it has any connection to the gcd. So consider the set of common divisors $d \mid a$ and $d \mid b$. All such d also divide

$$a - q_1b = 1 \cdot a + (-q_1) \cdot b = r_1$$

So these d also divides $r_2 = b - q_2r_1$, and indeed divide all the remainders, even $r_{n-1} = r_{n-3} - q_{n-1}r_{n-2}$. So all common divisors of a and b are divisors of r_{n-1}.

On the other hand, if d divides r_{n-1}, it divides $r_{n-2} = r_{n-1}q_n$, and thus divides $r_{n-3} = r_{n-2}q_{n-1} + r_{n-1}$, and so forth. Hence d divides a and b.

So the set of common divisors of a and b are equal to the set of divisors of r_{n-1}, so this algorithm really does give the gcd. □

As you might expect, the proof makes more sense if you try it out with actual numbers; for the theoretical view, see Exercise 2.5.14. Especially if you can find a and b for which the algorithm takes four or five steps, you will gain some insight.

2.4 The Bezout Identity

2.4.1 Backwards with Euclid

Now, before we get to the third characterization of the gcd, we need to be able to do the Euclidean algorithm *backwards*. This is sometimes known as the *Bezout identity*.

Definition 2.4.1 (Bezout identity)**.** A representation of the gcd d of a and b as a linear combination $ax + by = d$ of the original numbers is called an instance of the **Bezout identity**. (This is not unique.)

It is worth doing some examples. Perhaps you already have gotten one, probably by trial and error. For instance,

$$6 = 60 \cdot (-2) + 42 \cdot 3 \, .$$

The third characterization in Theorem 2.2.2 implies that doing this is *always* possible; $\gcd(a, b) = ax + by$ for some integers x and y. Doing the Euclidean algorithm backwards gets this.

Sometimes it's good to write the Euclidean algorithm down one side of a table, and then go backwards up the other side of the table to obtain an instance of the Bezout identity. Here's an example with the gcd of 8 and 5; follow it from top left to the bottom and then back up the right side. The middle column provides the necessary rewriting.

$8 = 1 \cdot 5 + 3$	$8 - 1 \cdot 5 = 3$	$1 = 2 \cdot 3 - 1 \cdot 5 = 2 \cdot (8 - 1 \cdot 5) - 1 \cdot 5 = 2 \cdot 8 - 3 \cdot 5$
$5 = 1 \cdot 3 + 2$	$5 - 1 \cdot 3 = 2$	$1 = 3 - 1 \cdot 2 = 3 - 1 \cdot (5 - 1 \cdot 3) = 2 \cdot 3 - 1 \cdot 5$
$3 = 1 \cdot 2 + 1$	$3 - 1 \cdot 2 = 1$	$1 = 3 - 1 \cdot 2$
$2 = 2 \cdot 1 + 0$		Go up this column...

This question of the Bezout identity is implemented in Sage as `xgcd(a,b)`, because this is also known as the **e*X*tended Euclidean algorithm**.

```
xgcd(60,42)
```

Or, $6 = -2 \cdot 60 + 3 \cdot 42$, as we saw above.

Example 2.4.2. Try to get the xgcd/Bezout identity for $\gcd(1492, 1066)$. You should get $2 = -5 \cdot 1492 + 7 \cdot 1066$.

Try the following Sage cell to check that it works.

```
xgcd(1492,1066)[1]*1492 + xgcd(1492,1066)[2]*1066
```

Sage note 2.4.3 (Remind how to get list elements)**.** Do you remember what the `[1]` means? What do you think the `[2]` means in this context?

2.4.2 Proving the final characterization

The final characterization of the greatest common divisor (Theorem 2.2.2) is that it is the least positive integer which can be written $au + bv$ for integers u, v. Let's prove that now.

- Call c the smallest such $au + bv$.

- By Proposition 1.2.6, any integer which divides a and b divides any such $au + bv$, so it divides the smallest such $au + bv = c$. In particular, the gcd of a and b (which we'll call d) divides c. So $d \leq c$.
- On the other hand, we know from the backward/extended Euclidean algorithm/Bezout identity that d can be written $d = ax + by$ for some integers x and y. Since c is the smallest such (positive) integer, $c \leq d$.
- Concluding that $d = c$ is the only way to avoid a contradiction!

2.4.3 Other gcd questions

You might also want to do *more than one* linear combination, for variety. How might we get another such representation?

Example 2.4.4 (Using Bezout to get another Bezout)**.** We used the backwards Euclidean algorithm to see that $6 = -2 \cdot 60 + 3 \cdot 42$. Let's use that to get another.

- Since 6 is itself a divisor of both 60 and 42, let's pick one (the smaller one!), 42, and write *it* as $42 = 7 \cdot 6$.
- Then we can really write

$$42 = 7 \cdot 6 = 7 \cdot (-2 \cdot 60 + 3 \cdot 42) \, ,$$

  since after all we just saw that was a way to represent 6!
- Now we plug this back into the original equation:

$$6 = -2 \cdot 60 + 3 \cdot 42 = -2 \cdot 60 + 3 \cdot (7 \cdot 6)$$

$$= -2 \cdot 60 + 3 \cdot (7 \cdot (-2 \cdot 60 + 3 \cdot 42))$$

If we simplify it out, that means $6 = -44 \cdot 60 + 63 \cdot 42$, which is indeed correct!

So, substituting a Bezout identity into itself yields more and more such identities. How many such identities are there? Is there a general form?

Another interesting question is that some gcds of large numbers are very easy to compute. What makes finding $\gcd(42000, 60000)$ so easy? If you're in a classroom, this is a perfect time to discuss.

On a related note, if $\gcd(a, b) = d$, could you make a guess as to a formula for $\gcd(ka, kb)$ (for $k > 0$)? Can you *prove* it in Exercise 2.5.16? (Hint: here is where our original definition *or* the Bezout version could be useful.)

2.4.4 Relatively prime

There is one final thing that the linear combination version of the gcd can give us. It is something you may think is familiar, but which can arise very naturally from the Bezout identity.

When do a and b have as greatest common divisor the *smallest* possible, $\gcd(a, b) = 1$? From this point of view, it is precisely when you can write $ax + by = 1$ for some integers x and y.

Think about this, though; it means that you can write *any* integer as a (linear) combination of a and b if their gcd is 1! This is a property I think people would have come up with no matter how the development of mathematics had gone; the pairs of integers such that you can write any number as a combination of them.

Definition 2.4.5 (Relatively Prime)**.** If the greatest common divisor of two numbers is one, we call these **relatively prime** numbers or **coprime** numbers.

Proposition 2.4.6. *Here are two interesting facts about coprime integers a and b:*

- *If $a \mid c$ and $b \mid c$, then $ab \mid c$.*
- *If $a \mid bc$, then $a \mid c$.*

Proof. The first is not too hard to prove, *if* you think in terms of Bezout. It does need a little cleverness.

- Remember that $ax + by = 1$ by definition.
- So $c = cax + cby$.
- Now write $c = kb$ and $c = \ell a$, and substitute them in the *opposite* parts of the previous line.
- This gives $c = (kb)ax + (\ell a)by$, and ab definitely divides both parts of this, so it divides the whole thing by our earlier proposition about divisibility.

We leave the second as an exercise (Exercise 2.5.19). □

It's also useful to try to find *counterexamples*! Can you find place where $\gcd(a, b) \neq 1$, $a \mid c$ and $b \mid c$, but ab does not divide c? (See Exercise 2.5.20.)

2.5 Exercises

1. Try stating and proving the division algorithm (Theorem 2.1.1) but for $b < 0$.

2. Can you find an n such that the possible remainders of a perfect square when divided by n are all numbers between zero and $n - 1$? If you can, how many different such n can you find? If not, can you prove there are none?

3. Write the gcd of 3 and 4 as a linear combination of 3 and 4 in three different ways. (Hint: trial and error.)

4. You can define the gcd of more than two numbers as the greatest integer dividing all of the numbers in your set. So, for instance, $\gcd(20, 30, 70) = 10$. Calculate the gcd of some hard-looking sets of three numbers by listing divisors.

```
gcd([3800,7600,1900])
```

Sage allows you to calculate arbitrary gcds like this.

5. Find the gcd of the four numbers 1240, 6660, 15540, and 19980 *without Sage*.

6. Prove that $\gcd(a, a + 2) = 1$ if a is odd and $\gcd(a, a + 2) = 2$ is a is even.

7. Let a be a positive integer. What is the greatest common divisor of a and $a + 1$? Prove it.

8. Use the Euclidean algorithm to find the gcd of 51 and 87, and then to write that gcd as a linear combination of 51 and 87. Do the same thing for 151 and 187. Find another way to write the gcd of one of these pairs as a linear combination.

9. Define the *least common multiple* of a and b to be the smallest positive number which is divisible by both a and b. Prove that the least common multiple of a and b is ab precisely when a and b are coprime.

10. Find the gcd of 151 and 187 using the Euclidean algorithm, then write it as a linear combination of these two numbers in two different ways.

11. Find the gcd of 500000001 and 5000001 in any way you see fit other than asking someone else.

12. Does the pattern you see in the following interact continue? How would you find a counterexample, how might you prove it?

```
@interact
def _(m=(3,[1..20]),n=(2,[1..20])):
    pretty_print(html("The gcd of $%s$ and $%s$ is
        $%s$"%(5*10^m+1,5*10^n+1,gcd(5*10^m+1,5*10^n+1))))
```

13. Find the gcd of three four digit numbers, none of which is divisible by ten.

14. To make the proof of the Euclidean algorithm, Algorithm 2.3.2, *very* complete, one would want to use induction to replace "and so forth" verbiage. Do so for practice with induction.

15. Prove that if a and c are coprime, and likewise b and c are coprime, then ab and c are coprime. (Hint: use Bezout.)

16. If $\gcd(a, b) = d$ and $k > 0$, prove a formula for $\gcd(ka, kb)$.

17. You probably know the Fibonacci numbers $1, 1, 2, 3, 5, 8, \cdots$, where $f_{n+2} = f_{n+1} + f_n$ and we number as $f_1 = 1$, $f_2 = 1$. Try applying the Euclidean algorithm to a pair of consecutive Fibonacci numbers? How long does it take compared to n? (For a more general approach see [C.1.1, Exercises 1.17-1.19].)

18. Try the above exercise again, but with a variant of the Fibonacci numbers where $f_{n+2} = f_{n+1} + 2f_n$. This would start $1, 1, 3, 5, 11, 21, \cdots$.

19. Prove that if a and b are coprime, and if $a \mid bc$, then $a \mid c$. (Hint: use the Bezout fact again.)

20. Find examples that contradict the facts about coprime integers (Proposition 2.4.6) if a and b *do* share a factor greater than 1.

21. We discussed *relatively* prime numbers in this chapter. Write down your own definition of a *prime* number. Then compare it with the book, a few internet sources, or some other authoritative source. Should 1 be considered prime? What about -1?

22. Search books and/or the Internet and find at least three *different* proofs that there is no largest prime number. (Ours, Theorem 6.2.1, is the oldest one we know of.) You don't have to understand all the details; they should be fairly different from each other, though. Do any of the proofs generate all primes *in order*?

Chapter 3

From Linear Equations to Geometry

So far, we have mostly investigated topics that will seem familiar even to the high school student; for instance, the gcd shows up in adding fractions with unequal denominators.

What makes number theory so interesting is that even a *slight* change in the questions we ask, or the way in which we approach them, can yield completely unexpected insights.

In this section, we will begin this process by going from the simple questions we started with into more subtle ones, largely motivated by a surprising connection with geometry.

3.1 Linear Diophantine Equations

The first goal for this chapter is to completely solve all 'Linear Diophantine Equations' (of two variables), generically

$$ax + by = c \text{ for } a, b, c \in \mathbb{Z}$$

They have been studied since the late Roman era (by Greeks, of course), but it turns out that a general solution for equations like $6x + 4y = 2$ were already known in the early medieval days by Indian mathematicians like Aryabhata. When, shortly after 1600, Bachet de Méziriac came up with the same answer, it was only the first in a long line of people coming up with this solution again and again. And that's the one we are doing today!

There are several main cases involved.

Theorem 3.1.1 (Solutions of Linear Diophantine Equations)**.** *We wish to solve $ax + by = c$ for all integers x, y that make it work. Let* $\gcd(a, b) = d$. *Then:*

1. *When c is not a multiple of d (or both a and b are zero) and c is not), there is no solution.*
2. *When a or b is zero (but not both), there are infinitely many solutions that require little work to obtain.*
3. *When $c = d$, there are infinitely many solutions, but you will need to first obtain* one *solution in order to generate the others.*

4. *When c is a nontrivial multiple of d, there are infinitely many solutions that are easiest to generate by means of a solution to $ax + by = d$.*

Proof. The details are in the following subsections.

1. When c is not a multiple of d: Subsection 3.1.1
2. When a or b is zero: Subsection 3.1.2
3. When $c = d$: Subsection 3.1.3
4. When c is a nontrivial multiple of d: Subsection 3.1.4

You should definitely follow the steps with specific simple numbers to see how each proof works. Examples 3.1.2 and 3.1.3 are good models. □

3.1.1 If c is not a multiple of $\gcd(a, b)$

What is the possibility of solving $ax + by = c$ in this case?

- Here, our previous theorems say this is impossible. Can you see why?
- For instance, if $a = 6$, $b = 9$, and $c = 5$.

3.1.2 If a or b is zero

- Say $b = 0$ – in which case $\gcd(a, b) = a$. Try $a = 55$.
- Then we are just solving $ax = c$, so it is true precisely when $a|c$, and then all pairs $\left(\frac{c}{a}, y\right)$ with integer y are solutions.
- If $a = 0$ the answer is analogous; write it down for yourself as practice!

3.1.3 If $c = \gcd(a, b)$

Suppose $a, b \neq 0$ and c actually *is* the gcd of a and b ... then there is some work to do. Follow along with $a = 60$, $b = 42$, and $c = 6$.

- Your first step should be to get that gcd via the Euclidean algorithm. Let's call it d.
- Then go backwards (i.e. Bezout identity 2.4.1) to get *one* solution (x_0, y_0). That is important, since now at least one $ax_0 + by_0 = c$ is known.
- Next, simply write down the whole solution set:

$$x = x_0 + \frac{b}{d}n,\ y = y_0 - \frac{a}{d}n \ \text{ for } n \in \mathbb{Z}\,!$$

Notice that a and b switch their 'affiliation' here from x and y to y and x. Also note that x and y have $\pm$ involved. It doesn't really matter which is which (switch $-n$ for n to see why), but if they have the same sign it is wrong. (When in doubt, try something and then check to see if the answers are right.)

- It's easy to check this works:

$$a\left(x_0 + \frac{b}{d}n\right) + b\left(y_0 - \frac{a}{d}n\right) = ax_0 + \frac{abn}{d} + by_0 - \frac{abn}{d} = c\,.$$

- Why does this give *all* solutions? Well, pick another solution (x, y), and let's show it has this form. Start with

$$ax + by = c = ax_0 + by_0, \text{ so that } \frac{a}{d}(x - x_0) = -\frac{b}{d}(y - y_0)\,.$$

  Now we use Proposition 2.4.6. Since $\frac{b}{d}$ divides the right side, it divides the left side. But since $\frac{b}{d}$ is coprime to $\frac{a}{d}$, then it must divide the *other* piece of the left side, so that

$$x - x_0 = k\frac{b}{d}, \text{ which means } x = x_0 + k\frac{b}{d}\,,$$

  which is exactly what we just said was the form of all solutions.

Example 3.1.2 (An easy example: $6x + 4y = 2$)**.** Trial and error tells us that $6x + 4y = 2$ can be solved with $x_0 = 1, y_0 = -1$. Thus the full answer is

$$x = 1 + \frac{4}{2}n,\ y = -1 - \frac{6}{2}n$$

or

$$x = 1 + 2n, y = -1 - 3n\,.$$

3.1.4 If c works, but is not the gcd

What if c is not the greatest common divisor? Then let $c = de$, where again d is the greatest common divisor.

- We still have the same solution as above for d, so we can find $d = ax_0 + by_0$ for some (x_0, y_0) as above.
- Now if we multiply the whole thing by e, we have that $de = ax_0e + by_0e$, or that $c = a(x_0e) + b(y_0e)$ is a solution.
- Now do *exactly the same thing* as above for the answer! The surprise is that $x = x_0e + \frac{b}{d}n, y = y_0e - \frac{a}{d}n$ still works, but it's easy to check again (see Exercise 3.6.2), with virtually the same check working as above. You don't need the e in the fractions because they will just cancel anyway.

Example 3.1.3. Try to do $15x - 21y = 6$, a slightly harder one. (Hint: $d = 3$; what are c and d?

3.2 Geometry of Equations

But just proving things are true and using them isn't enough. *Why* is it true, intuitively? I believe the right place to do this is in geometry. Try out the following interactive cell.

```
@interact
def _(a=slider(-10,10,1,6),b=slider(-10,10,1,4),
    c=slider(-20,20,1,2),viewsize=slider(3,20,1,5)):
```

```
p = plot(-(a/b)*x+c/b,-viewsize,viewsize,
    plot_points=200)
lattice_pts=[[i,j] for i in [-viewsize..viewsize] for
    j in [-viewsize..viewsize]]
plot_lattice_pts =
    points(lattice_pts,rgbcolor=(0,0,0), pointsize=2)
if mod(c,gcd(a,b))==0:
    line_pts = [coords for coords in lattice_pts if
        a*coords[0]+b*coords[1]==c]
    if line_pts==[]:
        plot_line_pts = Graphics()
    else:
        plot_line_pts =
            points(line_pts,rgbcolor=(0,0,1),
            pointsize=20)
    pretty_print(html("Showing␣solutions␣to␣
        $%sx+%sy=%s$␣in␣this␣viewing␣
        window"%(str(a),str(b),str(c))))
    show(p+plot_lattice_pts+plot_line_pts,
        figsize=[5,5], xmin=-viewsize,xmax=viewsize,
        ymin=-viewsize,ymax=viewsize)
else:
    pretty_print(html("The␣gcd␣of␣$%s$␣and␣$%s$␣is␣
        $%s$,␣which␣does␣not␣divide␣
        $%s$,"%(str(a),str(b),str(gcd(a,b)),str(c))))
    pretty_print(html("so␣no␣solutions␣to␣
        $%sx+%sy=%s$"%(str(a),str(b),str(c))))
    show(p+plot_lattice_pts,
        figsize=[5,5],xmin=-viewsize,xmax=viewsize,
        ymin=-viewsize,ymax=viewsize)
```

The little gray dots in the graph above are called the **integer lattice**; this is all the intersections of the lines $y = m, x = n$ for all integers m, n. There are many mathematical lattices, but this is the one we will focus on in this course.

Definition 3.2.1. The **integer lattice** is the set of points (m, n) for $m, n \in \mathbb{Z}$.

In the graphic, for instance $(-2, 3)$ is probably visible; however, note that $(-1, 1/2)$ should not be not a little dot, because it doesn't have integer values.

Now, since $ax + by = c$ may be thought of as a line (in fact, the line

$$y = -\frac{a}{b}x + \frac{c}{b}$$

with slope $-\frac{a}{b}$), we now have a *completely different interpretation* of the most basic number theory question there is, the linear Diophantine equation. It is simply asking, "When (for what a, b, c combinations) does the line hit this lattice? If it does, can you tell me all intersections?" If you play around with the sliders you will quickly see that things work out just as promised in the theorems.

But let's go a little deeper. There are three interesting insights we can get.

- First, Theorem 3.1.1 now expresses a very mysterious geometric idea, depending on whether
$$\gcd(a, b) \mid c$$
If so, then this line hits lots of the lattice points; if not, the line *somehow slides between every single one of them*! You can check this by keeping a, b the same and varying c in the interact above.

- Secondly, it makes the proof of why Theorem 3.1.1 gets all of the answers much clearer. If you have one answer (for instance, $(1, -1)$) and go right by the run and down by the rise in $\frac{a}{b}$ (our example was $a = 6, b = 4$), you hit another solution (perhaps here $(-3, 5)$) since it's still all integers and the slope was the line's slope.

  But wait, couldn't there be points in between? Sure. So make $\frac{a}{b}$ into lowest terms (e.g. $\frac{3}{2}$), which would be $\frac{a/d}{b/d}$. And this is the 'smallest' rise over run that works to keep you on the line and keep you on integer points.

- Third, it can help clarify the role of the solution which the Bezout identity (extended Euclidean algorithm) gives for $ax + by = c$. Namely, as pointed out in an article from 2013 in the American Math Monthly by S.A. Rankin, the "solution provided … lies nearest to the origin." Again, try the applet to convince yourself of this!

Although we won't pursue it, there is a question which this formulation in an online text brings up. Namely, given that the 'line's in question are themselves only pixellated approximations whose coordinates may not satisfy $ax + by = c$, what is the connection between the computer graphics and the number theory? See *How to Guard an Art Gallery* [C.5.7], Chapter 4, for an accessible take on this[1] from a number-theoretic viewpoint, as well as Exercise 3.6.15.

3.3 *Positive* Integer Lattice Points

Now that we have the geometric viewpoint, here is a more subtle question:

Question 3.3.1. Assume there exists a solution (hence infinitely many) to $ax + by = c$. How many such solution pairs (x, y) have x and y both positive?

This is similar to the conductor question. It is closely related to *integer programming*, something with industrial applications.

```
@interact
def _(a=slider(1,20,1,1), b=slider(1,20,1,1),
    c=slider(1,20,1,4)):
    ym = c/b + 1
    xm = c/a + 1
    p = plot(-(a/b)*x+c/b,-1,xm, plot_points = 200)
    lattice_pts = [[i,j] for i in [0..xm] for j in [0..ym]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0), pointsize=2)
    if mod(c,gcd(a,b))==0:
        line_pts = [coords for coords in lattice_pts if
            (coords[0]>0) and (coords[1]>0) and
            (a*coords[0]+b*coords[1]==c)]
        if len(line_pts)==0:
            pretty_print(html( 'Solutions to
                $%sx+%sy=%s$:'%(str(a),str(b),str(c))))
            pretty_print(html( 'No positive lattice points
                at all!'))
            show(p+plot_lattice_pts, figsize = [5,5], xmin
                = 0, xmax = xm, ymin = 0, ymax = ym)
```

[1] As well as several other topics in this text! But you'll have to read it to find out which ones.

```
        else:
            plot_line_pts = points(line_pts, rgbcolor =
                (0,0,1), pointsize=20)
            pretty_print(html('Solutions␣to␣
                $%sx+%sy=%s$:'%(str(a),str(b),str(c))))
            pretty_print(html('Number␣of␣positive␣lattice␣
                points␣=␣' + str(len(line_pts))))
            show(p+plot_lattice_pts+plot_line_pts, figsize
                = [5,5], xmin = 0, xmax = xm, ymin = 0,
                ymax = ym)
    else:
        pretty_print(html('Solutions␣to␣
            $%sx+%sy=%s$:'%(str(a),str(b),str(c))))
        pretty_print(html('No␣positive␣lattice␣points␣at␣
            all!'))
        show(p+plot_lattice_pts, figsize = [5,5], xmin =
            0, xmax = xm, ymin = 0, ymax = ym)
```

Let's explore this. How many such points are there in the following cases? Draw pictures by hand, or use the interact above.

- $x + y = 4$, $x + y = 5$, $x + y = 6$, ...
- $2x + y = 4$, $2x + y = 5$, $2x + y = 6$, ...
- $2x + 2y = 4$, $2x + 2y = 5$, $2x + 2y = 6$, ...
- $3x + y = 4$, $3x + y = 5$, $3x + y = 6$, ...

Can you get any good conjectures?

Note that what we are really asking is how many integer lattice points lie between the intercepts.

3.3.1 Solution ideas

One way to think about this is the *distance* between solutions. Let's assume that the equation is $ax + by = c$, and $\gcd(a, b) = 1$. Then, using out technique from last time, from the solution (x_0, y_0) we get a new solution $(x_0 + b, y_0 - a)$, so the distance between any two solutions is, by the Pythagorean Theorem,

$$\sqrt{[(x_0 + b) - x_0]^2 + [(y_0 - a) + y_0]^2} = \sqrt{a^2 + b^2}\,.$$

Our strategy is to ask:

- How many times does that distance fit between the *intercepts* of the line?

Does that strategy make sense? It doesn't give an exact answer, but should give a good ballpark estimate.

Let's calculate these things. You may want to follow it $a = 3$, $b = 2$, $c = 4$.

- The intercepts are $\frac{c}{a}$ and $\frac{c}{b}$, respectively.
- Using the Pythagorean Theorem again, we see that the whole length available is
$$\sqrt{\left(\frac{c}{a}\right)^2 + \left(\frac{c}{b}\right)^2} = \frac{c}{ab}\sqrt{a^2 + b^2}\,.$$
- The *ratio* of this total length and the length between solutions is thus $\frac{c}{ab}$.

That's a nice pat answer. There are two problems with it, though!

1. There is no guarantee that $\frac{c}{ab}$ is an integer! In fact, it usually won't be. For instance, with $2x + 3y = 10$, $\frac{10}{2\cdot 3} \approx 1.67$. So should the number of points be bigger than or less than this?

2. Secondly, even so it's not clear what the precise connection between $\frac{c}{ab}$ and the actual number of points is. $2x+3y = 5$ has one, and $2x+3y = 7$ has one, but $2x + 3y = 6$ doesn't. Yet $\frac{c}{ab}$ is about equal to one for all three of these. In fact, the number of points is thus not even monotone increasing with respect to c increasing, which is rather counterintuitive.

We will have to deal with each of these situations.

3.3.2 Toward the full solution

We can deal with each of these problems. To do so, we introduce a new function:

Definition 3.3.2 (Greatest integer function)**.** The *greatest integer function* (also called the *floor* function) is the function which takes a real number x and returns the next integer below it (or equal to it). We notate it $\lfloor x \rfloor$.

Example 3.3.3. A few examples should suffice to understand it:

$$\lfloor 1.5 \rfloor = 1\ ,\ \lfloor 1 \rfloor = 1\ ,\ \lfloor 1.99 \rfloor = 1\ ,\ \lfloor 0.99 \rfloor = 0\ ,\ \lfloor -.01 \rfloor = -1\ .$$

Now let's use this to rectify our problems.

1. To take care of the integer problem, we will just consider $n = \left\lfloor \frac{c}{ab} \right\rfloor$, the greatest integer function applied to $\frac{c}{ab}$.

2. Secondly, we simply recognize that there isn't a nice formula. *On average*, we should expect n lengths between integer points along the line segment in question (and hence as many as $n + 1$ lattice points, since a partition of n intervals has $n + 1$ endpoints associated to it).

Rather than give a general formula, we examine individual cases to show what to expect. This applet helps.

```
@interact
def _(c=[5..12]):
    a = 2
    b = 3
    ym = c/b + 1
    xm = c/a + 1
    p = plot(-(a/b)*x+c/b,-1,xm, plot_points = 200)
    lattice_pts = [[i,j] for i in [0..xm] for j in [0..ym]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
    if mod(c,gcd(a,b))==0:
        line_pts = [coords for coords in lattice_pts if
            (coords[0]>0) and (coords[1]>0) and
            (a*coords[0]+b*coords[1]==c)]
        if len(line_pts)==0:
            pretty_print(html('Solutions␣to␣
                $%sx+%sy=%s$:'%(str(a),str(b),str(c))))
            pretty_print(html('No␣positive␣lattice␣points␣
                at␣all!'))
```

```
            show(p+plot_lattice_pts, figsize = [5,5], xmin
                = 0, xmax = xm, ymin = 0, ymax = ym)
        else:
            plot_line_pts = points(line_pts, rgbcolor =
                (0,0,1),pointsize=20)
            pretty_print(html('Solutions to
                $%sx+%sy=%s$:'%(str(a),str(b),str(c))))
            pretty_print(html('Number of positive lattice
                points = ' + str(len(line_pts))))
            show(p+plot_lattice_pts+plot_line_pts, figsize
                = [5,5], xmin = 0, xmax = xm, ymin = 0,
                ymax = ym)
    else:
        pretty_print(html('Solutions to
            $%sx+%sy=%s$:'%(str(a),str(b),str(c))))
        pretty_print(html('No positive lattice points at
            all!'))
        show(p+plot_lattice_pts, figsize = [5,5], xmin =
            0, xmax = xm, ymin = 0, ymax = ym)
```

Let's focus on the case where a and b are relatively prime.

1. How could the fewest lattice points appear? That's when both the x- and y-intercepts actually are lattice points, because they do not have *positive* coordinates. So if c/a and c/b are both integers, then we get precisely

$$n - 1 = \left\lfloor \frac{c}{ab} \right\rfloor - 1$$

   lattice points. This is the transition between $2x + 3y = 5$ to $2x + 3y = 6$.

2. What if just *one* of the intercepts is a lattice point? Then, beginning at that point, there is definitely room for the full n lengths to appear, and you're guaranteed to get n lattice points, because we just said the *other* intercept isn't a lattice point, so the nth one must appear before that point. So here the formula is just plain old

$$n = \left\lfloor \frac{c}{ab} \right\rfloor .$$

   Compare $2x + 3y = 8$ and $2x + 3y = 9$ to the others above.

3. Finally, if neither c/a nor c/b is an integer, then you get n *or* $n+1$ lattice points (remember, $n = \lfloor \frac{c}{ab} \rfloor$). There's no nice formula for this otherwise; but notice that with $2x+3y = 11$ you do get $\lfloor \frac{11}{2\cdot 3} \rfloor + 1 = 2$ positive lattice points! (And it jumps back down to $\lfloor \frac{12}{2\cdot 3} \rfloor - 1 = 1$ at $c = 12$!

For more details, see the excellent book *The Geometry of Numbers* [C.3.15].

If $\gcd(a, b) \neq 1$, it is not too hard to show that any such line with respect to lattice points is the same as a line $a'x + b'y = c'$ for which $\gcd(a', b') = 1$. Which line would that be?

3.4 Pythagorean Triples

3.4.1 Definition

There are a lot of other interesting questions that one can ask about pure integers, and polynomial equations they might satisfy (so-called Diophantine

equations). However, *answering* many of those questions will prove challenging without additional tools, so we will have to take a detour soon. But one such question is truly ancient, and worth exploring more in this chapter.

It is also quite geometric. We just used the Pythagorean Theorem above, but you'll note that we didn't really care whether the hypotenuse was an integer there. Well, when is it? More precisely:

Question 3.4.1. When are all three sides of a right triangle integers?

Definition 3.4.2. We call a triple of integers x, y, z such that $x^2 + y^2 = z^2$ a **Pythagorean triple**.

There isn't necessarily evidence that Pythagoras thought this way about them. However, Euclid certainly did, and so will we. For that matter, we should also think of them as x, y, z that fit on the quadratic curve $x^2+y^2 = z^2$, given z ahead of time.

Let's try this out for a little bit. When do we get a triple? (Keep in mind that we will always expect the triple $(z, 0, z)$ and $(0, z, z)$ where $0^2 + z^2 = z^2$, but that's not really what we are interested in.)

```
@interact
def _(z=(2,[1..100])):
    f(x,y)=x^2+y^2-z^2
    max = z
    p = implicit_plot(f,(x,-1,max),(y,-1,max),plot_points
        = 200)
    lattice_pts = [[i,j] for i in [0..max] for j in
        [0..max]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
    curve_pts = [coords for coords in lattice_pts if
        f(coords[0],coords[1])==0]
    if len(curve_pts)==0:
        show(p+plot_lattice_pts, figsize = [5,5],
            aspect_ratio=1)
    else:
        plot_curve_pts = points(curve_pts, rgbcolor =
            (0,0,1),pointsize=20)
        show(p+plot_lattice_pts+plot_curve_pts, figsize =
            [5,5], aspect_ratio=1)
```

3.4.2 Characterizing Pythagorean triples

It seems quite random for which z we get a Pythagorean triple existing! (We'll return to *that* question later.) Let's see what triples are even possible.

First, it turns out we really only need to worry about the case when x, y, z are all relatively prime to each other.

Definition 3.4.3. A Pythagorean triple with x, y, z mutually relatively prime is called a **primitive Pythagorean triple**.

Proposition 3.4.4. *Any Pythagorean triple with two numbers sharing a factor can be reduced to a primitive triple.*

Proof. If $x = x'a$ and $y = y'a$, for instance, then

$$x^2 + y^2 = (x')^2a^2 + (y')^2a^2 = z^2$$

which means that $a^2 \mid z^2$, and hence that $a \mid z$ as well. The other cases are similar. (One can prove the last statement with the gcd and Bezout as well, but I trust you believe it for now. See below in Proposition 3.7.1.) □

So let's consider just the case of primitive triples. In just a little while we will discover we have the proof of a result, Theorem 3.4.5.

We can start with very elementary considerations of even and odd. By the previous proposition, x and y can't both be even.

I claim they can't both be odd, either. For if they were, we would have $x = 2k + 1$ and $y = 2\ell + 1$ for some integers k, ℓ, and then

$$(2k+1)^2 + (2\ell+1)^2 = 4\left(k^2 + \ell^2 + k + \ell\right) + 2$$

But this contradicts Proposition 2.1.4 with respect to the remainder of a perfect square when divided by four.

So we may assume without loss of generality that x is odd and y is even, (which means z is odd).

3.4.2.1 An intricate argument

So we have $\gcd(x, y, z) = 1$ and x, z are odd and y is even. Now we will do a somewhat intricate, but familiar, type of argument about factorization and divisibility.

Let's rewrite our situation as

$$y^2 = z^2 - x^2 \,.$$

The right-hand side factors as

$$z^2 - x^2 = (z - x)(z + x) \,.$$

Certainly $z - x$ and $z + x$ are both even, so that $z - x = 2m$ and $z + x = 2n$. But since their product is a square (y^2), then that product $2m \cdot 2n = 4mn$ is also a perfect square. Since y is even, $y = 2k$ for some $k \in \mathbb{Z}$ and $y^2 = 4k^2$, so $mn = k^2$ is a perfect square.

Let's look at these mysterious factors $m = \frac{z-x}{2}$ and $n = \frac{z+x}{2}$. Are they relatively prime? Well, if they shared a factor, then $x = m + n$ and $z = n - m$ also share that factor. But $\gcd(x, z) = 1$, so there are no such factors and

$$\gcd\left(\frac{z-x}{2}, \frac{z+x}{2}\right) = \gcd(m, n) = 1$$

Now recall that y is even. Letting $y = 2j$, we see that $y^2 = 4mn$ yields $j^2 = mn$; however, this time m and n are relatively prime!

At this point we need what may seem to be an intuitive fact about squares and division; if coprime integers make a square when multiplied, then they are each a perfect square. (See Proposition 3.7.2.) So $m = p^2$ and $n = q^2$ for some integers (obviously coprime) p and q.

This clearly implies that $j^2 = p^2q^2$, so $y = 2pq$.

3.4.2.2 The punch line

Now we can put everything together.

$$z - x = 2p^2, \; z + x = 2q^2, \;\text{ and } y = 2pq \,.$$

That is:

Theorem 3.4.5 (Characterization of primitive Pythagorean triples)**.** *For a primitive triple x, y, z, we have*

$$z = p^2 + q^2,\ x = q^2 - p^2,\quad \text{and } y = 2pq\ .$$

Further, since x is odd, p and q cannot both be odd or both even.

Definition 3.4.6. We say two integers p, q have **opposite parity** if one is even and the other is odd, and we say they have the **same parity** otherwise.

Algorithm 3.4.7. *We can find* all *primitive Pythagorean triples by finding coprime integers p and q which have opposite parity, and then using this formula! And we get* all *Pythagorean triples by multiplying.*

It's really worth trying to find these *by hand*; it gives one a very good sense of how this all works.

Of course, you could generate some by computer as well …

```
n=10
Generators=[(p,q) for p in range(1,n) for q in
    range(p+1,n) if (gcd(p,q)==1) and not
    (mod(p,2)==mod(q,2))]
for pairs in Generators:
    x = pairs[1]^2-pairs[0]^2; y = 2*pairs[0]*pairs[1]; z
        = pairs[0]^2+pairs[1]^2
    pretty_print(html('$%s$ squared plus $%s$ squared is
        $%s$ squared - $%s$'%(x,y,z,x^2+y^2==z^2)))
```

Remark 3.4.8. One can find many infinite subfamilies of Pythagorean triples. A nice brief article by Roger Nelsen [C.6.18] shows that there are infinitely many Pythagorean triples giving *nearly* isoceles triangles (where the smaller sides are just one unit different). What families can you find?

3.4.3 Areas of Pythagorean triangles

3.4.3.1 Which areas are Possible?

Historically, one of the big questions one could ask about such Pythagorean integer triangles was about its *area*. For *primitive* ones, the legs must have opposite parity (do you remember why?), so the areas will be integers. (For ones which are not primitive, the sides are multiples of sides with opposite parity, so they are certainly also going to have an integer area.)

So what integers work? You all know one with area 6, and it should be clear that ones with area 1 and 2 can't work (because the sides would be too small and because $2, 1$ doesn't lead to a triple); can you find ones with other areas?

```
n=10
Generators=[(p,q) for p in range(1,n) for q in
    range(p+1,n) if (gcd(p,q)==1) and not
    (mod(p,2)==mod(q,2))]
for pairs in Generators:
    x = pairs[1]^2-pairs[0]^2; y = 2*pairs[0]*pairs[1]; z
        = pairs[0]^2+pairs[1]^2
    pretty_print(html('The primitive triple $%s$ gives a
        triangle of area $%s$'%((x,y,z),x*y/2)))
```

It is worth asking why there are no odd numbers in the list so far. In fact, we can prove quite a bit about these things.

Remember, x and y can be written as $x = q^2 - p^2$ while $y = 2pq$, for relatively prime opposite parity $q > p$. Then the area *must* be

$$pq(q^2 - p^2) = pq(q + p)(q - p)\,.$$

So can the area be odd?

Proposition 3.4.9. *In a primitive Pythagorean triple given by the formula in Theorem 3.4.5, the four factors of the area*

$$pq(q^2 - p^2) = pq(q + p)(q - p)$$

must all be relatively prime to each other.

Proof. We already know that p and q are coprime.

The factors p and $p + q$ must also share no factors, since any factor they share is shared by $(p + q) - p = q$, but $\gcd(p, q) = 1$. The same argument will work in showing that p and $q - p$ are, as well as q and either sum.

If $q + p$ and $q - p$ share a factor, since they are odd it must be odd, *and* it must be a factor of their sum and difference $2q$ and $2p$. Since the putative factor is odd, it is coprime to 2, and so we can use Proposition 2.4.6 to say that it is a factor of both p and q, which is impossible unless said factor is 1. □

So one could analyze a number to see if it is possible to write as a product of four relatively prime integers as a starting point. E.g. $30 = 2 \cdot 3 \cdot 5 \cdot 1$ is the only way to write 30 as a product of four such numbers (assuming no more than one of those is 1!), and since $q + p$ must be the biggest, we must set $q + p = 5$. Quickly one can see that $q = 3, p = 2$ works with this, so there is such a triangle. What are the sides?

This turns out to be a *very deep* unsolved problem. This news update from the American Institute of Mathematics gives some background on the **congruent number problem**, which asks the related question of which Pythagorean triangles with *rational* side lengths give integer areas. This page in particular is interesting from our present point of view.

3.4.3.2 Which areas are square?

But we can ask another question, which led Fermat to some of his initial investigations into this theory.

Question 3.4.10. Namely, when is the area of a Pythagorean triple triangle a perfect square?

```
@interact
def _(n=20):
    Generators=[(p,q) for p in range(1,n) for q in
        range(p+1,n) if (gcd(p,q)==1) and not
        (mod(p,2)==mod(q,2))]
    list = []
    for pairs in Generators:
        x = pairs[1]^2-pairs[0]^2; y =
            2*pairs[0]*pairs[1]; z = pairs[0]^2+pairs[1]^2
        if is_square(x*y/2):
            pretty_print(html('The␣primitive␣triple␣
                $%s,%s,%s$␣gives␣a␣triangle␣of␣square␣area␣
                $%s$'%(x,y,z,x*y/2)))
```

```
            list.append((x,y,z))
    if not list:
        pretty_print(html("No triangles of square area up
            to $p,q\leq %s$!"%(n,)))
```

If you'll notice, we don't see to be getting a lot of these. In fact, none. What would we need to do to investigate this?

Remember, x and y can be written as $x = q^2 - p^2$ while $y = 2pq$, for relatively prime opposite parity $q > p$. Then the area *must* be

$$pq(q^2 - p^2) = pq(q + p)(q - p)\,.$$

Now, in the previous section, we showed each of these quantities was relatively prime to each other. So if the area is *also* a perfect square, then since they are coprime, they themselves are all perfect squares!

Now we will do something very clever. It is a proof strategy, similar to something the Greeks used occasionally, and it is something Fermat used for many of his proofs, called **infinite descent**. We are going to take that (hypothetical) triangle, and produce a triangle with strictly smaller sides but otherwise with the same properties – including integer sides and square area! That means we could apply the same argument to our new triangle, and then the next one … but the Well-Ordering Principle (1.2.1) won't allow that. So the original triangle was impossible to begin with.

So let's make that smaller triangle!

Proposition 3.4.11. *If a primitive Pythagorean triangle has area a perfect square, we can create another one of strictly smaller hypotenuse length.*

Proof. We know that $q + p$ and $q - p$ are (odd) squares. Call them u^2 and v^2. That means that we can write u and v as $\frac{u+v}{2} + \frac{u-v}{2}$ and $\frac{u+v}{2} - \frac{u-v}{2}$ (which are integers since u and v are odd).

Letting $a = \frac{u+v}{2}$ and $b = \frac{u-v}{2}$, we have that $q + p = (a + b)^2$ and $q - p = (a - b)^2$. Then a little algebra shows that $q = a^2 + b^2$ and $p = 2ab$. These are both squares, so $a^2 + b^2 = q = c^2$ (!), which defines a triangle with area $\frac{ab}{2} = \frac{2ab}{4} = \frac{p}{4}$, another perfect square.

But $c < z$. This is because $z = q^2 + p^2 = \left(c^2\right)^2 + p^2 = c^4 + p^2$, so that unless $p = 0$, c is strictly less than z. But $p = 0$ doesn't give a triangle at all! So we have our strictly smaller triangle satisfying the same properties. This process could be continued infinitely often – hence the name. □

Corollary 3.4.12. *No difference of perfect fourth powers can be a perfect square. That is,*

$$v^4 - u^4 = t^2$$

cannot be solved in integers.

Proof. In the proof of the proposition, we really showed that there is no pair p and q of (coprime) squares such that $q^2 - p^2$ is also a perfect square t^2; that is what we started with, after all. So, if $p = u^2$ and $q = v^2$ we have that

$$v^4 - u^4 = t^2$$

is impossible. □

In Exercise 3.6.9 you will use this to prove the famous first case of Fermat's Last Theorem:

$$x^4 + y^4 = z^4$$

is not possible for any three positive integers x, y, z. See also Subsection 14.2.2.

3.5 Surprises in Integer Equations

This chapter has discussed linear and quadratic Diophantine equations. As you can see, even relatively simple questions become *much* harder once you have to restrict yourself to integer solutions. And doing it without any more tools becomes increasingly unwieldy.

But there is one final example of a question we can at least touch on. Recall that Pythagorean triples come, at their heart, from the observation that $3^2 + 4^2 = 5^2$. This is an interesting coincidence with close numbers. So too, we can notice that 3^2 and 2^3 are only one apart, and 5^2 and 3^3 are only two units apart.

```
@interact
def _(k=(1,[-5..25])):
    f(x,y)=y^2-x^3-k
    p = implicit_plot(f,(x,-3,3),(y,-6,6),plot_points =
        200)
    lattice_pts = [[i,j] for i in [-3..3] for j in [-6..6]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
    curve_pts = [coords for coords in lattice_pts if
        f(coords[0],coords[1])==0]
    if len(curve_pts)==0:
        show(p+plot_lattice_pts, figsize = [5,5],
            aspect_ratio=1)
    else:
        plot_curve_pts = points(curve_pts, rgbcolor =
            (0,0,1),pointsize=20)
        show(p+plot_lattice_pts+plot_curve_pts, figsize =
            [5,5])
    pretty_print(html("Solutions␣of␣$x^3+%s=y^2$␣in␣this␣
        viewing␣window"%(k,)))
```

This is known as **Bachet's equation** or the **Mordell equation**. Louis Mordell, an early 20th-century mathematician, proved that there are only *finitely* many integer solutions to this sort of equation for a given k. However, finding them all, or even some (!) turns out to be quite tricky, especially since many have *no* solution. See this link for some tables of what *is* known.

It turns out that this, too, has incredibly deep connections to a concept we will not investigate called **elliptic curves**; given their importance in cryptography and theory, that is enough reason to study them. However, it is independently interesting that there are some Mordell equations which are solvable by more elementary means, and in Section 15.3 there is the opportunity to do a few. Here are some examples to whet your appetite.

- The history of the solution $25 + 2 = 27$ for $k = 2$ is interesting. Bachet himself, in his translation and commentary on Diophantus, talked about *rational* solutions. Fermat asked the English mathematician John Wallis (of infinite product fame) whether there were other solutions, and implied there were no others. Euler proved this, but using some hidden assumptions so that the proof was incomplete. (See Fact 15.3.4.)

- Euler's proof in 1738 that $9 - 1 = 8$ was the only nontrivial solution to $k = -1$, however, is correct. He uses the same method of *infinite descent* we saw in Proposition 3.4.11. (He even shows that there aren't even any

other *rational number* solutions to $y^2 - 1 = x^3$, all in the midst of a paper actually about demonstrating Exercise 3.6.9.)

This is also related to a *very* old question which was called Catalan's conjecture, yet again related to these funny little coincidences. Namely:

Question 3.5.1 (Catalan's Conjecture)**.** Eight and nine are consecutive perfect powers; are there others?

```
@interact
def _(end_range=10):
    pretty_print(html("Solutions through numbers and
        powers $%s$"%end_range))
    print [(x,p,y,q) for x in range(1,end_range) for y in
        range(1,end_range) for p in range(2,end_range) for
        q in range(2,end_range) if x^p+1==y^q]
```

This *was* called Catalan's conjecture because, as of 2002, it is Mihailescu's Theorem! The history of this question goes back to the 1200s; [C.3.17] has a nice overview of many important pieces of its history, and Wolfram MathWorld has an accessible introduction.

3.6 Exercises

1. For each of the following linear Diophantine equations, either find the form of a general solution, or show there are no integral solutions.

- $21x + 14y = 147$
- $30x + 47y = -11$

2. Check the details in Subsection 3.1.4.

3. Find all *simultaneous* integer solutions to the following system of equations. (Hint: do what you would ordinarily do in high school algebra or linear algebra! Then finish the solution as we have done.)

- $x + y + z = 100$
- $x + 8y + 50z = 156$

4. Compute the number of *positive* solutions to the linear Diophantine equation $6x + 9y = c$ for various values of c and compare to the analysis we did above.

5. Explore the patterns in the *positive* integer solutions to $ax+by = c$ situation above. For sure I want you to do this for the ones I mention there, but try some others and see if you see any broader patterns!

6. Prove that any line $ax+by = c$ which hits the integer lattice but $\gcd(a, b) \neq 1$ is the same as a line $a'x + b'y = c'$ for which $\gcd(a', b') = 1$, and explain why that means that without loss of generality the first topic doesn't need any more explanations.

7. Find a primitive Pythagorean triple with at least three digits for each side.

8. Prove that 360 cannot be the area of a primitive Pythagorean triple triangle.

9. Find a way to prove that $x^4 + y^4 = z^4$ is not possible for any three positive integers x, y, z. (Hint: use Corollary 3.4.12; this one is harder.)

10. We already saw that if x, y, z is a primitive Pythagorean triple, then exactly one of x, y is even (divisible by 2). Assume that it's y, and then prove that y is divisible by 4.

11. Under the same assumptions as in the previous problem, prove that exactly one of x, y, z is divisible by 3. (Combined with the previous exercise, this proves that every area of a Pythagorean triple triangle is divisible by 6. Is it also true that exactly one of x, y, z is divisible by 5?)

12. A Pythagorean triple satisfies $x^2 + y^2 = z^2$. Explore patterns for triples of positive integers which satisfy $x^2 - xy + y^2 = z^2$. If Pythagorean triples correspond to right triangles, what sort of triangles do these triples correspond to?

13. Find a (fairly) obvious solution to the equation $m^n = n^m$ for $m \neq n$. Are there other such solutions?

14. Show that

$$\gcd(x, y)^2 = \gcd(x^2, xy, y^2)$$

which we use in Proposition 3.7.2. You can try this using the set of divisors definition of gcd, or using the definition $\gcd(a, b, c) = \gcd(\gcd(a, b), c)$.

15. Explore Bresenham's algorithm in print or online. What is the connection to this chapter? How do *non*-solutions to linear Diophantine equations relate to actual solutions, in this context?

3.7 Two facts from the gcd

Here are two facts that seem really obvious but do need proofs. All can be done just with the gcd; kudos go to users Math Gems and coffeemath at math.stackexchange.com for most of these clever arguments.

Proposition 3.7.1 (When perfect squares divide each other)**.** *For integers a, z it is true that*

$$a^2 \mid z^2 \implies a \mid z$$

Proof. First, let $d = \gcd(a, z)$. Then we can write $z^2 = a^2 \cdot k$ for some integer k, and immediately write

$$(z')^2 d^2 = (a')^2 d^2 k$$

for some integers z' and a', by definition of gcd. (That is, $z = z'd$ and $a = a'd$.)

Cancelling the d^2 (yes, we *do* assume this property of integers) yields

$$(z')^2 = (a')^2 k \, .$$

Since $\gcd(a', z') = 1$, we have $a'x + z'y = 1$ for some $x, y \in \mathbb{Z}$; now we substitute for 1 in $a' \cdot 1 \cdot x$ (!) to get

$$a'(a'x + z'y)x + z'y = 1$$

Now we have that $a'^2 x + z'(a'xy + y) = 1$, so that $\gcd((a')^2, z') = 1$ as well. But of course $a' \mid (z')^2$. Clearly if a positive number *is* a divisor, but their greatest *common* divisor is 1, then that number is going to have to be 1 by definition of divisors. So $a' = 1$. (If a' was negative, the same argument for $-a'$ shows $-a' = 1$, so really $a' = \pm 1$.)

Hence $a = a'd = \pm d$, which is a divisor of z, we have the desired result. □

Proposition 3.7.2 (When the product of coprime numbers is a square)**.** *If integers* $j^2 = mn$ *and* $\gcd(m, n) = 1$*, then* m *and* n *are also both perfect squares.*

Proof. First, we will need a general fact about gcds:

$$\gcd(x, y)^2 = \gcd(x^2, xy, y^2)$$

See Exercise 3.6.14.

We know that $1 = \gcd(m, n) = \gcd(m, n, j)$, so

$$m = m \cdot \gcd(m, n, j) = \gcd(m^2, mn, mj) = \gcd(m^2, j^2, mj)$$

Now we use the fact, so that

$$m = \gcd(m, j)^2$$

. That's a perfect square.

The same argument with n and j yields $n = \gcd(n, j)^2$. □

Chapter 4

First Steps with Congruence

Our next big goal is a better notion of how to deal with divisibility and remainders, one we are all familiar with. That is the notion of *congruence*!

We will begin by reviewing that notion, and start asking the kinds of questions that one will be able to ask with this notion.

4.1 Introduction to Congruence

Let's start by a little calculation. What is the remainder of 25 when divided by 6?

```
25 % 6
```

In general, the command `x % m` computes "x modulo m", which is to say the remainder of x when you divide by m.

An alternate way to do this is with the command `mod(x,m)`.

```
mod(25,6)
```

In a moment this will be more desirable, but for now it is less so, because it creates a different kind of Sage object.

Because of the division algorithm, we *know* that there is a *unique* such remainder. If we call it r (so that `r = x % m`), then $0 \leq r < m$, which is very important. However, lots and lots of different numbers can have the same remainder:

```
[ x % 6 for x in [1, 7, 13, 19, 25, -5, -11, 6001, -17]]
```

(See Sage note 4.6.2 for this type of list construction.)

In mathematics, what we often do in such a situation where structure is shared is connect things with a **relation**.

A relation is a very general notion, and basically it exists once you define it; however, we will not pursue this further. Our relation will be called **congruence**, and it is massively important. It is also relatively new! We essentially use the same definitions and notation that Gauss came up with just two centuries ago.

Definition 4.1.1 (Congruence)**.** We say that a **is congruent to** b, or

$$a \equiv b \pmod{n}$$

precisely if $n \mid (a - b)$. We call n the **modulus**.

Often we can prove a small helping statement, usually called a lemma.

Lemma 4.1.2 (Congruence-Remainder)**.** *Saying $a \equiv b \pmod{n}$ is exactly the same thing as saying a and b leave the same remainder when divided by n.*

Proof. We can sketch the proof. It is a good exercise (see Exercise 4.7.12) to fill in the details.

- Write $a = nq + r$ and $b = nq' + r'$. (Why is this possible, what are the various symbols?) Then there are two steps (why do they suffice?)
- First, if $r = r'$ then there is a k such that $a - b = nk$, which means $a \equiv b \pmod{n}$. (Why?)
- The other direction is showing if $a - b = nk$ for some $k \in \mathbb{Z}$, then $r = r'$. This is a little harder; try thinking about getting the remainders on one side, and what $r \neq r'$ would imply with respect to n.

□

Example 4.1.3. In our case, saying $25 \equiv 1 \equiv -5 \pmod{6}$ is the same as saying $25 = 4 \cdot 6 + 1$ and $1 = 0 \cdot 6 + 1$ and $-5 = -1 \cdot 6 + 1$.

It's fun to use congruence as a conceptual assistant. Here are some examples of our previous thinking recast in this way.

- Recall the fact about remainders when dividing by four, Proposition 2.1.4. This is just saying that the only possibilities are

$$x^2 \equiv 0 \text{ or } 1 \pmod{4}$$

- Could you try to use this idea to think of possible last (decimal) digits of a perfect square? Which modulus would be helpful?
- What about cubes; what remainders are possible modulo 4? What last digits are possible?

4.2 Going Modulo First

Okay, that's all fun. But we need power, too. Here's an example of such power. Even though I'm not physically present, I can do amazing computations! Let me compute $2^{1000000000} \pmod{3}$! I'll do it instantaneously.

Ready for the answer? It's 1!

Perhaps you don't believe an absent author. We can check it with Sage:

```
%time 2^1000000000 % 3
```

```
1
```

Sage note 4.2.1 (Timing your work)**.** In a Sage worksheet, putting `%time` before a command tells you how long it took. Putting `%timeit` instead runs the command many times and gives a 'best of' timing. (This does not universally work in the embedded cells in the web version of this book.)

Hmm, but that took more than a few milliseconds – strange that I could do it so fast!

Sage note 4.2.2 (Too big of numbers)**.** If I add one more zero, it will throw a very nasty error, like `MemoryError: failed to allocate 1250000024 bytes`, because things are too big. We can quickly go beyond the bounds of what our computers can do in number theory!

Now consider that I did this huge computation *instantaneously* in my head. Surely I must be full of brains, like the Scarecrow in L. Frank Baum's *Oz* books?

Of course, the reason is not that I am clever, but that congruence can be turned into arithmetic! Unlike the Wizard, I will give away my secret. I just used the following useful property.

Fact 4.2.3. *If $a \equiv b \pmod{n}$, then $a^m \equiv b^m \pmod{n}$ no matter how huge m is.*

Proof. See Exercises 4.7.4 and 4.7.5. □

Now I do my first congruence computation:

$$2 \equiv -1 \pmod{3} \text{ and } (-1)^{1000000000} = 1 \,,$$

the latter like all even powers of negative one. Ta-dah!

What I've done is *first* think of the original number as in the congruence, and then taken its power.

4.2.1 Computer diversion

Sage can verify this approach is much faster, and even for much bigger powers. Here we will need to use the `mod(x,m)` syntax:

```
print mod(2,3)^1000000000; print
    mod(2,3)^10000000000000000000000000000030
```

Even the presumptively very, very big latter computation should be as fast as your internet connection.

Sage note 4.2.4 (Give it a name)**.** We can use `print` to output multiple things per line of code, though it only prints them. Of course, writing this a lot can get annoying. So instead we can assign our 'modulo integer' a name, like `b`, and then just do as normal. This makes it easy to do lots of interesting tests.

```
b=mod(2000,31)
b,b^1000,b^2000,b^3000,b^4000
```

The last command is what prints out.

Sage note 4.2.5 (Making tuples)**.** In this case, we put commas between things so that all of the stuff in the last row prints out. It's in parentheses because the commas create a *tuple* (a special Python way of making a list with certain nice properties).

Sage note 4.2.6 (Types matter)**.** What was computed above is not a trick; I definitely couldn't do 2000^{1000}, or even 16^{1000}, in my head. How does Sage do it? The answer lies in the kind of thing `b` really is, which confirms that Sage is using modular numbers, not normal integers.

```
b=mod(2000,31)
b, type(b)
```

In Python, we can ask for the `type` of anything.In this case, we asked to print out `b` and then to print out its type, which is definitely not an ordinary integer, and can be manipulated much more efficiently.

This was a lot of computer business. The point is that if the computer thinks it's a good idea to just think of the remainder before you do any arithmetic, maybe we should too.

4.3 Properties of Congruence

There are two main sets of propositions that make this possible. The proofs are not hard, and you may skip them on a first reading.

Proposition 4.3.1 (Congruence is an equivalence relation)**.** *Congruence is* ***reflexive****,* ***symmetric****, and* ***transitive****, which combine to make it an* ***equivalence relation****.*

- *For any* $a \in \mathbb{Z}$*,* $a \equiv a \pmod{n}$*.*
- *If* $a \equiv b \pmod{n}$*, then* $b \equiv a \pmod{n}$*.*
- *If it happens that* both $a \equiv b$ *and* $b \equiv c \pmod{n}$*, then* $a \equiv c \pmod{n}$ *as well.*

See any intro-to-proof text for more background. For our purposes, this means all the things you know are true about equality are also true about congruence (with a particular modulus n *picked, of course).*

Proof. We will show each of the properties, leaving some pieces to the reader (Exercise 4.7.6).

- (Reflexive) For any $a \in \mathbb{Z}$, $a \equiv a \pmod{n}$.
  - The definition of congruence means we want to show $n \mid (a - a)$.
  - But $a - a = 0$. So we claim $n \mid 0$.
  - Any questions?
- (Symmetric) If $a \equiv b \pmod{n}$, then $b \equiv a \pmod{n}$.
  - For the reader!
- (Transitive) If it happens that *both* $a \equiv b$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$ as well.
  - The definition of congruence means we want to show *if* $n \mid (a - b)$ and $n \mid (b - c)$, *then* $n \mid (a - c)$ as well.
  - We use the definitions to see $a - b = nk$ and $b - c = n\ell$ for some $k, \ell \in \mathbb{Z}$.
  - Add these two equations to get $a - c = n(k + \ell)$, which is the definition of $n \mid (a - c)$.

□

Proposition 4.3.2 (Congruence arithmetic is well-defined)**.** *Congruence is* ***well-defined*** *with respect to addition and multiplication. That is, if $a \equiv c$ and $b \equiv d$ (modulo some fixed modulus n):*

- $a + b \equiv c + d$
- $ab \equiv cd$

Proof. Let $a \equiv c$ and $b \equiv d$ (modulo some fixed n):

- $a + b \equiv c + d$
  - There must exist k and ℓ such that $a = c + kn$ and $b = d + \ell n$.
  - So $a + b = c + kn + d + \ell n = (c + d) + (k + \ell)n$.
  - So $a + b$ and $c + d$ must have the same remainder modulo n.
- $ab \equiv cd$
  - For the reader; see Exercise 4.7.7.

□

Just below, in Section 4.4, we will see that these propositions and the following fact mean we are ready to roll with modulo and integers.

Fact 4.3.3. *Any set that has an equivalence relation on it can be broken up into disjoint subsets called* ***equivalence classes****. It can be useful to considered these classes as elements of a set of all such classes.*

Proof. We consider this to be background; see any intro-to-proof text. □

So we can break up $\mathbb{Z}$ into disjoint subsets, and use well-definedness. If I want to do a computation, I can pick any number with the same remainder modulo n, and it will still work fine. (Hopefully I pick an easier number to work with!) Here is an example.

Example 4.3.4. For instance, $2 \equiv 5 \pmod{n}$ is the same thing as saying $5 \equiv 2 \pmod{n}$, and if $2 \not\equiv 6 \pmod{n}$, then $5 \not\equiv 6 \pmod{n}$ either.

Or instead of computing $2 \cdot 2 \cdot 2 \cdot 2$ modulo 3, I might choose $-1 \cdot -1 \cdot -1 \cdot -1$ instead, getting the same answer (modulo 3)

It won't always be that clear-cut, but that is the general idea.

4.4 Equivalence classes

Let's make the previous discussion a bit more rigorous.

Definition 4.4.1. Assume throughout that we have fixed a modulus n.

- We call *any* number congruent to a a **residue** of a.
- We call the collection of *all* residues of a the **equivalence class** of a.

- We denote this class by the notation

$$[a] = \{\text{all numbers congruent to } a \text{ modulo } n\} .$$

(Sometimes this is notated $[a]_n$, but the modulus is nearly always evident from the context.)

Example 4.4.2. For instance, the equivalence class we started with is

$$[1] = \{1, 7, 13, 19, 25, -5, -11, 6001, \ldots\}$$

or perhaps better written as

$$\{1 + 6n \mid n \in \mathbb{Z}\} = [1] .$$

The point is you can choose your favorite number in an equivalence class to serve as a *representative* for all of them. To connect to before, for the congruence relation, there are only finitely many classes, otherwise the division algorithm would be meaningless. After, all, there are only n possible remainders.

Let's solve the 'magic trick' above using this concept in a slightly different way.

$$2^{1000000000} = (2^2)^{500000000} = 4^{500000000} \equiv 1^{500000000} = 1 .$$

Example 4.4.3. Here is something which is *not* a legal manipulation.

$$2^{1000000000} \equiv 2^1 \equiv 2 .$$

Even though $1000000000 \equiv 1$ modulo 3, clearly the end result is wrong, because $2^1 \not\equiv 1 \pmod 3$, which was the right answer. In general, we have only see you can reduce in the base of a power; nothing is said about the exponent! (Later we'll see how to do reduction in the exponent under controlled circumstances – with a *different modulus*.

4.4.1 Residue systems

As you saw above, knowing the 'right' residue can be very helpful. Because of this, we make two sets of them for general use. We call a set of integers with precisely one for each equivalence class a **complete residue system** or **complete set of residues** for a given modulus.

- Usually, we just use the 'normal' remainders; this is called the set of **least nonnegative residues**. This is just what you think it is; for $n = 6$, it is $\{0, 1, 2, 3, 4, 5\}$, representing the set of equivalence classes $\{[0], [1], [2], [3], [4], [5]\}$. They are easy to think of and understand.
- The problem is that they get big! To calculate $4^{20} \pmod 6$, I need to reduce a lot, e.g.

$$4^{20} \equiv (4^2)^{10} \equiv 16^{10} \equiv 4^{10} \equiv (4^2)^5 \equiv 16^5 \equiv 4^5 \equiv (4^2)^2 \cdot 4 \equiv 4^2 \cdot 4 \equiv 4 \cdot 4 \equiv 4$$

It's at least a little easier on the ol' noggin to mostly use $4 \equiv -2 \pmod 6$ like this:

$$4^{20} \equiv (-2)^{20} \equiv ((-2)^2)^{10} \equiv 4^{10} \equiv (-2)^{10} \equiv ((-2)^2)^5 \equiv 4^5$$
$$\equiv (-2)^5 \equiv ((-2)^2)^2 \cdot (-2) \equiv 4^2 \cdot (-2) \equiv (-2)^3 \equiv -8 \equiv 4 .$$

Notice in the second one I never broke single digits! So we sometimes use the set of **least absolute residues**, the collection of representatives of each class which are closest to zero. In this case the least absolute residues are $\{-2, -1, 0, 1, 2, 3\}$ for $\{[4], [5], [0], [1], [2], [3]\}$.

4.5 Why modular arithmetic matters

4.5.1 Starting to see further

This has been fun and all. But *why* are we doing all this? There are two reasons.

The first is practical. Simply put, modular arithmetic makes it much easier to solve certain otherwise very difficult problems about integers, because we can check whether things are possible when we look at things just modulo n. For instance, we can prove things like:

- "The polynomial $x^3 - x + 1$ has no integer roots".
- "The curve $x^3 = y^2 - 7$ has no lattice points".

Most practical of all are the rules for using modular arithmetic.

- First off, always *first* reduce modulo n, then do our arithmetic (add, multiply, exponentiate). We have seen lots of examples of this.
- Secondly, always use whichever *residue* of a number modulo n is convenient for our purposes.

  For example, to add $[22] + [21]$ modulo 23 it might be smarter to use the residues $-1 \in [22]$ and $-2 \in [21]$. The answer $[-3]$ is of course the same as $[22 + 21] = [43] = [20]$ modulo 23.

```
mod(22,23)+mod(21,23)==mod(-3,23)
```

Sage note 4.5.1 (Checking equality)**.** We can check if two numerical expressions are equal using ==.

There are a few things to be aware of when doing this, of course. One very important such caveat is that with exponentiation, you can only replace the *base* with something in the same congruence class. Just to make sure you get this, on your own compare

$$2^3 \pmod 5,\ 7^3 \pmod 5,\ \text{ and } 2^8 \pmod 5\ .$$

These are quite different.

Referring to our earlier wording, we can assume $[a]^n$ is well-defined, but there is no guarantee that $a^{[n]}$ makes any sense.

The second reason for doing modular arithmetic is theoretical. We get a *new number system*! (See Chapter 8.) It's a number system which has new problems, new solutions, and new things to explore. And that's what we'll be doing from now on.

4.5.2 Taking powers

As one example of how modular arithmetic might matter a bit, let's examine the following algorithm for taking ridiculously high powers of numbers (modulo n). We first need the following interesting fact.

Fact 4.5.2. *For any integer a:*

1. $a^{2^1} = a^2$

2. $a^{2^2} = (a^2)^2$

3. $a^{2^3} = (a^{2^2})^2$

In general,

$$a^{2^n} = \left(a^{2^{n-1}}\right)^2$$

That is to say, each "power of a to a power of 2" is the square of the previous *"power of a to the previous power of 2".*

Proof. What does a^{2^n} even mean? By definition,

$$2^n = 2^{n-1} \cdot 2 = 2^{n-1} + 2^{n-1} ,$$

so a^{2^n} is the same as

$$a^{2^{n-1}+2^{n-1}} = a^{2^{n-1}} \cdot a^{2^{n-1}} = \left(a^{2^{n-1}}\right)^2$$

□

Example 4.5.3. In this case, it will be easier to do examples before stating the algorithm. To compute x^{20}, first we see that 16 is the highest power of 2 less than 20.

- Compute x^2 modulo n.
- Square *that* for $(x^2)^2 = x^{2^2} = x^4$ (modulo n).
- Then square twice more for $x^{2^3} = x^8$ and $x^{2^4} = x^{16}$; we reduce modulo n at each point.

Now write x^{20} as x to a *sum of powers of* 2;

$$x^{20} = x^{16+4} = x^{2^4+2^2} = x^{2^2} \cdot x^{2^4}$$

Then do this final multiplication modulo n as well. You might want to try it to see you get the same thing.

Example 4.5.4. Now let's get really explicit, and calculate $2^{23} \pmod{11}$. First,

$$23 = 2^4 + 2^2 + 2 + 1, \text{ so } 2^{23} = 2^{2^4} \cdot 2^{2^2} \cdot 2^2 \cdot 2 .$$

Now let's get the powers of 2 needed:

$$2^2 \equiv 4 \pmod{11},\ \left(2^2\right)^2 = 4^2 \equiv 5 \pmod{11},$$

$$\left(2^4\right)^2 = 5^2 \equiv 3 \pmod{11}, \text{ and } \left(2^8\right)^2 = 3^2 \equiv 9 \pmod{11}$$

So we get, as a computation one can do completely without a calculator,

$$2^{2^4} \cdot 2^{2^2} \cdot 2^2 \cdot 2 \equiv 9 \cdot 5 \cdot 4 \cdot 2 \equiv 18 \cdot 20 \equiv 7 \cdot 9 \equiv 63 \equiv -3 \equiv 8 \pmod{11}$$

Algorithm 4.5.5. *In general, we can compute x^k modulo n:*

- *Write the exponent $k = \sum_{i=1}^{\ell} k_i 2^i$, where each $k_i = 0$ or 1. (This is called the binary representation of k.)*
- *Compute x^2, x^4, x^8, and so forth as above, each time reducing modulo n.*

- *Multiply* $\prod_{i=1}^{\ell} x^{k_i 2^i}$ *together as in the examples above. Obviously, if* $k_i = 0$ *(such as for* $i = 3$ *in the* x^{20} *example) you skip it, as it just contributes one to the product.*

Those interested in efficiency should note that this requires roughly two times the number of binary digits of your number operations, or about $2\log_2(n)$ operations, as opposed to normal powers which might require n operations; in addition, you only deal with numbers at most size n^2, as opposed to gigantic ones, when you mod out after each step, so it requires very little memory.

4.6 Toward Congruences

Question 4.6.1. What are the possible last digits of a perfect cube? (This was touched on at the end of Section 4.1.)

We can think of this more systematically now. For instance, if the last digit of $x > 0$ is 3, then $x = 10m + 3$ for some integer m. That is, $[x] = [3] \pmod{10}$. So the cube would look like

$$x^3 = (10m+3)^3 = 1000m^3 + 900m^2 + 270m + 27 = 10(\text{ stuff } + 2) + 7$$

This would presumably have last digit 7.

We can ask Sage to answer this for all possible last digits very quickly:

```
[mod(i,10)^3 for i in [0..9]]
```

Sage note 4.6.2 (List comprehensions)**.** This programming structure is known as a **list comprehension**. Think of it as set builder notation

$$\{i^3 \pmod{10} \mid 0 \leq i < 10\}$$

That's the set of all cubes modulo `i`, for `i` between `0` and `9`. Sage replaces `0..9` with the integers from 0 to 9.

If you check, what this is doing is getting the (least nonnegative) residue modulo 10 of the cube of every possible last digit. Notice that we also *get* every possible last digit.

It's possible to think of this more generally. Since we just said the last digit is all we cared about, we could think of this as answering a related kind of question. For all last digits d, is there an x such that the following works?

$$x^3 \equiv d \pmod{10}$$

Definition 4.6.3. Any (integer) *equation* with congruence in place of equality is called a **congruence**.

As a result, the previous calculation says that there is a solution to the congruence $x^3 \equiv d \pmod{10}$ for all possible d. Another way to say this is that every number (equivalence class) modulo 10 has a cube root. For instance, the cube root of [7] is [3].

This is definitely not true in $\mathbb{Z}$; the usual cube root of 7 (where $7 \neq [7]$) is not even rational! This exemplifies the following fact.

Fact 4.6.4. *Things which are false for the integers might be true in modular arithmetic.*

However, it is worth thinking about the following, though I will leave "things" vague.

Fact 4.6.5. *Things which are true for the integers are normally true in modular arithmetic.*

Now let's try the same question again, but with a different modulus.

```
[mod(i,4)^3 for i in [0..3]]
```

This seems to imply that every equivalence class modulo 4 "has a cube root" except [2].

This is suggestive. Maybe the right generalization is to ask this.

Question 4.6.6. Given a modulus n, when is there a solution to

$$x^3 \equiv d \pmod{n}$$

Or, for what moduli does d have a cube root modulo n?

Once we've opened things up to one such congruence, the sky's the limit.

Question 4.6.7. Over the integers, there are only two solutions to $x^2 = x$, the familiar $x = 0$ and $x = 1$. This leads to another natural question we can ask in modular arithmetic.

Namely, what are solutions to the congruence

$$x^2 \equiv x \pmod{n}$$

for different moduli n?

Sage can help us explore this sort of question.

```
@interact
def _(n=(2,[0..100])):
    list=[x for x in [0..n-1] if (mod(x,n)==mod(x,n)^2)]
    pretty_print(html("The solutions to the congruence
        $x^2\equiv x$ (mod $%s$)"%(n,)))
    pretty_print(html("are "+str(list)))
```

Often, it seems we get the same answers as over the integers. But not always! Can you try to conjecture for which n we *do* get the same answer? (See Exercise 4.7.14.)

We begin to see that there are two aspects of solving congruences, which will come up again and again for us.

- Solving a given congruence
- Figuring out for which moduli a congruence has solutions (or how many or …)

Much of the course will return to these ideas, such as sooner in Chapters 5 and 7 and later in Chapter 17.

4.7 Exercises

1. Give the least absolute residues and the least nonnegative residues for $n = 21$.

2. Prove that 13 divides $145^6 + 1$ and 431 divides $2^{43} - 1$ *without* a computer (but definitely using congruence).

3. Compute $7^{43} \pmod{11}$ as in Subsection 4.5.2 *without* using Sage or anything that can actually do modular arithmetic. (You should never have to compute a number bigger than $(11-1)^2 = 100$, so it shouldn't be too traumatic.)

4. Use the properties of congruence (in Proposition 4.3.1) or the definition to show that if $a \equiv b \pmod{n}$, then $a^3 \equiv b^3 \pmod{n}$.

5. Use the properties of congruence (in Proposition 4.3.1, *not* the definition) and induction to show that if $a \equiv b \pmod{n}$, then $a^m \equiv b^m \pmod{n}$ for any positive m.

6. Finish the details of proving Proposition 4.3.1, especially the second part (symmetric).

7. Finish the details of proving Proposition 4.3.2.

8. Find and prove what the possible last decimal digits are for a perfect square.

9. Prove that if the sum of digits of a number is divisible by 3, then so is the number. (Hint: Write 225 as $2 \cdot 10^2 + 2 \cdot 10 + 5$, and consider each part modulo 3.)

10. Prove that if the sum of digits of a number is divisible by 9, then so is the number.

11. For which positive integers m is $27 \equiv 5 \pmod{m}$?

12. Complete the proof that having the same remainder when divided by n is the same as being congruent modulo n.

13. Find some a and n such that $a^n \pmod{6}$ equals $a^{n+6} \pmod{6}$, where $a \not\equiv 0, 1$ and $n \neq 0, 1$. Then try to find an example where they are *not* equal.

14. Explore, using the interact after Question 4.6.7 or 'by hand', for *exactly* which moduli n the only solutions to $x^2 \equiv x \pmod{n}$ are $x = [0]$ and $x = [1]$.

Chapter 5

Linear Congruences

There are many questions one can ask of the integers, and in the preceding material we have already encountered many, especially those asking for solutions of simple equations in one or two variables.

One can ask very similar questions (and many more) about the integers modulo n. So we will focus on *congruences*, which are simply equations modulo n (see Definition 4.6.3). To exemplify this, consider the following similar ideas:

- $2x + 3y = 5$ (solutions are pairs of integers)
- $2x + 3y \equiv 5 \pmod 7$ (solutions would be pairs of *equivalence classes* $[x], [y]$ modulo 7)
- $2x + 3y \equiv 5 \pmod n$ for any particular n (solutions would be triplets $[x], [y], n$, since it would depend on n)

Try comparing solutions to these by hand; what is similar about them, what is not?

In one sense these are actually a big improvement in the level of difficulty. After all, you just have to try x, y from 0 to 6 (the least nonnegative residues) in the congruence $2x + 3y \equiv 5 \pmod 7$.

On the other hand, if the congruence was modulo $n = 10^{100}$, that would be less desirable, especially if the techniques for $\mathbb{Z}$ proved not to be useful with a congruence.

If we slapped an x^2 in the middle of the congruence, it might very hard indeed to solve quickly. So in this chapter, we will stay focused on the simplest case, of the analogue to linear equations, known as **linear congruences** (of one variable). This includes systems of such congruences (see Section 5.3).

5.1 Solving Linear Congruences

Our first goal to completely solve *all* linear congruences $ax \equiv b \pmod n$. The most important fact for solving them is as follows.

Proposition 5.1.1.

$$ax \equiv b \text{ (mod } n\text{) has a solution precisely when } \gcd(a, n) \mid b\ .$$

Proof. The proof of this is pretty straightforward, as long as we recall when linear *Diophantine* (integer) equations have solutions.

The following are clearly equivalent:

- Solutions x to $ax \equiv b \pmod n$

- Solutions x to $n \mid ax - b$
- Solutions x, y to $ax - b = ny$
- Solutions x, y to $ax - ny = b$

And we know from Theorem 3.1.1 that this final equation has solutions precisely when $\gcd(a, n) \mid b$. □

Before going on, test yourself by checking which of the following four congruences has a solution and which ones don't.

- $7x \equiv 8 \pmod{15}$
- $6x \equiv 8 \pmod{15}$
- $7x \equiv 8 \pmod{14}$
- $6x \equiv 8 \pmod{14}$

5.1.1 The nitty-gritty of solving

Just like in linear algebra or calculus, though, it's not enough to know *when* you have solutions; you want to actually be able to *construct* solutions. If possible, one wants to construct *all* solutions. In this case, we can do it.

Proposition 5.1.2. *If we can construct one solution to the linear congruence $ax \equiv b \pmod{n}$, we can construct all of them.*

Proof. Consider the proof of Proposition 5.1.1 above. We don't care about y (other than that it exists, and it does). So if we have *one* solution to the congruence, that is the same as having a solution x_0, y_0 to the equation $ax - ny = b$.

But we already know what solutions to that look like, from Theorem 3.1.1. Looking just at the x components, the solutions are

$$x_0 + \frac{n}{d}t \qquad t \in \mathbb{Z} \text{ where } d = gcd(a, n) \ .$$

This argument also gives us the exact number of solutions, because letting t go from 0 to $d - 1$ will give all different solutions. □

Example 5.1.3. Let's solve

$$12x \equiv 9 \pmod{15} \ .$$

Here, $\gcd(a, n) = 3$ so we will have 3 solutions, all separated by $\frac{n}{d} = \frac{15}{3} = 5$.

We need one solution first. Trying by guess and check small values gives us

- $12(1) = 12 \not\equiv 9$,
- but $12(2) = 24 \equiv 9 \pmod{15}$.

So we may take $x = 2$ as our x_0. Then we add 5 to each of these, and we see that $x = [2], [7], [-3]$ all work.

Alternately,

$$2 + 5t, \ t \in \mathbb{Z}$$

is the general solution.

5.2 A Strategy For the First Solution

The previous proposition always works. However, it can be very tedious to find that *first* solution if the modulus is bigger. This section is devoted to strategies for *simplifying* a congruence so that finding such a solution is easier.

Fact 5.2.1 (Strategies that work for simplifying congruences)**.** *We have two main types of simplification we can do. First, there are two types of cancellation we can use.*

- *If a, b, and n all are divisible by a common divisor, we can cancel that out (keeping in mind that we still will need our final solution to be modulo n).*
- *If a and b share a common divisor which is coprime to the modulus, we can cancel it from a, b (only).*

See Propositions 5.2.4 and 5.2.5 for precise statements and proofs.

Secondly, there are two counterintuitive ways that may lead to a simpler congruence after taking remainders.

- *We could* multiply *a and b by something coprime to n. If, after reducing modulo n, that makes a or b smaller, then that was a good idea!*
- *We can* add *some multiple of n to b. Again, if that happens to make a and (the new) b share a factor, then that was a good idea!*

These steps may be applied in any order, though typically the first two are done as often as possible.

Example 5.2.2 (A big example)**.** Let's do a big problem exemplifying all the steps.

$$\text{Solve } 30x \equiv 18 \pmod{33} .$$

1. First, note that all three of the coefficients and modulus are divisible by 3. So right away we should simplify by dividing by 3. *But keep in mind* that our final solution will need to be modulo 33, not modulo eleven! We should still end up with $\gcd(30, 33) = 3$ total solutions, and if we don't, we have messed up somewhere.
2. Now we have $10x \equiv 6 \pmod{11}$. (Again, although this will have one solution modulo 11, we will need to get the other two solutions modulo 33.) Since 10 and 6 are both divisible by 2, and since $\gcd(2, 11) = 1$, we can divide the coefficients (not modulus) by 2 without any other muss.
$$5x \equiv 3 \pmod{11}$$
3. So take $5x \equiv 3 \pmod{11}$, and let's try to replace 3 by *another number congruent to 3 modulo 11* which would allow me to use the above steps again.
   - I could try $3 + 11 = 14$, but that gives
   $$5x \equiv 14 \pmod{11}$$
   and 14 doesn't share a divisor with 5 (from the $5x$).

- If I try $3 + 22 = 25$, giving

$$5x \equiv 25 \pmod{11}$$

then 25 does share a divisor with 5.

4. Now I can go back and reduce $5x \equiv 25 \pmod{11}$ to

$$x \equiv 5 \pmod{11}$$

And that's the answer!

5. Or is it? Remember in the first step that we started modulo 33, and that all the answers will be equivalent modulo 11. So we see that

$$x = 5 + 11t \text{ for } t \in \mathbb{Z}$$

will be the answer, which is the three equivalence classes $\{[5], [16], [27]\}$.

Does it check out?

```
[mod(30*x,33)==18 for x in [5,16,27]]
```

One final observation is that we avoided trial and error as long as possible. At various points we could have done so, but $x = 1$ and $x = 2$ wouldn't have worked right away, and I am lazy…

Example 5.2.3. Let's finish the previous example gain, but using the other possible counterintuitive step. That was the trick to multiply a and b by something which would reduce; ideally it would reduce $[a] \equiv [1]$.

- We were at $5x \equiv 3 \pmod{11}$.
- Multiplying $a = 5$ and $b = 3$ by 9, which is coprime to 11, gives us

$$45x \equiv 27 \pmod{11} .$$

- This reduces to $x \equiv 5$, and gives the same answer as before (provided we remember to get all possible answers *modulo 33*).

These should have solutions; try completely solving one of these on your own now, before moving on. The exercises provide other interesting practice.

- $7x \equiv 8 \pmod{15}$
- $6x \equiv 8 \pmod{14}$

Here are formal statements and proofs of the propositions we used.

Proposition 5.2.4 (Canceling, Part I)**.** *If $d \neq 0$, then $ad \equiv bd \pmod{nd}$ precisely for the same a, b, n as when $a \equiv b \pmod{n}$.*

Proof. Like many such proofs, you basically follow your nose.

- We write $ad \equiv bd \pmod{nd}$ as $ad - bd \mid nd$, or $ad - bd = k(nd)$ for some $k \in \mathbb{Z}$ We rewrite this as $d(a - b) = d(kn)$.
- Since $d \neq 0$,

$$d(a - b) = d(kn) \text{ is equivalent to saying } a - b = kn ,$$

which is of course by definition saying that $a \equiv b \pmod{n}$.

- Since all steps were equivalences, both statements are equivalent.

□

Proposition 5.2.5 (Canceling, Part II)**.** *If $d \neq 0$ and $\gcd(d, n) = 1$, then $ad \equiv bd \pmod{n}$ precisely for the same a, b, n as when $a \equiv b \pmod{n}$.*

Proof. We'll only sketch the proof; see Exercise 5.6.2.

- Use the definitions as above, starting with the *ad* situation.
- You should have that n divides some stuff, which is itself a product of d and other stuff.
- We had a proposition about coprimeness and division; what remains should yield us $a \equiv b \pmod{n}$

□

5.3 Systems of Linear Congruences

Here are three interesting problems which may seem totally unrelated at first:

- You have lots of volunteers at a huge campaign rally. Because you are very efficient at moving them, and you want to gauge how to group them when dispatching them to different size venues, you line them up in rows. When you do it by fives (with one left over), by sixes (two left over), and by sevens (with three left over). How many are there total?
- You're an ancient sky watcher, and have discovered that three heavenly bodies come to the region of the sky you care about with great regularity. Comet 1 comes every five years, starting next year. Comet 2 comes every six years, starting two years from now. Comet 3 comes every seven years, starting three years from now. When will they all come in the same year?
- You like math a lot. You want to know what integers x simultaneously solve the following three linear congruences:
  - $x \equiv 1 \pmod{5}$
  - $x \equiv 2 \pmod{6}$
  - $x \equiv 3 \pmod{7}$

  Can you find an answer to these by trial and error?

5.3.1 Introducing the Chinese Remainder Theorem

In Section 5.2, we were able to solve any *one* linear congruence completely. It's a good feeling.

But we know that this is a pretty restricted result. If you've had a course in linear algebra, you've tried to solve big systems over the reals or complex numbers; sometimes in real-life operations research problems, there can be hundreds of thousands of linear equations to solve simultaneously!

It turns out this is true for modular arithmetic too, especially in encryption standards. Can we solve a *system* of linear congruences? Of course, one could ask a computer to do it by simply checking all possibilities.

```
@interact(layout=[['a_1','n_1'],['a_2','n_2'],['a_3','n_3']])
def _(a_1=('\(a_1\)',1), a_2=('\(a_2\)',2),
    a_3=('\(a_3\)',3), n_1=('\(n_1\)',5),
    n_2=('\(n_2\)',6), n_3=('\(n_3\)',7)):
    try:
        answer = []
        for i in [1..n_1*n_2*n_3]:
            if (i%n_1 == a_1) and (i%n_2 == a_2) and
                (i%n_3 == a_3):
              answer.append(i)
        string1 = "<ul><li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li>"%(a_1,n_1)
        string2 = "<li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li>"%(a_2,n_2)
        string3 = "<li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li></ul>"%(a_3,n_3)
        pretty_print(html("The␣simultaneous␣solutions␣to␣
            "))
        pretty_print(html(string1+string2+string3))
        if len(answer)==0:
            pretty_print(html("are␣none"))
        else:
            pretty_print(html("all␣have␣the␣form␣"))
            for ans in answer:
                pretty_print(html("$%s$␣modulo␣
                    $%s$"%(ans,n_1*n_2*n_3)))
    except ValueError, e:
        pretty_print(html("Make␣sure␣the␣moduli␣are␣
            appropriate␣for␣solving!"))
        pretty_print(html("Sage␣gives␣the␣error␣message:"))
        pretty_print(html(e))
```

As one might expect, this is not the most promising solution strategy. If you dig into the code a bit you'll see that many cases aren't even treated properly, which could be very tedious to catch.

However, in considering systems of congruences, there is a famous theorem. This kind of simultaneous solution was apparently first considered by the Chinese mathematician Sun Tzu, about the same time as the late Greek mathematicians were coming up with what we now call Diophantine equations. A *very full* solution (see Subsection 5.5.1) was given by Qin Jiushao in the 13th century and rediscovered only in the 19th century in the West.

Theorem 5.3.1 (Chinese Remainder Theorem)**.** *Consider a* general *system of k (linear) congruences:*

- $x \equiv a_1 \pmod{n_1}$
- $x \equiv a_2 \pmod{n_2}$
- ...
- $x \equiv a_k \pmod{n_k}$

where all the n_i are mutually coprime. In this case, we have an algorithm *for solving the system.*

Proof. This will be done in a completely constructive fashion in Subsection 5.4.1. □

The name comes from the provenance, and is often abbreviated CRT. Whether any actual Chinese rulers used it to decide how many troops they had by lining them up in threes, fours, fives, etc. is questionable. However, many of the example problems in Qin's text mention divination, alignment of different calendars, and the like, so we can assume such problems were of practical interest as well as theoretical, even at that time. Similar questions of astronomical/astrological importance pepper the history of mathematics.

Finally, note that one can also go much further and do linear algebra modulo n, and this is a lot of what modern cryptography is about, not to mention the modern hard-core computational number theory Sage was largely invented to help do. We can't do everything in this text, but you should be aware that *everything* done in linear algebra has very interesting modulo n counterparts, as part of the theme of number theory showing the unity of mathematics.

5.3.2 The inverse of a number

To do this justice, we need a very useful preliminary concept.

Definition 5.3.2 (The Inverse of a Number)**.** The **inverse of a number** a modulo n is the least nonnegative solution of the congruence

$$ax \equiv 1 \pmod{n}$$

.

Example 5.3.3. For example, the inverse of 26 modulo 31 is the least nonnegative solution of

$$26x \equiv 1 \pmod{31}.$$

This is called the **inverse** because you can think of the solution as 26^{-1}, or $\frac{1}{26}$, in the numbers modulo $n = 31$.

Note that there is not always an inverse! Here are some questions to ponder regarding inverses.

Question 5.3.4.

- What connection do a and n need if we expect there to exist an inverse of a modulo n?
- How many inverses modulo n should a have, assuming it has one at all?

As a first step, try to find inverses to all the number you can modulo 10. Then do it again modulo 11.

The following Sage command computes the "inverse of 26 modulo 31".

```
inverse_mod(26,31)
```

Sage note 5.3.5 (Getting interactive Sage help)**.** You can look for more information on Sage commands (in a normal Sage session) by using question marks; try `inverse_mod`? and `inverse_mod`?? in a Sage notebook, command line interface, or SageMath Cloud. (This is not supported when embedded in a web page as in your text.)

The point is that this is definitely something we can compute, using the methods of last time of solving solitary linear congruences.

5.4 Using the Chinese Remainder Theorem

We will here present a *completely* constructive proof of the CRT (Theorem 5.3.1). That is, we will not just prove it can be done, we will show how to get a solution to a given system of linear congruences.

Keep in mind that this is a procedure that works. It may have a number of steps, but its power is not to be underestimated. After some careful examples, we'll see some other uses.

5.4.1 Constructing simultaneous solutions

Remember that we are trying to solve the system of equations $x \equiv a_i \pmod{n_i}$. It is important to confirm that all n_i are coprime in pairs. Then the following steps will lead to a solution. You will find basically this proof in any text; I use the notation in [C.1.1].

1. First, let's call the product of the moduli $n_1 n_2 \cdots n_k = N$.

2. Take the quotient N/n_i and call it c_i. It's sort of a "complement" to the ith modulus within the big product N.

3. Now find the inverse of each c_i modulo n_i. That is, for each i, find a solution d_i such that
$$c_i d_i \equiv 1 \pmod{n_i}$$
Notice that this is *possible*. You can't find an inverse modulo any old thing! But in this case, c_i is the product of a bunch of numbers, all of which are coprime to n_i, so it is also coprime to n_i, as required.

4. For each i, multiply the three numbers $a_i \cdot c_i \cdot d_i$.

5. Now we evaluate each of these products (indexed by i) modulo the various n_j. That looks bad, but most things cancel:

   - By definition, each c_j is divisible by n_i (except for c_i itself), so modulo n_i the product is
   $$a_j c_j d_j \equiv 0 \pmod{n_i} .$$
   - The product
   $$a_i c_i d_i \equiv a_i \cdot 1 \equiv a_i \pmod{n_i}$$

6. Now add all these products together to get our final answer,
$$x = a_1 c_1 d_1 + a_2 c_2 d_2 + \cdots + a_k c_k d_k .$$
For each n_i, we can do the sum modulo n_i too; the previous step shows this sum is
$$x \equiv 0 + 0 + \cdots + a_i + \cdots + 0 \pmod{n_i} .$$
So this is definitely a solution.

7. Any other solution x' has to still fulfill $x' \equiv a_i \equiv x \pmod{n_i}$, so $n_i \mid x' - x$ for all moduli n_i. Since all n_i are relatively prime to each other, $N \mid x' - x$ too (if $a \mid c$ and $b \mid c$ and $\gcd(a, b) = 1$, then $ab \mid c$). So $x' \equiv x \pmod{N}$, which means x is the *only* solution modulo N!

Clearly this needs an example.

Example 5.4.1 (A first CRT example)**.** Let's look at how to solve our original system using this method.

- $x \equiv 1 \pmod 5$
- $x \equiv 2 \pmod 6$
- $x \equiv 3 \pmod 7$

We'll follow along with each of the steps in Sage.
First, I'll make sure I know all my initial constants.

```
n_1, n_2, n_3 = 5,6,7
a_1, a_2, a_3 = 1,2,3
N = n_1*n_2*n_3
print n_1, n_2, n_3
print a_1, a_2, a_3
print N
```

Next, I'll write down all the c_i, the complements to the moduli, so to speak. Remember, $c_i = N/n_i$.

```
n_1, n_2, n_3 = 5,6,7
a_1, a_2, a_3 = 1,2,3
N = n_1*n_2*n_3
c_1,c_2,c_3 = N/n_1,N/n_2,N/n_3; c_1,c_2,c_3
```

Now we need to solve for the inverse of each c_i modulo n_i. One could do this by hand. For instance,

$$42d_1 \equiv 2d_1 \equiv 1 \pmod 5 \text{ yielding } d_1 = 3, \text{ since } 2 \cdot 3 = 6 \equiv 1 \pmod 5 .$$

But that is best done on homework for careful practice; in the text, we might as well use the power of Sage.

```
d_1=inverse_mod(42,5);
    d_2=inverse_mod(35,6);d_3=inverse_mod(30,7)
d_1,d_2,d_3
```

Now I'll create each of the big product numbers, as well as their sum.

```
n_1, n_2, n_3 = 5,6,7
a_1, a_2, a_3 = 1,2,3
N = n_1*n_2*n_3
d_1=inverse_mod(42,5); d_2=inverse_mod(35,6);
    d_3=inverse_mod(30,7)
a_1*c_1*d_1, a_2*c_2*d_2,a_3*c_3*d_3;
    a_1*c_1*d_1+a_2*c_2*d_2+a_3*c_3*d_3
```

Of course, we don't recognize 836 as our answer. But:

```
n_1, n_2, n_3 = 5,6,7
N = n_1*n_2*n_3
mod(836,N)
```

Let's try some more interesting moduli for an example to do on your own. Can you follow the template?

- $x \equiv 1 \pmod{6}$
- $x \equiv 11 \pmod{35}$
- $x \equiv 3 \pmod{11}$

Sage can also approach this in a similar way, as we saw earlier.

```
@interact(layout=[['a_1','n_1'],['a_2','n_2'],['a_3','n_3']])
def _(a_1=('\(a_1\)',1), a_2=('\(a_2\)',2),
    a_3=('\(a_3\)',3), n_1=('\(n_1\)',5),
    n_2=('\(n_2\)',6), n_3=('\(n_3\)',7)):
    try:
        answer = []
        for i in [1..n_1*n_2*n_3]:
            if (i%n_1 == a_1) and (i%n_2 == a_2) and
                (i%n_3 == a_3):
              answer.append(i)
        string1 = "<ul><li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li>"%(a_1,n_1)
        string2 = "<li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li>"%(a_2,n_2)
        string3 = "<li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li></ul>"%(a_3,n_3)
        pretty_print(html("The␣simultaneous␣solutions␣to␣
            "))
        pretty_print(html(string1+string2+string3))
        if len(answer)==0:
            pretty_print(html("are␣none"))
        else:
            pretty_print(html("all␣have␣the␣form␣"))
            for ans in answer:
                pretty_print(html("$%s$␣modulo␣
                    $%s$"%(ans,n_1*n_2*n_3)))
    except ValueError, e:
        pretty_print(html("Make␣sure␣the␣moduli␣are␣
            appropriate␣for␣solving!"))
        pretty_print(html("Sage␣gives␣the␣error␣message:"))
        pretty_print(html(e))
```

5.4.2 A theoretical but highly important use of CRT

Now, there are many, many useful things we can do with the CRT.

Proposition 5.4.2 (Converting to and from coprime moduli)**.** *Suppose that $X \equiv Y \pmod{N}$, and $N = \prod m_i$, where $\gcd(m_i, m_j) = 1$ for all $i \neq j$. Then we have two directions of equivalence between a congruence and a system of congruences.*

- *Certainly if N divides $X - Y$, so does a factor of N, so $X \equiv Y \pmod{m_i}$ for each of the relatively prime factors of N. Thus, solutions to the "big" congruence are also solutions to a system of many little ones.*

- *But the* CRT *allows me to reverse this process. The moduli in question are all coprime to each other, so if we are given a solution pair* (X_i, Y_i) *to each of the congruences*

$$X_i \equiv Y_i \pmod{m_i}$$

then when combined they will give one (!) solution of

$$X \equiv Y \pmod{N}$$

That means that *any question about congruences* is really a question about congruences modulo simple moduli (see Proposition 6.5.1 for a strong statement of this). We will use this fact again and again in the remainder of the text, and it is a huge reason why the Chinese Remainder Theorem is so intensely powerful.

5.5 More Complicated Cases

Solving linear congruences is a completely solved problem (up to computer power). Although one does not usually cover all extensions in an introductory course, the following subsections will introduce some, without full detail.

5.5.1 Moduli which are not coprime

What happens if, in a system of congruences, we don't have the enviable situation where all the n_i are relatively prime? Let's go back to the interact from before one last time, with some moduli which are not pairwise coprime, and see if we get anything.

```
@interact(layout=[['a_1','n_1'],['a_2','n_2'],['a_3','n_3']])
def _(a_1=('\(a_1\)',1), a_2=('\(a_2\)',2),
    a_3=('\(a_3\)',3), n_1=('\(n_1\)',5),
    n_2=('\(n_2\)',6), n_3=('\(n_3\)',7)):
    try:
        answer = []
        for i in [1..n_1*n_2*n_3]:
            if (i%n_1 == a_1) and (i%n_2 == a_2) and
                (i%n_3 == a_3):
              answer.append(i)
        string1 = "<ul><li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li>"%(a_1,n_1)
        string2 = "<li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li>"%(a_2,n_2)
        string3 = "<li>$x\equiv␣%s␣\\text{␣(mod␣
            }%s)$</li></ul>"%(a_3,n_3)
        pretty_print(html("The␣simultaneous␣solutions␣to␣
            "))
        pretty_print(html(string1+string2+string3))
        if len(answer)==0:
            pretty_print(html("are␣none"))
        else:
            pretty_print(html("all␣have␣the␣form␣"))
            for ans in answer:
                pretty_print(html("$%s$␣modulo␣
                    $%s$"%(ans,n_1*n_2*n_3)))
    except ValueError, e:
```

```
        pretty_print(html("Make␣sure␣the␣moduli␣are␣
            appropriate␣for␣solving!"))
        pretty_print(html("Sage␣gives␣the␣error␣message:"))
        pretty_print(html(e))
```

As previously mentioned, Qin discovered a very general answer for getting answers in this situation. An answer exists as long as $\gcd(n_i, n_j)$ divides $a_i - a_j$ for all i and j. Lebèsgue was the first to rediscover this in the modern era, in 1859.

5.5.2 The case of coefficients

Another case is that of congruences not of the form $x \equiv a \pmod{n}$, but of the form $Ax \equiv B \pmod{n}$. What can we say when our linear system has coefficients to the variable?

If you have simultaneous congruences with coefficients,

$$A_i x \equiv B_i \pmod{N_i}$$

then first write their individual solutions in the form $x \equiv a_i \pmod{n_i}$. Then you can use the CRT to get a solution of that system, which is also a solution of the 'big' system.

For instance, try now to solve

- $2x \equiv 2 \pmod{5}$
- $5x \equiv 4 \pmod{6}$
- $3x \equiv 2 \pmod{7}$

Surprised? Don't forget to get back to the original modulus!

See also Example 6.5.2 for combining these ideas with those of Proposition 5.4.2.

5.5.3 A practical application

Finally, there is a practical application. Suppose you are adding two *very* large numbers – too big for your computer! How would you do it? The answer is one can use the CRT, in particular the ideas of Proposition 5.4.2.

- First, pick a few mutually coprime moduli *smaller* than the biggest you can add on your computer.
- Then, reduce your two numbers x and y modulo those moduli and add the two huge numbers in each of those moduli.
- Then the CRT allows you to put $x + y$ modulo each of the moduli back together for a complete solution!

Needless to say, we won't do an actual example of this.

5.6 Exercises

1. Why do the latter two strategies in Fact 5.2.1 need no additional proof?

2. Complete the outline of the proof of Proposition 5.2.5.

3. We found solutions to $ax \equiv b \pmod{n}$ as congruence classes modulo n. But since $\gcd(a, n) = d$ is important here, it could be worth talking about how many congruence classes modulo n/d we have. Well, how many do we get? (If this sounds confusing, pick a specific problem and try it, then see if you get the same answer in general.)

4. Write down two linear congruences which do *not* have solutions modulo 15, but *do* have solutions modulo 16. (You do *not* have to solve them.)

5. We know that $b \equiv c \pmod{n}$ implies $ab \equiv ac \pmod{n}$ as well. Prove that the converse is true if $\gcd(a, n) = 1$, and give a counterexample where the converse fails if $\gcd(a, n) \neq 1$.

6. For each of the following linear congruences, find all of its solutions.

(a) $18x \equiv 42 \pmod{50}$

(b) $15x \equiv 9 \pmod{25}$

(c) $6x \equiv 3 \pmod{9}$

(d) $980x \equiv 1540 \pmod{1600}$

7. Solve the simultaneous system below. ([C.1.1], Exercise 3.8)

- $x \equiv 1 \pmod{4}$
- $x \equiv 2 \pmod{3}$
- $x \equiv 3 \pmod{5}$

8. Find an integer that leaves a remainder of 9 when it is divided by either 10 or 11, but that is divisible by 13.

9. When eggs in a basket are removed two, three, four, five, or six at a time, there remain, respectively, one, two, three, four, or five eggs. When they are taken out seven at a time, none are left over. Find the smallest number of eggs that could have been contained in the basket. (Brahmagupta, 7th century AD)

10. Find a problem on the internet about pirates quarreling over treasure (or monkeys over bananas) that could be solved using the CRT, and solve it.

11. Solve the system $4x \equiv 2 \pmod{6}$, $3x \equiv 5 \pmod{7}$, $2x \equiv 4 \pmod{11}$.

12. Solve the congruence $5x \equiv 22 \pmod{84}$.

13. Solve the simultaneous system $x \equiv 4 \pmod{6}$, $x \equiv 7 \pmod{15}$. Note that this doesn't fit our pattern, but you should still be able to solve this, since there are only two. (Hint: trial and error.)

Chapter 6

Prime Time

Now it's time to introduce maybe the most important concept in the whole course. It's one you are almost certainly already pretty familiar with. That is the concept of *prime* numbers.

Although we'll take a somewhat traditional route to introduce them, consider what precedes this chapter. We attacked linear congruences as far as we could via the concept of 'relatively prime'/'coprime'. But the thought should be gnawing at us of whether there is something deeper than simply not sharing factors other than one; what are the factors that are (or are not) shared in the first place? As mathematicians, we always want to ask whether there is a *simpler* notion available, or one that explains more.

We will see the fruit of this for linear congruences in Section 6.5, using the most powerful tool in our arsenal, Theorem 6.3.2. But once we have unleashed the power of primes, we will see and use them everywhere, such as in Chapters 22 and 12. Examining them more closely will lead to us some of the deepest mathematics of the book in Chapters 21 and 25.

So let's get started!

6.1 Introduction to Primes

6.1.1 Definitions and examples

Definition 6.1.1. A positive integer p greater than 1 is called **prime** if the only positive divisors of p are 1 and p itself.

Definition 6.1.2. If an integer $n > 1$ is not prime, it is called **composite**.

The first few primes are $2, 3, 5, 7, 11, \ldots$ That means $4, 6, 8, 9, 10, 12 \ldots$ are composite. But figuring out which numbers are prime is notoriously difficult. Indeed, we will spend significant time later on this question, such as in Chapter 12 and Chapter 21. So below, we introduce a few Sage functions for this.

Here are answers to questions you might have about primes that Sage could answer.

- Is a given number prime?

```
is_prime(6) # Is my number a prime?
```

- Is it at least a *power* of a prime?

```
is_prime_power(25) # Is my number a prime power?
```

- List some primes for me!

```
PR = prime_range(100) # What are all primes up to but
    not including 100?
print PR
```

- List the first n primes ...

```
PFN = primes_first_n(100) # What are the first 100
    primes?
print PFN
```

- Give me prime factors.

```
# What are the prime factors of a number?
factor( 2 * 3 * (2*3+1) * (2*3*(2*3+1)+1) *
    (2*3*(2*3+1)*(2*3*(2*3+1)+1)+1) )
```

Sage note 6.1.3 (Making comments)**.** As is typical in Python, comments on them are given after # signs.

6.1.2 Prime fun

Here's a little fun that starts us thinking along the lines of what's to come. Let's see whether we can generate some primes with a simple polynomial.

```
f(x)=x^2+x+41
@interact
def _(n=(0,[0..39])):
    pretty_print(html("Is␣$%s$␣for␣$x=%s$,␣which␣is␣$%s$,␣
        a␣prime␣number?"%(f(x),n,f(n))))
    print is_prime(f(n))
```

Of course, I'm cheating a little:

```
f(x)=x^2+x+41
f(40)
```

```
is_prime(f(40)),factor(1681)
```

In fact, we can prove that quite the opposite of what you might have thought with this example is true.

Fact 6.1.4. *There is no non-constant polynomial $f(x)$ with integer coefficients such that $f(x)$ is prime for all integers x.*

What is the reason no such polynomial can exist? It turns out to be *directly related* to our previous work on congruences. Namely, if $f(a) = p$ for some a, then for any $b \equiv a \pmod{p}$ we have $f(b) \equiv f(a)$ (by well-definedness of addition and subtraction) as well, so

$$f(b) \equiv f(a) \equiv p \equiv 0 \pmod{p}, \text{ or that } p \mid f(b) .$$

But the problem is that if $f(b)$ is *also* prime for *all* such b, then we have a polynomial which returns to height $f(x) = p$ periodically, and hence $\lim_{x\to\infty} f(x) \neq \pm\infty$, which is only possible if f is a constant polynomial.

This could be a big surprise – limits and calculus can be used in number theory! Even at this early stage, it is evident, but there will be more later, such as in Chapters 24 and 20.

The fact is not true for *multivariate* polynomials (see e.g. Wikipedia). Yikes!

A few more single-variable polynomials that do happen to generate a number of primes are below, though the second one takes a long time! Among other sites, Mathworld has lots and lots more information.

```
g(x)=8*x^2-488*x+7243
for n in [0..30]:
    print g(n),is_prime(g(n))
```

```
h(x)=x^6+1091
for n in [0..3906]:
    if is_prime(h(n)):
        print (n,h(n))
```

6.2 To Infinity and Beyond

6.2.1 Infinite primes

At this point it's a good idea to mention that the search for 100, or 1000, or how every many prime numbers is not hopeless; that is the content of Euclid's famous theorem on the infinitude of the primes.

Strictly speaking, he proves that no matter what n is, there is always a bigger prime $p > n$, which isn't exactly the same thing as a Cantoresque "infinite set of primes"! But we still say there are infinitely many prime numbers.

As usual, Joyce's web version of the original is a great resource. There are *many* proofs of this theorem, some of which would be corollaries of theorems later in this text. Most use some form of proof by contradiction, but there are exceptions, such as Saidak's proof from the American Math Monthly, which we will mention again in Section 21.1. One notable proof by Furstenberg even uses point-set topology, though this has been interpreted in a non-topological way as well.

Here is a slightly modernized version of Euclid's proof.

Theorem 6.2.1 (Infinitude of Primes)**.** *There is no upper bound on the size of the collection of prime numbers.*

Proof. Suppose that we have found exactly $n > 0$ prime numbers, $p_1, p_2, \ldots, p_n$. Find the smallest positive integer N which is a multiple of all of these simultaneously (we know at least one such number exists, since you could multiply them all together).

Then $N + 1$ is either prime, or it is not. (" EZ ".") If it is prime, then it is certainly different from the others, so we have increased the size of the set of primes.

If on the other hand it is not prime, then it has some prime divisor p. (This actually does require proof, and is Euclid's Book 7, Proposition 31. For us, it

is easiest to have it follow immediately from Theorem 6.3.2.) We claim p is not one of the p_i already known.

If it were, then if p is a divisor of both N and $N+1$, which means it is a divisor of 1 (see Exercise 2.5.7). This is absurd (“ ”). (Can you recall why?)

So p is not one of the original list, and is prime, so we have found a larger list than before. □

There are two things worth pointing out about this proof. First, Joyce points out that Euclid doesn't bother to mention that N is in fact the product of the primes in question. It's not necessary to mention, but using this characterization, the same proof would show the (weaker but also interesting) conclusion that there is no upper bound on the size of a set of mutually coprime positive integers. Secondly, as is typical, Euclid only proves this with a small n, rather than with some modern stand-in like ellipses; those interested in math history will be interested in how Wallis used this to his advantage in the Hobbes-Wallis controversy.

6.2.2 The sieve of Eratosthenes

Much later in the text we will talk some about efficient ways to tell if a number is prime, or to get prime numbers (see Chapter 12, for example). For now, all we will use is something usually known as the Sieve of Eratosthenes.

Algorithm 6.2.2 (Sieve of Eratosthenes)**.** *To check whether a number $n > 1$ is composite or prime, it suffices to divide by all primes $p \leq \sqrt{n}$. Anything that isn't divisible by these is prime.*

Proof. If n is not prime (composite), we can write $n = de$ for integers d and e both strictly between 1 and n. If both $d, e > \sqrt{n}$, then

$$n = de > \left(\sqrt{n}\right)^2 = n \ ,$$

a contradiction. □

This is indeed an algorithm, because it provides a specific procedure to identify primes up to a specific limit.

Example 6.2.3. To get all prime numbers up through 100, it suffices to remove any numbers divisible by $2, 3, 5$, or 7, as $\sqrt{100} < 11$.

Eratosthenes was a contemporary of Archimedes, and no slouch. He is best known for estimating the size of the Earth fairly accurately, amazingly so for the time. (Along the way, that puts the lie to those who would claim everyone thought the earth was flat until Columbus.)

6.3 The Fundamental Theorem of Arithmetic

6.3.1 Preliminaries and statement

Our biggest goal for this chapter, and the motive for introducing primes at this point, is the Fundamental Theorem of Arithmetic, or FTA. It should probably be called the Fundamental Theorem of Number Theory, but in older usage one said “arithmetic”, and the name has stuck.

Definition 6.3.1. A **factorization** of an integer is a way of writing it as a product of other integers. This nearly always refers to one of two things, which are mentioned explicitly if there is danger of ambiguity:

- A product into *prime* numbers, which can be called a **prime factorization**;
- A product into positive powers of primes, which can be called a **prime power factorization**.

Theorem 6.3.2 (Fundamental Theorem of Arithmetic)**.** *The following are true:*

- *Every $n > 1$ has a prime factorization.*
- *Every such factorization of a given n is the same if you put the prime factors in increasing order (uniqueness).*

More formally, we can say the following. Any positive integer $N > 1$ may be written as a product

$$N = \prod_{i=1}^{n} p_i$$

of primes, and further, if we can write a different such product

$$N = \prod_{j=1}^{m} q_m$$

then $m = n$ and a reordering of the q_j will make them the same as the p_i.

Proof. We will prove this in Subsection 6.3.2. □

Example 6.3.3. For instance:

- $30 = 2 \cdot 3 \cdot 5$
- $24 = 2 \cdot 3 \cdot 2 \cdot 2 = 2 \cdot 2 \cdot 2 \cdot 3$

Clearly (from normal experience, I mean) the only other possibilities are putting the primes in a different order. Why doesn't this work for $N = 1$?

Example 6.3.4. Usually we will implicitly assume the primes are in nondecreasing order, and write 3^2 instead of $3 \cdot 3$, so be ready for the notation

$$N = \prod_{i=1}^{n} p_i^{e_i}$$

for this same result; we only use the first notation for convenience for the first statement. (Sometimes when the context is clear, one can even write $N = \prod p$ or $N = \prod p^e$.)

For instance:

- $30 = 2^1 \cdot 3^1 \cdot 5^1$
- $24 = 2^3 \cdot 3^1$

Just to get this down, practice writing the following as a product of such *prime powers*.

- $N = 12100$
- $N = 1250$
- $N = 3072$

6.3.2 Proof of the FTA

This theorem is quite old, and of course Euclid has a nice proof of it, along with various lemmata (the plural of lemma, though I'll also use "lemmas" in this text) he needs to get there. The key ingredients are:

- If a number is prime, that *is* the prime factorization.
- If a number is composite, then it is *divisible* by some prime. (Euclid used this in his proof of the infinitude of primes, above.)
- This process can be continued and is finite.
- Any *other* way in which you can write the same number as a product of primes is just a reordering of the one obtained in the previous step.

The last step requires this lemma, which is Euclid's Book 7, Proposition 30.

Lemma 6.3.5. *If a prime p divides a product ab, then p divides at least one of a or b.*

Proof. Left to reader in Exercise 6.6.3; this is very closely related to Proposition 2.4.6. □

Corollary 6.3.6. *If a prime p divides any finite product of primes, then p divides at least one of them, i.e.*

$$p \mid \prod_{k=1}^{\ell} a_k \text{ implies } p \mid a_k \text{ for at least one } k$$

Proof. By induction, left to reader in Exercise 6.6.4. □

Okay, now we need the details.

Let's use induction on the size of N. So our base case is $N = 2$, which is of course prime and has a unique factorization 2^1.

First, let's suppose we have proved that all numbers up to N can be written as a product of primes (uniquely or not). Then we look at $N + 1$ to continue the induction.

- If $N + 1$ is prime, that is its prime factorization, as with 2.
- If not, then by induction we know it is composite, so $N + 1 = ab$, where $1 < a, b < N + 1$. (*Note why a, b are smaller!* Recall the proof of the Sieve 6.2.2.) In this case, a and b have prime decompositions $\prod p_i$ and $\prod q_j$, since they are less than $N+1$ but not 1, and so $N+1 = \prod p_i \prod q_j$.

By induction, this shows that a prime factorization exists.

It remains to be shown that such a factorization is unique. So first rewrite it as

$$N + 1 = \prod p_i$$

and now suppose that (once again by induction) we have written all numbers up to N *uniquely* as a product of primes. So let's look at another such representation,

$$N + 1 = \prod q_j$$

At this point we need Corollary 6.3.6. By definition, p_1 divides $N + 1$. Hence p_1 divides at least one of the q_j. But the only positive divisors of a prime are itself and 1, so $p_1 = q_j$.

Cancel these from the product to get two different representations of (the integer) $\frac{N+1}{p_1}$ as a product of primes. By the induction hypothesis, these *are* unique up to reordering, so multiplying both by p_1 to get $N+1$ should also be unique up to reordering.

By induction, we are done.

Two comments about this proof are in order. First (this is especially to the instructor), this may seem like a different kind of induction proof, because we do not simply assume N has a (unique) factorization, but that all $n \leq N$ do. It is not; the statement we are proving is just different. Rather than proving "N has a factorization" we are proving "all numbers less than N have a factorization". In particular, there is no pedagogical need for a separate notion of 'strong induction', in my view.

Second, if you are familiar with other algebraic structures, it is very important to note this theorem is *not* true for every algebraic system! (Not even for every such system that is like the integers in other ways.) Most interesting examples of this are just beyond the level of this course. For those who *must* know what this means now, try Exercise 6.6.25.

6.4 First consequences of the FTA

The impact of the FTA is so great, I cannot overstate its significance. This section collates a few examples, but you will see them throughout the text, as well as in the next section, when we connect it back to congruences.

Most importantly, lots of theorems now have *reasons*, not just proofs. This is an important point about mathematics! You will (re)prove a few things in the Exercises to see this. This ability boils down to the fact that $\gcd(a,b) = 1$ now means that a and b do not share any common *prime* factors. Here is a first example to give the feel.

Example 6.4.1. If $a \mid c$, $b \mid c$, and $\gcd(a,b) = 1$, then $a = \prod p_i$ and $b = \prod q_j$ but none of the p_i can be any of the q_j (or the gcd would include that prime).

Since by the FTA $c = \prod r_k^{e_k}$ where the r_k are distinct, the p_i must be some of the r_k and the q_j must be different ones, so that $\prod p_i q_j$ still divides c.

So if $a \mid c$, $b \mid c$, and $\gcd(a,b) = 1$, then $ab \mid c$, which is part of Proposition 2.4.6.

As another example, the proofs from Section 3.7 become far simpler. We can prove the first here, and save the second for Exercise 6.6.11.

Example 6.4.2. Let's show that $a^2 \mid z^2$ implies $a \mid z$.

Solution. To begin, let's write $a = \prod p^e$. Then

$$a^2 = \prod p^e \cdot \prod p^e = \prod p^{e+e} = \prod p^{2e}$$

Similarly,

$$z = \prod q^f \text{ implies } z^2 = \prod q^{2f}$$

If these two numbers divide each other, then we can separate each prime out, so that

$$p^{2e} \mid q^{2f}$$

where $p = q$. But this just means $2e < 2f$, so $e < f$ as well.

This is true for all the primes p dividing a, so $p^e \mid (p=q)^f$ for all such p; multiplying these together shows that

$$a = \prod p^e \mid \prod_{p=q} q^f \mid \prod q^f = z$$

as desired.

The reader should note that for such proofs, the implicit use of Corollary 6.3.6 is crucial along with the FTA.

Nearly as important is that *computing* all kinds of things becomes easier. If we let $a = \prod_{i=1}^{n} p_i^{e_i}$ and $b = \prod_{i=1}^{n} p_i^{f_i}$, where it's possible that e_i or f_i is zero at times, then we can often get formulas for various combinations of a and b.

Definition 6.4.3. Given two numbers $x \leq y$, we let the **maximum** and **minimum** be defined by

$$\max(x, y) = y \text{ and } \min(x, y) = x$$

with the clear extension to a min or max of a set consisting of more than two numbers.

Then we have formulas of this kind.

- $$ab = \prod_{i=1}^{n} p_i^{e_i + f_i}$$
- $$\gcd(a, b) = \prod_{i=1}^{n} p_i^{\min(e_i, f_i)}$$
- Assuming $b \mid a$,
$$a/b = \prod_{i=1}^{n} p_i^{???}$$
(See Exercise 6.6.7.)

Another use of the FTA is to help us do in a *systematic* way results that were probably first obtained by extremely ad-hoc methods. As an example, it is likely that you have seen a proof that $\sqrt{2}$ is irrational, and it probably used mostly the concept of "evenness". But we can prove that $\sqrt{m} \notin \mathbb{Q}$ (for m not a perfect square) in a very similar fashion.

Most deeply, it gives us a *canonical* way to describe every integer in terms of simpler integers, and gives a measure of simplicity. We'll exploit this some much later, such as in Chapter 24.

Here are some ways to calculate these things in Sage. Simply replace the numbers you are interested in.

```
prime_divisors(693)
```

```
factor(693)
```

Note that the first of these functions gives just a list of the prime divisors, while the second one gives the full prime power factorization.

Finally, let's note that depending on the context, the computational, proof, or other tools won't all be necessary. To demonstrate that, let's introduce some useful additional notation.

Definition 6.4.4. For p prime, we say that $p^k \parallel n$ precisely when $p^k \mid n$ *but* p^{k+1} does not divide n.

Definition 6.4.5. We write $n!$ for the product of the integers from 1 to n, called n **factorial**.

As an example, $5^2 \parallel 75$. The prime factorization of a number clearly gives you information about this.

```
factor(factorial(20))
```

Since $2^{18} \parallel 20!$ and $5^4 \parallel 20!$, we can conclude that 20! ends with exactly 4 zeros merely from the prime factorization, which we could certainly get without multiplying it out (though in this case Sage does that first). We can check this:

```
factorial(20)
```

6.5 Applications to Congruences

6.5.1 Factoring the modulus

The reason the fundamental theorem is so useful for congruences is that prime *powers* (for different primes) are automatically relatively prime to each other. So in using the Chinese Remainder Theorem (Theorem 5.3.1) we don't have a spend time looking for coprime factors; we can just factor into prime powers using the Fundamental Theorem of Arithmetic. So here is a useful repositioning of Proposition 5.4.2.

Proposition 6.5.1 (Converting to and from prime powers)**.** *Suppose that $X \equiv Y \pmod N$, and $N = \prod p_i^{e_i}$. Then we have two directions of equivalence between a congruence and a system of congruences.*

- *Certainly if N divides $X - Y$, so does a factor of N, so $X \equiv Y \pmod{p_i^{e_i}}$ for each of prime power factors of N. (Once again, solutions to the "big" congruence are also solutions to a system of many little ones.*
- *But by their nature, the prime powers in a factorization are all coprime to each other, so if we are given a solution pair (X_i, Y_i) to each of the congruences*

  $$X_i \equiv Y_i \pmod{p_i^{e_i}}$$

  then when combined they will give a solution of

  $$X \equiv Y \pmod N$$

That means that *any question about congruences* is really a *question about congruences modulo prime powers*. We will use this fact again and again in the remainder of the text, and it is a huge reason why the CRT is so intensely powerful.

Similarly, referring to Subsection 5.5.2, what if one has *one* complicated congruence with coefficients and a composite modulus N?

$$Ax \equiv B \pmod N$$

Just take $N = p_1^{e_1} \cdots p_k^{e_k}$ and then solve all the congruences $Ax \equiv B \pmod{p_i^{e_i}}$ first. Then use the CRT to 'patch' them together for a final solution. This is a little tedious, but certainly doable.

Example 6.5.2. Let's solve

$$21x \equiv 31 \bmod 180$$

this way.

- What are the individual congruences?
- Can we solve them?
- Can we then put them back together?

6.5.2 Moduli which are prime powers

When it comes to *linear* congruences, these consequences of the Chinese Remainder Theorem and FTA allow us to reconsider the prime power case with a more subtle tool. Assume that in solving a bunch of congruences

$$x \equiv a_j \pmod{n_j}$$

we would like to start by solving congruences

$$x \equiv a_j \pmod{p^e}$$

where p^e divides n_j.

The general approach is then to first solve modulo p, in the hope that it could lead to a solution modulo p^e. Consider this extended example.

Example 6.5.3 (Prime Power Congruences)**.** Let $f(x) = 2x - 3$. The only solution of $2x \equiv 3 \pmod 5$ is clear; it is $x = [4]$. How might we get solutions (mod 25) from this? Here are some steps.

- First, any solution of $2x \equiv 3 \pmod{5^2}$ is also a solution of $2x \equiv 3 \pmod 5$. So $x \equiv 4 + 5k \pmod{25}$ for some k, since $[4] = \{4 + 5k | k \in \mathbb{Z}\}$.
- Plugging $4 + 5k$ in the congruence yields

  $$2x \equiv 2(4 + 5k) \equiv 2 \cdot 4 + 2 \cdot 5k \equiv 3 \pmod{25},$$

  or, rearranging (but keeping everything unmultiplied),

  $$3 - 2 \cdot 4 \equiv 2 \cdot 5k \pmod{5^2}.$$

- Now, we know that $5 \mid 3 - 2 \cdot 4$, because we already know that 4 solved our original congruence:

  $$3 \equiv 2 \cdot 4$$

  So we can cancel out 5 from the entire congruence to get that

  $$\frac{3 - 2 \cdot 4}{5} \equiv 2k \pmod 5.$$

  This simplifies to $-1 \equiv 2k \pmod 5$, which has solution $k \equiv 2 \pmod 5$.
- Hence, using this k and plugging it back in to get a solution to $2x \equiv 3 \pmod{5^2}$, we get

  $$4 + 5k = 4 + 5 \cdot 2 = 14 \pmod{5^2}$$

  as the solution. And indeed $2 \cdot 14 = 28 \equiv 3 \pmod{25}$.

Example 6.5.4. Let's do it all again, more tersely, to get a solution to $2x \equiv 3$ modulo $5^3 = 125$.

- I already know that [14] is the solution to $2x \equiv 3 \pmod{5^2}$.
- That means that a solution to $2x \equiv 3 \pmod{5^3}$ must look like $14 + 5^2k$.
- Plugging this in gives me $2(14 + 5^2k) \equiv 3 \pmod{5^3}$, which rearranges to
$$2 \cdot 5^2 k \equiv 3 - 2 \cdot 14 \pmod{5^3} .$$
- Since we know that 14 solves $2x \equiv 3 \pmod{5^2}$, that means (by definition of congruence) that
$$5^2 \mid 3 - 2 \cdot 14 ,$$
so we can divide "all three sides" of the last congruence by 5^2.
- This yields
$$2k \equiv \frac{3 - 2 \cdot 14}{5^2} \equiv \frac{-25}{5^2} \equiv -1 \pmod{5} .$$
- Solving this yields, of course, $k \equiv 2 \pmod{5}$, so
$$x \equiv 14 + 5^2 \cdot 2 \equiv 64 \pmod{125} ,$$
and indeed $2 \cdot 64 = 128 \equiv 3 \pmod{125}$ works.

You can do this as often as you like, and (properly interpreted) it will yield all solutions of your congruence modulo p^e, one step at a time. We'll see a generalization of this in Section 7.2.

6.6 Exercises

1. A number such as 11, 111, 1111 is called a **repunit**. Clearly eleven is a prime repunit. Find two more, and say how you did it.

2. Find the prime numbers less than 100 using the Sieve of Eratosthenes (6.2.2). Make sure you actually draw it! Every math student should do this once, and *only* once.

3. Prove Lemma 6.3.5; if a prime p divides a product ab, then p divides at least one of a or b.

4. Prove Corollary 6.3.6; if a prime p divides any finite product of primes, then p divides at least one of them.

5. Prove that if $\gcd(a, b) = 1$ and $a \mid bc$ then $a \mid c$ as well, using the FTA.

6. Prove using the FTA that if $\gcd(a, b) = d$ then $\gcd\left(\frac{a}{d}, \frac{b}{d}\right) = 1$.

7. Assuming $b \mid a$, find a formula to complete
$$a/b = \prod_{i=1}^{n} p_i^{???}$$
and prove it using the FTA.

8. How would you describe a factorization of a *rational* number? Do you think you could extend the Fundamental Theorem of Arithmetic to this case? If so, how? If not, why would it not be appropriate?

9. Show that if a and b are positive integers and $a^3 \mid b^2$, then $a \mid b$.

10. Show that if $p^a \parallel m$ and $p^b \parallel n$, then $p^{a+b} \parallel mn$.

11. Prove Proposition 3.7.2 using the FTA; if $\gcd(m, n) = 1$ and mn is a perfect square, then so are m and n.

12. Is it possible for $n!$ to end in exactly five zeros?

13. Show that $\log_{10}(5)$ is irrational.

14. Show that $3^{2/3}$ is irrational.

15. By hand, find the prime factorizations of 36, 756, and 1001. Use these to find their pairwise gcds.

16. By hand, find the gcd of $2^2 \cdot 3^5 \cdot 7^2 \cdot 13 \cdot 37$ and $2^3 \cdot 3^4 \cdot 11 \cdot 31^2$.

17. By any method you like, find the prime factorizations of $2^{24}-1$ and 10^8-1, as well as their gcd.

18. Prove that the only solutions of $x^2 \equiv x \pmod{p}$ are $x = [0]$ and $x = [1]$, if p is a prime. (Refer to Question 4.6.7; this and the next exercise answer Exercise 4.7.14.)

19. Try to decide for *exactly* which composite moduli n the previous question is true. (Refer to the interact in Question 4.6.7; this and the previous exercise answer Exercise 4.7.14.)

20. Find solutions to $3x - 4 \equiv 0 \pmod{49}$ and $\pmod{343}$ using the method in Subsection 6.5.2, starting with modulus seven.

21. Fill in the details of Example 6.5.2.

In the next few exercises, recall the definition of *least common multiple* (or lcm) from Exercise 2.5.9.

22. Find the pairwise least common multiples (recall Exercise 2.5.9) in the previous few exercises.

23. Find a formula for the lcm using the FTA:

$$\operatorname{lcm}(a, b) = \prod_{i=1}^{n} p_i^{???}$$

24. Prove that if $a, b > 0$ then $\gcd(a, b)\operatorname{lcm}(a, b) = ab$ using the FTA.

25. Let $\mathbb{Z}[\sqrt{-5}]$ be the set of all numbers of the form $a + b\sqrt{-5}$ for $a, b \in \mathbb{Z}$. Find two different factorizations of $N = 6$ in this set (known as a ring), , all the factors of which are not ± 1.

Chapter 7

First Steps With General Congruences

There is a lot more one can say about solving congruences. However, congruences also play a crucial role in solving all manner of other number-theoretic problems.

In this chapter, we collate a significant number of interesting results that the congruence framework affords us. Among them are some of the most important results we have access to at this early stage, including Fermat's Little Theorem and Lagrange's Theorem on polynomials.

7.1 Exploring Patterns in Square Roots

Just as in high school algebra one moved from linear functions to quadratics (and found there was a *lot* to say about them!), this is the next natural step in number theory. We will focus on congruences. We haven't abandoned integers! But it turns out that questions about quadratic polynomials with integers are much, much harder, and are better pursued after studying the relatively simple (and computable) cases of quadratic congruences. Much later, we will return to a full investigation of this.

You may recall that we looked at one particular quadratic congruence in Question 4.6.7 and Exercise 4.7.14, and saw that the solution depended at least partly on the modulus in Exercises 6.6.18 and 6.6.19. So we will examine these slightly simpler-sounding questions keeping in mind the structure of the modulus, not so much the actual answers.

Question 7.1.1. Consider the following questions.

- For what prime p does -1 have a square root?
- For what integers n does 1 have more square roots than just ± 1?

These questions are exactly equivalent to the following quadratic congruence questions.

- Is there a solution to
$$x^2 \equiv -1 \pmod{p} \text{ or } x^2 + 1 \equiv 0 \pmod{p} \text{ ?}$$
- Is there a solution to
$$x^2 \equiv 1 \pmod{n} \text{ (or equivalently } x^2 - 1 \equiv 0 \pmod{n})?$$

Let's look at each of these in turn. The interacts are merely an aid; it is quite possible use pencil and paper to explore these as well.

- An interact for which primes -1 has a square root:

```
@interact
def _(p=(13,prime_range(10,100))):
    pretty_print(html("Values␣of␣$x^2+1$␣mod␣%s"%(p,)))
    pretty_print(html("<ul>"))
    for m in [0..p-1]:
        pretty_print(html("<li>$%s^2+1\equiv␣%s\\text{␣
            (mod␣}%s)$</li>"%(m,mod(m,p)^2+1,p)))
    pretty_print(html("</ul>"))
```

- An interact for when 1 has more square roots than just ± 1 – a rather tricky question:

```
@interact
def _(n=(12,[10..100])):
    pretty_print(html("Values␣of␣$x^2-1$␣mod␣%s"%(n,)))
    pretty_print(html("<ul>"))
    for m in [0..n]:
        pretty_print(html("<li>$%s^2-1\equiv␣%s\\text{␣
            (mod␣}%s)$</li>"%(m,mod(m,n)^2-1,n)))
    pretty_print(html("</ul>"))
```

What do you get? See Exercise 7.7.1; writing ideas in the margin of a physical book or in a small text document on a computer are both awesome.

7.2 From Linear to General

In this section, we will take two ideas we already used with linear congruences, and see how they can be modified to apply in any polynomial situation.

7.2.1 Combining solutions

One of the most important things we can do is study congruences with prime (power) modulus, because we can combine their solutions to get solutions for any congruences when we combine the Chinese Remainder Theorem and Fundamental Theorem of Arithmetic (recall Proposition 6.5.1). Even more interestingly, we can combine the *numbers* of solutions.

Informally, if you want to get the total number of solutions of a polynomial congruence, just write the modulus as a product of prime powers $n = \prod_{i=1}^{k} p_i^{e_i}$, find out how many solutions the congruence has with each prime power modulus, then multiply those numbers for the total number of solutions.

Example 7.2.1. For instance, if $f(x) \equiv 0$ has 2 solutions modulo 3, 1 solution modulo 5, and 3 solutions modulo 7, it would have $2 \cdot 1 \cdot 3 = 6$ solutions modulo $105 = 3 \cdot 5 \cdot 7$.

We will state this for the general case of a coprime factorization of n, though again the prime power factorization is usually the most useful.

Fact 7.2.2. *Let $n_1, n_2, \cdots, n_k$ be a set of k moduli, all of which are relatively prime to each other. Suppose that for some polynomial $f(x)$ you know that there are N_i (congruence classes of) solutions to*

$$f(x) \equiv 0 \pmod{n_i} .$$

Then the congruence

$$f(x) \equiv 0 \left(mod \prod_{i=1}^{k} n_i \right) \text{ has } \prod_{i=1}^{k} N_i \text{ total solutions.}$$

Proof. From among the N_i solutions to each congruence, choose a set $a_1, a_2, \cdots a_k$ of solutions of

$$f(x) \equiv 0 \pmod{n_i}$$

for all i. This is a set of k numbers a_i modulo n_i. Now we can u se , which means that we can use the Chinese Remainder Theorem (Theorem 5.3.1) to get one number a such that $a \equiv a_i \pmod{n_i}$ for all i.

Since modular arithmetic is well-defined, and since polynomials are *exclusively* composed of addition and multiplication, $f(a)$ must be equivalent (modulo $\prod_{i=1}^{k} n_i$) to the number which the CRT gives from the system $x \equiv 0 \pmod{n_i}$. This is, of course, zero, so we obtain one solution to

$$f(x) \equiv 0 \left(\mod \prod_{i=1}^{k} n_i \right) .$$

Do this for each combination of solutions to the individual congruences. Clearly all such solutions must be different modulo $\prod_{i=1}^{k} n_i$. How many are there? Simply multiply how many there are for each n_i to get the total number of combinations of solutions. If there are N_i solutions modulo n_i, we would get $\prod_{i=1}^{k} N_i$. There aren't any *additional* answers, because any answer to the 'big' congruence automatically also satisfies the 'little' ones; if $\prod_{i=1}^{k} n_i \mid f(a)$, then certainly $n_i \mid f(a)$ as well. □

7.2.2 Prime power congruences

We have already discussed prime power congruences in Subsection 6.5.2. Recall that in Examples 6.5.3 and 6.5.4 we took the (obvious) solution of $2x \equiv 3 \pmod 5$ (namely, $x = [4]$), and from it relatively easily got solutions (mod 25) and even (mod 125).

But that is essentially the same as asking for solutions to $2x-3 \equiv 0$, a linear congruence. Let's see if we can generalize this for more general polynomial congruences.

The key was taking the *already known* fact $5 \mid 3 - 2 \cdot 4$ and then cancelling out 5 from the entire congruence to get that

$$\frac{3 - 2 \cdot 4}{5} \equiv 2k \pmod 5 .$$

We were able to solve the resulting congruence $-1 \equiv 2k \pmod 5$, which had solution $k \equiv 2 \pmod 5$. Finally, we plugged that back in to get a solution to $2x \equiv 3 \pmod{5^2}$, which was

$$4 + 5k = 4 + 5 \cdot 2 = 14 \pmod{5^2}$$

as the solution.

But can we use this to get solutions to more advanced congruences as well, like the simple quadratics we've started exploring in this chapter? The answer is yes, with a minor caveat. The preceding discussion was just a basic form of the following.

Theorem 7.2.3 (Hensel's Lemma)**.** *For p prime, suppose you already know a solution equivalence class x_{e-1} (mod p^{e-1}) of the (polynomial) congruence*

$$f(x) \equiv 0 \text{ (mod } p^{e-1})$$

Assume the technical condition that $\gcd(p, f'(x_{e-1})) = 1$*. Then there is a solution to*

$$f(x) \equiv 0 \text{ (mod } p^e)$$

of the form

$$x_e = x_{e-1} + kp^{e-1}$$

where k satisfies

$$\frac{f(x_{e-1})}{p^{e-1}} + k \cdot f'(x_{e-1}) \equiv 0 \text{ (mod } p).$$

Proof. If p and $f'(x_{e-1}))$ are relatively prime, then any linear congruence of the form $kf'(x_{e-1}) \equiv a \pmod p$ can be solved. Since x_{e-1} is a known zero of $f(x)$ for modulus p^{e-1}, we know that as an *integer* (not modulo anything) $p^{e-1} \mid f(x_{e-1})$.

This means that if we set $a = -\frac{f(x_{e-1})}{p^{e-1}}$, there will indeed be a solution k to the congruence to be solved in the statement. Then the only question becomes why $x_e = x_{e-1} + kp^{e-1}$ is actually a solution to $f(x) \equiv 0 \pmod{p^e}$.

To see this, think of f as a polynomial with terms of the form $c_i x^i$; then $f(x_{e-1} + kp^{e-1})$ can be expanded out term-by-term. Each term will look like

$$c_i(x_{e-1} + kp^{e-1})^i = c_i x_{e-1}^i + c_i(x_{e-1}^{i-1} \cdot kp^{e-1}) \cdot i + \text{ terms with at least } p^{2(e-1)}$$

Since $e \geq 2$ in this context, all those extra terms will be divisible by at least p^e and hence irrelevant in that modulus, so we have terms like

$$c_i x_{e-1}^i + c_i \cdot i x_{e-1}^{i-1} \cdot kp^{e-1} \pmod{p^e}$$

(Most treatments are similar to this, e.g. [C.1.1], with a binomial theorem-esque treatment, but one could also think of it as Taylor expansion, such as in [C.1.13].)

We're nearly done. Recall Proposition 5.2.4 where we are allowed to cancel a nonzero divisor from "all three sides" of a congruence. That motivates dividing each term and the modulus by p^{e-1} to get

$$\frac{c_i x_{e-1}^i}{p^{e-1}} + c_i \cdot i x_{e-1}^{i-1} \cdot k \pmod p$$

Now add up the terms for all i and it should be pretty clear from the form that we have

$$\frac{f(x_{e-1})}{p^{e-1}} + f'(x_{e-1}) \cdot k$$

Since we just showed this is divisible by p, we multiply everything by p^{e-1} (this time actually using Proposition 5.2.4) and get $f(x_e) \equiv 0 \pmod{p^e}$ as desired. □

Example 7.2.4. Let's use this to take solutions to $x^2 + 1 \equiv 0 \pmod 5$ and get solutions modulo 25 and 125.

First, by inspection the solutions modulo 5 are [2], [3] (or [±2]). So solutions modulo 25 will look like $3 + k \cdot 5$ or $2 + k \cdot 5$. Further, $f'(x) = 2x$, so for either solution modulo 5 the technical derivative condition is met.

Let $x_1 = 3$. Then the condition for k is

$$\frac{f(x_1)}{5} + k \cdot (2x_1) \equiv 0 \pmod 5$$

which simplifies to $2 + 6k \equiv 0$, which solves to $k \equiv -2 \equiv 3$. Then our solution to the congruence modulo 25 would be

$$x_2 = x_1 + 3 \cdot 5 \equiv 18 \pmod{25}$$

And indeed $18^2+1 = 325$ is divisible by twenty-five. Try the same procedure with $x_1 = 2$ to get the solution $x_2 \equiv 7$. (See also Example 16.1.4.)

Example 7.2.5. The same process with $e = 3$ and $x_2 \equiv 7$ yields, as a condition for k,

$$\frac{7^2 + 1}{25} + 14k \equiv 0 \pmod 5$$

This gives $k = 2$, and indeed

$$x_3 = x_2 + 2 \cdot 5^2 = 57$$

yields

$$57^2 + 1 = 3250 \equiv 0 \pmod{125}$$

This is a very powerful technique. What is most interesting is that this is even interpretable as Newton's method in calculus. How? Note that the result above can be rearranged as

$$x_e = x_{e-1} - \frac{f(x_{e-1})}{f'(x_{e-1})}$$

since $p^{e-1} \mid f(x_{e-1})$ and the technical condition is tantamount to saying $f'(x_{e-1})$ has an inverse. (Unlike in the Newton case, it is also possible for there to be solutions here if $\gcd(p, f'(x_{e-1})) \neq 1$, but only if $\frac{f(x_{e-1})}{p^{e-1}}$ itself is also divisible by p; we omit details of this case.)

If you didn't notice this, don't feel bad! In the linear case, where $f(x) = 2x - 3$, the derivative was just $f'(x) = 2$ and it was not at all obvious that anything more than a trick was involved. Still, it's another fascinating place where ideas from calculus can invade the world of number theory.

7.3 Congruences as Solutions to Congruences

We need to start applying these ideas more. In Section 7.1 we explored the number of solutions to $x^2 - 1 \equiv 0 \pmod n$ for arbitrary n. It should be clear we expect *at least* two solutions, but why are there sometimes more? Could we ever get a solution that is comprehensible?

```
@interact
def _(n=(12,[10..110])):
    counter = 0
    pretty_print(html("Values␣of␣$x^2-1$␣mod␣%s"%(n,)))
    pretty_print(html("<ul>"))
```

```
    for m in [0..n]:
        pretty_print(html("<li>$%s^2-1\equiv␣%s\\text{␣
            (mod␣}%s)$</li>"%(m,mod(m,n)^2-1,n)))
        if mod(m,n)^2-1==0:
            counter += 1
    pretty_print(html("</ul>"))
    pretty_print(html("There␣are␣$%s$␣solutions␣to␣
        $x^2-1\equiv␣0$␣(mod␣$%s$)."%(counter,n)))
```

Since x^2-1 is a polynomial, our knowledge of Fact 7.2.2 suggests we should try to answer this by looking at different *prime power* moduli first, then multiply the answers.

The key idea we will use is this:

- We know that (for a prime p) $p \mid x^2-1 = (x-1)(x+1)$ implies $x \equiv \pm 1 \pmod{p}$.
- More generally, $p^e \mid (x-1)(x+1)$ implies p divides $x-1$ or $x+1$.

So we should just look at various p^e.

If p is odd (and hence greater than two), the two possibilities $p \mid x-1$ and $p \mid x+1$ are mutually exclusive, so *all* the factors of p in p^e need to divide the same one. So $p^e \mid (x+1)$ or $p^e \mid (x-1)$ are the only possibilities ($x \equiv [\pm 1]$) and there are two solutions.

However, if $p = 2$ then $2 \mid x-1$ and $2 \mid x+1$ is definitely possible, so there could be more solutions.

- We know that ± 1 are still the only solutions for 2^2 and 2^1; in the latter case $+1 \equiv -1$, so there is actually only *one* solution in this case.
- However, for 2^3 it's possible that $2 \mid (x+1)$ and $2^2 \mid (x-1)$, or vice versa, so that $2^2 \pm 1 = 3, 5$ are also solutions to the congruence.
- For higher powers of 2 this sort of thing can happen, too. For instance, one could have $2 \mid (x+1)$ and $2^4 \mid (x-1)$. However, it's not possible that $2^2 \mid (x+1)$ and $2^3 \mid (x-1)$ because numbers two apart can't both be divisible by four. So the only other possibility is that $2 \mid (x+1)$ and $2^{e-1} \mid (x-1)$, or vice versa, which is a total of four solutions.

That means we get a very intriguing answer.

Fact 7.3.1. *Let k be the number of* odd *primes that divide n. Consider the congruence $x^2 - 1 \pmod{n}$. Then:*

- *There are 2^k solutions if n is odd.*
- *There are $1 \cdot 2^k$ solutions if n is divisible by 2 but not by 4.*
- *There are $2 \cdot 2^k = 2^{k+1}$ solutions if n is divisible by 4 but not by 8.*
- *There are $4 \cdot 2^k = 2^{k+2}$ solutions if n is divisible by 8.*

Proof. Use Fact 7.2.2 and the argument above. □

What does this have to do with the title of this section? Let's recast the result.

- There are 2^k solutions if $n \equiv 1 \pmod{2}$, or $n \equiv 1, 3, 5, 7 \pmod{8}$.

- There are 2^k solutions if $n \equiv 2 \pmod 4$, or $n \equiv 2, 6 \pmod 8$.
- There are $2 \cdot 2^k = 2^{k+1}$ solutions if $n \equiv 4 \pmod 8$.
- There are $4 \cdot 2^k = 2^{k+2}$ solutions if $n \equiv 0 \pmod 8$.

This is only the first of many such results.

7.4 Polynomials and Lagrange's Theorem

We've seen several times in this chapter that although one can have theorems of various kinds for congruences, polynomials seems to behave very nicely – even to the point of allowing us to prove statements about the *integer* output of polynomials!

At the same time, it's clear that for good behavior, there is no substitute for prime moduli; the results in the previous sections really confirm this. So how can we combine polynomials and prime modulus?

Theorem 7.4.1 (Lagrange's Theorem for Polynomials)**.** *If p is prime and $f(x)$ is a non-trivial polynomial with integer coefficients of degree d, then there are at most d congruence classes of solutions modulo p.*

Proof. This proof is fairly detailed, so feel free to try it out with specific numbers. It proceeds via induction on the degree d of the polynomial.

First, consider the case where there are *no* solutions to $f(x) \equiv 0 \pmod p$. Then there is nothing further to prove, since $0 \leq d$ for any polynomial. This actually proves a base case, for if the degree is $d = 0$ then $f(x) = c$ for $c \neq 0$. (If $c = 0$ we have the trivial polynomial, which is not a covered case.)

For another base case, suppose that the degree $d = 1$. Then we have $ax + b \equiv 0 \pmod p$, which is the same as $ax \equiv -b \pmod p$. In this case $\gcd(a, p) = 1$ and there is exactly one solution by Proposition 5.1.2 (if $ax + b$ is actually going to have a linear term, otherwise $p \mid a$).

Now we'll use some induction. Let's assume that all polynomials with degree e less than d have at most e solutions modulo p.

- So assume that f has degree d, i.e.

$$f(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0$$

  We already dealt with the case where f has no solutions, so assume that $f(b) \equiv 0 \pmod p$ for at least one congruence class $[b]$.

- Remember the factorization

$$\left(x^k - b^k\right) = (x - b)\left(x^{k-1} + \cdots + b^{k-1}\right)$$

  (We could have used this to prove Fact 4.2.3.) Now let's apply that to

$$f(x) - f(b) \equiv f(x) \equiv$$

$$\left(a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0\right) - \left(a_d b^d + a_{d-1} b^{d-1} + \cdots + a_1 b + a_0\right) =$$

$$a_d\left(x^d - b^d\right) + a_{d-1}\left(x^{d-1} - b^{d-1}\right) + \cdots + a_1(x - b)$$

  to get

  $(x - b) \cdot$ (A bunch of stuff, but factored out and hence of lower degree) .

- Now consider the condition that $f(x) \equiv 0$. Based on this, it can be written in two ways, recalling that $f(b) \equiv 0$:
  - $f(x) \equiv 0$
  - $f(x) \equiv f(x) - f(b) \equiv (x-b) \cdot \text{Stuff}(x)$

  Therefore
  $$f(x) \equiv (x-b) \cdot \text{Stuff}(x) \equiv 0 \pmod{p}$$
  implies that p divides the product of $x-b$ and the stuff.
- The "Stuff" function must be a polynomial of degree less than d, so we can assume there are at most $d-1$ solutions to it modulo p. There is only one extra way to divide $x-b$, so there are at most d solutions available for $f(x)$, including $x \equiv b$.
- But $f(x)$ was an arbitrary polynomial of degree d, so it works for all polynomials of degree d.

So by induction, it works for any polynomial. □

We just saw this result isn't true for general moduli. In Section 7.3 we got as many as 2^{k+2} solutions to $x^2-1 \equiv 0$ for moduli that looked like $8p_1p_2\cdots p_k$. We would expect only *two* with Lagrange's Theorem.

But whatever the solution to the $x^2 \pm 1$ problems are modulo a *prime*, there cannot be more than 2 solutions to them! If we find two solutions, we have all of them. This proves to be quite useful to keep things from going crazy when we are trying to investigate congruences; if we keep the modulus prime, we will be okay.

Of course, we also might not even get all the solutions possible in theory. We might not even get two in some instances of a quadratic polynomial, since $x^2+1 \equiv 0$ doesn't have a solution modulo three (just try all three options). The following interact investigates this a bit more.

```
@interact
def _(n=(13,prime_range(100))):
    counter = 0
    pretty_print(html("Zero␣values␣of␣$x^2+1$␣mod␣
        %s"%(n,)))
    pretty_print(html("<ul>"))
    for m in [0..n-1]:
        if mod(m,n)^2+1==0:
            pretty_print(html("<li>$%s^2+1\equiv␣%s\\text{␣
                (mod␣}%s)$</li>"%(m,mod(m,n)^2+1,n)))
            counter += 1
    pretty_print(html("</ul>"))
    pretty_print(html("There␣are␣$%s$␣solutions␣to␣
        $x^2+1\equiv␣0$␣(mod␣$%s$)."%(counter,n)))
```

Maybe that's not so surprising, since we don't have zeros of x^2+1 over the real numbers either. Could there be connections or parallels?

7.5 Wilson's Theorem and Fermat's Theorem

Polynomials aren't the only types of formulas we will see. Here, we introduce two famous theorems about other types of congruences modulo p (a prime) that will come in very handy in the future.

7.5.1 Wilson's Theorem

Theorem 7.5.1 (Wilson's Theorem)**.** *If p is a prime, then*

$$(p-1)! \equiv -1 \pmod{p} .$$

Proof. If $p = 2$ this is very, very easy to check. So assume $p \neq 2$, hence $p-1$ is even.

Now we will think of all the numbers from 1 to $p-1$, which will be multiplied.

For each n such that $1 < n < p-1$, we know that n has a unique inverse modulo p. Pair up all the numbers between (not including) 1 and $p-1$ in this manner.

- For instance, if $p = 11$, pair up $(2, 6)$, $(3, 4)$, $(5, 9)$, and $(7, 8)$.

Then multiplying out $(p-1)$ factorial, we can reorder the terms thus, and notice the cancellation:

$$(p-1)! \equiv 1 \cdot 2 \cdot 3 \cdots (p-2) \cdot (p-1) \equiv 1 \cdot a \cdot a^{-1} \cdot b \cdot b^{-1} \cdots (p-1)$$

$$\equiv 1 \cdot 1 \cdot 1 \cdots 1 \cdot (p-1) \equiv (p-1) \equiv -1 \pmod{p}$$

- For instance, if $p = 11$, we pair up

  $$10! \equiv 1 \cdot 2 \cdots 9 \cdot 10 \equiv 1 \cdot (2 \cdot 6) \cdot (3 \cdot 4) \cdot (5 \cdot 9) \cdot (7 \cdot 8) \cdot 10$$

  which simplifies to

  $$10! \equiv 1 \cdot 1 \cdot 1 \cdot 1 \cdot -1 \pmod{p}$$

Beautiful! The only loose end is that perhaps some number pairs up with itself, which would mess up that all the numbers pair off nicely.

- However, in that case, $a^2 \equiv 1 \pmod{p}$, so by definition $p \mid (a-1)(a+1)$.
- Since p is prime, it must divide one or the other of these factors, and in either case $a \equiv 1$ or $a \equiv p-1$.
- But we were not pairing off 1 or $p-1$, so this can't happen.

□

One exercise below is to show that Wilson's theorem fails for $p = 10$. That is, that $(10-1)! \not\equiv -1 \pmod{10}$. So does it work or not for other moduli?

```
@interact
def _(n=range_slider(2,100,1,(3,9))):
    for modulus in [n[0]..n[1]]:
        pretty_print(html("$(%s-1)!\equiv %s$ (mod
            $%s$)"%(modulus,
            mod(factorial(modulus-1),modulus), modulus)))
```

7.5.2 Fermat's Little Theorem

If one explores a little with powers of numbers modulo p a prime, one usually notices some pattern of those powers. This is the best-known, and soon we'll reinterpret it in a powerful way.

Theorem 7.5.2 (Fermat's Little Theorem)**.** *If* $\gcd(a,p) = 1$ *for* p *a prime, then*

$$a^{p-1} \equiv 1 \ (mod\ p)\,.$$

Proof. Sketch of proof (to fill in, see Exercise 7.7.8):

- If $\gcd(a,p) = 1$ and p is prime, show that $\{a, 2a, 3a, \ldots, (p-1)a, pa\}$ is a complete residue system (mod p).
  - That is, show that the set $\{[a], [2a], [3a], \ldots, [pa]\}$ is the same as the complete set of residues $\{[0], [1], [2], \ldots, [p-1]\}$, though of course in a different order.
- If p is prime and p does not divide a, then

$$a \cdot 2a \cdot 3a \cdots (p-1)a \equiv 1 \cdot 2 \cdot 3 \cdots (p-1) \pmod{p}\,.$$

- Now use Wilson's Theorem.

□

There are other ways to prove it as well. There is a beautiful approach in terms of counting necklaces or strings of pearls which requires essentially no number theory, but rather basic ideas from **combinatorics**, the discipline of counting well. We'll see a more abstract approach after we introduce the concept of **groups** in Chapter 8; see Exercise 9.6.1. So despite the innocuous appearance of this result as a corollary of another theorem, do not be fooled. It is incredibly powerful.

7.6 Epilogue: Why Congruences Matter

Although we will spend some significant time working on solving congruences, I haven't forgotten deeper questions. To see how congruences can impact this, recall the search in Section 7.1 for primes p such that

$$x^2 \equiv -1 \pmod{p}$$

has a solution. Take a look at this table and see if you can find something.

```
@interact
def _(n=20):
    yeslist=[]
    nolist=[]
    for p in prime_range(3,n):
        res = 0
        for res in [0..p]:
            if mod(res,p)^2+1 == 0:
                yeslist.append(p)
                break
        else:
            nolist.append(p)
```

```
    t = [['exist', 'do not exist']] + [[a,b] for (a,b) in
        map(None,yeslist,nolist)]
    for item in t:
        for i in range(len(item)):
            if item[i] is None:
                item[i]=''
    pretty_print(html("Solutions to $x^2\equiv -1$ (mod
        $p$) for $2\le p\le %s$:"%n))
    pretty_print(html(table(t, header_row = True, frame =
        True)))
```

Question 7.6.1. Do you see a pattern related to some kind of congruence? (This one should be more apparent than in Section 7.3; see also Exercise 7.7.10.)

The reason I point this kind of thing out is not just because I can, but because it shows simple congruence patterns can have a big result. We will prove a result about *integers*, assuming something about *congruences*.

Recall our search through Mordell/Bachet curves, and let's look at the particular case $y^2 = x^3 + 7$.

```
f(x,y)=y^2-x^3-7
@interact
def _(viewsize=slider(3,40,1)):
    p = implicit_plot(f, (x,-viewsize,viewsize),
        (y,-2*viewsize,2*viewsize), plot_points = 200)
    lattice_pts = [[i,j] for i in [-viewsize..viewsize]
        for j in [-2*viewsize..2*viewsize]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
    curve_pts = [coords for coords in lattice_pts if
        f(coords[0],coords[1])==0]
    if len(curve_pts)==0:
        show(p+plot_lattice_pts, figsize = [5,5])
    else:
        plot_curve_pts = points(curve_pts, rgbcolor =
            (0,0,1),pointsize=20)
        show(p+plot_lattice_pts+plot_curve_pts, figsize =
            [5,5])
    pretty_print(html("Solutions of $x^3+7=y^2$ in this
        viewing window"))
```

It's amazing how the curve slips between every integer lattice point... So it seems that a perfect square can't *ever* be exactly seven more than a perfect cube. Is this true? Here's where congruences come into play.

Proposition 7.6.2 (Showing a Mordell curve has no integer point)**.** *There is no integer x such that $y^2 = x^3 + 7$, so there are no integer points on this curve.*

Proof. First let's consider the case where x is even. Then $2 \mid x$, so $8 \mid x^3$. That means $y^2 \equiv 7 \pmod 8$.

```
[i^2 for i in Integers(8)]
```

Unfortunately, the only perfect squares mod (8) seem to be 0, 1, and 4. So this is not possible.

What about if x is odd? Then y must be even, since x^3 and 7 are odd. So let's examine whether $x \equiv 1 \pmod 4$ or $x \equiv 3 \pmod 4$, the next two options.

- If $x \equiv 3 \pmod 4$, then $x^3 \equiv 27 \equiv 3 \pmod 4$, so $x^3 + 7 \equiv 10 \equiv 2 \pmod 4$. But we already know from earlier that perfect squares are only 0 or 1 modulo 4, so that's not possible.
- So it must be the case that $x \equiv 1 \pmod 4$.

Now we do a trick like that of completing the square:

$$y^2 = x^3 + 7 \Rightarrow y^2 + 1 = x^3 + 8 \Rightarrow y^2 + 1 = (x+2)(x^2 - 2x + 4)$$

Let's analyze this carefully in the following argument.

- If $x \equiv 1 \pmod 4$, then $x + 2 \equiv 3 \pmod 4$.
- So not only is $x + 2$ an odd number, but also it must be divisible by a prime q of the form $4n+3$. (Otherwise all its primes look like $4n+1 \equiv 1$, the product of which would also be $\equiv 1 \pmod 4$.)
- If q divides $x+2$, it (naturally) divides $(x+2)(x^2-2x+4)$ as well. But if it divides $(x+2)(x^2-2x+4)$, it must then divide y^2+1, since they're equal.
- However, our exploration said that a prime of the form $4n+3$ can't divide y^2+1! So, assuming this is true, $x \equiv 1 \pmod 4$ doesn't work either.

As a note, we will eventually prove the result of exploration in Fact 13.3.2, but you may want to try to find an "elementary" proof in Exercise 7.7.10. □

Enough said; congruences are amazingly powerful.

7.7 Exercises

1. Pick one, and really do some exploration and write about it. See Section 7.1 for more information.

- Do exploration to try to find a criterion for which primes p there are square roots of -1. You will have to examine primes less than 10 by hand to make sure you are right!
- Do exploration to find out anything you can about how many square roots of 1 there are for a given n.

2. Figure out how many solutions $x^2 \equiv x \pmod n$ has for $n = 5, 6, 7$, and then compute how many solutions there are modulo 210.

3. Find solutions to $x^2 + 8 \equiv 0 \pmod{121}$ using the method above in Theorem 7.2.3.

4. Solve $f(x) = x^3 - x^2 + 2x + 1 \equiv 0 \pmod{5^e}$ for $e = 1, 2, 3$.

5. Show that Wilson's Theorem fails for $p = 10$ and check that it works for $p = 11$ by computing 11! and then reducing.

6. In the same setup as in Wilson's Theorem, what is the value of $(j-2)!$, depending upon the modulus?

7. Use Fermat's Little Theorem to help you calculate each of the following *very* quickly:

- $512^{372} \pmod{13}$
- $3444^{3233} \pmod{17}$

- $123^{456} \pmod{23}$

8. Prove Fermat's Little Theorem using the steps in Theorem 7.5.2, or any way you would like.

9. Prove that Wilson's Theorem always fails if the modulus is not prime. Hint: use the fact that the modulus n then has factors m other than 1 or n.

10. Prove that it is impossible for $p \mid x^2 + 1$ if a prime p has $p \equiv 3 \pmod 4$ – that is, if p is of the form $4n + 3$. (Hard.)

11. Prove that $x^2 + y^2 = p$ has no (integer) solutions for prime p with that same form.

12. Show that $y^2 = x^3 + 999$ has no (integer) solutions.

Chapter 8

The Group of Integers Modulo n

This chapter does not do any number theory, per se. Yet it is at the heart of the text. We introduce two powerful methods to deal with integers modulo n – visualizing them graphically, and the language of group theory.

There is no prerequisite in either case; do not feel worried if you have not encountered algebraic structures like groups before. We will only take and introduce what we need, and refer back to fundamental properties often.

8.1 The Integers Modulo n

8.1.1 Definition

It is time for us to finally define what we have been working with for quite a while now.

Definition 8.1.1 (Integers Modulo n)**.** For a positive integer n, the set of equivalence classes of integers modulo n is called the **integers modulo** n. We denote it $\mathbb{Z}_n$. That is,

$$\mathbb{Z}_n = \{[0], [1], [2], \cdots, [n-2], [n-1]\} .$$

In the case where $n = p$ is a prime, we usually write $\mathbb{Z}_p$. (For those who have had an abstract algebra course, this may be different notation than you have used, but we will be consistent with this usage.)

This friendly number system will become a good acquaintance, if not friend, throughout the rest of the course. We'll explore it soon, but first let's see some of the basic properties.

As it turns out, $\mathbb{Z}_n$ has several very interesting properties. Like all of our number systems in this class, you can add and multiply elements of $\mathbb{Z}_n$ (we call something like that a **ring**). This is true because of our earlier proof of well-definedness for addition and multiplication in Proposition 4.3.2.

As a first step in visualizing, we can make an addition table. This is not very interesting. But in some sense, it is interesting that it isn't interesting. Does that make any sense?

```
@interact
def addition_table_(n=(11,[2..50])):
    P=[[mod(a,n)+mod(b,n) for a in [0..n-1]] for b in
        [0..n-1]]
```

```
    pretty_print(html("The␣addition␣table␣for␣modulus␣
        $%s$"%(n,)))
    pretty_print(html(table(P, header_row = True, frame =
        True)))
```

The top row and left column may be considered as a list of a and b. Any ideas about patterns here?

It's also possible to make a multiplication table. This makes things a little more interesting.

```
@interact
def _(n=(11,[2..50])):
    P=[[mod(a,n)*mod(b,n) for a in [0..n-1]] for b in
        [0..n-1]]
    pretty_print(html("The␣multiplication␣table␣for␣
        modulus␣$%s$"%(n,)))
    pretty_print(html(table(P, frame=True)))
```

Again, notice that the columns and rows are both from 0 to $n-1$; this is standard. For now we'll usually just use the set of least nonnegative residues to represent $\mathbb{Z}_n$; recall that this is $\{[0],[1],[2],\ldots,[n-2],[n-1]\}$.

Are there any patterns you notice here?

There is at least one observation that is curious. For some moduli, the only zeros are where we expect them, in the top row and left column. For others, they are in other spots.

8.1.2 Visualization

What's even better is to see this visually! I still can't get over how easy it is for me to do this in Sage (and other math programs); it is so cool that even my non-mathematician wife says, "What's that – it's neat!"

```
@interact
def multiplication_table_plot(n=(7,[2..50])):
    P=matrix_plot(matrix(n,[mod(a,n)*mod(b,n) for a in
        srange(n) for b in srange(n)]),cmap='jet')
    show(P,figsize=7)
```

How does one interpret this? The a row and b column give the color corresponding to $a \cdot b \pmod{p}$. That means the first (0th) column is the color for $a \cdot 0 = 0$ and the second (1st) column gives the colors of each element $a \cdot 1 = a$ of $\mathbb{Z}_n$. Since zero times anything is zero, that gives us a lot of that color along two edges.

Can you see the difference between prime and composite moduli better now?

8.1.3 Inverses

Let's focus on the tables/graphs for when $n = p$ a prime. There's at least one interesting observation we can make about them. *Every* row and every column (other than the ones corresponding to 0) has the entry 1 in it. (That's the deepest nonzero blue in the default coloring.) You can't necessarily say this about other numbers. Let's translate this into notation.

Fact 8.1.2. *Every nonzero element of $\mathbb{Z}_n$ has an inverse.*

Proof. If $\gcd(a, n) = 1$, then $ax \equiv b \pmod{n}$ has a unique solution in $\mathbb{Z}_n$. So if $n = p$ is prime, then $\gcd(a, p) = 1$ always, except for $a \equiv 0$.

Now we let $b = 1$, and finding x becomes the same as finding an *inverse* element of a. So for prime moduli, every non-zero element has a unique inverse in $\mathbb{Z}_p$. □

(In algebraic nomenclature, this means $\mathbb{Z}_p$ is a **field**, yet another example of bizarre but fun math terminology.)

What was the command again to get an inverse?

```
inverse_mod(26,31)
```

It turns out there is an even easier way to get at this in Sage than the one I used last time! In retrospect, it makes sense.

```
c = mod(26,31)
c^-1
```

```
c = mod(26,31)
c*c^-1
```

Go back to the graphics or tables. Can you "see" that there is (exactly one) inverse for every non-zero element of $\mathbb{Z}_p$?

8.2 Powers

Let's continue to restrict ourselves to looking at $\mathbb{Z}_p$, the integers modulo some prime p, for a bit longer. This will enable us to get a little more detailed in our exploration. We eventually want to explore solutions to congruences modulo primes and prime powers.

Let's begin by exploring powers. Powers are particularly important, since polynomials are constructed from them. The following interact allows exploration of powers a^n modulo p for various primes p and bases a. Notice I have *not* yet brought in the colors.

```
@interact(layout=[['p','a']])
def _(p=(7,prime_range(50)),a=(3,[0..50])):
    b=mod(a,p)
    top=ceil(2*p/10)*10
    pretty_print(html("If␣we␣look␣at␣some␣of␣the␣powers␣of␣
        $%s$"%(a,)))
    pretty_print(html("modulo␣the␣prime␣$%s$,␣we␣
        get:"%(p,)))
    pretty_print(html("<ul>"))
    for m in [0..top]:
        pretty_print(html("<li>$%s^{%s}\equiv␣%s\\text{␣
            (mod␣}%s)$</li>"%(a,m,b^m,p)))
    pretty_print(html("</ul>"))
```

Do you see any patterns? It's probably a little early to try to come up with potential theorems, but there should be at least some patterns you see. Do you maybe even see any theorems we have already proved in here?

One of the biggest patterns is hard to see in this format, but is the simplest. Given a prime p, you should get get the same answers for $a \equiv a' \pmod{p}$. (This is the essence of the earlier proof about a polynomial not having only prime output, Fact 6.1.4.) So we should really just restrict ourselves to looking at $0 \leq a < p$.

8.2.1 Returning to visualizing

Still, this is a lot of data to assimilate. Is there some way to think about it differently?

This next interact is *super*-cool, because it combines the short, *color-coded* format with the much less familiar material of powers.

```
@interact
def power_table_plot(p=(7,prime_range(50))):
    P=matrix_plot(matrix(p-1,[mod(a,p)^b for a in
        range(1,p) for b in srange(p)]),cmap='jet')
    show(P,figsize=6)
```

The default coloring needs some explanation, as they are not the same as in the previous example. The a row and b column gives the color corresponding to $(a+1)^b \pmod{p}$. That means the first (0th) column is the color for $a^0 = 1$ and the second (1th) column gives the colors of each element $a^1 = a$ of $\mathbb{Z}_p$. For instance, $(2, 4)$ corresponds to $3^4 \equiv 4 \pmod{7}$ in the initial example. Notice this color corresponds to the *row 3*, because of the numbering.

(As far as I know, this representation first appears in Wagon and Bressoud's excellent computational number theory text [C.3.7]. The PascGalois project has related visualizations.)

Sage note 8.2.1 (Colorful options)**.** If you don't like the colors, you can change the word in the quotes after the word `'cmap'`; if you get rid of that, it will be a grayscale plot, which is most appropriate for vision-impaired users. Some others you could try are `'Oranges'` or `'hsv'` or … Well, see the next Sage cell if you *really* want to know all of them!

```
for c in colormaps:
    print c
```

What color patterns can you see here? To say it another way, what potential theorems do you see? (Again, do you see any that we already have discussed?)

In a classroom or self-study situation, I strongly recommend thinking about this until coming up with some nice potential theorem regarding whether there are any patterns in $a^b \pmod{p}$ that hold for all p or all a or all b, or *something*.

8.3 Essential Group Facts for Number Theory

Many of the bookkeeping issues which arise in number theory can be made much easier by changing our language and introducing a small amount of abstraction. That abstraction is the concept of 'group'. These notes will introduce this concept in the most basic way possible, with only the minimum needed to translate many difficult arguments into simpler language.

8.3.1 Step-by-step notions to the definition

We will take an approach that starts with the familiar and adds properties until we reach our goal.

8.3.1.1 Sets

Sets are just what you think. They are collections of (mathematical) stuff.

In our uses of groups, we will exclusively be concerned with sets that are collections of numbers, like P, the set of primes, and $\mathbb{Z}$, the set of integers, or $\mathbb{Z}_n$, the set of equivalence classes of integers modulo n. But it's helpful to think more generally.

8.3.1.2 Binary operations

A **binary operation** is a set with a multiplication table on it. That's it.

Usually books call it $*$ or something like that, and then define a binary operation on the set S to be a function from $S \times S$ to S.

- Usually this would be (say) normal addition or multiplication on numbers, though it could also be subtraction.
- On the other hand, if S is the set of continuous functions on $\mathbb{R}$, the operation could be composition of functions, $f \circ g$.

Notice that if our set is $\mathbb{Q}$ and our operation is division, we don't have a full table. The essential thing is that it's a set with a table or rule for the operation.

8.3.1.3 Closed operations

A binary operation is called **closed** if you don't get anything outside the set with your operation. This is important because it's easy to break this.

- If you are *adding* two positive numbers, for instance, you always get a new positive number.
- Is this still true if you subtract two positive numbers from each other?
- This also can happen with division, right? You have to look at $\mathbb{Q}$, and then you have to be careful because of the previous point.
- For a more complicated example, let S be the set of 2x2 matrices with determinant 1; if you add two of them, your determinant might change a lot.
- On the other hand, if you *multiply* two such matrices, you're golden; the determinant will still be 1.

8.3.1.4 Associative operations

An operation is **associative** if it doesn't matter how you put parentheses in.

This is not an algebra course, so I won't harp on this – everything we do will satisfy it in obvious ways. But it's worth noting that exponentiation is *not* associative, so it's not a trivial condition.

Example 8.3.1.

$$2^{(2^3)} = 2^8 = 256 \text{ but } (2^2)^3 = 4^3 = 64\,.$$

8.3.1.5 Identity

Much more important is whether your operation has an **identity element**.

You have seen this many times before in addition and multiplication.

$$a + 0 = a = 0 + a \text{ and } a \cdot 1 = a = 1 \cdot a\,.$$

When we turn this into abstract math, we say that an identity for a general operation $*$ on a set S is an element, conveniently called e, which has the very nice property that if you $*$ by it, you get the same thing back.

- That is, $e * a = a = a * e$ for any $a \in S$.
- The identity matrix under matrix multiplication is another example.
- By the way, if there is an identity, there's only one, which is sometimes useful to know.

Example 8.3.2. Here is a more interesting example. Let your set be the set of all rotations of a square which leave it facing the same way. That is, rotation by 90 degrees to the left, 180 degrees right, etc. (Think of a child's block sorter.)

- The binary operation would be to do one rotation, and then the other one.
- Then an identity element e of this is just *to leave the block alone*!

This is sort of weird at first, but an extremely important example.

8.3.1.6 Inverses

Almost there! Let's keep thinking about that last example. Say I turn the block 90 degrees to the right, then I realize I made a horrible mistake and want to get back to the original position. Is there anything I can do, short of buying a new square block?

Of course there is! Just turn it back 90 degrees to the *left*. So if I call the first move $90R$ and the second one $90L$, I can say that $90R * 90L = e$, since the net effect is the same.

Generalizing this, if a is an element of your set S and there is another element a' such that

$$a * a' = e = a' * a\,,$$

then we call a' an inverse of a.

- The absolute prototype of this is negative numbers. That is, for any number n, if you add $-n$, then you get zero!
- The same thing happens a lot; for matrix multiplication, the inverse matrix would be the operation inverse.
- For rational numbers, the reciprocal would be the inverse.

But notice that in both of these cases not every mathematical object *has* an inverse with respect to every operation! A matrix with determinant zero does not have an inverse matrix. In $\mathbb{Q}$ under multiplication, zero has no inverse.

8.3.2 What is a group?

Definition 8.3.3 (Group)**.** If a set and binary operation on that set is closed an associative with identity and inverses for *every* element, we call that set a **group**.

Example 8.3.4. The most excellent examples of this are the following:

- $\mathbb{R}, \mathbb{Q}, \mathbb{Z}$ under addition with zero as identity
- The sets $\mathbb{R}, \mathbb{Q}$ *except* zero (written as $\mathbb{Q} \setminus \{0\}$) under multiplication with 1 as identity
- $\mathbb{Z}_n$ under addition with $[0]$ as identity. For example, in $\mathbb{Z}_3$, every element has an inverse; $[0]' = [0]$, $[1]' = [2]$, and $[2]' = [1]$, because $[0] + [0] = [0] = [1] + [2]$.

Remark 8.3.5. If we are talking about any old group, we just call it G.

Also, after a while, it gets boring to always type $*$, and instead we just use normal multiplication notation, writing $x * y = xy$.

Example 8.3.6 (A preview of what's to come)**.** A final example is just a preview of what's to come.

- We noted that $\mathbb{Q} \setminus \{0\}$ is a group under multiplication, with 1 as the identity.
- Is there something analogous for $\mathbb{Z}_n$? Indeed there is, and we will see it soon. But notice that things will be more complicated.
- For instance, in $\mathbb{Z}_3$, both $[1]$ and $[2]$ have multiplicative inverses (in fact, themselves), so $\mathbb{Z}_3 \setminus \{[0]\}$ works, just like $\mathbb{Q} \setminus \{0\}$ does.
- But in $\mathbb{Z}_4$, $[2]$ also does not have a multiplicative inverse, so it would not make sense to say that $\mathbb{Z}_4 \setminus \{[0]\}$ is a group.
- That extra complication is one reason we need to think hard about these things!

8.3.3 Properties of groups we will need

The reason for introducing groups in a course which does not presume previous exposure to algebra is that is just makes things simpler. We will start here with familiar facts in a new guise, and then work our way to some facts which will prove invaluable.

8.3.3.1 Solutions to equations

Since a group has inverses, we can solve very simple 'linear' equations in them. This is stated as

$$a * x = b \text{ is solved by } x = a' * b (= a^{-1} * b) \ .$$

For instance, over $\mathbb{R}$, $a + x = b$ always has a solution for any real numbers a, b. We just take $x = b + (-a)$, where $-a$ is the inverse for the group operation of a (as mentioned above).

More important to us is the fact that in $\mathbb{Z}_n$, there are solutions. The operation is still $+$, so we have $a + x \equiv b \bmod(n)$ solved by $x \equiv (b + (-a)) \bmod(n)$.

This doesn't seem much more interesting, but you will see soon why this concept is so important.

8.3.3.2 Finite groups

A group can have finitely many or infinitely many elements. Most of our normal ones, such as $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, matrix groups, are infinite.

But the ones we'll use in this text will mostly have finitely many elements. This is because we are counting each equivalence class, like $[0], [1], [2]$ in (mod 3) arithmetic, as just one element.

A group with finitely many elements is called, unimaginatively, a **finite group**.

8.3.3.3 Order of a group

Definition 8.3.7. The number of elements of a finite group is called the **order** of the group.

For any old group G, we use $|G|$ as notation for its order.

Example 8.3.8. So if we are talking about $\mathbb{Z}_3$, it has 3 elements, so it has order 3 (unsurprisingly) and we write $|\mathbb{Z}_3| = 3$.

8.3.3.4 Order of an element

This is a tougher concept. Suppose you have some element, such as $[1] \in \mathbb{Z}_3$. If you just keep adding $[1]$ to itself, eventually you get back to zero, right? After all,

$$[1] + [1] + [1] \equiv [0] \text{ (mod 3)}.$$

Take a finite group G with order $|G| = n$. We will bring the concept of order to elements, not just groups.

First, list all elements of the group:

$$\{e = x_1, x_2, \ldots, x_n\}$$

Now let's take an element x, and start operating on it by itself. What I mean by this is listing $x, x*x = x^2, x^3, \ldots$. (Don't be confused by the power notation alternating with addition notation; $\mathbb{Z}_n$ has two operations, so we keep $+$ there, but in a general group we use multiplicative notation.)

Here is the key. There are only finitely many elements in the group, so by x^{n+1} at the latest, at least two of these 'powers' will be *equal*. (This argument, that if you fit $n+1$ objects into n slots then there must be a repeat, is called the **pigeonhole principle**, among other names.)

To be concrete, let's say $x^s = x^t$, with $s < t$. Now we can do a very curious thing. Take the *inverse* of x, written x^{-1}. If we multiply *it* together s times, we get $(x^{-1})^s$ which we can write x^{-s}. Then multiply $x^s = x^t$ by x^{-s};

$$x^{-s}x^s = x^{-s}x^t, \text{ or } e = x^{t-s}\ .$$

We are almost there! This means there is a positive integer k such that $x^k = e$. By the Well-Ordering Principle (Axiom 1.2.1), there is a *least* such integer. This integer, associated to a specific element of the group, is what we have been aiming for.

Definition 8.3.9. For a group element $x \in G$, the least integer k such that $x^k = e$ is called the **order** of the element x. We write it $|x|$, by analogy with the order of a group.

Example 8.3.10. For example, in $\mathbb{Z}_6$, look at the element $[4]$.

$[4]+[4]+[4]+[4]+[4]+[4] \equiv [0] \bmod(6)$, but $[4]+[4]+[4] \equiv [0] \bmod(6)$ too.

So 6 is a possible order, but clearly 3 is the smallest number of times that will yield $[0]$, so $|[4]| = 3$.

8.3.3.5 The connection

Here comes the coolest part, where we connect the two concepts of order. We will definitely use Theorem 8.3.11 in proving various theorems.

Take a look at any old element $x \in G$. If x has order m, then there are (at least) m distinct elements of G,

$$\{x, x^2, x^3, \ldots, x^{m-1}, e\} .$$

Now take *any* other element not in this subset, y, and look at the set

$$\{xy, x^2y, x^3y, \ldots, x^{m-1}y, ey = y\} ;$$

Note that these are also all distinct elements of the group. Are any of them also included in the first set (powers of x)?

Suppose that some $x^s y$ is the same as some x^t. That would mean $x^s y = x^t$, so multiplying by x^{-t} we get

$$x^{s-t}y = e$$

That would mean $y = x^{t-s}$, a contradiction since we said y isn't a power of x. Hence the new elements form a disjoint set from the previous set.

Now find an element z not in either set, and do the same thing. Then the set

$$\{xz, x^2z, x^3z, \ldots, x^{m-1}z, ez = z\}$$

will be disjoint from the other sets, and all its elements will still be distinct. Since G is finite, eventually doing this process again and again will fill out G completely.

Theorem 8.3.11 (Lagrange's Theorem on Group Order)**.** *The order of any element x of G divides the order of the group itself. We can write this as*

$$|x| \mid |G|$$

Proof. Examine the above argument. We have a number of subsets of G, all of size m, which exactly fill out G, which has size n. This forces that m divides n as integers. □

Example 8.3.12. For example, above we saw that $[4] \in \mathbb{Z}_6$ has order 3, and of course $\mathbb{Z}_6$ itself has order 6. You can check for yourself that 3 divides 6, so that $|[4]| \mid |\mathbb{Z}_6|$.

(We already had a theorem by this name, but that doesn't usually stop whoever names theorems from giving them names. Lagrange was one of the most important mathematicians of the eighteenth century, among other posts as Euler's successor in Berlin at the court of Frederick the Great.)

8.3.3.6 Cyclic groups

There is another, simpler concept to keep in mind.

- If G has order $|G| = n$ and there is some element $x \in G$ such that x has order $|x| = n$ as well, then it must go through all the possible other elements of G before hitting $x^n = e$.
- This element, whose powers run through all n elements of G, is called a **generator** of the group.
- Any group that has a generator (again, an element whose powers hit *all* elements of the group) is called a **cyclic group**.

It is pretty clear, I hope, that $\mathbb{Z}_n$ is a cyclic group with generator $[1]$, for any n. But not every group is cyclic! See Exercises 8.4.8 and 8.4.9.

There can be more than one generator; going back to $\mathbb{Z}_6$, note that

$$[1] + [1] + [1] + [1] + [1] + [1] \equiv [0] \text{ and } [5] + [5] + [5] + [5] + [5] + [5] \equiv [0] .$$

Other elements are in between (e.g. $[2] \equiv [1] + [1] \equiv [5] + [5] + [5] + [5]$).

8.3.3.7 Abelian groups

This won't come up too much, but it is important to note that most of the groups we will encounter in this course have one additional special property.

Namely, it doesn't matter what order you do the operation in. (Such an operation is called **commutative**.)

- For instance, clearly (in *any* $\mathbb{Z}_n$) it is true that $[1] + [2] = [2] + [1]$, or really for any elements at all.
- Not all groups have this property; you may recall that multiplying matrices in two different orders may yield two different answers.

Any group which has this property, that $a * b = b * a$ for all $a, b \in G$, is called an **Abelian group**. Just keep it in mind!

8.4 Exercises

1. Write out the addition table for $\mathbb{Z}_{11}$ completely, by hand.

2. Write out the multiplication table for $\mathbb{Z}_{11}$ completely, by hand.

3. Find some conjecture/pattern to state about multiplication tables, based on any of the interacts in this chapter.

4. Find some conjecture/pattern to state about values of $a^n \pmod p$, for p prime and $0 \leq n < p$ you discovered using the interact in Subsection 8.2.1. This could be *anything* profounder than

$$a^0 \equiv 1 \pmod p \text{ or } 1^n \equiv 1 \pmod p$$

for all prime p and for all n, but should at least be some pattern you tested for a number of values.

5. Give an example of a non-closed binary operation.

6. In Example 8.3.2, what is the order of the group element which is rotation by ninety degrees to the left? What is the order of rotation by 180 degrees?

7. Consider a similar setup to Example 8.3.2, but with a regular hexagon. If R is rotation of the hexagon by sixty degrees to the right, verbally describe R^{-1}. How would you describe R^3 verbally? What is the order of R?

8. Give an informal argument that $\mathbb{Q}$ is not cyclic.

9. Give an example of a cyclic group which is not finite.

10. (Only if you have some experience with matrices.) Find two 2×2 matrices A and B which have non-zero determinant such that $A \cdot B \neq B \cdot A$. Conclude that the group of 2×2 matrices with non-zero determinant is *not* Abelian. (It *is* a group, because all such matrices have an inverse matrix.)

Chapter 9

The Group of Units and Euler's Function

9.1 Groups and Number Systems

There is a lot that the integers modulo n can teach us. We can approach new horizons by rethinking the problems we have just studied.

9.1.1 Solving linear equations – again

What is a group, again? As we saw in Section 8.3, a group is *any* 'number system' where we can solve linear equations.

Example 9.1.1. Here are some familiar group examples.

- The integers modulo n, $\mathbb{Z}_n$, is a group under addition. As an example, $3 + x \equiv 2 \pmod 4$ has a solution.
  - Namely, we use the (group) inverse, $-3 \equiv 1$, to solve it, so that
  $$x \equiv 2 + (-3) \equiv 2 + 1 \equiv 3 \pmod 4$$
  is the solution.
- Similarly, we can solve equations like $\frac{2}{3} \cdot x = 5$ over the rational numbers. Why? Because $\frac{2}{3}$ has a (group) inverse in the group $\mathbb{Q} \setminus \{0\}$ (under multiplication), namely $\left(\frac{2}{3}\right)^{-1} = \frac{3}{2}$, and
  $$x = 5 \cdot \frac{3}{2}$$
  does indeed solve this equation.

Let us use this idea to help us with solving *congruences* modulo n. Using the above framework, I should be able to solve

$$43x \equiv 2 \pmod{997}$$

by using something like $a = 43^{-1}$, the notation we saw before.

That would get us

$$x \equiv 2a \equiv 2 \cdot 43^{-1} \pmod{997} .$$

Let's try this in Sage.

```
a=mod(43,997)
x=2*a^-1
a^-1
pretty_print(html("$a$ is $%s$"%a))
pretty_print(html("$a^{-1}$ is $%s$"%a^-1))
pretty_print(html("$2a^{-1}$ is $%s$"%x))
```

This checks out, of course:

```
mod(43*742,997)
```

We can similarly try to solve with a composite modulus:

$$53y \equiv 29 \pmod{100}$$

using $b = 53^{-1}$, so that

$$y \equiv 29 \cdot b \equiv 29 \cdot 53^{-1} \pmod{100}.$$

```
y=29*mod(53,100)^-1
pretty_print(html("$y$ is $%s$"%y))
```

```
y=29*mod(53,100)^-1
53*y
```

9.1.2 A new group

9.1.2.1 The group of units

So solving this should often be possible. But it can't *always* work, otherwise I could use it to solve something like

$$52y \equiv 29 \pmod{100}$$

and we already know this does not have a solution. We can't just use this idea willy-nilly, indeed, there isn't a 52^{-1} in this case.

Hence we introduce a new group – and it's even a simple set to define.

Definition 9.1.2. We let U_n, the **group of units modulo** n, be the set of equivalence classes $[a]$ modulo n such that $\gcd(a, n) = 1$.

This will be the set where we *are* allowed to do inverses, and hence to solve things easily. Before going on, figure out for yourself the elements of U_5 and U_8.

Now, naming something doesn't guarantee it's useful, or that it performs as claimed! So we need to check some things from Definition 8.3.3.

Proposition 9.1.3. *The group of units is really a group.*

Proof. First, this is certainly a set. Since we earlier proved that any two elements of a residue class have the same gcd with a modulus, the definition makes sense, and we know how to check if something is in it.

Next, the set is associative with respect to multiplication, because it's really the same as multiplication over $\mathbb{Z}$. The identity element [1] is likewise inherited from $\mathbb{Z}$.

Finally, we do need to check whether the multiplication is *closed* on this set. After all, it's not obvious that if $ax \equiv 1$ and $bx \equiv 1$ have solutions, then so does $(ab)x \equiv 1$! But if $\gcd(a, n)$ and $\gcd(b, n)$ are both 1, then ab will also be coprime to n, which is all that is needed. All in all, that means U_n really and truly is a group. □

We can even give a formula of sorts for the inverse (this should work in any group).

Fact 9.1.4. *The inverse of ab is $b^{-1}a^{-1}$.*

Proof. First, b^{-1} and a^{-1} exist, so $(b^{-1})(a^{-1})$ exists. Next, if $ab \cdot x \equiv 1$, then

$$(b^{-1}a^{-1})(ab)x \equiv (b^{-1}a^{-1}) \cdot 1$$

so

$$(b^{-1} \cdot 1 \cdot b)x \equiv b^{-1}a^{-1}, \text{ which gives } x \equiv b^{-1}a^{-1}$$

□

9.1.2.2 More facts and examples

The terminology **units** makes sense too. If you are in a number system with addition *and* multiplication, then a unit is an element that has a multiplicative inverse.

Example 9.1.5. Here are some examples of units.

- In the integers, ± 1 are the units.
- More unusual is the set of complex numbers (!), which all are units except 0. (In fact, the inverse of $r\left(\cos(\theta) + i\sin(\theta)\right)$ is $\frac{1}{r}\left(\cos(-\theta) + i\sin(-\theta)\right)$.).
- And U_n is the set of all the integers modulo n that have multiplicative inverses. By our previous investigations, we know this is when $ax \equiv 1 \pmod{n}$ has a solution. Since multiplication *is* the operation, there are inverses!

Naturally, it can take a while to list all these guys, but it's worth doing. Try it for $n = 10$, $n = 11$, and $n = 12$ by hand.

Sage has commands to list the group of units and give the order of the group.

```
@interact
def _(n=22):
    pretty_print(html("The␣units␣of␣$\\mathbb{Z}_{%s}$␣
        are"%n))
    pretty_print(html(
        Integers(n).list_of_elements_of_multiplicative_group())
        )
    pretty_print(html("There␣are␣$%s$␣of␣
        them."%Integers(n).unit_group_order()))
```

Sage note 9.1.6 (Reminder to try things out)**.** Remember, you can use these yourself by using these commands, or by cutting and pasting them in a Sage notebook, SageMath Cloud, or command line interface. They are tedious to type, though!

```
Integers(50).list_of_elements_of_multiplicative_group()
```

```
Integers(50).unit_group_order()
```

9.2 The Euler Phi Function

We give the size of the group of units (mod n) a special name.

Definition 9.2.1. We give the order of U_n the name $\phi(n)$. That is, by definition,

$$\phi(n) = |U_n| \ .$$

This is the so-called Euler ϕ function. It can also be written phi, it is pronounced 'fee', and it's occasionally notated φ just for fun.

One of the most fun things to do with basic number theory is to explore new concepts with pencil and paper – because it really is tractable.

Question 9.2.2. Do you see any patterns on the value of $\phi(n)$?

9.2.1 Euler's theorem

So far this is a relatively abstract concept. What follows is not abstract at all, but very, very useful! Let's follow the following argument to see what we can find out about $\phi(n)$.

Recall the notion of the order of an element (Definition 8.3.9). So any random element $[a] \in U_n$ (for some n) has an order. For instance, the order of $[2]$ in U_7 is 3, because $[2]^1$ and $[2]^2$ are not 1, but $[2]^3 \equiv 8 \equiv 1 \pmod 7$.

This means we can apply the things we learned about orders, in particular Theorem 8.3.11 of Lagrange. It stated that the order of any element of a finite group divides the order of the group itself.

Think about what this implies for orders in $|U_n|$. First, $|a|$ divides $|U_n|$. (For instance, in the example above, 3 divides 6.) That can be rewritten as

$$|a| \ \mid \ \phi(n), \text{ or } \phi(n) = k|a|$$

for some positive integer k.

Finally, let's apply this fact to powers of a.

$$a^{\phi(n)} = a^{k|a|} = (a^{|a|})^k \equiv 1^k \equiv 1 \pmod n$$

This is very interesting; without it, all we would know is that $a^{|a|} \equiv 1$ because that's the definition of what 'order' means. With it, we have proved one of the many celebrated theorems of Leonhard Euler:

Theorem 9.2.3 (Euler's Theorem)**.** *If* $\gcd(a, n) = 1$*, then* $a^{\phi(n)} \equiv 1$ *(mod* n*).*

Proof. See the preceding paragraphs. □

Try verifying Euler's Theorem for $n = 12$ and $n = 11$ for some simple a such as $a = 3$ or $a = 5$. Can you see how to recover Fermat's Little Theorem from Euler's Theorem, as a special case? (See Exercise 9.6.1.)

9.3 Using Euler's Theorem

Euler's Theorem has many uses. We will begin with its use in computation.

9.3.1 Inverses

Use it to compute inverses mod (n), with just a little cleverness. If

$$a^{\phi(n)} \equiv 1 \pmod{n} ,$$

then certainly multiplying both sides by a^{-1} yields

$$a^{\phi(n)-1} \equiv a^{-1} \pmod{n} .$$

We can check this using Sage.

```
@interact
def _(a=3,n=10):
    a=mod(a,n)
    try:
        b = a^-1
        pretty_print(html("$%s^{-1}$ is $%s$ and
            $%s^{\phi(%s)-1}=%s^{%s-1}$ is also $%s$"%(a,
            b, a, n, a, euler_phi(n), a^(euler_phi(n)-1))))
    except:
        pretty_print(html("Don't forget to pick an $a$
            that actually has an inverse modulo $n$!"))
```

Example 9.3.1. Let's pick a congruence we wanted to solve earlier, like

$$53y \equiv 29 \pmod{100}$$

and try to solve it this way. Instead of all the stuff we did before, we could just multiply both sides by the inverse of 53 in this form.

$$53y \equiv 29 \pmod{100}$$

$$53^{\phi(100)-1} \cdot 53y \equiv 53^{\phi(100)-1} \cdot 29 \pmod{100}$$

Now using Theorem 9.2.3, we get

$$1 \cdot y \equiv 29 \cdot 53^{\phi(100)-1} \pmod{100} .$$

One could conceivably do this power by hand using our tricks for powers; using a computer, it would look like the following in Sage.

```
mod(29*53^(euler_phi(100)-1),100)
```

This answer jells with our previous calculation. Better, I didn't have to solve a *different* linear congruence in order to solve my original one; I just had to have a way to do multiplication mod (n).

Sage note 9.3.2 (Euler phi in Sage)**.** Notice that Sage has a command to get the Euler phi function, namely `euler_phi(n)`. This doesn't have the direct connection to the group, but is easier to use than `Integers(n).unit_group_order()`.

9.3.2 Using Euler's theorem with the CRT

We can use this to do Chinese Remainder Theorem systems much more easily, as long as we have access to ϕ.

Remember the algorithm for the CRT, where we tried to solve systems like this:

- $x \equiv a_1 \pmod{n_1}$
- $x \equiv a_2 \pmod{n_2}$
- $\cdots$

There, we had to calculate many solutions to congruences of the form

$$\frac{N}{n_i} x \equiv 1 \pmod{n_i}\,.$$

(This was to get the d_i numbers.) Our new information means that this inverse is just

$$\left(\frac{N}{n_i}\right)^{-1} \equiv \left(\frac{N}{n_i}\right)^{\phi(n_i)-1}\,,$$

since we are looking at a congruence modulo n_i.

So the things in the final solution which looked like

$$a_i \cdot \frac{N}{n_i} \cdot \left(\frac{N}{n_i}\right)^{-1}$$

can be thought of as

$$a_i \cdot \frac{N}{n_i} \cdot \left(\frac{N}{n_i}\right)^{\phi(n_i)-1} = a_i \left(\frac{N}{n_i}\right)^{\phi(n_i)}\,,$$

which is much cooler and simpler! So the answer to the general system is just

$$x \equiv \sum_{i=1}^{k} a_i \left(\frac{N}{n_i}\right)^{\phi(n_i)} \pmod{N}\,.$$

```
a_1,a_2,a_3 = 1,2,3
n_1,n_2,n_3 = 5,6,7
N=n_1*n_2*n_3;N
```

```
mod(a_1*(N/n_1)^(euler_phi(n_1)) +
    a_2*(N/n_2)^(euler_phi(n_2)) +
    a_3*(N/n_3)^(euler_phi(n_3)),N)
```

Sage note 9.3.3 (More complex list comprehension)**.** It's possible to do the previous work more concisely, no matter how many congruences you have, if you know a little Python and a little something called a 'list comprehension' (recall Sage note 4.6.2.

```
sum([mod(a*(N/n)^(euler_phi(n)),N) for (a,n) in
    [(a_1,n_1),(a_2,n_2),(a_3,n_3)]])
```

But that's not necessary for our purposes.

Example 9.3.4. We can do this one step even better. Take a huge system like

- $3x \equiv 7 \pmod{10}$
- $2x \equiv 5 \pmod{9}$
- $4x \equiv 1 \pmod{7}$

Can we find solutions for this using the same mechanism? Yes, and without too much difficulty now.

Since one can solve $bx \equiv c \pmod{n}$ with

$$x \equiv b^{\phi(n)-1} \cdot c\,,$$

any likely system of congruences with coprime moduli

$$b_i x \equiv c_i \pmod{n_i}$$

where N is the product of the moduli could be solved by

$$x \equiv \sum_{i=1}^{k} \left(b_i^{\phi(n_i)-1} c_i \right) \left(\frac{N}{n_i} \right)^{\phi(n_i)} \pmod{N}\,.$$

Let's use this to solve this system.

```
c_1,c_2,c_3 = 7,5,1
m_1,m_2,m_3 = 10,9,7
M=m_1*m_2*m_3
b_1,b_2,b_3 = mod(3,M),mod(2,M),mod(4,M)
d_1,d_2,d_3 = mod(M/m_1,M),mod(M/m_2,M),mod(M/m_3,M)
b_1^(euler_phi(m_1)-1)*c_1*d_1^(euler_phi(m_1)) +
    b_2^(euler_phi(m_2)-1)*c_2*d_2^(euler_phi(m_2)) +
    b_3^(euler_phi(m_3)-1)*c_3*d_3^(euler_phi(m_3))
```

Notice that we make as much stuff modulo M to begin with as possible. Even for bigger numbers, asking Sage to first make things modular is a big help – it takes essentially no time!

Example 9.3.5. We can demonstrate this with a much larger example, picking essentially random large primes to compute with. (It's worth trying to time this – recall that we can use `%time` for this in notebooks, see Sage note 4.2.1.)

```
c_1,c_2,c_3 = 7,5,1
m_1,m_2,m_3 = random_prime(10000), random_prime(20000),
    random_prime(30000)
M=m_1*m_2*m_3
b_1,b_2,b_3 = mod(3,M),mod(2,M),mod(4,M)
d_1,d_2,d_3 = mod(M/m_1,M),mod(M/m_2,M),mod(M/m_3,M)
pretty_print(html("Our␣primes␣are␣$%s$,␣$%s$,␣and␣
    $%s$"%(m_1,m_2,m_3)))
b_1^(euler_phi(m_1)-1)*c_1*d_1^(euler_phi(m_1)) +
    b_2^(euler_phi(m_2)-1)*c_2*d_2^(euler_phi(m_2)) +
    b_3^(euler_phi(m_3)-1)*c_3*d_3^(euler_phi(m_3))
```

```
c_1,c_2,c_3 = 7,5,1
m_1,m_2,m_3 = random_prime(10^8), random_prime(2*10^8),
    random_prime(3*10^8)
M=m_1*m_2*m_3
b_1,b_2,b_3 = mod(3,M),mod(2,M),mod(4,M)
d_1,d_2,d_3 = mod(M/m_1,M),mod(M/m_2,M),mod(M/m_3,M)
pretty_print(html("Our primes are $%s$, $%s$, and
    $%s$"%(m_1,m_2,m_3)))
b_1^(euler_phi(m_1)-1)*c_1*d_1^(euler_phi(m_1)) +
    b_2^(euler_phi(m_2)-1)*c_2*d_2^(euler_phi(m_2)) +
    b_3^(euler_phi(m_3)-1)*c_3*d_3^(euler_phi(m_3))
```

9.4 Exploring Euler's Function

One of the neatest things about $\phi(n)$, beyond it being quite useful for things we are familiar with (congruences), is that it is a prototype for the many functions there are in number theory. So we will look at it in a bit more depth.

Let's get some more conjectures about values of $\phi(n)$. Finding patterns is fun!

One pattern we saw is Theorem 9.2.3, that if $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

But there are some other places one might look for patterns, now that one has done some number theory. These are questions the Fundamental Theorem of Arithmetic just begs us to ask, regarding a possible formula.

Question 9.4.1.

- Given a prime p, is there a formula for $\phi(p^e)$?
- If m and n are coprime, is there a relation between $\phi(mn)$ and $\phi(m)$ and $\phi(n)$?

What happens in the latter case for $n = 15$ and $m = 16$? Can you do it *by hand*?

There are a lot of other interesting questions one can ask about this function which aren't directly related to a formula.

Question 9.4.2. For instance, one can ask:

- When does $\phi(n) \mid n$?
- When (if ever) does $\phi(m) \mid \phi(n)$? (See Exercise 9.6.15.)
- Given m, for how many integers n it is true that $\phi(n) = m$?
- Are there infinitely many n for which $\phi(n)$ ends in zero? (See Exercise 9.6.14.)

One can also ask questions about new, related functions. For instance, let $f(n) = \phi(n)/n$. Can you find a formula? Where is this function equal to certain values, such as $f(n) = 1/2$? (See Exercise Group 9.6.11–9.6.13.)

Quite surprisingly, there is an *additive* result as well – try adding up

$$\sum_{d|n} \phi(d)$$

for small values of n to seek a pattern!

```
@interact
def _(n=range_slider(2,150,1,(2,20))):
    top = n[1]
    bottom = n[0]
    cols = ((top-bottom)//10)+1
    T = [cols*['$n$','$\phi(n)$']]
    list = [[i,euler_phi(i)] for i in range(n[0],n[1])]
    list.extend((10-(len(list)%10))*['',''])
    for k in range(10):
        t = [item for j in range(cols) for item in
            list[k+10*j] ]
        T.append(t)
    pretty_print(html(table(T,header_row = True, frame =
        True)))
```

Remark 9.4.3. Before moving on to some proofs in the next section, we *highly* encourage all readers to explore a lot of this – perhaps using the interact above. It's simply not the same to just prove, and even less so to read a someone else's proof. To really understand these (or other) things in mathematics, one must get a feel for them.

9.5 Proofs and Reasons

In this text, we try to strike a balance between exploration and proof. The point is that number theory is *both* of these things. Exploration is wonderful, but we will see a number of times where we really do need the proof to avoid error. Nonetheless, do not start this section before really trying things!

In a good proof, the *techniques* will not just prove that things are true, but lend *insight* into why they are true. The proofs here have this trait.

9.5.1 Computing prime powers

With some effort above, you should have seen a pattern for $\phi(p^e)$. Let's prove this.

Fact 9.5.1.

$$\phi(p^e) = p^e - p^{e-1} = \left[1 - \frac{1}{p}\right] p^e$$

Proof. What we want is the number of positive numbers (!) coprime to p^e and less than p^e.

The most important point is that any number which is not coprime to p^e must share a prime factor with it, which must be p. Likewise, any number divisible by p is not coprime to p^e, so this is a necessary and sufficient condition.

Now we just need to count these numbers. But all the numbers less than or equal to p^e which have a factor of p are just the multiples of p, which occur every pth element. Since p^e itself is the p^{e-1}th such multiple, there are exactly p^{e-1} such integers *not* coprime to p^e.

Subtract; there are

$$p^e - p^{e-1}$$

element which *are* coprime. □

9.5.2 Multiplicativity

The most interesting proof is that of this fact about ϕ applied to certain products. Later (Definition 18.1.3) we will see this has proved that ϕ is **multiplicative**.

Fact 9.5.2.

$$\phi(mn) = \phi(m) \cdot \phi(n) \text{ IF } \gcd(m, n) = 1$$

Proof. Take the integers from 1 to mn and arrange them in an array like so – n rows, m columns:

$$\begin{array}{cccc} 1 & 2 & 3 & \dots & m \\ m+1 & m+2 & m+3 & \dots & 2m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (n-1)m+1 & (n-1)m+2 & (n-1)m+3 & \dots & nm \end{array}$$

Now notice that only some of the columns correspond to elements of U_m. Namely, the columns with $km+\ell$ where $\gcd(\ell, m) = 1$ correspond. The others cannot have elements coprime to m. Thus there are $\phi(m)$ columns like this where all elements are coprime to m; we focus on these.

Now we look at the other direction, rows. Within *each* such column, there are all possible classes in $\mathbb{Z}_n$. Why?

- Suppose that two elements of the ℓ column are the same equivalence class.
- Then $km + \ell \equiv k'm + \ell \pmod{n}$.
- And then $km \equiv k'm$ and we can cancel m, since we already know it is coprime to n.
- That leads to $k \equiv k'$, so they are in the same row as well as the same column (hence, the same element).

That means that each relevant column has exactly $\phi(n)$ elements in it which are coprime to n, so that we get $\phi(m)\phi(n)$ in total! □

Example 9.5.3. It can be easier to see with an example, say $n = 15$.

```
@interact
def _(m=(5,[2..10]),n=(3,[2..10])):
    T = [['$[%s]$'%i for i in [1..m]]]
    for k in range(n):
        t = []
        for i in [1+k*m..m+k*m]:
            if gcd(i,m*n)==1:
                t.append('$%s$ !'%i)
            else:
                t.append('$%s$'%i)
        T.append(t)
    pretty_print(html(table(T, header_row=True,
        frame=True)))
```

In the interact, the actual units modulo mn are marked with exclamation points. If you pick an m and n which aren't coprime, you'll see how the exclamation points don't come in the right amounts.

Again, since there are $\phi(m)$ columns with $\phi(n)$ elements in them, all coprime to both m and n, that means there are $\phi(m)\phi(n)$ elements coprime to mn, which proves what we wanted.

9.5.3 Addition Formula

If you were diligent in your exploration, you will have discovered that

$$\sum_{d|n} \phi(d) = n \, .$$

We will prove this carefully, using subsets. We will gain insight of a combinatorial nature – that there are two ways to count n, one of which is precisely about finding numbers coprime to divisors of n.

To really understand this proof, it is best to follow along with $n = 15$.

Fact 9.5.4.

$$\sum_{d|n} \phi(d) = n$$

Proof. In order to show this, we will take $\{1, 2, 3, \ldots, n\}$ and divide it up into subsets with different gcd with n. This will total up to n different possibilities, and there will be $\phi(d)$ different possibilities for each possibly gcd.

Indeed, the only possibilities for greatest common divisor with n are the various divisors d of n, which we will call $\{d \mid n\}$, so each subset corresponds to one of these. So our sets look like

$$\{a \mid \gcd(a, n) = 1 = d_0\}, \ \{a \mid \gcd(a, n) = d_1\}, \ldots, \{a \mid \gcd(a, n) = n = d_k\} \, .$$

Let's look at these sets more carefully.

- Each one consists of numbers sharing divisor d_i with n. So, if we wanted to, we could divide *all* the numbers in the ith set by d_i.
- That new set will be the set of numbers $1 \leq b \leq \frac{n}{d_i}$ coprime to $\frac{n}{d_i}$.
- So the size of the set of numbers with gcd d_i with n is the same as the number of things coprime to $\frac{n}{d_i}$.

More precisely, if we look at *all* the sets in question, they are the same in size as these sets:

$$\{b \mid \gcd(b, n/1) = 1\}, \ \{b \mid \gcd(b, n/d_1) = 1\}, \ldots, \{b \mid \gcd(b, 1) = 1\} \, .$$

The old sets were all disjoint, so even though these new sets themselves are different, their *sizes* (or cardinalities) are the same as before. So

$$n = \phi(n) + \phi(n/d_1) + \phi(n/d_2) + \cdots + \phi(1) \, .$$

But the set of numbers $\frac{n}{d_i}$ for *all* divisors d_i of n is *also* the set of all divisors of n! So we can rewrite the sum as desired,

$$n = \sum_{d|n} \phi(d)$$

□

Some readers will want to know this will be revisited in a far more sophisticated way in Example 23.2.3.

9.5.4 Even more questions

There are lots of other interesting questions to tackle. Go back to the beginning of Section 9.4 and look at some of the questions you didn't yet explore. You now have the tools you need to tackle such questions, and even to prove things about them. The structure of ϕ is very regular!

9.6 Exercises

1. Prove Theorem 7.5.2 as a corollary of Theorem 9.2.3.

2. Prove that if p is prime, then $a^p \equiv a \pmod{p}$ for every integer a.

3. Formally prove that $\phi(p) = p - 1$ for prime p, by deciding which $[a] \in \{[0], [1], [2], \ldots, [p-2], [p-1]\}$ have $\gcd(a, p) = 1$.

4. Verify Euler's Theorem by hand for $n = 15$ (note that $\phi(15) = 8$, and remember that $a^8 = ((a^2)^2)^2$ so we can use modulo reduction at each squaring).

5. Get the inverse of 29 modulo 31, 33, and 34 using Euler's Theorem.

6. Evaluate without a calculator $11^{49} \pmod{15}$ and $139^{112} \pmod{27}$.

7. Solve the congruence $33x \equiv 29 \pmod{127}$ and $\pmod{128}$.

8. Solve as many of the systems of congruences we already did Exercises 5.6 using the Chinese Remainder Theorem and Euler's Theorem as you need in order to understand how it works. Follow the models closely if necessary.

9. Use the facts from Section 9.5 to create a general formula for $\phi(N)$ where $N = \prod_{i=1}^{k} p_i^{e_i}$. Then prove it by induction.

10. Compute the ϕ function evaluated at 1492, 1776, and 2001.

Let $f(n) = \phi(n)/n$.

11. Show that $f(p^k) = f(p)$ if p is prime.

12. Find the smallest n such that $f(n) < 1/5$.

13. Find all n such that $f(n) = 1/2$.

14. Prove whether there are infinitely many values of ϕ that end in zero.

15. *Conjecture* whether there are any relations between m and n that might lead $\phi(m)$ to divide $\phi(n)$.

16. Look up the *Carmichael conjecture* about ϕ. What does it say, and what is the current status of this conjecture?

17. Use the ideas that proved ϕ was multiplicative (Subsection 9.5.2) to see whether you can finally solve A First Problem. Especially think of making a table.

Chapter 10

Primitive Roots

There is deeper structure in the group of units than one might at first suspect. This chapter explores that structure.

To start off, remember our search for patterns in the powers of $a \pmod n$? That is, we looked for patterns in $a^b \bmod(n)$. One of the things we discovered was Fermat's Little Theorem, which was that the first and last columns of the following graphic were the same color (representing one).

There is lots left to discover, though. Can you find more?

```
@interact
def power_table_plot(p=(7,range(2,50))):
    P=matrix_plot(matrix(p-1,[mod(a,p)^b for a in
        range(1,p) for b in
        srange(euler_phi(p)+1)]),cmap='jet')
    show(P,figsize=6)
```

Sage note 10.0.1 (Reminder for colormaps)**.** Remember, to get a gray-scale plot, just remove the part with `cmap='jet'` etc.

Have you made the observation that sometimes we get *all* colors in a single row? This means that (at least sometimes) $a^b \pmod n$ goes through *every single number* when we do enough powers a^b.

It turns out that this concept has a name, and is the last of the big concepts of basic congruence number theory.

10.1 Primitive Roots

10.1.1 Definition

Definition 10.1.1. We say that $a \in U_n$ is a **primitive root** of n when a^b runs through all elements of U_n for $1 \leq b \leq \phi(n)$.

Or, you can say it hits all the possible colors in the interact! For *composite* n, this won't mean all colors per se, just all colors that represent units. So for such moduli, we shrink the number of rows down for this final interact. This has rows that are the elements of U_n, but certainly not labeled correctly.

```
@interact
def _(modulus=(7,range(2,50))):
```

```
show(matrix_plot(matrix(euler_phi(modulus),
    [mod(a,modulus)^b for a in range(1,modulus) for b
    in srange(euler_phi(modulus)+1) if
    gcd(a,modulus)==1]),cmap='jet',figsize=6))
```

Sage note 10.1.2 (Filtering list comprehensions)**.** We are only looking at units here. The syntax `[x for y in range(1,mod)if func(x)]` takes list comprehensions to another level, by 'filtering'. This allows us to remove from the list anything which doesn't fit what we want. In this case, we removed non-units; `gcd(a,mod)==1` was required.

10.1.2 Two characterizations

Proposition 10.1.3. *There are two equivalent ways to characterize/define a primitive root of n among numbers such that* $\gcd(a, n) = 1$.

- *We say that a is a* ***primitive root*** *of n if a^b yields every element of U_n.*
- *We say that a is a* ***primitive root*** *of n if the* order *of a is $\phi(n)$.*

Proof. Why are these true? Recalling the terminology from Section 8.3, the first one means that U_n is a **cyclic group** (one all of whose elements are powers of a specific element), and that a is a **generator** of that group. This is the more advanced point of view.

The second point of view also uses the group idea of the order of an element. Remember, this is the smallest number of times you can multiply something by itself and get 1 as a result. What would this idea mean without using the terminology of groups? With that viewpoint, k is the order of a if $a^k \equiv 1 \pmod{n}$ and $a^b \not\equiv 1$ for $1 \leq b < k$. □

10.1.3 Finding primitive roots

As a first exercise, the gentle reader should figure out the orders of some elements of some small groups of units. For $n \in \{5, 7, 8, 9, 10, 12, 14, 15\}$, try exploring U_n. There should be at least some primitive roots.

- Were *all* elements primitive roots?
- Did all of these groups have them?
- Is it particularly fun to look?

It's useful to try looking for primitive roots by hand. However, it's better to know whether one should *bother* to look, and hence to try to prove things about orders in general.

10.2 A Better Way to Primitive Roots

10.2.1 A useful lemma

In order to find primitive roots, we might want a better approach than simply trying every single power of a for every a until we find one.

Example 10.2.1 (A motivating example)**.** Let's walk through an example to motivate a new approach, using a small modulus.

Take some number n with a $\phi(n)$ with a few factors. Say, $n = 11$ and $\phi(11) = 10$. Okay, we know that *every* element $a \in U_{11}$ will have

$$a^{10} \equiv 1 \pmod{11} ,$$

but which elements don't reach the unit *before* the tenth power?

Well, we know that the order of an element has to divide $\phi(11) = 10$, so we could try a^2 and a^5; no other a^k could yield 1. In fact, if those aren't $\equiv 1$, there aren't any other possible orders out there, so that a would work as a primitive root.

- Let's try this with $a = 2$.

  $$2^2 \equiv 4 \not\equiv 1 \pmod{11} \text{ and } 2^5 = 32 \equiv -1 \not\equiv 1 \pmod{11} ,$$

  so 2 must be a primitive root.

- What about with $a = 3$?

  $$3^5 = 9 \cdot 9 \cdot 3 \equiv (-2)^2 \cdot 3 \equiv 12 \equiv 1 \pmod{11}$$

  so 3 is not a primitive root modulo eleven.

The moral is that we didn't have to check *all* ten possible powers to decide whether a was a primitive root modulo eleven.

Now we formalize this, and in fact rephrase it in a slightly more efficient way.

Sage note 10.2.2 (How Sage does primitive roots)**.** As far as I understand the code, this is how even Sage tests for finding primitive roots.

Lemma 10.2.3 (Testing for Primitive Roots)**.** *An element $a \in U_n$ is a primitive root if and only if*

$$a^{\phi(n)/q} \not\equiv 1 \text{ in } U_n \text{ for each prime } q \mid \phi(n) .$$

Proof. If a is in fact a primitive root, then $\phi(n)$ is the *smallest* number k such that $a^k \equiv 1$, so certainly for numbers smaller than $\phi(n)$, like $\phi(n)/q$, those powers shouldn't be $\equiv 1$.

On the other hand, if a isn't a primitive root, then its order k must be a proper divisor of $\phi(n)$.

Now look at the prime divisors q of $\phi(n)/k$.

- For such a divisor,
  $$q \mid \phi(n)/k$$
- So $qk\ell = \phi(n)$ for some ℓ.
- That means $k\ell = \phi(n)/q$.
- But then $\phi(n)/q$ is a multiple of k.

So since $a^k \equiv 1$, then certainly

$$a^{k\ell} = a^{\phi(n)/q} \equiv 1 \pmod{n}$$

as well, which completes the proof. □

This proof is a little terse, so let's unpack this test. Essentially, we change two things from the initial idea of trying *all* divisors of $\phi(n)$:

- Instead of trying powers which are divisors of $\phi(n)$, we try powers which are $\phi(n)$ divided by divisors. So 2^5 becomes $2^{10/2}$ and 3^2 becomes $3^{10/5}$. That seems like it's not doing anything other than rewriting, but at least it organizes things differently.
- Then, instead of having to try all $\phi(n)/d$, we use a trick to just need *prime* divisors d. (See the proof.)

Doing some examples slowly will help it make sense.

```
@interact
def _(n=(19,[2..100]),a=3):
    phi=euler_phi(n)
    pds=prime_divisors(phi)
    if gcd(a,n)!=1:
        pretty_print(pretty_print(html("Make␣sure␣$a$␣and␣
            $n$␣are␣relatively␣prime!")))
    else:
        a = mod(a,n)
        pretty_print(pretty_print(html("Is␣$%s$␣a␣
            primitive␣root␣of␣$%s$?"%(a,n))))
        pretty_print(pretty_print(html("The␣prime␣divisors␣
            of␣$\phi(%s)$␣are␣$%s$"%(n,',',join([str(pd)
            for pd in pds])))))
        pretty_print(pretty_print(html("The␣powers␣are␣"+'␣
            and␣'.join(['$%s^{%s/%s}\equiv␣
            %s$'%(a,phi,pd,a^(phi/pd)) for pd in pds]))))
        pretty_print(pretty_print(html("And␣the␣order␣of␣
            a=$%s$␣is␣a.multiplicative_order()=$%s$"%( a ,
            a.multiplicative_order()))))
```

10.2.2 Using the test lemma

If you try various n and various attempts at primitive roots a, you will see that this really works. Make sure you are trying a that are actually coprime to n, though! As it turns out, there aren't very many things to try, since $\phi(n)$ in general doesn't have a lot of prime divisors, even if n is a fairly large prime.

Why not try it by hand for $n = 17$? There is only one prime divisor of $\phi(17)$, which makes things easier. Fill in this table, where PR means primitive root.

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PR?	No															No

This lemma also makes easy some statements that would otherwise be quite hard. For instance, you should (Exercise 10.6.2) see how to use the test lemma to prove that if a is a primitive root of n, then so is a^{-1} (modulo n).

Here's something harder, to show the power of this approach.

Proposition 10.2.4. *If a is a primitive root of n, then so is $n - a$ if $4 \mid \phi(n)$.*

Proof. Let's think in terms of powers. If $a^{\phi(n)/q} \not\equiv 1$, then

$$(n-a)^{\phi(n)/q} \equiv (-a)^{\phi(n)/q} \equiv (-1)^{\phi(n)/q} a^{\phi(n)/q} .$$

So, as long as $\phi(n)/q$ is even for all prime divisors of $\phi(n)$, the two powers (the one of a and the one of $n-a$ are the same.

Since $\phi(n)$ is already even, the only possible odd $\phi(n)/q$ comes from $q = 2$. □

10.3 When Does a Primitive Root Exist?

Recall your experimentation in Subsection 10.1.3. You should have discovered that there is not *always* a primitive root.

Fact 10.3.1. *There is no primitive root for $n = 12$.*

Proof. □

This is also the case for $n = 8$ (Exercise 10.6.3).So, when *do* we have primitive roots?

10.3.1 Primitive roots of powers of two

We'll start this investigation by proving that most powers of 2 do *not* have primitive roots. The following should give you an error.

```
power=25
primitive_root(2^power)
```

Proposition 10.3.2. *For $k > 2$, there are no elements of U_{2^k} that have order $\phi(2^k) = 2^{k-1}$, because the highest order they can have 2^{k-2}.*

Proof. Assume $n = 2^k$ for $k > 2$. (For $n = 2$ and $n = 4$, there are primitive roots – check this if you haven't already). In Exercise 10.6.3 we show that $n = 8$ does not have a primitive root. In particular, each element of U_8 has order $2^{3-2} = 2$, so that $a^2 \equiv 1 \pmod 8$ for all $a \in U_8$. Think of this as a base case for induction on k.

Now assume by induction that for $n = 2^k$ it is true that no element has order higher than 2^{k-2}. I.e.,

$$a^{2^{k-2}} \equiv 1 \pmod{2^k}.$$

By definition of divisibility, that means for *any* odd number a, we have that

$$a^{2^{k-2}} = 1 + 2^k \cdot m$$

for some integer m.

Next, let's look at what happens to everything in modulus 2^{k+1}. We want that

$$a^{2^{(k+1)-2}} = a^{2^{k-1}} \equiv 1 \pmod{2^{k+1}}.$$

While it's easy to get 2^{k+1} from 2^k, the only way to easily get $a^{2^{k-1}}$ from $a^{2^{k-2}}$ is by squaring. (Recall Fact 4.5.2 where we found powers quickly by using $(a^{2^e})^2 = a^{2^{e+1}}$.)

So we write $a^{2^{k-1}}$ as a square, substitute the above, and look at the remainders.

$$a^{2^{k-1}} = \left(a^{2^{k-2}}\right)^2 = (1 + 2^k m)^2 = 1 + 2^{k+1}m + 2^{2k}m^2$$

$$= 1 + 2^{k+1}(m + 2^{k-1}m^2) \equiv 1 \bmod 2^{k+1}$$

By induction we are done; because the highest possible order of an element is less than ϕ, there are no primitive roots modulo 2^k for $k > 2$. □

Fact 10.3.3. *It turns out that* ± 5 *have order* 2^{k-2} *in* U_{2^k}.

Proof. We won't prove this, but it is easy if you use just a little group theory, and one can demonstrate it for a given example.

```
@interact
def _(power=5):
    a = mod(5,2^power)
    pretty_print(html("Powers␣of␣5␣modulo␣$2^{%s}$␣
        are"%power))
    print [a^i for i in [1..2^(power-1)]]
```

□

10.3.2 Two important lemmas

There follow two important lemmas[1] for working with primitive roots, whose proofs are valuable exercises.

10.3.2.1 How the lemmas work

Lemma 10.3.4. *Suppose* p *is prime and the order of* a *modulo* p *is* d*. If* b *and* d *are coprime, then* a^b *also has order* d *modulo* p*.*

Proof. See Exercise 10.6.5. □

Lemma 10.3.5. *Suppose* p *is prime and* d *divides* $p-1$ *(and hence is a possible order of an element of* U_p*). There are at most* $\phi(d)$ *incongruent integers modulo* p *which have order* d *modulo* p*.*

Proof. See Exercise 10.6.6. □

Before using them a lot, we should unpack these results a little bit. Here is a first taste.

Fact 10.3.6. *If there is one primitive root of* n*, then there are actually* $\phi(\phi(n))$ *of them.*

Proof. We will only deal with the case of $n = p$ prime.

In Lemma 10.3.4, let the order of a be $p-1$. Then a is a primitive root modulo p, *and* so is a^b for every b coprime to $p-1$. Since there are $\phi(p-1)$ of these, it satisfies the claim. □

It works; let's check this out.

```
@interact
def _(p=(41,prime_range(100))):
    a=mod(primitive_root(p),p)
    pretty_print(html("$%s$␣is␣a␣primitive␣root␣of␣$%s$,␣
        with␣order␣$%s$"%(a,p,p-1)))
    L=[(i,a^i,(a^i).multiplicative_order()) for i in
        range(2,p-1) if gcd(i,p-1)==1]
    for item in L:
        pretty_print(html("$%s^{%s}\equiv␣%s$␣also␣has␣
            order␣$%s$␣(and␣$\gcd(%s,%s)=1$)"%(a, item[0],
            item[1], item[2], item[0], p-1)))
```

[1] Or lemmata, but who's counting?

10.3.2.2 How the lemmas (don't) fail

To continue, let's pick a non-prime number we know something about to see how many numbers we have with a given order.

We saw in Proposition 10.3.2 that powers of 2 don't have cyclic groups of units, but they do have lots of elements with the next smallest possible order. So, for example, for $n = 32$ we can look at whether powers b coprime to that order of such an element are in fact also elements with the same order.

```
@interact
def _(n=5):
    pretty_print(html("Modulo␣$2^%s"%n))
    a=mod(5,2^n)
    L=[(i,a^i,(a^i).multiplicative_order()) for i in
        range(1,a.multiplicative_order()) if
        gcd(i,a.multiplicative_order())==1]
    for item in L:
        pretty_print(html("$%s^{%s}\equiv␣%s$␣has␣order␣
            $%s$␣(and␣$\gcd(%s,%s)=1$)"%(a, item[0],
            item[1], item[2], item[0],
            a.multiplicative_order())))
```

The interact confirms that it works; indeed, Lemma 10.3.4 should be true whether p is prime or not, though I won't ask you to prove it.

Lemma 10.3.5 also seems to be working; there are exactly $\phi(8) = 4$ powers here, which have order eight.

The problem in deciding if there are primitive roots, though, is that there might be some element of the same order as the ones above which is not actually one of them! This code finds them.

```
@interact
def _(n=5):
    pretty_print(html("Modulo␣$2^%s"%n))
    a=mod(-5,2^n)
    L=[(i,a^i,(a^i).multiplicative_order()) for i in
        range(1,a.multiplicative_order()) if
        gcd(i,a.multiplicative_order())==1]
    for item in L:
        pretty_print(html("$%s^{%s}\equiv␣%s$␣has␣order␣
            $%s$␣(and␣$\gcd(%s,%s)=1$)"%(a, item[0],
            item[1], item[2], item[0],
            a.multiplicative_order())))
```

In some sense there are 'extra' elements with order 8 when $n = 32$. If you have eight elements of order eight, and obviously at least one element of order 1, in U_{32}, then it is impossible to have the required eight elements of order sixteen one would need for there to be a primitive root modulo 32. (Why? Because $8 + 1 + 8 > 16 = |U_{32}|$.) In essence, the fact that this can't happen for a prime modulus is why primitive roots *do* exist in that case.

10.4 Prime Numbers Have Primitive Roots

We use many of the same techniques and ideas in by proving that every prime number p has a primitive root. Let's check that this claim is true for at least *some* primes.

```
L=[(p,primitive_root(p)) for p in prime_range(100)]
for item in L:
    pretty_print(html("A primitive root of $%s$ is
        $%s$"%(item[0],item[1])))
```

So at least we get a primitive root for the first 25 primes.

Theorem 10.4.1. *Every prime has a primitive root. In other words, the order $p-1$ group U_p is always cyclic.*

Proof. The key to the proof is to try to write $\phi(p) = p-1$ in two different ways:

1. $p-1 = \phi(p) = \sum_{d|p-1} \phi(d)$
2. $p-1 = \sum_{d|p-1} |\{a \in U_p \mid a \text{ has order } d\}|$

Note that the first fact is simply Fact 9.5.4 for $n = p-1$.

The second equation makes sense too. We already proved (in Theorem 8.3.11) Lagrange's result that the order of any element of a group divides the order of the group, so the only possible orders of elements in U_p are positive divisors of $p-1$. Since *every* element of U_p has *some* order, the $p-1$ elements thereof are divided up into (disjoint) sets of these different orders.

To finish the proof, we then just need to prove that the only possibility is the number of elements of U_p of order d is $\phi(d)$. Proving this is Claim 10.4.2. Then there will certainly be at least one element of maximal order. □

Before we finish the proof by examining the claim, we need some discussion. First, let's see what these sets look like for two examples – one where we know we have a primitive root, and one where we know we don't.

Here is the list of sets of different order elements for $n = 41$:

```
for d in divisors(40):
    L=[]
    for a in range(1,41):
        if mod(a,41).multiplicative_order()==d:
            L.append(a)
    pretty_print(html("There are $%s=\phi(%s)$ elements of
        order $%s$ - "%(len(L),d,d)+str(L)))
```

But here is the list of sets for $n = 32$; there aren't any for the highest possible order, and all the other sets have orders exact multiples of $\phi(d)$.

```
for d in divisors(euler_phi(32)):
    L=[]
    for a in range(1,32):
        if mod(a,2)==1 and
            mod(a,32).multiplicative_order()==d:
            L.append(a)
    if len(L)==euler_phi(d):
        pretty_print(html("There are $%s=\phi(%s)$
            elements of order $%s$ - "%(len(L),d,d)+str(L)))
    else:
        pretty_print(html("There are $%s\\neq\phi(%s)$
            elements of order $%s$ - "%(len(L),d,d)+str(L)))
```

For another set of ideas, recall that if g is a primitive root of p, by definition $g^{p-1} \equiv 1$ but no previous positive power is. Assuming p is an odd prime, then $p-1$ is even, and we could try to separate out the odd and even powers

$$g, g^3, g^5, \ldots \text{ and } g^2, g^4, g^6, \ldots$$

and compare them or their products.

- Can you see why the inverse of an even power of a primitive root is also an even power?
- Do you think an odd power (greater than one) of a primitive root g could be a *different* primitive root g'? Why or why not? What about even powers of a given primitive root – could they be primitive roots, at least in principle?

Now let's prove our claim.

Claim 10.4.2. *If p is prime, the number of elements of U_p of order d is $\phi(d)$.*

Proof. Assume that p is prime. For any of the divisors d of $p-1$ (not just $p-1$ itself), the size of the set

$$|\{a \in U_p \mid a \text{ has order } d\}|$$

certainly can't be *bigger* than $\phi(d)$, by Lemma 10.3.5. On the other hand, *every* element of U_p has some order! And by Lemma 10.3.4, once we find one a with order d, then all the powers of a coprime to d also have that order, so there are $\phi(d)$ of them.

So the entire proof boils down to finding at least one element a with order d for each potential order d.

Suppose that any of the sets for d (such as the set of *primitive* roots for $d = p-1$) is empty. Then the elements which 'would have' had order d in our group of units have to be 'distributed' among the other sets. That's $\phi(d)$ elements.

But we know that none of the sets corresponding to a divisor d is bigger than $\phi(d)$, and

$$\sum_{d|p-1, d<p-1} \phi(d) < p-1 \, ;$$

yet all of the $p-1$ elements in U_p must be in one of the sets. This doesn't make sense unless there is at least one element in each of the sets of elements with order d, so in particular there is at least one of each potential order, which means there are the maximal number with each order. □

The proof above makes it evident that the real place primality is used is in the crucial lemmas 10.3.5 and 10.3.4. If you are still curious to see how this works, you can explore more below.

```
@interact
def _(n=(25,[0..100])):
    for d in divisors(euler_phi(n)):
        L=[]
        for a in range(1,n):
            if gcd(a,n)==1 and
                mod(a,n).multiplicative_order()==d:
                L.append(a)
        if len(L)==euler_phi(d):
```

```
            pretty_print(html("There␣are␣$%s=\phi(%s)$␣
                elements␣of␣order␣$%s$␣-␣
                "%(len(L),d,d)+str(L)))
        else:
            pretty_print(html("There␣are␣$%s\\neq\phi(%s)$␣
                elements␣of␣order␣$%s$␣-␣
                "%(len(L),d,d)+str(L)))
```

10.5 A Practical Use of Primitive Roots

We will soon begin talking about cryptography and related matters. Before we do so, we will preview our computational needs by using primitive roots to solve some congruences in a cool way.

Suppose you want to solve a more mysterious congruence than the basic ones we have tackled thus far. Here are two examples:

- $x^4 \equiv 13 \pmod{19}$
- $7^x \equiv 6 \pmod{17}$

You can think of the first one as finding a higher root modulo n, and the second one as finding a *logarithm* modulo n.

10.5.1 Finding a higher root

Here's one way to solve the first congruence. First, find a primitive root modulo 19. Obviously we could just ask Sage, or use Lemma 10.2.3 with trial and error. In the not too distant past, the back of every number theory text had a table of primitive roots!

```
primitive_root(19)
```

Now what we will do is try to represent *both* sides of

$$x^4 \equiv 13 \pmod{19}$$

as powers of that primitive root.

The easy part is representing x^4; we just say that $x \equiv 2^i$ for some (as yet unknown) i, so

$$x^4 \equiv \left(2^i\right)^4 \equiv 2^{4i} .$$

The harder part is figuring out what power of 2 gives 13. Again, there is no shortcut, though books in the past had huge tables of them and powers (for easy reference). In practice, one would have all powers of a given primitive root available for use ahead of time.

```
a=mod(2,19)
L=[(i,a^i) for i in range(2,19)]
for item in L:
    if item[1]!=13:
        pretty_print(html("$%s^{%s}\equiv␣%s\\not\equiv␣
            13$"%(a,item[0],item[1])))
    else:
        pretty_print(html("$%s^{%s}\equiv␣%s$␣-␣
            hooray!"%(a,item[0],item[1])))
        break
```

By substituting the primitive roots in for x^4 and 13, we can say that

$$x^4 \equiv 13 \pmod{19}$$

becomes

$$2^{4i} \equiv 2^5 \pmod{19}\ .$$

This is a much more familiar type of problem. How would we have solved this in high school? You would solve it this way, with equations (not congruences):

$$2^{4i} = 2^5 \Rightarrow 4i = 5 \Rightarrow i = 5/4\ .$$

We will try to do something very similar here.

What is very important is that this congruence is, in some sense, really no longer a congruence in $\mathbb{Z}_{19}$. To be precise, everything in sight is really in U_{19}, a cyclic group of order $\phi(19) = 18$. But a cyclic group of order 18 would just the same as thinking modulo eighteen! So we can take out the exponents, just like in precalculus, but do things (mod 18):

$$4i \equiv 5 \pmod{18}\ .$$

(See Exercise 10.6.12.)

Sadly, this does not have a solution. But we figured it out without taking every fourth power out there! Indeed, doing that confirms our result:

```
[mod(i,19)^4 for i in range(2,19)]
```

Example 10.5.1. Let's try the same congruence modulo 17 instead – that is, can we solve

$$x^4 \equiv 13 \pmod{17}\ ?$$

Here, a primitive root is 3, and it turns out that $3^4 \equiv 13$, so we can try. This gives

$$3^{4i} \equiv 3^4 \pmod{17} \Rightarrow 4i \equiv 4 \pmod{16}\,,$$

which definitely does have solutions.

In fact, there are four solutions $(1, 5, 9, 13)$ to the reduced congruence

$$i \equiv 1 \pmod{4}$$

so there are four solutions $(3^1, 3^5, 3^9, 3^{13})$ to the original congruence. Let's check this:

```
a = mod(3,17)
[(a^b)^4 for b in [1,5,9,13]]
```

You can even see it at work for more complicated things.

Example 10.5.2. If we try solving $x^6 \equiv 8 \pmod{49}$, we'll need a primitive root of 49; 3 works. I can find out what power 3^i of 3 yields 8:

```
a = mod(3,17)
x=mod(primitive_root(49),49)
L=[(i,x^i) for i in range(2,euler_phi(49))]
for item in L:
    if item[1]!=8:
```

```
        pretty_print(html("$%s^{%s}\equiv␣%s\\not\equiv␣
            8$"%(a,item[0],item[1])))
    else:
        pretty_print(html("$%s^{%s}\equiv␣%s$␣-␣
            hooray!"%(a,item[0],item[1])))
        break
```

So we write $x = 3^i$ for some as yet unknown i, and get

$$3^{6i} \equiv 3^{36} \pmod{49},$$

which gives us

$$6i \equiv 36 \pmod{\phi(49) = 42}$$

and this reduces to

$$i \equiv 6 \pmod{7}.$$

So $i = 6, 13, 20, 27, 34, 41$ all work, which means that $x = 3^i \equiv 43, 10, 16, 6, 39, 33$ all should work.

```
[mod(d,49)^6 for d in [43,10,16,6,39,33]]
```

10.5.2 Discrete logarithms

Similarly, we can try to solve logarithmic examples like

$$7^x \equiv 6 \pmod{17}.$$

Indeed, solving this problem is an example of what is called a **discrete logarithm** problem. Such problems are apparently very, very hard to solve quickly, but (!) no one has every actually *proved* this.

Example 10.5.3. A primitive root modulo 17 is 3, and we can check that $7 \equiv 3^{11} \pmod{17}$ and $6 \equiv 3^{15} \pmod{17}$. Then, replacing these, we see that

$$3^{11x} \equiv 3^{15} \pmod{17}$$

yields

$$11x \equiv 15 \pmod{16};$$

since $3 \cdot 11 = 33 = 32 + 1$, we see that 3 and 11 are inverses modulo 16, so $x \equiv 3 \cdot 15 \equiv 45 \equiv 13 \pmod{16}$. And indeed, it checks out with Sage.

```
mod(7,17)^13==6
```

Sage note 10.5.4 (Reminder on equality)**.** To check whether two things are equal, remember that you can just use `==` with the two expressions and see if you get `True` or `False`.

Example 10.5.5. Let's try to solve $16^x \equiv 13 \pmod{19}$.

Recall that 2 is a primitive root of 19, and obviously $16 = 2^4$. It might look harder to represent 13; of course we could do it with the computer, but note that $13 + 19 = 32 = 2^5$. Sometimes we really can do them by hand!

Thus our congruence becomes

$$2^{4x} \equiv 2^5 \pmod{19}$$

which yields

$$4x \equiv 5 \pmod{18}.$$

We already saw in Subsection 10.5.1 that this has no solutions, so neither does the original congruence.

10.6 Exercises

1. Find primitive roots of 18, 23, and 27 using Lemma 10.2.3 to test various numbers.

2. If a is a primitive root of n, prove that a^{-1} is also a primitive root of n.

3. Show that there is no primitive root for $n = 8$.

4. Find two primitive roots of 81 using the Euler ϕ criterion Lemma 10.2.3 (that is, by hand).

5. Suppose p is prime and the order of a modulo p is d. Prove that if b and d are coprime, then a^b also has order d modulo p. (Hint: actually write down the powers of a^b, and figure out which ones could actually be 1.)

6. Suppose p is prime and d divides $p-1$ (and hence is a possible order of an element of U_p). Prove that at most $\phi(d)$ incongruent integers modulo p have order d modulo p. Hint: Lagrange's (polynomial) Theorem 7.4.1.

7. Find the orders of *all* elements of U_{13}, including of course the primitive roots, if they exist. Then verify Claim 10.4.2.

8. Challenge: assuming p is prime, prove that there are exactly $\phi(p-1)$ primitive roots of p if there is at least one. (Don't use Claim 10.4.2.)

9. Challenge: Assume that a is an odd primitive root modulo p^e, where p is an odd prime (that is, both a and p are odd). Prove that a is also a primitive root modulo $2p^e$.

10. Solve $x^6 \equiv 4 \pmod{23}$.

11. Solve $x^4 \equiv 4 \pmod{99}$ by writing this as the combination of two congruences which *can* be solved with primitive roots, and then using Subsection 5.4.1 to put them back together.

12. If $x \equiv y \pmod{\phi(n)}$, show that $a^x \equiv a^y \pmod{n}$. Hint: Theorem 9.2.3.

13. Find all solutions to the following. Making a little table of powers of a primitive root modulo 23 first would be a good idea.

- $3x^5 \equiv 1 \pmod{23}$
- $3x^{14} \equiv 2 \pmod{23}$
- $3^x \equiv 2 \pmod{23}$
- $13^x \equiv 5 \pmod{23}$

14. For which positive integers a is the congruence $ax^4 \equiv 2 \pmod{13}$ solvable?

15. Conjecture what the product of *all* primitive roots modulo p (for an odd prime p) is, modulo p. Prove it! (Hint: one of the results in Subsection 10.3.2 and thinking in terms of the computational exercises might help.)

Chapter 11

An Introduction to Cryptography

We are now ready for some applications. This chapter introduces cryptography, as well as the prototype for a cool mathematical encryption system and other similar topics. In Chapter 12, we will also discuss practical issues in implementing these – namely, finding *huge* primes and factoring *huge* composite numbers.

By 'huge' I mean something substantially bigger than this.

```
print next_prime(randrange(2^99))
print next_prime(randrange(2^300))
```

Those are peanuts by today's standards. But with the tools we've developed up to this point, we are ready for them.

11.1 What is Cryptography?

Cryptography is not just the science of making (and breaking) codes, as a dictionary might have it. It is the *mathematical* analysis of the tools of secrecy, from both the perspective of someone keeping a secret and that of the person trying to figure it out. Sometimes it is also called *cryptology*, while sometimes that term is reserved for a wider meaning.

There are two kinds of codes.

- There are codes which disguise information and are intended to remain secret! (Especially for those needing private communication.)
- There are codes encapsulating information in a convenient format, not needing secrecy. (Especially to allow for error checking.)

Mathematicians use the word **code** to indicate information is being stored, reserving the term **cipher** to talk about a way to protect that information. So, what we do when learning about this is some of each, though mostly about ciphers.

11.1.1 Encoding and decoding

There are many ways to **encode** a message. The easiest one for us (though not used in practice in exactly this way) will be to simply represent each letter

of the English alphabet by an integer from 1 to 26. It is also easy to represent both upper- and lowercase letters from 1 to 52.

We'll use the following embedded cell to turn messages into numbers and vice versa. You encode a plaintext message (no spaces, in quotes, for our examples) and decode a positive integer.

```
def encode(s): # Input must be in quotes!
    s = str(s).upper()
    return sum((ord(s[i])-64)*26^i for i in range(len(s)))

def decode(n):
    n = Integer(n)
    list = []
    while n != 0:
        if n%26==0:
            list.append(chr(64+26))
            n -= 1
        else:
            list.append(chr(n%26+64))
        n //=26
    return ''.join(list)
```

Let's try to encode the letter "q".

```
encode('q')
```

Sage note 11.1.1 (Always evaluate your definitions)**.** If the previous cell doesn't work, then you may need to evaluate the first one in this section again. If anything in this chapter ever gives a `NameError` about a global name `encode`, you probably need to reevaluate some previous cell. Most likely, the one with `def encode`!

The process of decoding (or to **decode**) is similar.

```
decode(17)
```

This should be straightforward. Too straightforward, perhaps. What are some issues here?

- First, notice that I didn't bother separating lower and uppercase letters.
- Also, no matter how complicated you get, with just a one-to-one correspondence, there are only a few possibilities for each letter. So if you know the human language in question, you can just start guessing which encrypted number stands for its most common letter.
- Can you think of other drawbacks?

That means that, in practice, we need to do a few other things. One thing that is commonly done is to make longer blocks of letters, and then turn *those* into numbers. After all, presumably there are a lot more three-letter (or longer) possible blocks of letters in English than would make it too easy to decrypt them. (Can you think of exceptions, though?)

For pairs, we will represent the first letter as a number from 1 to 26, and the second letter as 26 times the letter number (think of it as base 26). Remember that A=1, B=2, etc.

Now compare the following two encodings of "The best day of the year" and see which one might be easier to decipher.

```
[encode(letter) for letter in 'Thebestdayofthisyear']
```

```
encode('cb');decode(3+26*2)
```

```
[encode(pair) for pair in
    ['th','eb','es','td','ay','of','th','is','ye','ar']]
```

Whereas there are many 5s in the first encoding, which you could guess were Es, the second one has only one repeat (though knowing English, one might guess it was 'Th'). For this reason, it's important to point out we haven't made anything secret yet, we've just encoded.

With three letter blocks, there are then already $26^3 = 17576$ possibilities.

```
encode('zab');decode(26+1*26+2*26^2)
```

One could use this to encode the famous phrase INT HEB EGI NNI GWA STH EWO RDX. In this case, we use an extra X to fill out the space from a famous quote.

To be fair, when filler of this type is used, it would more often be used in the middle to confuse things. In addition, one might recombine the message in various ways. We will, however, usually keep our whole message together as one item, since we want to understand the mathematical aspects most, rather then real cryptography.

11.2 Encryption

We will spend most of our time talking about **enciphering**, or **encrypting**, messages. Such **encryption** is the difficult part, after all, the details of which we want to keep secret.

What is cool about modern ciphers is that we actually *expect* that any eavesdropper will know how we do the encryption; they just don't know the **key**, which is the specific numbers we use to perform our mathematical encryption.

Reversing this process (hopefully only done by the person you want to receive your message!) is called **decryption**. Sometimes you need a different set of numbers to decrypt, in which case we distinguish between the **encryption key** and the **decryption key**.

Sage note 11.2.1 (Reminder to evaluate definitions)**.** Don't forget to evaluate this so we can use words as messages instead of just numbers.

```
def encode(s): # Input must be in quotes!
    s = str(s).upper()
    return sum((ord(s[i])-64)*26^i for i in range(len(s)))

def decode(n):
    n = Integer(n)
    list = []
    while n != 0:
        if n%26==0:
```

```
            list.append(chr(64+26))
            n -= 1
        else:
            list.append(chr(n%26+64))
        n //=26
    return ''.join(list)
```

11.2.1 Simple ciphers

In the past, one would usually assume that both the sender and the receiver keep their keys secret (seems reasonable!), which is called **symmetric key cryptography**. The symmetry is that both people need to keep it secret. One early example of this supposedly goes back to C. Julius Caesar. To encrypt a message, first convert it to numbers, and then add three to each number ('wrapping around' as in modular arithmetic if needed), and convert back to letters.

```
message='MathIsCool'
secret=[encode(letter) for letter in message]
secret
```

It's pretty clear that 1=A here, for instance. Now let's add three to each. The second letter should get to 4=D, for instance.

```
code=[(x+3)%26 for x in secret]
print code
print ''.join([decode(n) for n in code])
```

What did I do here? Again, this is just modular arithmetic, modulo 26, so I added 3 mod (26).

11.2.2 Decryption and inverses

How will I decrypt it, if I get this mysterious message? Here is the main point about mathematical ciphers; they need to come from operations that have *inverses*! So in number theoretic ciphers, they'll need to come from (somehow) invertible operations.

In this case, the operation is modular addition, which certainly has inverses. If your encoded numerical message is x, your key is a, and you are working modulo (n), then your encrypted message m is

$$m \equiv x + a \bmod(n)$$

To get x back, you just use the additive inverse to a modulo n, which is $-a$.

Since -3 is the inverse of 3, this one is easy to decipher.

```
''.join([decode((x-3)%26) for x in code])
```

We could list the key here as a pair (a, n), with $a = 3$ and $n = 26$.

As noted above, one can do something similar with bigger numbers, in blocks of two. In the next Sage cell, the code requires a message with an even number of letters; can you make it more flexible?

```
message='Mathiscool'
secret=[encode(message[2*i:2*i+2]) for i in
    range(len(message)/2)]
secret
```

11.2.3 Getting more sophisticated

Let's do something a little more interesting to encrypt our 'secret' about how cool math is. What else has inverses?

Well, sometimes multiplication mod (n) does! We could make a cipher that gets m by performing

$$m \equiv ax + b \pmod{n} .$$

Here, let's choose $a = 5$ and $b = 18$; we'll use $n = 677$, the next prime after 26^2, since we have blocks of two letters each.

```
code=[(5*x+18)%(next_prime(26^2)) for x in secret]
print code
print ''.join([decode(n) for n in code])
```

Now the key is listed as a triple, $(a, b, n) = (5, 18, 677)$. How do we invert this?

To get from $ax + b$ back to x, ordinarily we would subtract b and then divide by a. Now we are working over $\mathbb{Z}_n$, so is that possible? We'll need our first 'extra' condition.

Fact 11.2.2. *To make modular encryption by a linear function workable, we need* $\gcd(a, n) = 1$. *In that case there is a number* a' *such that*

$$a(a') \equiv 1 \ (mod\ n) ,$$

so we can decode via

$$m \longmapsto a'(m - b) \equiv x \ (mod\ n) .$$

To decode this particular example, then, we need to first subtract 18, then multiply by an inverse to 5 (mod 677) (which turns out to be 271):

```
''.join([decode(271*(x-18)%677) for x in code])
```

You should get 'MathIsCool' or whatever message you originally used.

The proof of the pudding is in the eating. There's no way I get the original message back unless this works!

11.2.4 Linear algebra and encryption

There is another way of using blocks of size two or more, which we won't pursue very far, but which is a big piece of how standard encryption works (see here and here). Let's look at our message again.

```
message='Mathiscool'
secret=[encode(letter) for letter in message]
secret
```

Now, in blocks of two, I will *change my numbers* by turning the first one into the *sum* of the numbers modulo 26 and leaving the second one alone. So for the second block $(20, 8)$, I will change that block to $(28, 8)$, which modulo 26 becomes $(2, 8)$.

```
[(secret[i]+secret[i+1])%26 if i%2==0 else secret[i] for i
    in range(len(secret))]
```

This turns out to be the same thing as multiplying the corresponding list of *vectors* of length two by a matrix!

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

To invert this cipher, we would need an inverse to this matrix *modulo 26*. (People don't do something quite so naive, as there aren't too many inverses modulo 26, but for our purposes this suffices.)

In any case, this is another connection to the rest of mathematics! And it is a huge reason why linear algebra over finite algebraic structures is *very* important in security.

11.2.5 Asymmetric key cryptography

Finally there is another type of encryption, which is rather different. There exists the possibility that *everybody* knows the key to encrypt, while only the legitimate person knows how to decrypt. This is called **asymmetric key cryptography**.

This idea may seem odd. But in practice today, people really do just post their encryption keys on the Internet! In the live book, this links the public key of a fairly well-known open-source software advocate, for example.

In theory, anyone who wants to send Person XYZ a secure message could use this key, but only Person XYZ can decrypt it – convenient! Such an implementation of an asymmetric system is called **public-key** cryptography, although of course it's only the encryption key that is actually public.

In this chapter, we will see examples of both symmetric and asymmetric systems, but the main point is to lead up to the mathematics of basic public key systems.

11.3 A Modular Exponentiation Cipher

To prepare for discussion of a famous public-key system, we will first discuss a (symmetric) system that leads to it. This system needs yet another invertible number theory procedure, one that we finally should be comfortable with.

That procedure is *modular exponentiation* as cipher. Recall that we have methods to solve modular exponentials (such as primitive roots). That means we have the tools to tackle these subtle techniques.

Sage note 11.3.1 (Another reminder to evaluate definitions)**.** Don't forget to evaluate this so we can use words as messages instead of just numbers.

```
def encode(s): # Input must be in quotes!
    s = str(s).upper()
    return sum((ord(s[i])-64)*26^i for i in range(len(s)))
```

```
def decode(n):
    n = Integer(n)
    list = []
    while n != 0:
        if n%26==0:
            list.append(chr(64+26))
            n -= 1
        else:
            list.append(chr(n%26+64))
        n //=26
    return ''.join(list)
```

11.3.1 The Diffie-Hellman method

In the cell below, we will pick a few numbers relevant to this method. To use it, we will need a prime number p, and some legitimate exponent e that won't mess things up too badly. (Also, suppose our secret is still that math is cool.)

What do I mean by 'won't mess things up too badly?' Recall that when we solved

$$x^3 \equiv 5 \bmod(17) \text{ as } 3^{3i} \equiv 3^5 \bmod(17)$$

we ended up in the world of $\phi(17) = 16$ and solved

$$3i \equiv 5 \bmod(16)\,.$$

This required a solution i to exist, which wouldn't happen for all possible numbers!

In order to keep using these ideas easily, we will pick an exponent *coprime* to $\phi(p)$.

Now, here is the algorithm (see also Algorithm 11.3.3). I just take my message (as a number) and raise it to the e power modulo p. It's as simple as that!

In the cell below, we pick a convenient e and p.

```
p=29 # a prime number
e=9 # a number coprime to euler_phi(p)=p-1=28
message='MathIsCool'
secret=[encode(letter) for letter in message]
code=[mod(x,p)^e for x in secret]
print code
print ''.join([decode(n) for n in code])
```

Here I picked $p = 29$ since it's close to 26, and more or less arbitrarily picked an exponent $e = 9$.

Note the steps. I first had to encode "MathIsCool" to numbers. Then I *exponentiated* each number in the coded version, modulo 29. To be precise, I sent each number

$$a \to a^9 \pmod{29}\,.$$

Remark 11.3.2. Notice that decoding the secret message code is not so useful anymore! (What would we do with the number 28 as an output, for instance?) So we usually just stick with the numbers.

Leaving aside for the moment that the letter A will now have the unfortunate property that it always stays 1, and hence basically unencrypted (this

is because we are doing a toy example), how on earth would we ever decrypt this? Do we have a way to invert

$$a^9 \pmod{29}$$

in any way?

Naturally, we do! We will use exponentiation *again* to do so. We just need something that solves

$$\left(a^9\right)^f \equiv a \pmod{29} ,$$

or more concisely

$$a^{9f} \equiv a^1 \pmod{29} .$$

(We can think of f as a power that inverts the original power 9.).

From our earlier discussion, this is just a solution to

$$9f \equiv 1 \pmod{28}$$

and we know we can find this. In the cell below, we do it computationally, but you could do this one 'by hand'.

```
f=mod(e,p-1)^-1 # the multiplicative inverse mod p-1 (!)
    to our encryption power
print f
print ''.join([decode(x^f) for x in code])
```

This method of encryption is known as the Diffie-Hellman, or D-H, method (named after its originators, who proposed it in the mid-70's).

11.3.2 A bigger example

Now we will do a more real example of this. Notice how important it was that we chose an initial exponent e that was coprime to $\phi(p) = p - 1$.

```
message='heymathiscooleverybody'
secret=encode(message)
secret
```

For convenience, I'll just take the next prime bigger than my message.

```
p=next_prime(secret)
print p
print factor(p-1)
```

Next, I pick an exponent. Not every exponent will work! Beforehand I factored $p - 1$ so I could find something coprime to it.

```
e=10103 # a number coprime to p-1
code=mod(secret,p)^e
code
```

The encrypted message is now just one number. Now we need the decryption key. Luckily, that's just as easy as taking an inverse modulo $p - 1$:

```
f=mod(e,p-1)^-1
print f
print ''.join(decode(code^f))
```

Here is one more extended Sage example. Here, the interesting point is that I allow Sage to pick a prime *for* me using `next_prime()`.

```
message='mathisreallycoolanditshouldntbeasecret'
secret=encode(message)
p=next_prime((secret)^5)
e=677 # hopefully coprime to p-1
code=mod(secret,p)^e
f=mod(e,p-1)^-1
pretty_print(html("My␣encoded␣message␣is␣$%s$"%secret))
pretty_print(html("A␣big␣prime␣bigger␣than␣that␣is␣
    $%s$"%p))
pretty_print(html("And␣I␣chose␣exponent␣$%s$"%e))
pretty_print(html("The␣encrypted␣message␣is␣$%s$"%code))
pretty_print(html("The␣inverse␣of␣$%s$␣is␣$%s$"%(e,f)))
pretty_print(html("And␣the␣decrypted␣message␣turns␣out␣to␣
    be:"))
print ''.join(decode(code^f))
```

11.3.3 Recap

Here is the formal explanation of our first awesome encryption scheme.

Algorithm 11.3.3 (Diffie-Hellman Encryption)**.** *To encrypt using this method, do the following.*

- *Turn your message into a number x.*
- *Pick a prime p (presumably greater than x).*
- *Pick an exponent e such that* $\gcd(e, p-1) = 1$.
- *Encrypt to a secret message by taking*

$$m = x^e \pmod{p}\ .$$

Here are the steps for decryption.

- *Find an inverse modulo $p-1$ to e, and call it f.*
- *Decrypt (if you want) by taking*

$$m^f \equiv \pmod{p}$$

- *Celebrate in your opponent's destruction.*

Proof. Why does this work? First, note that our condition on f is equivalent to

$$ef \equiv 1 \pmod{p-1}$$

. Then we can simply compute that

$$m^f \equiv (x^e)^f \equiv x^{ef} \equiv x^1 \equiv x \pmod{p}$$

which verifies that we get the original message back. □

Feel free to use the following Sage cells to see what happens with your own short messages.

```
@interact
def _(message='mathiscool',e=677):
    secret=encode(message)
    p=next_prime(100*(secret))
    if gcd(e,p-1) != 1:
        pretty_print(html("Looks␣like␣$%s$␣isn't␣coprime␣
            to␣the␣prime!␣Try␣another␣one."%e))
    else:
        code=mod(secret,p)^e
        try:
            f=mod(e,p-1)^-1
        except:
            pretty_print(html("Looks␣like␣$%s$␣is␣not␣
                coprime␣to␣the␣prime␣we␣chose,␣$%s$"%(e,p)))
        pretty_print(html("My␣encoded␣message␣is␣
            $%s$"%secret))
        pretty_print(html("A␣big␣prime␣bigger␣than␣that␣is␣
            $%s$"%p))
        pretty_print(html("And␣I␣chose␣exponent␣$%s$"%e))
        pretty_print(html("The␣encrypted␣message␣is␣
            $%s$"%code))
        pretty_print(html("The␣inverse␣of␣$%s$␣is␣
            $%s$"%(e,f)))
        pretty_print(html("And␣the␣decrypted␣message␣turns␣
            out␣to␣be:"))
        print ''.join(decode(code^f))
```

Or you can choose a prime on your own.

```
@interact
def _(message='hi',p=991,e=677):
    secret=encode(message)
    if is_prime(p) and gcd(p,e)==1 and p>secret:
        e=677 # hopefully coprime to p-1
        code=mod(secret,p)^e
        try:
            f=mod(e,p-1)^-1
        except:
            pretty_print(html("Looks␣like␣$%s$␣is␣not␣
                coprime␣to␣the␣prime␣we␣chose,␣$%s$"%(e,p)))
        pretty_print(html("My␣encoded␣message␣is␣
            $%s$"%secret))
        pretty_print(html("A␣big␣prime␣bigger␣than␣that␣is␣
            $%s$"%p))
        pretty_print(html("And␣I␣chose␣exponent␣$%s$"%e))
        pretty_print(html("The␣encrypted␣message␣is␣
            $%s$"%code))
        pretty_print(html("The␣inverse␣of␣$%s$␣is␣
            $%s$"%(e,f)))
        pretty_print(html("And␣the␣decrypted␣message␣turns␣
            out␣to␣be:"))
        print ''.join(decode(code^f))
    elif not is_prime(p):
        pretty_print(html("Pick␣a␣prime␣$p$!"))
    elif p <= secret:
```

```
        pretty_print(html("Make␣sure␣your␣prime␣is␣bigger␣
            than␣your␣secret,␣$%s$"%secret))
    else:
        pretty_print(html("Make␣sure␣that␣$gcd(p,e)=1$!"))
```

Sage note 11.3.4 (Compute what you need)**.** Remember, you can always compute anything you need. For instance, if you for some reason didn't pick a big enough prime, you can use the following command to find one.

```
next_prime(11058)
```

Remark 11.3.5. In 2015, Whitfield Diffie and Martin Hellman won the Turing Award for their contribution, the highest award in computer science.

11.3.4 A brief warning

Remember, the key that makes it all work (thanks to Fermat's Little Theorem/Euler's Theorem) is that exponents of congruences mod n live in the world of congruences mod $\phi(n)$, as long as they are numbers coprime to $\phi(n)$. That's why $\gcd(e, p-1) = 1$ is important.

Here's an example of how not choosing your exponent wisely can go wrong.

```
message='hi' # needs to be in quotes
secret=encode(message)
p=991 # needs to be bigger than secret
e=2 # NOT coprime to p-1
code=mod(secret,p)^e
code
```

Sage note 11.3.6 (Change values right in the code)**.** You don't have to have a Sage cell with little boxes for interacting to change the values! Try changing them above to encode your own secret.

Assuming you followed along, so far, so good; it got encrypted. But what happens when we try to decrypt?

```
f=mod(e,p-1)^-1
message,secret,code,decode(code^f) # prints all the steps
```

You should have gotten an error (in fact, a `ZeroDivisionError`, which should sound relevant). It turns out not even to be possible to go backwards. Be warned that you must know the mathematics to use cryptography wisely.

11.4 An Interesting Application: Key Exchange

There is a quite useful application of D-H called **key exchange**. Here is the basic concept.

Two people trying to pass information (universally called Alice and Bob) want to decide on a secret key for using some encryption routine. Since all we really care about are the numbers, once we've encoded, we should just assume the key is a number.

Unfortunately, Alice and Bob know that someone may be listening in on their decision. Instead of trying to send a secret key only one of them has chosen, they try to create a secret key *together* using (essentially) public means. Here's how it works.

Algorithm 11.4.1 (Diffie-Hellman key exchange)**.** *Here are the steps.*

- *First, Alice and Bob jointly pick a big prime p and a base for exponentiation g, presumably with $1 < g < p$. This doesn't need to be secret.*
- *Now, they each* secretly *choose an exponent; maybe Alice chooses m and Bob chooses n.*
- *The key step: Each of them exponentiates g to their secret power, modulo p.*
- *Then they pass off these numbers to each other, and once again exponentiate the* other *person's number to their* own *secret power, modulo p.*

The resulting numbers are the same and give the secret key.

Proof. The two numbers are $(g^m)^n = g^{mn}$ and $(g^n)^m = g^{nm}$, which are the same, and certainly are so modulo p. □

Example 11.4.2. Alice and Bob pick $p = 991$ and $g = 55$, and then (separately) pick $m = 130$ and $n = 123$. Then they compute the powers g^m and g^n modulo p.

```
p=991
g=mod(55,p)
m=130
n=123
Alice_does=g^m
Bob_does=g^n
Alice_does,Bob_does
```

Alice and Bob have different numbers now, but after doing their powers after the exchange, the numbers should be the same.

```
Bob_does^m,Alice_does^n
```

Note the code takes one power to the `m` and the other power to the `n`.

Thus, now they have a secret key ($g^{mn} = g^{nm}$) they can easily compute but which a spy in the middle cannot. Feel free to try this with your own numbers you pick!

```
@interact
def _(p=(991,prime_range(1000)),g=55,m=130,n=123):
    g=mod(g,p)
    pretty_print(html("If␣you␣jointly␣picked␣$p=%s$␣and␣
        base␣$g=%s$"%(p,g)))
    pretty_print(html("Then␣separately␣picked␣secret␣
        powers␣$m=%s$␣and␣$n=%s$"%(m,n)))
    pretty_print(html("Your␣publicly␣traded␣info␣would␣be␣
        $%s^{%s}\equiv␣%s$␣and␣$%s^{%s}\equiv␣
        %s$"%(g,m,g^m,g,n,g^n)))
    pretty_print(html("But␣the␣secret␣joint␣key␣would␣be␣
        $%s^{%s\cdot␣%s}\equiv␣%s$"%(g,m,n,g^(m*n))))
```

This gives an encryption key useful to both Alice and Bob, to protect from any potential Eve who might be listening in. (That's Eve for eavesdropping, believe it or not – also a universal person in these stories.)

On the down side, if Eve not only is listening, but actually has access to Alice and Bob's transmissions and can change them, she can actually add *her own* exponent ℓ to the game, so that she pretends to have secret key $g^{m\ell}$ with Alice and secret key $g^{n\ell}$ with Bob. Both of their keys' security is now compromised. Such a situation is known as a "Man in the Middle" attack. There is no obvious way to stop such an attack with this algorithm, if Eve has that much power. (See Exercise 11.8.5.)

11.5 RSA Public Key

Sage note 11.5.1 (We keep reminding you)**.** Remember, this cell contains the commands used to make numbers from letters, so always evaluate it before doing any en/decoding.

```
def encode(s): # Input must be in quotes!
    s = str(s).upper()
    return sum((ord(s[i])-64)*26^i for i in range(len(s)))

def decode(n):
    n = Integer(n)
    list = []
    while n != 0:
        if n%26==0:
            list.append(chr(64+26))
            n -= 1
        else:
            list.append(chr(n%26+64))
        n //=26
    return ''.join(list)
```

In order to deal with some of the issues of symmetric systems, we will now introduce the most famous public-key system. Recall that this means we have an encryption key that is easy for anybody at all to use, but is very difficult to undo unless you know the secret. (Sometimes this is called a **trapdoor** system, because it's easy to fall in but it's hard to get back out unless you know where the secret passageway is!)

The formal name for the system in this section is "Rivest, Shamir, Adleman" or RSA, for the three folks who developed it in the late 1970s. The acronym continues to be the name of the security company they cofounded, owned by EMC when this was written.

11.5.1 The background

The idea behind RSA is to make Diffie-Hellman, which relies only upon Theorem 7.5.2 and primes, into a system which involves Euler's Theorem (9.2.3). We want to do so, but not so heavily as to make the computation too expensive. (With the advent of mobile devices, it turns out that this has once again become a big issue, so much so that even RSA or similar methods are being replaced with more sophisticated ones involving things like the Mordell equation, known as elliptic curves.)

It turns out that the easiest way to keep computation easy while sticking with exponentiation is to choose as a modulus a large integer n with only *two prime factors*, instead of *one* large prime p as we did before. For instance:

```
p=89
q=97
n=p*q
pretty_print(html("Multiply␣the␣primes␣$%s$␣and␣$%s$␣to␣
    get␣our␣modulus␣$%s$"%(p,q,n)))
```

Exponents here live in the world of $\phi(n)$. We can easily compute this using Fact 9.5.2 (so that $\phi(n) = (p-1)(q-1)$). So the computations are going to be easy for us, assuming we know p and q.

But they will not be so easy to compute without that knowledge, for which we need to have the prime decomposition of n. In particular, for reasonably large n, that means $\phi(n)$ is essentially secret to anyone who isn't tough enough to factor n.

Remark 11.5.2. At least that's what people currently believe; if it isn't true, we are in deep trouble security-wise, as we will see later.

As an example, in the early 1600s, Fermat believed $2^{32}+1$ was prime. It took until 1732 and the genius of Euler to factor $2^{32}+1$ as follows, which shows the one hundred sixteenth prime is the *smaller* of two factors.

```
2^32+1,factor(2^32+1),nth_prime(116)
```

Hence $n = 2^{32}+1$ wouldn't have been a bad n to choose in the early 1700s, since it would take a *lot* of trial and error to get to the one hundred sixteenth prime!

11.5.2 The practice of RSA

That's the preliminaries. From now on, we do exactly the same thing as before, choosing an e coprime to $\phi(n)$, etc. This time, though, instead of keeping e secret, we let anybody know it (along with n, which we have to let people know anyway).

Example 11.5.3. With the same primes, let's choose $e = 71$, because that is coprime to $\phi(89 \cdot 97) = \phi(89)\phi(97) = 88 \cdot 96 = 8448$.

```
p=89
q=97
n=p*q
phi=euler_phi(n)
e=71
pretty_print(html("Multiply␣the␣primes␣$%s$␣and␣$%s$␣to␣
    get␣our␣modulus␣$%s$"%(p,q,n)))
pretty_print(html("Are␣$e=%s$␣and␣$\phi(%s)=%s$␣
    coprime?"%(e,n,phi)))
gcd(e,phi)==1
```

We compute an inverse mod $\phi(n)$ just as before, which will be (as before) our decryption key. Since we are able to compute $\phi(n)$, it isn't hard to get an inverse for e; if you only knew n, though, it would be very hard to do this (for reasonably large n).

```
f=mod(e,phi)^-1;f
```

Now, just like with D-H, I raise my message (number) to the power e to encrypt, and raise to the power f to decrypt an encrypted message. Here are all the steps together!

```
@interact
def _(message='hi',p=89,q=97,e=71):
    secret=encode(message)
    n = p*q
    phi = (p-1)*(q-1)
    if gcd(n,e)==1 and n>secret:
        code=mod(secret,n)^e
        try:
            f=mod(e,phi)^-1
            pretty_print(html("My encoded message is
                $%s$"%secret))
            pretty_print(html("A big product of primes
                bigger than that is
                $pq=%s\cdot%s=%s$"%(p,q,n)))
            pretty_print(html("(which means my secret
                $\phi(n)=\phi(%s\cdot%s)=(%s-1)(%s-1)$ is
                $%s$)"%(p,q,p,q,phi)))
            pretty_print(html("And I chose exponent
                $%s$"%e))
            pretty_print(html("The encrypted message is
                $%s^{%s}\equiv%s$"%(secret,e,code)))
            pretty_print(html("The inverse of $%s$ modulo
                $%s$ is $%s$"%(e,phi,f)))
            pretty_print(html("And the decrypted message
                turns out to be:"))
            print ''.join(decode(code^f))
        except:
            pretty_print(html("Looks like $%s$ is not
                coprime to $\phi(%s)=%s$"%(e,n,phi)))
    elif gcd(phi,e)!=1:
        pretty_print(html("Make sure that
            $gcd(\phi(n),e)=1$!"))
    elif n <= secret:
        pretty_print(html("My encoded message is
            $%s$"%secret))
        pretty_print(html("Make sure that $pq=%s\cdot
            %s=%s$ is bigger than your secret"%(p,q,n)))
```

11.5.3 Why RSA works

Now we have an encryption method where anyone can encrypt. The modulus n (not written as pq) and e are both published, and anyone who wants to send a message of length n or less just exponentiates. You just have to be sure that $\phi(n)$ and e are coprime for it to be defined properly.

Algorithm 11.5.4 (RSA encryption algorithm)**.** *In order to encrypt a message x via RSA with public key (n, e), you do*

$$x^e \pmod n .$$

In order for the owner of the key to decrypt a message m, they do

$$m^{e^{-1}} = m^f \pmod{n}$$

for any f solving $ef \equiv 1 \pmod{\phi(n)}$.

Proof. Since

$$ef \equiv 1 \pmod{\phi(n)}$$

we have $ef = k\phi(n) + 1$ for some integer k. Hence

$$(x^e)^f = x^{ef} = x^{k\phi(n)+1} = (x^{\phi(n)})^k x^1 \equiv 1^k x \equiv x \pmod{n}$$

and it all works out, we recover the original message. □

And if someone nefarious were to try to decrypt this, they would need access to f somehow, or something equivalent to it mathematically. That would mean solving

$$ef \equiv 1 \pmod{\phi(n)}$$

for f without actually knowing what $\phi(n)$ is!

Naturally, that is pretty easy to compute in the cases above. But in real life?

```
p=next_prime(randrange(2^50))
q=next_prime(randrange(2^50))
n=p*q # needs to be bigger than secret
pretty_print(html("The␣first␣part␣of␣my␣key,␣$%s$,␣is␣the␣
    product␣of␣my␣secret␣primes"%n))
```

The n in the cell above is the product of two primes – but would you like to try to compute $\phi(n)$ by hand? Without knowing the actual primes, it could be very difficult to figure out $\phi(n)$, which you probably need to get f.

Realistic examples have much larger primes than this, say 100 digits. But let's see what would happen next in a 'real' example.

```
message='mathiscool' # needs to be in quotes
secret=encode(message) # needs to be less than n
pretty_print(html("My␣message␣is␣$%s$␣numerically"%secret))
```

Hopefully the randomness of the p and q I picked didn't keep n from being greater than the numerical value of the message.

Now we pick the other piece of our key, e. Believe it or not, it doesn't really seem to matter (though no one has proved this) what e is; documentation for a widely used RSA implementation says this:

> The modulus size will be *num* bits, and the public exponent will be e. Key sizes with $num < 1024$ should be considered insecure. The exponent is an odd number, typically 3, 17 or 65537.

Well, I figure 17 is easier to use than 65537 but less obvious than 3. Let's check that it's coprime to the modulus of the key.

```
phi=euler_phi(n)
e=17 # needs to be coprime to phi
pretty_print(html("And␣I␣can␣check␣whether␣$e=17$␣is␣
    coprime␣to␣$\phi(%s)$"%n))
gcd(phi,e)==1
```

If you get False above (I did once in a while during testing), then just pick a different e. (Only evaluate this cell if you have to!)

```
e=65537 # needs to be coprime to phi
pretty_print(html("Second␣try␣-␣is␣$e=65537$␣coprime␣to␣
    $\phi(%s)$?"%n))
gcd(phi,e)==1
```

Once we have our key, away we go!

```
code=mod(secret,n)^e
pretty_print(html("My␣encoded␣message␣is␣$%s$"%secret))
pretty_print(html("A␣big␣product␣of␣primes␣bigger␣than␣
    that␣is␣$n=%s$"%n))
pretty_print(html("And␣I␣chose␣exponent␣$%s$"%e))
pretty_print(html("The␣encrypted␣message␣is␣$%s^{%s}\equiv␣
    %s$"%(secret,e,code)))
```

Crack that! Who knows what $\phi(n)$ is?
But if I know it, I can calculate the inverse of e:

```
f=mod(e,phi)^-1
pretty_print(html("My␣original␣primes␣were␣$%s$␣and␣
    $%s$"%(p,q)))
pretty_print(html("So␣
    $\phi(n)=(%s-1)(%s-1)=%s$"%(p,q,phi)))
pretty_print(html("Which␣makes␣$f=%s$"%f))
pretty_print(html("And␣the␣decrypted␣message␣turns␣out␣to␣
    be:"))
print ''.join(decode(code^f))
```

11.6 RSA and (Lack Of) Security

There are some elementary security issues we can now discuss with RSA. Remember, we aren't learning to be security experts here, and there are far more powerful techniques available! But these are some underlying fundamentals.

Sage note 11.6.1 (A final reminder to evaluate definitions)**.** Don't forget to evaluate this so we can use words as messages instead of just numbers.

```
def encode(s): # Input must be in quotes!
    s = str(s).upper()
    return sum((ord(s[i])-64)*26^i for i in range(len(s)))

def decode(n):
    n = Integer(n)
    list = []
    while n != 0:
        if n%26==0:
            list.append(chr(64+26))
            n -= 1
        else:
            list.append(chr(n%26+64))
        n //=26
    return ''.join(list)
```

11.6.1 Beating the man in the middle

First, remember one problem with Diffie-Hellman key exchange (Section 11.4). Someone who can control your messages can actually fake them. This can't happen with public-key systems (at least not as easily). Here's why.

Suppose I want to let someone verify I am who I say I am. In a public-key system, I never need to let f get known, so I encode my signature *with* f itself as the exponent!

First, I just turn my signature into a number.

```
signature='Crisman'
code=encode(signature)
```

Then I raise it to the power of the *secret key* f, the inverse of the *public key* e.

```
p=89
q=97
n=p*q
phi=euler_phi(n)
e=71
f=mod(e,phi)^-1
secret=mod(code,n)^f
secret
```

Now anyone in the world can check my signature by raising this version of the signature to the public power e modulo n.

```
secret^e; decode(secret^e)
```

The reason this works is because

$$ef \equiv 1 \pmod{\phi(n)}$$

and $ef = fe$ in a commutative setting:

$$\left(Name^f\right)^e = (Name)^{ef} \equiv Name^1 \equiv Name \pmod{n}$$

Naturally, implementing this is somewhat more complex in real life (e.g. padding is used), but it is one major digital signing method implemented on many secure systems.

Interestingly, this concept also can be used in the opposite way. Suppose that someone sends a message using their public signature as above – a message which later turns out to implicate him or her in illegal activity, a scandal, offensive behavior, etc. The author may wish to repudiate this message, but (at least in principle) the digital signature cannot be repudiated in the same way as other types of messages. (Of course, one can always say that one's private key was stolen, so it's not foolproof!) I am indebted to my colleague, Russ Tuck, for this observation.

11.6.2 A cautionary tale

Lest you think we are now completely secure, let me warn you about one possible problem. Remember how we said above that it seems not to matter too much what e is? Well, that is sort of true, and sort of untrue.

Suppose we chose to send a message using the following primes and randomly (?) chosen exponent e. (Notice that if $\gcd(e, \phi(pq)) \neq 1$, this code wouldn't have worked at all.)

```
message='hiphop'
secret=encode(message)
p=197108347
q=591324977
e=52665067560570823
n=p*q
phi=(p-1)*(q-1)
code=mod(secret,n)^e
f=mod(e,phi)^-1
pretty_print(html("My␣encoded␣message␣is␣$%s$"%secret))
pretty_print(html("A␣big␣product␣of␣primes␣bigger␣than␣
    that␣is␣$pq=%s\cdot%s=%s$"%(p,q,n)))
pretty_print(html("(which␣means␣my␣secret␣
    $\phi(n)=\phi(%s\cdot␣%s)=(%s-1)(%s-1)$␣is␣
    $%s$)"%(p,q,p,q,phi)))
pretty_print(html("And␣I␣chose␣exponent␣$%s$"%e))
pretty_print(html("The␣encrypted␣message␣is␣
    $%s^{%s}\equiv%s$"%(secret,e,code)))
pretty_print(html("The␣inverse␣of␣$%s$␣modulo␣$%s$␣is␣
    $%s$"%(e,phi,f)))
pretty_print(html("And␣the␣decrypted␣message␣turns␣out␣to␣
    be:"))
print ''.join(decode(code^f))
```

The above cell just does the RSA algorithm for a particular case, verifying it works.

Now suppose Alice has sent Bob this message using Bob's impressive RSA key (above) of

$$(n, e) = (116555088756283019, 52665067560570823) .$$

Let me impersonate Eve, trying to snoop. On a hunch (or, as [C.1.3] puts it, after attending a seminar at a decryption conference), I figure I don't have much to lose by just trying random arithmetic, so I decide to just keep taking eth powers of the encrypted text (which was already raised to the eth power once).

```
trial_decrypt=code
for i in [1..25]:
    trial_decrypt=trial_decrypt^e
    print ''.join(decode(trial_decrypt))
```

What's this? You should see a meaningful message appear. Eve would barely have to do anything to decrypt this!

11.6.3 The explanation

This circumstance may seem mysterious, but it really is related to mathematics we already used a number of times before. Remember that we could find an inverse for a modulo n by just taking powers of a, because

$$a^{-1} \equiv a^{\phi(n)-1} \pmod{n}$$

Similarly, for any possible message m and public key e, there will always be some power k of e such

$$m^{e^k} \equiv m^1 \pmod{n}$$

which is the same as

$$e^k \equiv 1 \pmod{\phi(n)}$$

For this to happen, we would have to coincidentally have that not only $\gcd(e, n) = 1$ (which we always pick), but also that $\gcd(e, \phi(n)) = 1$. Then Euler's Theorem 9.2.3 says that the order of e modulo $\phi(n)$ is a divisor of $\phi(n)$, so we will sometimes find e where that order is a *small* divisor of $\phi(n)$.

Of course, in real life this would only happen randomly, so you could just protect against it by checking the order of e modulo $\phi(n)$. Here's how I created this not-quite-random example!

```
g = 7  # Pick something coprime to n
print gcd(g,phi)
i = mod(g,phi) # look at it mod phi(n)
print i.multiplicative_order()
print factor(i.multiplicative_order())
```

```
j=i^( 11 * 13 * 37 * 1879 * 22973)  # take it to as high a
    power I can to reduce the order
print j.multiplicative_order() # make sure this is small
print gcd(j,phi) # check we still have the right gcd
print j
```

What was the problem here? The issue is that we had a $\phi(n)$ such that its group of units had elements of tiny order in *its* group of units. (Two levels deep here!)

More precisely, we had a $\phi(n)$ such that $U_{\phi(n)}$ had elements of very small order in it, so that

$$e^{verysmallorder} \equiv 1 \pmod{\phi(n)}$$

was possible. How can we avoid this?

11.6.4 A solution

When we found elements of big order (primitive roots, for prime modulus) in Chapter 10, we relied on having the original modulus p being prime. We did not tell the whole story, but we did do enough of what happens with other moduli to know that we should suspect that having a small number of primes to small powers should let us keep finding elements of big order. (For instance, we saw that 2^n had elements pretty close to being primitive roots.)

And we do know something about $\phi(n)$. Namely, since $n = pq$ is the product of two primes, we know that $\phi(n) = (p-1)(q-1)$ is also the product of two numbers. It would be too much to hope for *those* to be prime! After all, $p-1$ and $q-1$ will both be even, since p and q will be odd primes.

However, it's possible to pick p and q so that $p-1 = 2p'$ and $q-1 = 2q'$, where p' and q' are both prime. In that case

$$\phi(n) = \phi(pq) = \phi(p)\phi(q) = 2p'2q' = 4p'q'$$

so that $\phi(n)$ at least has order four times a product of (still big) prime numbers.

We will not prove it, but it turns out this is enough to guarantee the existence of elements of orders $p'-1$ and $q'-1$ in $U_{\phi(pq)}$, just like we had elements of order $p-1$ in U_p. To be precise, we get elements of order

$$p'-1 = \frac{p-1}{2} - 1 \text{ and } q'-1 = \frac{q-1}{2} - 1$$

if

$$\frac{p-1}{2} \text{ and } \frac{q-1}{2}$$

are both prime. Here is an example of this with very small p and q.

```
n = 7*11
phi = euler_phi(n)
[mod(i,phi).multiplicative_order() for i in [1..phi] if
    gcd(i,phi)==1]
```

Going backwards, we are looking for prime numbers p', q' such that $2p' + 1, 2q' + 1$ are also prime, and then we use $p = 2p' + 1$ and $q = 2q' + 1$ in RSA, finding an exponent that has big order in $U_{\phi(n)}$. In this example, $p' = 5$ and $q' = 3$.

Such primes p' and q' are called **Germain primes**, for French mathematician Sophie Germain – the only female number theorist of note before the twentieth century, and definitely an important figure.

Research into security of number-theoretic cryptography is ongoing. There are practical points as well; as just one example, one ePrint discovered that 0.2% of a large set of public keys have "secret keys [which] are accessible to anyone who takes the trouble" to try to find them. Other studies have found even more – often because of poor randomness.

Another interesting vulnerability is that there is a significant (in practice, not in theory) chance that two RSA keys will share a (prime) factor. In another study it was found that not only did a nontrivial number of apparently unrelated keys share a factor (enabling their complete factorization), many keys were the same! These would still be hard to factor, but as the authors says, "[g]iven cryptographic key sizes, we would not expect to see devices generate a single duplicated key for the population sizes we examined if the keys were generated with sufficient entropy." This chapter is just a small taste of issues to consider, and no substitute for having a real security professional!

11.7 Other applications

These are just the most typical and famous encryption systems used in introductory number theory texts; there is a huge amount of active research into the mathematics of cryptography, much of which uses rather more advanced mathematics. The important point is that we have observed some of the basic issues to consider in such systems. Another good system to check out which has mathematics at the same level is the El-Gamal system.

There are also tons of other cryptographic applications one can do which are not directly about encryption. Two of my favorites are finding ways to flip a coin over the Internet and how to find out if someone makes more money than you without them revealing their actual salary. For now, we just share one secret.

11.7.1 Secret sharing

Suppose that three people work at a company with some trade secret, all of whom have clearance to know about the secret's details. However, the company wants to avoid one of the three being bought off by a competitor and revealing it in an act of corporate espionage.

The company needs to devise a system where, in order to actually gain access to the details of the trade secret, one needs *two* of the people involved. In a movie, you would have an impressive safe with three locks; each person would have a separate key to one of the locks, and the safe would be constructed so that any two of the keys would open it.

But real cryptography is not the movies! For one thing, the data is probably electronic, so it's really something we need to do digitally. Cryptography provides the perfect way to deal with these issues. What we will do is indeed give each person a key – a digital encryption key, of course. (The exposition is based on the one in the book I first learned it from, [C.1.4], which does this in full generality.)

Algorithm 11.7.1 (Secret Sharing)**.** *Suppose the trade secret is digitally represented as a large number K. Here are steps to create three different keys; access to any two of these will allow access to K.*

- *Choose some prime $p > K$.*
- *Choose three numbers $m_1 < m_2 < m_3$ which are:*
  - *mutually coprime and coprime to p, i.e.* $\gcd(m_i, m_j) = 1$ *and* $\gcd(m_i, p) = 1$.
  - *AND such that*
$$m_1 m_2 > p m_3$$
- *Let $M = m_1 m_2$.*
- *Now choose some $t < M/p$ at random. Then the keys are as follows:*
  - *We have a modified secret*
$$K_0 = K + tp$$
  - *Person i gets the key*
$$k_1 = K_0 \ (mod\ m_i)$$

Proof. What good do these do us? Well, the Chinese Remainder Theorem allows us to reconstruct K_0 modulo $m_i m_j$ with any two keys k_i and k_j. That may not seem like a lot; that just gives us things to within multiples of $m_i m_j$.

But by our choice of $M = m_1 m_2 > p m_3$, we know that $M/p > m_3$ (and hence $M/p > m_i$ as well). So

$$K_0 = K + tp < p + tp = (t+1)p \leq (M/p)p = M$$

And certainly if $K_0 < M$, then $K_0 < m_i m_j$, since M is the *smallest* such product. So the Chinese Remainder Theorem allows us to reconstruct K_0 uniquely, and then $K = K_0 - tp$!

Finally, note that just one person doesn't have enough information to get K, since that just tells that

$$K_0 \equiv k_i \pmod{m_i},$$

so that

$$K_0 = k_i + \ell m_i$$

for all ℓ modulo m_i. □

Obviously, we'll want to see this in action.

Example 11.7.2. This is a slight modification of [C.1.4, Example 7.8]. Suppose your secret was $K = 4$. Let's pick $p = 7$, and numbers $11, 13, 16$.

```
K=4
p=7
m1,m2,m3=11,13,16
```

We'll check quickly that $m_1 m_2 > p m_3$:

```
m1*m2>p*m3
```

So $M = 11 \cdot 13 = 143$, and we can pick $t = 15$ more or less randomly as being less than $M/p = 143/7 = 20\frac{3}{7}$.

```
M=m1*m2
t=15
```

So $K_0 = K + tp = 4 + 15 \cdot 7 = 109$:

```
K_0=K+t*p
```

This gives keys k_i, which are K_0 modulo m_i:

```
k1,k2,k3 = mod(K_0,m1),mod(K_0,m2),mod(K_0,m3)
k1;k2;k3
```

Note that in our example, we can check all the conditions in the proof by hand, but with industrial-size numbers that would not be possible.

Now let's actually reconstruct the secret K in these cases. First, let's see that any two people do have enough information.

```
CRT(10,5,m1,m2); CRT(10,13,m1,m3);CRT(5,13,m2,m3)
```

```
109-t*p
```

Great!

One might suspect that a lone person, without one of the other secret sharers, might be able to just 'guess' which of the various solutions was right in this very small example, but it's actually not clear which is the correct K_0. If you get only one chance, you might not want to try to be lucky!

```
[10+i*m1 for i in [0..10]];[5+i*m2 for i in
    [0..10]];[13+i*m3 for i in [0..10]]
```

As a note, we should point out that this secret sharing method doesn't just protect against someone defecting. It also provides protection against one of the three becoming incapacitated somehow. If all three were necessary to unlock the secret, the company is an illness or death or resignation away from its secret being irretrievably lost without a system of this type.

Finally, it is not terribly hard to extend this to a system that works with spreading a secret known to n in such a way that k of the n sharers are needed to access the secret. (Again, see [C.1.4].)

11.8 Exercises

1. Do all the encryptions and/or encodings in Sections 11.1 and 11.2 'by hand'.

2. Encrypt your name using an affine method ($ax+b$) with key $(5, 6, 29)$ (don't worry about letters), and decrypt BXHBI.

3. Create your own $ax+b \pmod n$ system of encryption and bring an encrypted message to class.

4. Use the Diffie-Hellman method of encryption to encrypt a short (three to five character) message with a $26 < p < 50$ 'by hand' (i.e. without Sage but with a calculator). Be prepared to explain your choice of e and p, and calculate that $ef \equiv 1 \pmod{p-1}$ by hand.

5. Draw a diagram and show that if Eve has control of both communications in D-H key exchange (Algorithm 11.4.1), she can intercept and decrypt *all* message.

6. Do this two-parter:

- Suppose you discovered that the message 4363094, where $p = 7387543$, actually represented the (numerical) message 2718. What steps might you take to try to discover e?
- Suppose that you discovered in the previous part by hard work that $e = 35$. Now quickly decrypt the message 6618138.

7. Pick two primes between 1000 and 2000 and create a public key (n, e) for them. What is the decryption key f? Show your work.

8. Suppose that $n = 9211$ and $e = 539$.

- Encrypt a (short) message.
- Find the decryption key f for this situation, and decrypt your message.
- Use f to sign your name!

9. Come up with your own RSA public-key system by choosing p and q and e as appropriate, but with $n > 10000$; then encrypt a short numerical message and hand in *only* the public key (n, e) and the encrypted message. (Your instructor's job will be to crack it!)

10. Construct a secret and share it in the way described in Algorithm 11.7.1.

11. Learn about a symmetric key cryptosystem in common use. Do you own any devices which use it?

12. Learn about the El-Gamal public key encryption method. How is it implemented? What mathematics used there is similar to what is used in this chapter? What is different?

13. Learn about the Advanced Encryption Standard. How is the mathematics used there different from what is used in this chapter?

Chapter 12

Some Theory Behind Cryptography

Cryptography is fun in and of itself. However, there are powerful theoretical issues at play throughout – as evidenced by the ever-increasing number of publications in this area.

With this in mind, we pick two of the many theoretical questions to address. Certainly we can only touch on basic questions, but the reader will be gratified to see how much variety there is even in this!

- How do we find all these big primes, anyway?
- How can we be sure it's not so easy to break the codes – such as by factoring big numbers?

12.1 Finding More Primes

As we have seen, it is not terribly hard to find lots of small primes easily. One can use Sieve of Eratosthenes, or make numbers coprime to known primes and then factoring them.

The problem is that almost every effort to find lots of *big* ones has been stymied. Primes simply do not follow nice enough rules to find them easily. This is despite the fact that they seem to follow *very* nice rules on average, which we will explore in later chapters.

12.1.1 Fermat primes

Here is an interesting historical example. Recall (Subsection 11.5.1) that our friend Pierre de Fermat thought that numbers of the form $2^{2^n}+1$ would always be prime – numbers such as 5, 17, and 257.

Definition 12.1.1. We call numbers of the form $F_n = 2^{2^n}+1$ **Fermat numbers**.

However, in 1732 Euler proved that this is not true for $n = 5$:

```
for n in [0..7]:
    pretty_print(html("If␣$n=%s$,␣then␣
        $2^{2^n}+1=2^{2^%s}+1=%s$␣factors␣as␣
        $%s$"%(n,n,2^(2^n)+1,factor(2^(2^n)+1))))
```

Nobody knows if there are any more primes in this sequence. Even the prime factors seem to be *quite* large, though.

There is a special test called Pépin's test that tests Fermat numbers for primality. It is equivalent to checking whether 3 is a primitive root of $2^{2^n}+1$. Proving it is just a little beyond us right now, so we will not address it here (see Subsection 17.5.2, though).

12.1.2 Primes from Fermat numbers

However, we can at least prove what seems obvious above – namely, that lots of primes come out of Fermat numbers. First, we need a lemma.

Lemma 12.1.2. *If $\ell = jk$, and k is even, then $2^\ell - 1$ factors as*

$$2^\ell - 1 = 2^{jk} - 1 = \left(2^j + 1\right)\left(\left(2^j\right)^{k-1} - \left(2^j\right)^{k-2} + \left(2^j\right)^{k-3} - \cdots + \left(2^j\right) - 1\right)$$

Proof. Multiply and/or apply a little induction. (See Exercise 12.7.1.) □

Example 12.1.3. For instance, $2^6 - 1 = 63$ factors as

$$2^{3\cdot 2} - 1 = (2^3 + 1)(2^3 - 1)$$

which corresponds to the factorization $9 \cdot 7$. Similarly, $2^{12} - 1 = 4095$ factors as

$$2^{3\cdot 4} - 1 = (2^3 + 1)(2^9 - 2^6 + 2^3 - 1)$$

which corresponds to the factorization $9 \cdot 455$.

Proposition 12.1.4 (Fermat numbers are coprime)**.** *$F_n = 2^{2^n}+1$ and $F_m = 2^{2^m}+1$ are coprime if $m \neq n$.*

Proof. First, notice that any two Fermat numbers are very closely related to each other; if $n < m$, then $F_n - 1$ divides $F_m - 1$. In fact, one is a power of the other:

$$2^{2^m} = \left(2^{2^n}\right)^{2^{m-n}}$$

Because of this, using Lemma 12.1.2 with $j = 2^n$ and $k = 2^{m-n}$ (which is certainly even), we get

$$2^{2^m} - 1 = \left(2^{2^n} + 1\right)\left(\left(2^{2^n}\right)^{2^{m-n}-1} - \left(2^{2^n}\right)^{2^{m-n}-2} + \cdots + \left(2^{2^n}\right)^1 - 1\right)$$

This implies the divisibility relationship

$$F_n = 2^{2^n} + 1 \mid 2^{2^m} - 1 = F_m - 2$$

so any number d that divides F_n also divides $F_m - 2$. Now we do a standard trick; combining all of the above facts, any divisor of F_n which also divides F_m must divide $F_m - (F_m - 2) = 2$, so a common divisor of F_n and F_m could only be two or one.

But both Fermat numbers are odd, so the gcd must be 1. □

12.1.3 Mersenne primes

Another early attempt at finding big primes was an idea of Marin Mersenne. Mersenne was a Minim monk who not only acted as a clearinghouse for scientific knowledge in early 17th century France (particularly between Pascal, Fermat, Descartes, Roberval, and their friends) but also wrote major theological and music theoretical treatises of his own.

Mersenne suggested that one try searching for primes of the form $2^p - 1$, where p is itself prime.

Definition 12.1.5. In general, numbers of the form $M_n = 2^n - 1$ are called **Mersenne numbers**. If they are prime, they are called **Mersenne primes**.

```
for p in prime_range(100):
    pretty_print(html("If␣$p=%s$,␣then␣$2^p-1=2^{%s}-1=%s$␣
        factors␣as␣$%s$"%(p,p,2^p-1,factor(2^p-1))))
```

Certainly this doesn't always give primes, but it's not bad. You can help the world search for more Mersenne primes if you leave your personal computer on and connected to the Internet, via the Great Internet Mersenne Prime Search (GIMPS). Random computers in labs at the University of Central Missouri and UCLA have found some of the largest known primes this way.

The most recent one (as of this writing) was found in January 2016! The largest known such primes are *very* large; this one has over twenty-two million digits. GIMPS even won a monetary prize for finding these huge ones; they shared it with many of the people who made it possible.

These primes are far too large and are not common enough to use for serious applications, but nonetheless help us investigate ideas about primes. Interestingly, although it is not necessarily an application one might think of, searching for them can also help more mundane hardware testing. A good example of this is that computing the GIMPS program uncovered a bug in a major Intel chip. Number theory can push our hardware (and software!) beyond our imagination.

The reason implementing something like this is conceivable is because of a special test which applies just to numbers $2^p - 1$.

Algorithm 12.1.6 (Lucas-Lehmer test)**.** *Let $x_0 = 4$ and let p be prime. To test whether $2^p - 1$ is prime, create the list of numbers*

$$x_{n+1} = \textit{residue of } x_n^2 - 2 \textit{ modulo } 2^p - 1$$

Do this $p - 2$ times; if the result is divisible by $2^p - 1$ (i.e., is zero modulo $2^p - 1$), then $2^p - 1$ is in fact prime.

Example 12.1.7. With $p = 5$ and $2^p - 1 = 31$, we would start with $x_0 = 4$; doing it $5 - 2 = 3$ times gives:

1. $4^2 - 2 = 14$ modulo 31 is 14
2. $14^2 - 2 = 194$ modulo 31 is 8
3. $8^2 - 2 = 62$ modulo 31 is 0

And of course 31 is indeed prime.

```
@interact
def _(p=(71,prime_range(100))):
    test = 4
    num = 2^p-1
    for i in range(p-2):
        test=(test^2-2)%num
    pretty_print(html("The test says "+str(bool(test==0))))
    pretty_print(html("And in fact $2^{%s}-1=%s$ primality
        is "%(p,num)+str(is_prime(num))))
```

Proving this is slightly beyond our capabilities right now.

12.1.4 Primes from Mersenne numbers

Although we didn't prove the test, we *can* prove the lesser result that Mersenne numbers are coprime, which (just as with the Fermat numbers) can give us a lot of interesting prime *factors*.

Proposition 12.1.8 (Mersenne numbers are coprime)**.** *Mersenne numbers* $2^p - 1$ *and* $2^q - 1$ *with coprime exponents are themselves coprime.*

Proof. By way of contradiction, let $d > 1$ be the gcd of the two numbers $2^p - 1$ and $2^q - 1$. Let's investigate the order of $2 \neq 1$ in U_d. (Before reading more, think about why 2 is even in this group.)

By definition of divisibility,

$$2^p \equiv 1 \pmod{d} \text{ and } 2^q \equiv 1 \pmod{d}$$

By group theory (use Theorem 8.3.11) we know that $2^k \equiv 1$ means that k is a multiple of the order $|2|$ of the element 2. Since they both satisfy this, p and q are both multiples of $|2|$.

Since p and q are coprime, though, the only possibility for $|2|$ is that $|2| = 1$. This is a contradiction, so our assumption that $d > 1$ was wrong. □

See this video featuring Holly Krieger, by Numberphile for an interesting take on this. Namely, all Mersenne numbers after $2^6 - 1$ (even the ones where p is not prime!) have a new prime divisor.

12.2 Primes – Probably

Primality testing is full of little tidbits like in the previous section, and tantalizingly devoid of easy methods that work for all special cases. Indeed, none of these paths lead us to reliable, reasonably fast discovery of large primes for cryptographic purposes, nor do other computationally infeasible methods like using Wilson's Theorem or other even stranger formulas (some of which appear later in the text).

Instead, what is typically done is to pick a number, and then use tests on it that do *not* guarantee primality!

Why would this work? The idea is that if you use *enough* tests that do not guarantee primality *but* have a quite low false positive rate in practice, then the number you have is more likely to be a prime than the chance that your computer made an arithmetic error.

This is astonishing, but true; and if you end up with a number that likely to be prime, you can always check it with one of the various slower tests I will not describe.

12.2.1 Pseudoprimes

We start this discussion with our visual representation of powers (see Subsection 8.2.1).

```
@interact
def _(p=(11,prime_range(100))):
    P=matrix_plot(matrix(p,[mod(a,p)^b for a in [1..p] for
        b in srange(1,p+1)]),cmap='gist_earth')
    P.show(figsize=6)
```

Notice again here that Fermat's Little Theorem is visible in the second-to-last column. The graphic has been expanded, so that the last column is a slight restatement thereof, true for all a:

$$a^p \equiv a \pmod{p}.$$

(See Exercise 12.7.3.)

This is useful, as it works for all input. We can now use it to state a test for possible primality:

Fact 12.2.1. *If there is an a such that $a^n \not\equiv a \pmod{n}$, then n must be composite.*

So if $a^n \equiv a \pmod{n}$ for a given n, it's at least possible that n is prime.

Definition 12.2.2. If $a^n \equiv a \pmod{n}$, we say n passes the **base a test**.

Definition 12.2.3. If $a^n \equiv a \pmod{n}$ but n is not prime, we say it is a **pseudoprime** base a.

That is to say, if a number satisfies Fermat's Little Theorem, we think it is likely enough to be prime to call it a pseudoprime.

It turns out that everyone from the ancient Chinese to Leibniz used this test for the base $a = 2$ to assert numbers are prime. And it doesn't do a bad job. As some former students pointed out, it's sort of like internet date matching for primes; it doesn't always work but can succeed reasonably often.

```
@interact
def _(n=100):
    pretty_print(html("Here␣are␣the␣numbers␣through␣$%s$␣
        that␣pass␣the␣base␣2␣test"%n))
    pretty_print(html("along␣with␣whether␣they␣are␣
        actually␣prime"))
    for i in [2..n]:
        if mod(2^i,i)==2:
            pretty_print(html("$2^{%s}\equiv␣2\\text{␣(mod␣
                }%s)$␣and␣the␣primality␣of␣$%s$␣is␣
                %s"%(i,i,i,is_prime(i))))
```

We can change the numbers in the range above to check for more – say up to 1000, which allows exploring the following question.

Question 12.2.4. Are there any numbers which satisfy the base a test and are *not* prime?

To the surprise of many in the world of numbers, the answer is yes. The numbers $n = 341$, $n = 561$, and $n = 645$ turn out to fall in that category.

```
pretty_print(html("We␣factor␣$341$␣and␣get␣
    $%s$"%factor(341)))
pretty_print(html("We␣factor␣$561$␣and␣get␣
    $%s$"%factor(561)))
pretty_print(html("We␣factor␣$645$␣and␣get␣
    $%s$"%factor(645)))
```

That's still not bad – out of 171 total such potential pseudoprimes base 2, only 3 of them actually are not prime, or about one and three quarters percent.

Perhaps unfortunately to cryptographers (though interestingly to pure mathematicians!), it turns out that there are infinitely many such pseudoprimes.

Fact 12.2.5. *If n is a pseudoprime (base 2), then so is $2^n - 1$.*

We will get this result as a corollary of something stronger soon (see Corollary 12.4.3 and Theorem 12.4.2).

All the Fermat and Mersenne numbers pass the base 2 test, incidentally, though they are all quite large compared to a typical number you might try.

12.2.2 Prime impostors, and how to avoid them

If we want to check things out more carefully, we can try to test for primality with a different base. In the next cell, we choose $a = 3$.

```
for n in [341,561,645]:
    pretty_print(html("$3^{%s}\equiv␣%s\\text{␣(mod␣
        }%s)"%(n,mod(3,n)^n,n)))
```

As you can see, this exposes 341 and 645 as fakes. What about 561? Let's try that one with base 5 as well.

```
@interact
def _(p=(5,prime_range(50))):
    for pr in prime_range(next_prime(p)):
        pretty_print(html("$%s^{561}\equiv␣%s\\text{␣(mod␣
            }561)"%(pr,mod(pr,561)^561)))
```

Hmm, that's interesting. What if I add some primes, like 7 or 11? Try it above.

In the next cell, I get systematic. We should expect output if 561 doesn't pass the base a test for some a.

```
@interact
def _(p=(5,prime_range(1000))):
    pretty_print(html("The␣primes␣up␣to␣$%s$␣for␣which␣
        $561$␣fails␣the␣base␣$p$␣test:"%p))
    for pr in prime_range(next_prime(p)):
        if mod(pr,561)^561!=pr:
            pretty_print(html("$%s^{561}\equiv␣%s\\text{␣
                (mod␣}561)$"%(pr,mod(pr,561)^561)))
```

It appears that $p^{561} \equiv p \bmod 561$ for every prime p! Let's prove it.

Fact 12.2.6. *The number* 561 *is a pseudoprime for every prime base p.*

Proof. By the Theorem 5.3.1,

$$a^{561} \equiv a \pmod{561}$$

if and only if the same congruence holds for all the prime power factors of 561. We know that

$$561 = 3 \cdot 11 \cdot 17 \,,$$

so these are the ones to check.

Remember, the exponents for these congruences live in the $(\text{mod } \phi(p))$ world, so we just need to check what 561 is in each of those worlds. We get:

- $561 \equiv 1 \pmod{16 = 17 - 1}$
- $561 \equiv 1 \pmod{2 = 3 - 1}$
- $561 \equiv 1 \pmod{10 = 11 - 1}$

That is, for $p = 3, 11, 17$ we see

$$a^{561} \equiv a^1 \pmod{p}$$

Using the Chinese Remainder Theorem, this congruence is *always* true! □

Definition 12.2.7. We call a number which is pseudoprime to every base a, but is not a prime number a **Carmichael number**, in honor of the first person to actually produce such numbers, Robert Carmichael (in 1912).

So is 561 a Carmichael number? We saw the factorization above, but here it is again:

```
factor(561)
```

The proof of Fact 12.2.6 suggests that to find a Carmichael number, we might want to look at n which are a product of primes p_i such that $n - 1 \equiv 1$ in the exponent world of p_i. It turns out that this is true, and we can prove something even more specific.

Proposition 12.2.8 (Korselt's Theorem)**.** *Carmichael numbers are precisely those composite n for which n is a product of at least two distinct primes p_i (no squares)*

$$n = p_1 p_2 p_3 \cdots p_k \text{ with } p_i \neq p_j$$

such that

$$p_i - 1 \mid n - 1$$

for all the prime factors.

Proof. Prime numbers satisfy almost all the conditions trivially. To show that 561 is a Carmichael number we used this idea in the form $n \equiv 1 \pmod{\phi(p_i)}$ for all three prime factors, and essentially that is the entire proof that such numbers are Carmichael numbers.

We will not prove the other half of this theorem (that all Carmichael numbers have this form). It is not hard, however, using a slight variant on the Euler ϕ function one can acquire from investigating U_n for composite n. □

Example 12.2.9. Here is another Carmichael number.

```
n=29341
pretty_print(html("$%s$ is composite with factorization
    $%s$, but"%(n,factor(n))))
for fact,pow in factor(n):
    pretty_print(html("$%s^{%s}\equiv %s\\text{ (mod
        }%s)$"%(fact,n,mod(fact,n)^n,n)))
pretty_print(html("and"))
for fact,pow in factor(n):
    pretty_print(html("$%s\equiv %s\\text{ (mod
        }\phi(%s)=%s)$"%(n, mod(n,euler_phi(fact)), fact,
        euler_phi(fact))))
```

12.3 Another Primality Test

For a long time it was open whether we might be lucky and there are finitely many Carmichael numbers. However, as was proved in the mid-nineties, there are infinitely many Carmichael numbers.

So now what? Can we find other ways to reliably get primes?

12.3.1 Another pattern

To answer this, we turn to another result visible in our modular power graphic.

```
@interact
def _(p=(11,prime_range(100))):
    P=matrix_plot(matrix(p-1,[mod(a,p)^b for a in [1..p-1]
        for b in srange(1,p)]),cmap='gist_earth')
    P.show(figsize=7)
```

As usual, Fermat's Little Theorem is the right-hand column. What's that pattern in the middle column?

Theorem 12.3.1 (The Square Root of Fermat's Little Theorem)**.**

$$a^{(p-1)/2} \equiv \pm 1 \pmod{p} \text{ for any odd prime modulus } p \nmid a$$

Proof. Since $a^{p-1} \equiv 1$ we know that $a^{(p-1)/2}$ is a solution to $x^2 \equiv 1$. (Note that p is odd> so $(p-1)/2$ makes sense.)

We can rewrite and factor the congruence $x^2 \equiv 1$ as $p \mid x^2-1 = (x+1)(x-1)$ and given that p is prime, that means $p \mid x-1$ or $p \mid x+1$.

Then $x \equiv \pm 1 \pmod{p}$. (This is restated in Subsection 16.1.1.) Since $a^{(p-1)/2}$ is one such solution, then $a^{(p-1)/2} \equiv \pm 1 \pmod{p}$. □

What is the use for us of this theorem? Think similarly to the pseudoprime situation. Imagine we are testing some number n for primality, but we then find that

$$a^{(n-1)/2} \not\equiv \pm 1 \pmod{n},$$

then that number is definitely not prime.

Let's try this on our pesky Carmichael number, once again starting with base $a = 2$.

```
mod(2,561)^((561-1)/2)
```

Not again! Try another base – maybe $a = 3$?

```
mod(3,561)^((561-1)/2)
```

Phew, this works, as $3^{(561-1)/2} \not\equiv \pm 1$ (and 561 is not prime). So this criterion *does* help us test at least a little better.

12.3.2 Miller's test

A slightly stronger variant of this test is called **Miller's test base** a for primality.

Algorithm 12.3.2 (Miller's test for base a)**.** *We will proceed by dividing and then checking a congruence.*

- *Begin with taking $n-1$; divide it by two, and then check the power*

$$a^{(n-1)/2} \pmod{n}$$

  If the result is -1 we say n passes Miller's test. If the result is not ± 1, we say it fails Miller's test (since if n is prime, the result would certainly be ± 1). If the result is $+1$, we continue.

- *Assuming $a^{(n-1)/2} \equiv 1$, we continue by dividing the power itself by two and then taking a to that new power. Once again, if the result is -1 we say n passes the test, and if it is not ± 1, we say it fails.*

- *If the result is $+1$, continue dividing the power by two, check the result. If we arrive at the point where we have divided $n-1$ by all possible powers of two and the result is still ± 1, then we say n passes the test.*

Example 12.3.3. Let's see a few examples of this. First, here is a number pseudoprime base 2 – but it does not pass this test, which is good since it's composite.

```
n=1387
pretty_print(html("We␣know␣$%s$␣is␣composite␣because␣it␣
    factors␣as␣$%s$"%(n,factor(n))))
pretty_print(html("Let's␣check␣$2^{(%s-1)/2}$␣modulo␣$%s$:␣
    it's␣$%s$"%(n,n,mod(2,n)^((n-1)/2))))
```

Looking good … But let's try another pseudoprime number (a Mersenne number, in fact) to see if it passes, just to be sure.

```
n=2047
pretty_print(html("We␣know␣$%s$␣is␣composite␣because␣it␣
    factors␣as␣$%s$"%(n,factor(n))))
pretty_print(html("Let's␣check␣$2^{(%s-1)/2}$␣modulo␣$%s$:␣
    it's␣$%s$"%(n,n,mod(2,n)^((n-1)/2))))
```

As we can see, this shows that $n = 2047$ passes the first part of Miller's test base 2, and that there is no further to go because $(2047-1)/2 = 1023$ is odd. So, as far as we know thus far, 2047 is prime.

Let's try this same test with an actual prime.

```
n=1009
pretty_print(html("We␣know␣$%s$␣is␣prime␣because␣it␣
    factors␣as␣$%s$"%(n,factor(n))))
pretty_print(html("Let's␣check␣$2^{(%s-1)/2}$␣modulo␣$%s$:␣
    it's␣$%s$"%(n,n,mod(2,n)^((n-1)/2))))
pretty_print(html("Let's␣check␣$2^{(%s-1)/2/2}$␣modulo␣
    $%s$:␣it's␣$%s$"%(n,n,mod(2,n)^((n-1)/2/2))))
```

We see that this passes Miller's test the first time, but we keep going since we got $\equiv 1$; the second time we got $\equiv -1$, so we stop and hope it's prime. (It is, in this case!)

12.4 Strong Pseudoprimes

Since composite numbers can pass this primality test too, this can get frustrating if we don't organize. So we come up with another name.

Definition 12.4.1. We call a composite number n that passes Miller's test base a a **strong pseudoprime base** a.

The bad news is that strong pseudoprimes exist, as we saw above with $n = 2047$. In fact, we can prove a theorem about it, which is analogous to Fact 12.2.5 and has it as an implication.

Theorem 12.4.2. *If n is a pseudoprime base 2, then $2^n - 1$ is a* strong *pseudoprime base 2.*

Proof. Let n be composite and odd, but it passes the test

$$2^n \equiv 2 \pmod{n}$$

Since n is odd, we can cancel 2, and get

$$2^{n-1} \equiv 1 \pmod{n}$$

Rewrite this as $2^{n-1} - 1 = nk$ for some (odd) integer k, which yields

$$(2^n - 1) - 1 = 2^n - 2 = 2\left(2^{n-1} - 1\right) = 2nk$$

Now apply Miller's test to $2^n - 1$.

$$2^{(n-1)/2} = 2^{nk} = (2^n)^k \equiv 1^k \equiv 1 \pmod{2^n - 1}$$

It passes Miller's test.

All that remains is to show $2^n - 1$ is composite if n is composite; this is a fairly straightforward extension of Lemma 12.1.2 (see Exercise 12.7.2). □

Corollary 12.4.3. *We have proved that if n is a pseudoprime base a, so is $2^n - 1$. (This is Fact 12.2.5.)*

Proof. All we need is that $(\pm 1)^2 = 1$. □

Corollary 12.4.4. *There are infinitely many strong pseudoprimes (and hence regular pseudoprimes) base 2.*

Proof. Just keep doing $2^n - 1$. □

Example 12.4.5. For instance, we now know that $2^{341}-1$ must fall in that category, and since the second number below is odd, this confirms it.

```
n=2^341-1
print mod(2,n)^((n-1)/2),(n-1)/2
```

But there are *not* strong Carmichael numbers! In fact:

Theorem 12.4.6. *If n is an odd composite positive integer, then n passes Miller's test for at most $(n-1)/4$ bases a between 1 and $n-1$.*

Proof. We will not do this proof, as it is somewhat long, but it is accessible to us at this time. It counts numbers of solutions of $x^\ell - 1$ modulo various prime powers and combines them with the Chinese Remainder Theorem to give a good counting argument. □

Needless to say, no one could compute that many bases to prove primality for any realistic n! But Rabin used this fact to suggest a test for a *probable prime* with probability of failure less than $\left(\frac{1}{4}\right)^k$ for any desired k.

Algorithm 12.4.7 (Miller-Rabin (probabilistic) primality test)**.** *Run Miller's test for k different bases less than $n-1$. If a number passes all of them, the probability of failure is less than $\left(\frac{1}{4}\right)^k$.*

For 100 primes, this is the probability that would come out.

```
(1./4)^100
```

So if you run the test for 100 primes, you are in pretty decent shape.

You can also always use some slow test to prove primality. That is what is called a **certificate of primality**, and although you may not believe it, programs that reliably generate reasonably large (100-200 digits, right now) primes and can verify it are hot items on the virtual shelves of those who care about such things.

Finally, let's see this in action. Remember that we wanted keys larger than 1024 bits for at least a semblance of security in RSA? Here we go with a start:

```
p=next_probable_prime(randrange(2^1024))
q=next_probable_prime(randrange(2^1024))
n=p*q
pretty_print(html(p))
pretty_print(html(q))
pretty_print(html(n))
```

The p and q we get above are just probable primes. Verifying them could take a little longer! Here, we try it with just one of them.

```
p=next_probable_prime(randrange(2^1024))
%time is_prime(p)
```

Sage note 12.4.8 (Reminder about timing)**.** Don't forget, you could use `%time is_prime(p)` to time this operation in a worksheet or Sage command line.

12.5 Introduction to Factorization

Let's take a last crack at issues directly related to cryptography. (That doesn't mean that other stuff we do in this text is unrelated – oh no! Especially the geometry is connected. But we will not make direct connections.)

We will focus on the main *attack* on the RSA algorithm, namely *factorization*.

12.5.1 Factorization and the RSA

Let's look at another toy RSA problem to get a sense of what is going on. First, I choose a modulus $n = 899$. I will also use Sage to verify it has two prime factors, without telling you what they are.

```
n=899
pretty_print(html("There␣are␣$%s$␣prime␣factors␣and␣their␣
    powers␣are␣$%s$␣and␣$%s$."%(len(n.factor()),
    n.factor()[0][1], n.factor()[1][1])))
```

Then I choose an exponent to raise my secret message by ...

```
e=13
pretty_print(html("We␣choose␣$n=%s$␣and␣exponent␣$e=%s$,␣
    and␣verify␣that␣$gcd(e,\phi(n))=1$:␣
    %s"%(n,e,1==gcd(e,euler_phi(n)))))
```

I haven't told you $\phi(n)$, but this guarantees it is coprime to my (public) encryption key, which I have chosen to be $e = 13$. Now we can encode our message, $x = 11$.

```
x=11
message=mod(x,n)^e
message
```

Now, how could we hope to crack this sinister message? (Assume that `euler_phi(899)` is just too huge for Sage to do.) Well, we *do* know $n = 899$ and that $e = 13$. That could help. Remember, if we knew p and q, we could easily calculate $\phi(n)$ without even using Sage, which should be enough.

Question 12.5.1. Can you quickly now factor n *without* using Sage?

Remark 12.5.2. Be smart about it. Think strategically; how should I have chosen a public modulus n to make this hard to do? How should p and q relate?

Hopefully you figured out p and q. Then we just need to find an inverse modulo $\phi(n) = (p-1)(q-1)$ to get our *decryption* key.

Sage note 12.5.3 (Trying your primes yourself)**.** You can fill in the values you got for `p` and `q` here to make things work. Try it!

```
p=
q=
f=inverse_mod(e,(p-1)*(q-1))
f
```

And we decrypt:

```
message^f
```

This gives us the original message $x = 11$ again.

This simple example makes it clear why factorization, not just looking for primes, might be important. To be truthful, many researchers in factorization simply do it to stay one step ahead of the other side, who is presumably also researching factorization – so to some extent it is an arms race.

But factorization is also inherently interesting mathematically! Here is an interesting statement, as an example.

Fact 12.5.4. *If I know $\phi(n)$ and n, and know that n is a product of exactly two distinct primes, I can easily compute them both.*

Proof. Of course, if we know $\phi(n)$, we already can crack the code, but who cares; maybe we are given $\phi(n)$ and n and want the factorization. Here is the short proof.

Suppose the (as yet unknown) primes are p and q. Then expand our formula to

$$\phi(n) = (p-1)(q-1) = pq - p - q + 1 = n - (p+q) + 1$$

We now can represent both $p+q$ and pq as formulas in n and $\phi(n)$:

- $p + q = n - \phi(n) + 1$
- $pq = n$

Where might we have a formula with $p+q$ and pq? That should seem familiar ...

$$(x-p)(x-q) = x^2 - (p+q)x + pq$$

So we can simply use the quadratic formula on this expression to get the values for p and q!

$$p, q = \frac{(p+q) \pm \sqrt{(p+q)^2 - 4pq}}{2} = \frac{n - \phi(n) + 1}{2} \pm \frac{\sqrt{(n-\phi(n)+1)^2 - 4n}}{2}$$

□

Example 12.5.5. Continuing the example above,

$$x^2 - (899 - 840 + 1)x + 899 = x^2 - 60x + 899 = 0$$

gives

$$x = \frac{60 \pm \sqrt{60^2 - 4(1)(899)}}{2(1)} = 30 \pm \frac{\sqrt{3600 - 3596}}{2} = 30 \pm 1 = 29, 31$$

12.5.2 Trial division

The first, and oldest, method of factoring is one you already know, and maybe used a few minutes ago – **trial factorization**. It is the method we used with the Sieve of Eratosthenes; you just try each prime number, one by one.

Do you remember what the highest number you would have to try in order to factor n by trial division is? (Can you prove it?)

The following algorithm does this very naively (and slowly, even for trial division). Let's try to talk through what each step does.

Sage note 12.5.6 (Code for trial division)**.** This is one of the few places where it really is important to follow the code. That said, the details of the syntax are not as important as the algorithm – unless you want to harness the power of computers more effectively!

```
def TrialDivFactor(n):  #  We define the function
    p = next_prime(1)     #  We start off by testing the
        next prime after 1
    top = ceil(math.sqrt(n))  # This was proved to be the
        biggest number we need
    while p < top:        #  As long as the prime is less
        than that bound, we keep going
        if mod(n,p)==0:   #  In this case, p divides n and
            we're done!
            break         #  This is Python's way of
                saying we are done searching
        p=next_prime(p)   #  Otherwise, we try the next
            prime until we're done looking
    if n==1:              #  We probably could have
        checked for this right away
        print "1 is not prime"   #  Well, 1 is not a prime!
    elif p==n:            #  If we get all the way through
        and end with a prime...
        print n,"is prime"  #   Then our number was prime
    elif mod(n,p)==0:     #  But otherwise... (!)
        print n,"factors as",p,"times",n/p  #  We have a
            factorization!
    else:                 #  And finally...
        print n,"is prime"      #  We must have gotten
            lucky.
```

Algorithm 12.5.7 (Trial Factorization)**.** *To factor n, first enumerate the primes in ascending order $p_1, p_2, \cdots p_k$, where p_k is the largest prime less than or equal to $\sqrt{n}$. For each prime in order, check whether $p_i \mid n$. If it does, terminate by returning p_i and n/p_i; otherwise n must be prime.*

Now let me verify it works on easy examples. Remember, we are just looking for factors at this point, not complete factorizations.

```
for z in range(1,18):
    TrialDivFactor(z)
```

Okay, so this seems reasonable. But it's a little more problematic when you try to do large numbers, where large means "bigger than you can do by hand, but nowhere close to the size we looked at in general." I'll actually time how long it takes.

```
@interact
def _(n=6739815371):
    TrialDivFactor(n)
    timeit('TrialDivFactor(%s)'%n)
```

Sage actually has this implemented in a much faster way, primarily by using optimized integers and a special version of Python that allows turning it into *much* faster code in the C language (Cython). Notice that it just returns a single factor – another slight speedup.

```
@interact
def _(n=6739815371):
    print n.trial_division()
    timeit('%s.trial_division()'%n)
```

That's roughly one thousand times faster for the initial example! Naturally, it's possible to speed up even more. But note that getting the full factorization slows us back down; after all, one has to check that the remaining factor is prime (or factor it, if it isn't).

```
@interact
def _(n=6739815371):
    print n.factor()
    timeit('%s.factor()'%n)
```

Even for this smaller number it takes some actual time – here is where one sees the difference between different implementations of the same algorithm.

```
timeit('TrialDivFactor(997*991)')
```

```
timeit('(997*991).trial_division()')
```

```
timeit('(997*991).factor()')
```

12.5.3 Starting in the middle

So much for trial division! But we have other tools at our disposal.

Some of you might have tried something other than straight trial factorization when attacking $n = 899$ from our earlier problem. Reason this way; since we know that someone is trying to protect a secret, they probably are not going to pick a number with primes like 3 and 5 in it. After all, that would be too easy to factor.

In fact, it stands to reason that the primes p and q should be relatively large compared to n – so why not start in the middle?

This was Fermat's idea for factoring larger numbers. However, he didn't just start with primes in the middle; for one thing, if your number is even somewhat big and you don't have a computer or huge list of primes, how would you know where to start? So Fermat became clever, as always, and used an algebraic identity to help himself along.

Fact 12.5.8. *Write $n = ab$, with $a > b$, and assume n is odd. Then we can write n as a* difference *of two square numbers.*

Proof. Namely, n is the difference of the squares of $s = \frac{a+b}{2}$ and $t = \frac{a-b}{2}$:

$$s^2 - t^2 = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2 = \frac{a^2}{2} - \frac{a^2}{2} + \frac{b^2}{2} - \frac{b^2}{2} + \frac{2ab}{4} + \frac{2ab}{4} = ab = n$$

□

Remark 12.5.9. Why is it fine to assume n is odd in these circumstances?

This may seem like an obscure identity to us, but at the time (and even well into the last century) such identities were the bread and butter of algebra, before we had tools like computers to help us along.

So what Fermat did is to try this identity backwards. Here is his strategy.

Algorithm 12.5.10 (The Fermat factorization algorithm)**.** *To find a factor for a number n, begin by seeking a perfect square s^2 bigger than n, but still as close as possible. Now, do the following until you succeed, increasing s by one each time.*

- *Check whether $s^2 - n$ is* itself *a perfect square t^2.*
- *That means we essentially turned*

$$s^2 - t^2 = n \text{ around into } s^2 - n = t^2$$

Once you succeed, then s and t are not *the factors of n; rather, they are*

$$a = s + t \text{ and } b = s - t\,.$$

Proof. It should be clear why a and b are the factors. But how do we know this algorithm terminates?

Assuming you started with s as instructed, eventually you will reach $s = (n+1)/2$, which is much larger than $\sqrt{n}$. But then $((n+1)/2)^2 - n = \frac{n^2+2n+1-4n}{4} = ((n-1)/2)^2$. You should check that this gives us the trivial factorization $n = n \cdot 1$, though! (See Exercise 12.7.9) □

Here is an implementation – again, assuredly slow, but at least verbose in its explanation – of this strategy. We simply start with the next s above the square root of n, and just keep trying $s^2 - n$ again and again for bigger and bigger s.

```
def FermatFactor(n,verbose=False):
    if n%2==0:
        raise TypeError,"Input must be odd!"
    s=ceil(math.sqrt(n))
    top=(n+1)/2
    while is_square(s^2-n)==0:
        if verbose:
            print s,"squared minus",n,"is",s^2-n,"which is
                not a perfect square"
        s=s+1
    t=sqrt(s^2-n)
    print "Fermat found that",s,"squared minus",t,"squared
        equals",n
    if s^2==n:
        print "So",n,"was already a perfect
            square,",s,"times",s
    elif s<top:
        print
            "So",s+t,"times",s-t,"equals",(s-t)*(s+t),"which
            is",n
    elif s==top:
        print "So Fermat did not find a factor, which
            means",n,"is prime!"
```

Example 12.5.11. Before we move on, let's try to factor 143 and 93 using this algorithm. Remember, we start with $s^2 - n$, where s is the next integer above $\sqrt{n}$, and see if it is a perfect square; then we increase s by one each time.

After we do them by hand, we can see what Sage does with them to check.

```
FermatFactor(143,verbose=True)
```

Well, we struck gold on the first try here! That happens if your number is the product of two primes which are two apart. (Such primes are known as twin primes, and have some interesting stories. Among other things, calculating with them helped find a bug in the Pentium computer chip in 1995; see Subsection 22.3.2.)

```
FermatFactor(93,verbose=True)
```

As you can see, we probably would have been better off with trial division for $n = 93$. It's obvious that it's divisible by 3, but that takes a long time to reach from the middle.

12.6 A Taste of Modernity

Now, these methods are the beginnings of how people *really* factor big numbers. Typically, one does trial division up to a certain size (maybe the first few hundred or thousand primes), then perhaps some modification of Fermat to make sure that there aren't any factors close to the square root if you are attacking something like RSA where that would otherwise be advantageous.

Then what?

There are many answers, some of which involve cool things called **continued fractions** or **number fields**. We won't really touch on those topics, but the previous section *did* talk about something very useful in the previous sections.

Namely, we could come up with some *probabilistic/random* methods! That's right, we are going to try to find a factor *randomly*!

12.6.1 The Pollard Rho algorithm

Here is the essence of this random approach; it is highly recursive, like many good algorithms.

Algorithm 12.6.1 (Generic routine for "random" factoring)**.** *Follow these steps.*

- *Pick some polynomial that will be easy to compute mod* (n)*.*
- *Plug in an essentially random seed value. (Often the seed is* 2 *or* 3*.)*
- *Compute the polynomial's value at the seed.*
- *If that has a non-trivial gcd with n, we have a factor. Otherwise, plug the value* back into *the polynomial, and repeat (and hope it eventually succeeds).*

Below is code for the method we'll discuss in this section. It has a modification to the generic algorithm which I will discuss below..

```
def PollardRhoFactor(n,kstop=50,seed=2):
    d=1
    a,b=seed,seed
    k=1
    def f(x):
        return (x^2+1)%n
    while (d==1 or d==n):
        a = f(a)
        b = f(f(b))
        d=gcd(a-b,n)
        k=k+1
        if k>kstop:
            pretty_print(html("Pollard␣Rho␣breaking␣off␣
                after␣$%s$␣rounds"%k))
            break
    if d>1:
        pretty_print(html("Pollard␣Rho␣took␣$%s$␣
            rounds"%k))
        pretty_print(html("The␣number␣it␣tried␣in␣the␣last␣
            round␣was␣$%s$,␣which␣shares␣factor␣
            $%s$"%(a-b,d)))
        pretty_print(html("And␣$%s$␣is␣a␣factor␣of␣$%s$␣
            since␣$%s\cdot␣%s=%s$"%(d,n,d,n/d,d*(n/d))))
```

The essence of the method is that by plugging in the values of the polynomial modulo n, we are generating a 'pseudo-random' sequence of numbers. And a 'pseudo-random' sequence might be *better* than the sequences we used for trial division or Fermat factorization, precisely because it will hit some small(ish) factors and some other large random factors. It might also be good that it could give us numbers which, although not a factor of n, might at least *share* a factor with n.

Example 12.6.2. Here are some details. Let $x_0 = 2$ and $f(x) = x^2 + 1$ (these could be different, but they are a typical first choice). We create the following sequence of x_i:

- $x_0 \equiv 2$
- $x_1 \equiv f(x_0)$
- $x_2 \equiv f(x_1)$
- etc., all (mod n).

For instance, doing it with $n = 8051$, we get

```
var('x')
@interact
def _(seed=2,n=8051,poly=x^2+1,trials=(10,[2..50])):
    f(x)=poly
    for i in range(trials):
        pretty_print(html("$x_{%s}=%s$"%(i,seed)))
        seed = (ZZ(f(seed)) % ZZ(n))
    pretty_print(html("$x_{%s}=%s$"%(i+1,seed)))
```

Now, these might be kind of random, but we will not actually try to find common divisors of these numbers with n.

Instead, we will try to see if all the *differences* $x_i - x_j$ share a common factor with n, using the (highly efficient to compute) gcd. That is a lot more opportunities! And hopefully it's just as (or more) 'random', and just as effective at finding factors.

However, that is too many to check quickly. So instead one modifies this one last time, with the following modification to the algorithm.

First, remember that polynomials are well-defined in modular arithmetic, and so the sequence of results will eventually repeat modulo any particular modulus d, since there are finitely many possible x_i. In particular, if d is a divisor of n, then if $\gcd(x_i - x_j, n) = d$ is a shared divisor found by the pair x_i and x_j, then not only will it be the case that

$$x_i \equiv x_j \pmod{d}$$

but it will also be the case for any number of iterations of f that

$$f^n(x_i) \equiv f^n(x_j) \pmod{d}$$

which means the sequence (modulo d, the common divisor) repeats itself every $j - i$ terms.

Let $k = j - i$ (or the first multiple of this which is greater than i); then the congruence

$$x_k \equiv x_{2k} \pmod{d}$$

will have to be true as well, so all the x_i, x_j pairs which come from the first one to share a divisor can be checked by checking just this *one* $\gcd(x_{2k} - x_k, n)$.

Algorithm 12.6.3 (Pollard Rho factoring algorithm)**.** *Follow these steps.*

- *Pick some polynomial $f(x)$ that will be easy to compute mod (n) (typically $x^2 + 1$).*
- *Plug in an essentially random seed value x_0. (Often the seed is 2 or 3.)*
- *Compute the polynomial's value at the seed, x_1.*
- *Continue plugging in $f(x_i) = x_{i+1}$, modulo n.*
- *For each k we check whether*

$$1 < gcd(x_{2k} - x_k, n) = d < n\,.$$

Proof. We will not formally prove this, as it does not always work. However, probabilistically (just like with Miller-Rabin) it should succeed for k in the neighborhood of the size of the square root of the smallest factor of n. So if n has a big, but not too big, divisor, this test should help us find that divisor. □

Example 12.6.4. In the example above, the numbers we would get for the first three rounds are

- $x_2 - x_1 = 26 - 5 = 21$
- $x_4 - x_2 = 7474 - 26 = 7448$
- $x_6 - x_3 = 871 - 677 = 194$

These factor as follows:

```
factor(21),factor(7448),factor(194)
```

and have the following gcds with 8051:

```
gcd(8051,21),gcd(8051,7448),gcd(8051,194)
```

So this would catch the four factors $3, 7, 19$, and 97, which is not bad, and indeed the final one caught a *common* divisor with 8051.

```
PollardRhoFactor(8051)
```

Remark 12.6.5. This method is usually called the Pollard rho method because it is due to John Pollard and because a very imaginative eye can interpret the x_i eventually repeating (mod d) (in the example, $d = 97$) as a tail and then a loop, i.e. a Greek ρ.

```
PollardRhoFactor(991*997)
```

Notice that sometimes the rho method doesn't come up with an answer quickly, or at all (it took 50 rounds without success here). I could up this to a lot more. So what do you do then – bring out the big guns? Not at all – just as with primality testing, you just change your starting point to try again!

```
PollardRhoFactor(991*997,seed=3)
```

In general, there are other such probabilistic algorithms, and they are quite successful with factoring numbers which might have reasonably sized but not gigantic factors. According to Wikipedia:

> The rho algorithm's most remarkable success has been the factorization of the eighth Fermat number by Pollard and Brent. They used Brent's variant of the algorithm, which found a previously unknown prime factor. The complete factorization of F_8 took, in total, 2 hours on a UNIVAC 1100/42.

Well, this was in the 70s. But still, it's not bad! Things don't automatically work quickly even now.

```
PollardRhoFactor(2^(2^8)+1,1000000) # one million rounds!
```

Hmm, what now? Let's change the seed.

```
PollardRhoFactor(2^(2^8)+1,1000000,seed=3)
```

Luckily, we have bigger guns at our disposal in Sage (especially in the component program Pari), that polish thing off rather more quickly.

```
factor(2^(2^8)+1)
```

A little better than two hours on a mainframe, or even on this computer, I hope you'll agree.

Sage note 12.6.6 (Building interacts)**.** Real factorization algorithms use several different methods to attack different types of factors. We can try to simulate this in a basic way by creating a Sage interact. Evaluate the first cell to define things, then the second one, which is the interact.

```
def TrialDivFactor(n):
    p = next_prime(1)
    top = ceil(math.sqrt(n))
    while p < top:
        if mod(n,p)==0:
            break
        p=next_prime(p)
    if n==1:
        print "1 is not prime"
    elif p==n:
        print n,"is prime"
    elif mod(n,p)==0:
        print n,"factors as",p,"times",n/p
    else:
        print n,"is prime"

def FermatFactor(n,verbose=False):
    if n%2==0:
        raise TypeError,"Input must be odd!"
    s=ceil(math.sqrt(n))
    top=(n+1)/2
    while is_square(s^2-n)==0:
        if verbose:
            print s,"squared minus",n,"is",s^2-n,"which is
                not a perfect square"
        s=s+1
    t=sqrt(s^2-n)
    print "Fermat found that",s,"squared minus",t,"squared
        equals",n
    if s^2==n:
        print "So",n,"was already a perfect
            square,",s,"times",s
    elif s<top:
        print
            "So",s+t,"times",s-t,"equals",(s-t)*(s+t),"which
            is",n
    elif s==top:
        print "So Fermat did not find a factor, which
            means",n,"is prime!"
```

```
@interact
def _(n=991*997,method=['trial','Fermat','Pollard Rho']):
    if method=='trial':
        TrialDivFactor(n)
```

```
    if method=='Fermat':
        FermatFactor(n)
    if method=='Pollard Rho':
        PollardRhoFactor(n)
```

An interact is just a Sage/Python function, except with `@interact` before it. There are many different input widgets you can use; this one demonstrates using a list and an input box which takes any input. See the interact documentation or Quickstart for many examples and more details.

If you think this sort of thing is cool, the Cunningham Project is a place to explore. I particularly like their Most Wanted lists. The idea is this.

> The Cunningham Project seeks to factor the numbers $b^n \pm 1$ for $b = 2, 3, 5, 6, 7, 10, 11, 12$, up to high powers n.

Another interesting resource is Sage developer Paul Zimmermann's Integer Factoring Records. Finally, Wagstaff's *The joy of factoring* [C.3.11] has *tons* of awesome examples and procedures – far too many, really, as well as an excellent discussion of how to speed up trial division etc.

12.7 Exercises

1. Check the multiplication needed in Lemma 12.1.2.

2. Prove the statement of Lemma 12.1.2 in the case that ℓ is odd.

3. Explain why the extension to Fermat's Little Theorem just before Fact 12.2.1 is true.

4. Check that 1729 and 2821 are Carmichael numbers.

5. Find a Carmichael number of the form $7 \cdot 23 \cdot p$ for a prime p.

6. Use either the Fermat or Mersenne coprime facts 12.1.4,12.1.8 to provide a different proof that there are infinitely many primes.

7. Pick some 4-6 digit numbers that don't share a factor with $30030 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$. Find factors by trial division (Algorithm 12.5.7).

8. Do the same with Fermat Factorization (Algorithm 12.5.10). Try to create a number that takes five steps with Fermat *and* with trial division.

9. Verify the last bit of the proof of The Fermat factorization algorithm.

10. Try using Pollard Rho on a large number you create out of a few big primes (not *too* big!) with different seeds. Can you get it to take longer than a few turns? Get your prize numbers; now try factoring again with this method where you have changed the polynomial to $x^3 + 1$ or something else other than $x^2 + 1$.

Chapter 13

Sums of Squares

We have now more or less exhausted a lot of what we can do with linear questions. With that in mind, we return to other considerations. As a warmup for this and ensuing chapters, consider the following question.

Question 13.0.1. Take a positive integer n and try to write it as $n = a^2 + b^2$ for $a, b \in \mathbb{Z}$. For which n is this possible, for which is it not?

It seems that Albert Girard already knew the answer to this question in the first quarter of the 17th century, and Fermat discovered it a couple years later as well. A full proof of the answer to this question did not come until Euler (no surprise here) about six score years later.

Remark 13.0.2. Girard is an interesting figure, less well-known than his contemporaries; he apparently was the first to use our modern notation for trigonometric functions, and spent his adult life in the Netherlands escaping religious persecution as a Protestant in France. Euler is well known for being a rather conventional religious family man amidst the Enlightenment court of Frederick the Great, and for taking a lot of teasing from Voltaire and the king (among other things, for being partly blind at the time). As with most things about Fermat's personal life, it's less well known that he also had a religious side; in [C.6.12] a well-known classicist translates a moving poem about the dying Christ written in honor of one of Fermat's friends.

So try out the question! Some things to think about while you try this:

- Are any special types of numbers easier to write in this way than others?
- Is there any way of generating new such numbers from old ones?
- If some types of numbers are *not* a sum of squares, how might you prove this?

A separate question to at least keep track of is this.

Question 13.0.3. *Assuming you can indeed write it in this way, how many ways* you can write a number as a sum of squares?

This chapter is completely devoted to continuing to address questions about writing numbers as a sum of two squares. It will lead us a little far afield, *of necessity*, to ask (and start to answer) questions about congruences again. Much of this chapter will be devoted to a *geometric* proof that certain numbers are indeed representable as as sum of two squares. This chapter is a perfect illustration of one of the main themes of this text – the *unity* of mathematics.

13.1 Some First Ideas

13.1.1 A first pattern

Let's assume you've done some exploration on your own. Here's a first pattern that you may have noticed, similarly to patterns in the past.

Fact 13.1.1. *If $n \equiv 3 \pmod 4$, then n is not writeable as a sum of squares.*

Proof. You should be able to prove this pretty easily based on things you already know about squares modulo 4. (See Exercise 13.7.1) □

The next thing to note is that Sage has a nice command to tell us an answer.

```
two_squares(29)
```

If a representation doesn't exist, we get an error. If it does, Sage returns two numbers (a, b) such that $a^2 + b^2 =$ your number.

In the next cell, I pick a number for which $n \equiv 1 \pmod 4$, but this number is *not* writeable. Thus Fact 13.1.1 doesn't just take care of all cases.

```
two_squares(21)
```

Fact 13.1.2. *There are $n \equiv 0, 1, 3 \pmod 4$ which are not representable as a sum of two squares.*

Proof. Show that 12, 21, and 6 are not. (See Exercise 13.7.2) □

You can use this interact to avoid the errors.

```
@interact
def _(n=29):
    try:
        a,b = two_squares(n)
        pretty_print(html("We can write
            ${0}={1}^2+{2}^2$".format(n,a,b)))
    except ValueError:
        pretty_print(html("${0}$ is not a sum of two
            squares".format(n)))
```

Sage note 13.1.3 (Handling errors)**.** Most computer languages have a way to "handle" errors if we don't want to think of them as errors. In Python, this is the try/except syntax you see above. Basically, we are trying to use the two squares command, but if it hiccups, we instead just print a nice message.

Remark 13.1.4. We have already addressed a very special case of writing numbers as a sum of squares. In fact, in Theorem 3.4.5 we saw a precise characterization of when a *perfect square* is a sum of two squares. We will mention this again briefly in Subsection 14.2.2.

13.1.2 Geometry

Next, we can interpret this question very differently, relying on our geometric intuition. This graphic helps us visualize the problem.

```
var('x,y')
@interact
def _(n=(5,range(100))):
    viewsize=ceil(math.sqrt(n))+2
    g(x,y)=x^2+y^2
    p = implicit_plot(g-n, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    lattice_pts = [[i,j] for i in [-viewsize..viewsize]
        for j in [-viewsize..viewsize]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
    curve_pts = [coords for coords in lattice_pts if
        g(coords[0],coords[1])==n]
    if len(curve_pts)==0:
        show(p+plot_lattice_pts, figsize = [5,5], xmin =
            -viewsize, xmax = viewsize, ymin = -viewsize,
            ymax = viewsize, aspect_ratio=1)
    else:
        plot_curve_pts = points(curve_pts, rgbcolor =
            (0,0,1),pointsize=20)
        show(p+plot_lattice_pts+plot_curve_pts, figsize =
            [5,5], xmin = -viewsize, xmax = viewsize, ymin
            = -viewsize, ymax = viewsize, aspect_ratio=1)
```

In this graph, $n = a^2 + b^2$, then n is the square of the radius of a circle which has (a, b) as the coordinates of a point. So the sum of squares problem is actually a geometric one!

That is, we can rewrite our questions like this.

Question 13.1.5.

- Which circles around the origin have lattice points, and which ones do not?
- If a circle has lattice points, how many does it have?

We will choose to address these questions by connecting to geometry. There are many ways; for instance, in Section 20.1 we will connect to calculus ideas in number theory.

13.1.3 Connections to some very old mathematics

The following identity was, separately, already known to Diophantus (remember Diophantine equations?) around 250, to Brahmagupta (about whom more later) around 600, and to Leonardo of Pisa (known also as Fibonacci) around 1250.

Fact 13.1.6 (Brahmagupta-Fibonacci identity)**.**

$$\left(a^2 + b^2\right)\left(c^2 + d^2\right) = (ac - bd)^2 + (ad + bc)^2$$

Proof. Multiply and cancel. □

This sort of identity may seem amazing to us, but to people used to needing lots of symbolic manipulation, it was just part of a toolkit by the time number theory began ascending with Fermat or Euler.

What is useful about this identity is that it implies the following.

Fact 13.1.7. Products *of numbers writeable as sums of squares may* also *be written as sums of squares!*

Proof. Use 13.1.6 above. □

```
@interact
def _(m=(13,[0..100]),n=(8,[0..100])):
    try:
        a,b = two_squares(m)
        c,d = two_squares(n)
        pretty_print(html("We␣know␣we␣can␣write␣
            ${6}={0}\cdot␣{1}$␣as␣
            $({2}^2+{3}^2)({4}^2+{5}^2)$".format(m, n, a,
            b, c, d, m*n)))
        pretty_print(html("But␣it␣is␣also␣writeable␣as␣
            $({0}\cdot{1}-{2}\cdot{3})^2␣+␣
            ({0}\cdot{3}+{1}\cdot{2})^2␣=␣
            {4}^2+{5}^2={6}$".format(a, c , b , d ,
            abs(a*c-b*d), a*d+b*c,m*n)))
    except ValueError:
        pretty_print(html("Please␣pick␣numbers␣that␣are␣
            both␣writeable␣as␣a␣sum␣of␣two␣squares"))
```

A final question for the reader is to ponder why this means that we can really reduce the question to whether *primes* are writeable as a sum of squares.

13.2 Primes Can Be Written in at Most One Way

Most of the rest of this chapter is dedicated to proving what we can about how to write numbers as sums of squares. We will begin our proofs by talking about *how many* ways we can write some numbers as a sum of squares. Namely, we'll connect sums of squares to factorization.

Remember that the Brahmagupta-Fibonacci identity says that if two numbers are in sums of two squares, so is their product. Remarkably, we can sort of do this backwards.

First, we need to say what we might mean as writing a number as a sum of squares in two essentially different ways. Compare

$$13 = 3^2 + 2^2 = 2^2 + 3^2$$

to the situation

$$25 = 5^2 + 0^2 = 3^2 + 4^2$$

We say the latter ways are **essentially different**. Now we see how to

Fact 13.2.1. *If an odd number N is writeable in two essentially different (*nonnegative*) ways as a sum of two squares, then $N = yz$, where $y, z > 1$ and y, z are themselves writeable as two squares' sum.*

Proof. Assume first that

$$N = a^2 + b^2 = c^2 + d^2$$

with a, c odd and b, d even. Then let

$$k = \gcd(a - c, d - b) \text{ and } n = \gcd(a + c, d + b) \text{ (both are even)}$$

and

$$\ell = \frac{a-c}{k} = \frac{d+b}{n} \text{ and } m = \frac{a+c}{n} = \frac{d-b}{k}\,.$$

Then we get that

$$N = \left[\left(\frac{k}{2}\right)^2 + \left(\frac{n}{2}\right)^2\right](m^2 + \ell^2)$$

There are some details here, especially in terms of verifying all these numbers exist, but they mostly just use the definitions of gcd and parity. See Exercise Group 13.7.8–13.7.10 □

Example 13.2.2. For $N = 25$, what are a, b, c, d?

Then $k = \gcd(2, 4) = 2$, $n = \gcd(8, 4) = 4$ which means $\ell = 1$ and $m = 2$, yielding

$$25 = \left[\left(\frac{2}{2}\right)^2 + \left(\frac{4}{2}\right)^2\right](1^2 + 2^2) = 5 \cdot 5$$

And so 25 is a product of numbers themselves writeable as a sum of two squares.

It is now nearly trivial to prove the following.

Proposition 13.2.3. *A prime is writeable in zero or one (positive) way as a sum of two squares.*

Proof. This is clear for $p = 2$, and if $p \equiv 3 \pmod 4$ this is part of Fact 13.1.1.

It remains to consider the case of $p \equiv 1 \pmod 4$.

- On the one hand, if p is writeable in two different ways, it factors.
- But prime numbers don't factor nontrivially.
- So there is just one way to do it.

□

We can see this visually below, in that each of the circles with radius square root a prime either has no lattice points, or has only positive lattice points that are (a, b) and (b, a) for one a and b.

```
var('x,y')
@interact
def _(n=(5,prime_range(150))):
    viewsize=ceil(math.sqrt(n))+.5
    g(x,y)=x^2+y^2
    p = implicit_plot(g-n, (-1,viewsize), (-1,viewsize),
        plot_points = 100)
    lattice_pts = [[i,j] for i in [-1..viewsize] for j in
        [-1..viewsize]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
```

```
    curve_pts = [coords for coords in lattice_pts if
        g(coords[0],coords[1])==n]
    if len(curve_pts)==0:
        show(p+plot_lattice_pts, figsize = [5,5], xmin =
            -1, xmax = viewsize, ymin = -1, ymax =
            viewsize, aspect_ratio=1)
    else:
        plot_curve_pts = points(curve_pts, rgbcolor =
            (0,0,1),pointsize=20)
        show(p+plot_lattice_pts+plot_curve_pts, figsize =
            [5,5], xmin = -1, xmax = viewsize, ymin = -1,
            ymax = viewsize, aspect_ratio=1)
```

13.3 A Lemma About Square Roots Modulo n

We'll continue our formal investigation of what numbers are in sums of two squares by taking a look at a lemma seemingly unrelated to this. Later we'll see that square roots of negative one in $\mathbb{Z}$ (not $\mathbb{Z}_n$) are connected to sums of squares as well, so this is not a completely implausible connection.

Before we do this, let's codify something we already have discussed, e.g. in Fact 7.3.1 or Section 7.6.

Definition 13.3.1. We say that a number a has a **square root modulo** n if there is some number x with

$$x^2 \equiv a \pmod{n} .$$

As an example, here is an alternate proof of Exercise 7.7.10.

Fact 13.3.2. *For an odd prime p, the only way there is a square root of -1 modulo p is if $p \equiv 1 \pmod 4$.*

Proof. We will use group theory to prove this.

Assume there *is* a square root, so that

$$u^2 \equiv -1 \pmod{p} .$$

Then the order of u in U_p is four, since

$$u^4 = (u^2)^2 \equiv (-1)^2 = 1 .$$

We know that the order

$$| U_p | = p - 1$$

but then Lagrange's (group theory) Theorem 8.3.11 says that four divides $p-1$.

Given that, the only possible kind of prime p solving this is the form $4k + 1$. □

Remember, this means there can't be a square root of minus one if $p \equiv 3 \pmod 4$. Of course, it also only means that there *might* be one if $p \equiv 1 \pmod 4$, so we certainly need the following lemma to confirm there *is* one. (See its use in Subsection 16.1.1, where we combine everything into Fact 16.1.2.)

Lemma 13.3.3. *If $p \equiv 1 \pmod 4$, then there actually does exist a square root of -1 modulo p. That is, there is a u such that*

$$u^2 \equiv -1 \pmod{p} .$$

Proof. Before we start the proof, recall Theorem 7.5.1, that $(p-1)! \equiv -1 \pmod{p}$ for primes. Do you remember our proof? We *paired up* all the numbers from 2 to $p-2$ in pairs of multiplicative inverses (mod p), thus:

$$(p-1)! = 1 \cdot 2 \cdot 2^{-1} \cdot 3 \cdot 3^{-1} \cdots (p-1) \equiv (p-1) \equiv -1 \pmod{p}.$$

Our strategy for this proof will be similar.

Assume Wilson's Theorem is true. Now pair up the numbers from 1 to $p-1$ in a *different* way, in pairs of additive inverses (mod p):

$$(p-1)! = 1 \cdot (p-1) \cdot 2 \cdot (p-2) \cdot 3 \cdot (p-3) \cdots \frac{p-1}{2} \cdot \frac{p+1}{2} =$$

$$\left[1 \cdot 2 \cdot 3 \cdots \frac{p-1}{2}\right] \cdot \left[(p-1) \cdot (p-2) \cdots \frac{p+1}{2}\right].$$

This makes sense because $(p-1)/2$ is an integer halfway between 1 and p, as p is odd.

If we rethink things (mod p), we can rewrite this in a more suggestive way.

- Let $\left(1 \cdot 2 \cdot 3 \cdots \frac{p-1}{2}\right)$ be called f (this is also $\left(\frac{p-1}{2}\right)!$, of course).
- Then

$$\left[1 \cdot 2 \cdot 3 \cdots \frac{p-1}{2}\right] \cdot \left[(p-1) \cdot (p-2) \cdots \frac{p+1}{2}\right]$$

$$\equiv f \cdot \left[-1 \cdot -2 \cdot -3 \cdot -\frac{p-1}{2}\right]$$

$$\equiv f \cdot (-1)^{\frac{p-1}{2}} \left[1 \cdot 2 \cdot 3 \cdots \frac{p-1}{2}\right] \equiv (-1)^{\frac{p-1}{2}} f^2.$$

Remember that our hypothesis is $p \equiv 1 \pmod{4}$. Then $p = 4k+1$, so $\frac{p-1}{2} = 2k$ is even, which means

$$-1 \equiv f^2 \text{ or } f^2 \equiv -1 \pmod{p}$$

What is neat about this proof is that it shows there are precisely *two* square roots of negative one (as Lagrange's (polynomial) Theorem 7.4.1 suggests), and we even have a formula for them:

$$\pm\left(\frac{p-1}{2}\right)!$$

□

Somehow this is a satisfying answer. We can check that these really are square roots of -1 using Sage.

```
@interact
def _(p=(13,[q for q in prime_range(200) if q%4==1])):
    k=mod(factorial((p-1)/2),p)
    pretty_print(html("The potential square roots of $-1$
        are $\pm
        \left(\\frac{%s-1}{2}\\right)!=%s,%s\\text{ (mod
        }%s)$"%(p,k,-k,p)))
    pretty_print(html("And we can compute that
        ${0}^2\equiv{1}$ and ${2}^2\equiv{3}$ modulo
        ${4}$".format(k,k^2,-k,(-k)^2,p)))
```

13.4 Primes as Sum of Squares

In the past few sections, one of the many things you may have conjectured about sums of squares is that *every* prime of the form $p = 4k + 1$ can be represented as the sum of two squares. (We discussed why limiting the question to primes was sufficient.) It turns out this is *true*, and we will spend most of the remainder of this chapter proving it (in the manner of [C.1.1, Chapter 10.6], though expanded greatly to avoid any direct reference to Minkowski's Theorem). At the end of the chapter, we'll combine it with the observation about primes of the form $p = 4k + 3$ to see exactly which numbers can be thus represented.

13.4.1 A useful plot

First, let's look at the following plot on the integer lattice. As you can see, I am plotting certain points on the circle $x^2 + y^2 = n$, with $n = 5$ to begin. I have done some 'magic' to turn the square root of $-1 \pmod n$ into these points. Before telling you the magic, this graphic will help us get ready to see it.

```
var('x,y')
@interact
def _(p=(5,[q for q in prime_range(200) if q%4==1])):
    u=mod(factorial((p-1)/2),p)
    viewsize=ceil(math.sqrt(p))+2
    g(x,y)=x^2+y^2
    plot1 = implicit_plot(g-p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[1]-u*coords[0])%p==0]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
        [5,5], xmin = -viewsize, xmax = viewsize, ymin =
        -viewsize, ymax = viewsize, aspect_ratio=1)
```

To be precise, I've used this square root of -1 to create the regularly spaced grid of blue points. You can think about it as a bunch of corners of parallelograms.

Remark 13.4.1. Sometimes we generically call things like the set of blue dots a **lattice**, though we will usually use the word lattice only to refer to the integer lattice of the black dots. A general lattice is something related to a concept from linear algebra – vectors generated by a basis, except instead of being vectors over $\mathbb{Q}$ or $\mathbb{R}$, they are over $\mathbb{Z}$.

Here is how we constructed the blue grid.

- Assume that p is our prime and $k = \left(\frac{p-1}{2}\right)!$ is our square root of negative one.
- The blue points all are of the form $(ak + bp, a)$ for all integers a, b.

For one final preliminary, let's define one more thing for any old point (x, y) in the integer lattice (and especially for our blue dots).

Definition 13.4.2. We call the **norm** of a point (x, y) the sum of squares, $N(x, y) = x^2 + y^2$.

13.4.2 Primes which are sums of squares

We are now ready to state our big theorem for the section. (See Fact 14.1.6 for a quite different proof.)

Theorem 13.4.3. *Every prime p of the form $4k + 1$ can be written as a sum of squares.*

Proof. The proof is fairly long. Here is the strategy; the first step will be detailed in Subsection 13.4.3 and Subsection 13.4.4.

Suppose we find some blue dot $(ak + bp, a)$ such that

$$0 < N(ak + bp, a) = a^2 + (ak + bp)^2 < 2p\,.$$

Then we know, modulo p, that

$$N(ak+bp, a) = a^2+(ak+bp)^2 \equiv a^2+(ak)^2 \equiv a^2+a^2k^2 \equiv a^2-a^2 \equiv 0 \pmod{p}\,,$$

so p in fact divides the norm of the point $(ak + bp, a)$.

So we have that $0 < a^2 + (ak + bp)^2 < 2p$ *and* that $p \mid a^2 + (ak + bp)^2$, meaning the only possibility is that $p = a^2+(ak+bp)^2$, which gives p explicitly written as a sum of squares. □

Example 13.4.4. For instance, with $p = 5$, we have that $k = \left(\frac{5-1}{2}\right)! = 2! = 2$, so we need to find a point $(a, 2a + 5b)$ such that $a^2 + (2a + 5b)^2 < 2p$. Guess and check with $a = 1$ and $b = 0$ gives us

$$N(1, 2 \cdot 1 + 5 \cdot 0) = 1^2 + (2 \cdot 1 + 5 \cdot 0)^2 = 5 < 2 \cdot 5 = 10$$

so this point should work, and this does give the correct statement that

$$5 = 1^2 + 2^2\,.$$

What remains to be shown is that there actually *is* such a blue dot.

13.4.3 Visualizing the proof

To prove the theorem that for any $p = 4k + 1$ we can write it as a sum of squares, we need to prove there is a blue dot (somewhere) that is not at the origin but also has norm smaller than $2p$. We will prove this by heavy reference to graphics, but all claims also make sense algebraically. Sometimes we need help to be able to think about more involved proofs.

```
@interact
def _(p=(5,[q for q in prime_range(200) if q%4==1])):
    u=mod(factorial((p-1)/2),p)
    viewsize=floor(sqrt(2*p))+2
    g(x,y)=x^2+y^2
    plot1 = implicit_plot(g-p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    plot2 = implicit_plot(g-2*p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
```

```
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[1]-u*coords[0])%p==0]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=10)
    show(plot1+plot2+plot_grid_pts+plot_lattice_pts,
        figsize = [5,5], xmin = -viewsize, xmax = viewsize,
        ymin = -viewsize, ymax = viewsize, aspect_ratio=1)
```

We include a variation on the graphic to make this visually clear. The bigger circle is the one we care about now – it has formula $x^2 + y^2 = 2p$, so radius $\sqrt{2p}$. If we find a blue point inside that circle but not at the origin, then the argument in the proof sketch shows it must be on the smaller circle.

Very strangely, the best way to do this is by considering the *areas* of the various circles, and showing that they are so big you just *must* have a blue point in it (but not at the origin). Let's see how this works.

The area of the bigger circle, which has radius $\sqrt{2p}$, is $\pi(\sqrt{2p})^2 = 2\pi p$. Since $\pi > 2$, we have that $2\pi > 2(2) = 4$, which mean that the area of the bigger circle is bigger than $4p$.

```
@interact
def _(p=(5,[q for q in prime_range(200) if
    q%4==1]),triangles_on=False):
    u=mod(factorial((p-1)/2),p)
    viewsize=2*p
    g(x,y)=x^2+y^2
    plot1 = implicit_plot(g-p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    plot2 = implicit_plot(g-2*p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    plot3 = line([[0,0], [2*p-2*Integer(u),2], [2*p,0],
        [2*Integer(u),-2], [0,0]], rgbcolor=(1,0,0))
    plot4 = line2d([[0,0], [2*p,0]], rgbcolor=(1,0,0),
        linestyle='--')
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[1]-u*coords[0])%p==0]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=10)
    plot_lattice_pts2 = points([[2*coords[0],2*coords[1]]
        for coords in lattice_pts], rgbcolor =
        (0,1,0),pointsize=20)
    if triangles_on:
        show(plot1+plot2+plot3+plot4 + plot_grid_pts +
            plot_lattice_pts+plot_lattice_pts2, xmin =
            -viewsize/2, xmax = viewsize, ymin =
            -viewsize/2, ymax = viewsize/2, aspect_ratio=1)
    else:
        show(plot1+plot2+plot_grid_pts + plot_lattice_pts
            + plot_lattice_pts2, xmin = -viewsize/2, xmax =
            viewsize, ymin = -viewsize/2, ymax =
            viewsize/2, aspect_ratio=1)
```

What we do now is to create a **sublattice** of the blue dots, which we will color green. (This is just a subset of a lattice which still otherwise satisfies the conditions for being a lattice.)

To create the green sublattice, take all blue dots, and just double their coordinates. (Naturally, each green dot is still a blue dot.)

Next, we take a look at the triangles made by the different colored dots. (You can click on `triangles_on` in the interact above to see them in red.) Compare the *thinnest* such triangles.

- The thinnest triangle made by *blue* dots would be from the origin and the points $(p, 0)$ (with $a = 0, b = 1$) and $(k, 1)$ (with $a = 1, b = 0$).
- The thinnest triangle made by the *green* dots has width $2p$ (from the origin to $(2p, 0)$, the previous point doubled) and height 2 (to the point $(2k, 2)$, which is $(k, 1)$ doubled).

The green dot triangle has area $4p/2$ – so the parallelogram with the solid red lines made of two of them has area $4p$. This area means it is *smaller* than the bigger circle.

This proof is very visual, so before we move on, make sure you believe all of this. Then we will analyze the exact areas involved more closely to finish. Remember, we are trying to prove that there is a blue point inside the bigger blue circle, but away from the origin.

13.4.4 Finishing the proof

Let's take stock.

- We've created circles of various sizes to find points in, and two lattices to examine.
- The area of the circle is *more* than the area ($4p$) of the smallest parallelogram made by green dots.
- Because all points inside the parallelogram (not just green, blue, or lattice points) will repeat outside of it in another parallelogram, $4p$ is the biggest area you can have and *not* repeat some point.
- So, the circle, having a bigger area, must have two points (not necessarily blue points, just points on the plane) which are repeated by the shifting of this parallelogram (called a **fundamental region**).

This may sound a little suspicious, so let's be sure about it.

Claim 13.4.5. *The circle has two points of some kind repeated by shifting the fundamental region.*

Proof. Call the parallelogram in red L. The circle is composed of all the pieces of the circle which lie in different parallelograms (comprised of green dots) congruent to L.

Let's 'move' the pieces in the circle to the corresponding part of L. Now suppose there are *not* two points which are repeated in the big circle. Then this movement of pieces of the circle to L is one-to-one.

If that movement is one-to-one, then the pieces of the circle must at most fill up L, but the circle has a *bigger* area than L! This is not possible, since moving doesn't change area, and thus there *are* two points which are repeated. □

Now let's continue the proof of the main Theorem 13.4.3. To start, take two points that are repeated in the circle; call them v and w. Then if we consider the points as vectors, $v - w$ is *itself* a green point, since the difference one shifts them by must be one of the obvious directions of the parallelogram in order to be a repeat.

By the construction of the green points, that means $(v - w)/2$ is a *blue* point. This point can't be the origin, since v and w are different!

Further, this blue point is inside the big circle (so its norm is less than $2p$). Why?

- Since the circle is nicely symmetric about the origin, the point $-w$ is also in the circle.
- The midpoint of the line segment connecting v and $-w$, both points in the big circle, is in fact $(v - w)/2 = \frac{v+(-w)}{2}$.
- Circles are convex, so this blue point being between v $-w$ means it's in the big circle. So we have found a blue point other than the origin in the blue circle.

Here is the picture of how to find the blue point in the circle. The black points are v, w, and $-w$, and you see the midpoint of the line is indeed blue.

```
@interact
def _(p=(5,[q for q in prime_range(200) if q%4==1])):
    u=Integer(mod(factorial((p-1)/2),p))
    big = math.floor(math.sqrt(2*p))
    viewsize=2*p
    g(x,y)=x^2+y^2
    plot1 = implicit_plot(g-p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    plot2 = implicit_plot(g-2*p, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    plot3 = line([[0,0], [2*p-2*u,2], [2*p,0], [2*u,-2],
        [0,0]], rgbcolor=(1,0,0))
    plot4 = line2d([[0,0],[2*p,0]], rgbcolor=(1,0,0),
        linestyle='--')
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[1]-u*coords[0])%p==0]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=10)
    big_lattice_pts = [[2*coords[0],2*coords[1]] for
        coords in lattice_pts]
    plot_lattice_pts2 = points(big_lattice_pts, rgbcolor =
        (0,1,0),pointsize=20)
    w = []
    v = []
    mw = []
    for i in [1..2*p-1]:
        for coords in [l for l in big_lattice_pts if
            l!=[0,0]]:
            if (i+coords[0])^2+(coords[1]-1)^2 < 2*p:
                for coords2 in [k for k in big_lattice_pts
                    if k!=[0,0] and k!=coords]:
```

```
                    if (i+coords2[0])^2 + (coords2[1]-1)^2
                        < 2*p:
                        w = [i+coords[0],coords[1]-1]
                        v = [i+coords2[0],coords2[1]-1]
                        mw = [-a for a in w]
                        break
            if w: break
        if w: break
    if not v:
        for i in [j for j in [u..p+u]]:
            for coords in [l for l in big_lattice_pts if
                l!=[0,0]]:
                if (i+coords[0])^2+(coords[1]-1)^2 < 2*p:
                    for coords2 in [k for k in
                        big_lattice_pts if k!=[0,0] and
                        k!=coords]:
                        if (i+coords2[0])^2 +
                            (coords2[1]-1)^2 < 2*p:
                            w = [i+coords[0],coords[1]-1]
                            v = [i+coords2[0],coords2[1]-1]
                            mw = [-a for a in w]
                            break
                if w: break
            if w: break
    if not v:
        print [p-u..2*p-u]
        for i in [j for j in [p-u..2*p-u]]:
            if i==6: print i
            for coords in [l for l in big_lattice_pts if
                l!=[0,0]]:
                if coords==[-4,2] and i==6: print coords
                if (i+coords[0])^2+(coords[1]+1)^2 < 2*p:
                    for coords2 in [k for k in
                        big_lattice_pts if k!=[0,0] and
                        k!=coords]:
                        if coords2==[-2,-4] and i==6 and
                            coords==[-4,2]: print coords2
                        if
                            (i+coords2[0])^2+(coords2[1]+1)^2
                            < 2*p:
                            w = [i+coords[0],coords[1]+1]
                            v = [i+coords2[0],coords2[1]+1]
                            mw = [-a for a in w]
                            break
                if w: break
            if w: break
    P1=point(v,pointsize=20,rgbcolor=(0,0,0))
    P2=point(w,pointsize=20,rgbcolor=(0,0,0))
    Z=point(mw,pointsize=20,rgbcolor=(0,0,0))
    plot5 = line([v,mw],rgbcolor=(0,0,0))
    plot6 =
        point(((v[0]+mw[0])/2,(v[1]+mw[1])/2),pointsize=20)
    show(plot1+plot2+plot3+plot4 + P1+P2+Z+plot5+plot6 +
        plot_grid_pts + plot_lattice_pts +
        plot_lattice_pts2, figsize = [5,5], xmin =
        -viewsize/2, xmax = viewsize, ymin = -viewsize/2,
        ymax = viewsize/2, aspect_ratio=1)
```

Sage note 13.4.6 (Examining code is good for you)**.** This is *by far* the longest code we've seen up to this point. It is a brute force check of all movements of all points in the parallelogram to find two points in the bigger circle. Can you think of ways to make it more efficient?

Believe it or not, we've concluded the proof – whew!
Why was this so hard? I can think of three reasons.

- First, we are trying to prove something about squares by proving something about square roots. It works, but it means there will be many steps.
- Secondly, we are not just algebraically proving it exists by solving an equation; we are forced to prove our square root exists with *inequalities*, which brings another set of complication.
- Third, we are looking not just at any old inequalities, but truly geometric ones, and so we must gain insight that way – worthwhile, but stretching.

Remark 13.4.7. Many more theorems of this kind can be proved using these techniques – the names of Minkowski and Blichfeldt show up in further generality, which we are intentionally avoiding. Those who have had some physics may have heard of Minkowski before, as his work nearly beat Einstein to the notion of special relativity.

13.5 All the Squares Fit to be Summed

There is one loose end. What are *all* the numbers we can represent as a sum of squares?

For instance, why are some composite numbers of the form $4k+1$ *not* writeable as the sum of two squares? Also, many even numbers are representable – how do we tell *which* even numbers are writeable? We conclude our discussion by proving the full statement, after a couple of preliminary lemmas.

Lemma 13.5.1. *If N has only primes of the form $4k+1$ and 2 as factors, it is writeable as a sum of two squares.*

Proof. Each of those primes is representable, so we can use Brahmagupta-Fibonacci identity to write the intermediate products that way; hence all such products are representable. □

Example 13.5.2. Consider this:

$$2 \cdot 13 \cdot 17 = 442 = \left(1^2+1^2\right)\left(3^2+2^2\right) \cdot 17$$

$$= \left[(1 \cdot 3 - 1 \cdot 2)^2 + (1 \cdot 2 + 1 \cdot 3)^2\right]\left(4^2+1^2\right)$$

$$= \left(1^2+5^2\right)\left(4^2+1^2\right) = (1 \cdot 4 - 5 \cdot 1)^2 + (1 \cdot 1 + 5 \cdot 4)^2 = 1^2 + 21^2$$

Lemma 13.5.3. *If N only has prime factors of the form $4k+3$ to even powers, it is writeable as a sum of two squares.*

Proof. First, p^2 is trivially always representable, since $p^2 = p^2 + 0^2$. Now, rather than using Fact 13.1.6, simply multiply this by N/p^2 (which itself must be a perfect square). □

Example 13.5.4. Consider this:

$$35802 = 442 \cdot 3^4 = \left(1^2 + 21^2\right) 3^2 \cdot 3^2$$

$$= 1^2 \cdot 3^2 \cdot 3^2 + 21^2 \cdot 3^2 \cdot 3^2 = 9^2 + 189^2$$

Theorem 13.5.5. *N can be written as a sum of two perfect squares precisely if it has only even powers (including zeroth powers) of any primes of the form $4k+3$.*

Proof. From 13.5.1 and 13.5.3, the only case left to consider if N has a prime of the form $p = 4k+3$, but to an odd power. This seemed to be the bottleneck in our exploration.

By way of contradiction, suppose that it is possible to write

$$N = a^2 + b^2 \, .$$

First, divide this equation by any factors of p common to both sides to get

$$M = c^2 + d^2$$

The power of p we divided by (so that $N = Mp^k$) must be an even power, since each term on the right-hand side is a perfect square and can only contribute even powers of primes by the Fundamental Theorem of Arithmetic.

Since N had an odd power of p, we know M still has an odd power of p dividing it, yet $p \nmid c, d$.

Take everything modulo p to get the congruence

$$0 \equiv c^2 + d^2 \pmod{p} \, .$$

Since $p \nmid c$, we can multiply this congruence by $\left(c^{-1}\right)^2$ to get

$$0 \equiv 1 + \left(c^{-1}\right)^2 d^2 \Rightarrow -1 \equiv \left(c^{-1} d\right)^2 \pmod{p}$$

This is a contradiction, as there is no square root of -1 modulo p by Fact 13.3.2! □

If this still seems too neat and dried, it can be instructive to get insight by plugging in different n below. When do you get an error, when not?

```
n=20
factor(n), two_squares(n)
```

(As a bonus, can you turn this into an interactive cell? See Sage note 12.6.6.)

13.6 A One-Sentence Proof

There is a completely different approach to this problem which has gained some notoriety. Often one wants multiple approaches in order to understand a problem more deeply; here, we have picked a geometric approach.

It happens that D. Zagier provided the culmination of a series of proofs using only sets and functions, and that proof takes *only one sentence* to write down! This is reproduced from the famous article [C.6.2] with the following title:

Proposition 13.6.1 (A One-Sentence Proof that Every Prime $p \equiv 1 \pmod 4$ is a Sum of Two Squares)**.**

Proof. The involution on the finite set

$$S = \{(x, y, z) \in \mathbb{N}^3 \mid x^2 + 4yz = p\}$$

defined by

$$(x, y, z) \to \begin{cases} (x + 2z, z, y - x - z) & \text{if } x < y - z \\ (2y - x, y, x - y + z) & \text{if } y - z < x < 2y \\ (x - 2y, x - y + z, y) & \text{if } x > 2y \end{cases}$$

has exactly one fixed point, so $|S|$ is odd and the involution defined by $(x, y, z) \to (x, z, y)$ also has a fixed point. □

In Exercise Group 13.7.17–13.7.21, you will be asked to verify the various statements that this proof depends on. Although perhaps it is not the easiest single sentence after all, it is still fun – fun enough that you can watch a couple videos about it from Numberphile!

13.7 Exercises

1. Prove that if $n \equiv 3 \pmod 4$, then n cannot be written as a sum of two squares (13.1.1).

2. Prove Fact 13.1.2.

3. Show that if $n \equiv 7 \pmod 8$, then n cannot be written as a sum of *three* perfect squares.

4. Find two numbers that can be written as a sum of three squares in two different ways (where different really means different, not $1^2 + 0^2 + 0^2 = 0^2 + 1^2 + 0^2$).

5. Find as many integers n as possible which are only writeable as a sum of squares via $n = a^2 + a^2 = 2a^2$, i.e. n is not writeable as a sum of *distinct* squares.

6. Verify Fact 13.1.6 by hand (i.e. write all the algebra out).

7. Let $r_2(n)$ be the number of different ways to write n as a sum of two squares, where *every* different way (not just essentially different) is counted. For instance,

$$r_2(2) = 4 \text{ because } (-1, 1), (-1, -1), (1, 1), (1, -1) \text{ all work.}$$

Prove that

$$r_2\left(2^m\right) = 4 \text{ for all } m\ .$$

Let N be odd, and let $N = a^2 + b^2$ and $n = c^2 + d^2$, where the pairs (a, b) and (c, d) are both positive and not the same or just switched in order. Verify the following to finish the proof of Fact 13.2.1.

8. It's okay to assume that a and c are odd and b and d are even.

9. If this is the case, show that $k = \gcd(a-c, b-d)$ and $n = \gcd(a+c, b+d)$ are both even.

10. Assuming the previous two exercises, show that $\frac{a-c}{k} = \frac{b+d}{n}$ and $\frac{b-d}{k} = \frac{a+c}{n}$.

Pick four random (to you) three digit numbers which are *not* of the form $4k+3$.

11. Decide whether these numbers are a sum of two squares without using Sage.

12. Pick two of those numbers and write them in all possible ways as a sum of two squares.

13. Show a positive integer k is the *difference* of two squares if and only if $k \not\equiv 2 \pmod 4$.

14. Prove that if $n \equiv 12 \pmod{16}$, show that n cannot be written as a sum of two squares.

15. Is there any congruence condition modulo 6 for which a number cannot be written as a sum of two squares?

16. Referring to the proof of the main theorem: Check that the pictures you get from some other primes with these lattices really work.

Check every piece of the Zagier proof (Proposition 13.6.1).

17. The set S is finite. Try figuring out what S is for $p = 5$ or $p = 13$, the smallest such primes.

18. Each (x, y, z) has exactly one of the three things to go to.

19. The function in question is an **involution**. That is, if you take the output and apply the function a second time, you get your original (x, y, z) back (this is a little tougher).

20. If (x, y, z) goes to (x, y, z) then it turns out that $(x, y, z) = (1, 1, \frac{p-1}{4})$ (you will probably need to use the definition of S for this, and remember that we assume $p \equiv 1 \pmod 4$).

21. That if the map $(x, y, z) \to (x, z, y)$ has a point which is fixed (the output is same as input) then this, combined with the definition of S, means that p is writeable as the sum of two squares.

Chapter 14

Beyond Sums of Squares

There are many fascinating topics that sums of squares connect to. This chapter gives some interesting points of view on several.

14.1 A Complex Situation

14.1.1 A new interpretation

Let's see another to interpret sums of squares. Suppose first that, as before,

$$n = a^2 + b^2 \ .$$

Then, if we let the symbol i stand for a (putative) square root of negative one, so that $-1 = i^2$, we could legitimately factor the equation:

$$n = a^2 - (i^2 b^2) = (a + bi)(a - bi)$$

Example 14.1.1. For instance, we could *factor* the prime number thirteen!!!

```
print 3^2+2^2; print (3+2*i)*(3-2*i)
```

It turns out that there is a beautiful connection between the theory of numbers representable as a sum of two squares and the following beautiful definition.

Definition 14.1.2. The **Gaussian Integers** $\mathbb{Z}[i]$ may be defined as the set

$$\mathbb{Z}[i] = \{a + bi \mid a, b \in \mathbb{Z}\}$$

This does assume that we can have such a symbol i with $i^2 = -1$.

Remark 14.1.3. These are named after C.F. Gauss, who explored them a great deal, though others were at least incipiently aware of them. There are so many stories about Gauss that one can hardly know where to begin, and he will come up again when we continue exploring prime numbers in Section 21.2; perhaps most relevant to our work is that he actually published about Gaussian integers!

If we bring back our lattice of integer points, we can think of such numbers as being points on the lattice, where the coordinate point $(3, 2)$ corresponds to $3 + 2i$, one of the 'factors' of 13. I'll plot both 'factors' below.

```
lattice_pts = [[i,j] for i in [-5..5] for j in [-5..5]]
plot_lattice_pts =
    points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
show(plot_lattice_pts + point([3,2],pointsize=30) +
    point([3,-2],pointsize=30),aspect_ratio=1)
```

There are many amazing questions to ask about this, and wonderful connections to abstract algebra. For example, the factorization of $a^2 + b^2$ yielding i, a "square root of negative one" over the integers, should mean we aren't surprised that there is a connection with "square roots modulo n". More to our purposes, there is a direct connection to writing numbers as a sum of squares, and we will show that it can be done more directly in the next section.

14.1.2 Revisiting the norm

How can we decide whether the verb "factor" becomes the legitimate word factor in such a number system? The reason we can is because *prime* numbers can be defined for this new system as well.

Fact 14.1.4. *Prime numbers in the Gaussian integers are of the following three possible forms:*

- *The form $p = 4n + 3$ for a (normal) integer prime*
- *The form pi for p of the same form*
- *The factors with respect to i of the (normal) primes of the form $p = 4n+1$ and $p = 2$*

We know the latter exist because we can write them as the sum of two squares.

Viewing these so-called **Gaussian primes** is fun. Many authors have created beatiful graphics such as this.

```
@interact
def _(viewsize=10):
    lattice_pts = [[i,j] for i in [-viewsize..viewsize]
        for j in [-viewsize..viewsize]]
    plot_lattice_pts =
        points(lattice_pts,rgbcolor=(0,0,0),pointsize=2)
    GG.<I> = GaussianIntegers()
    Gaussian_primes = [ x for x in lattice_pts if
        GG(x[0]+x[1]*I).is_prime() ]
    plot_Gaussian_primes =
        sum([polygon([(G[0]+1/2,G[1]+1/2),
        (G[0]+1/2,G[1]-1/2), (G[0]-1/2,G[1]-1/2),
        (G[0]-1/2,G[1]+1/2)],alpha=.6) for G in
        Gaussian_primes])
    show(plot_Gaussian_primes+plot_lattice_pts,
        aspect_ratio=1)
    pretty_print(html("Plot␣of␣Gaussian␣primes␣with␣
        coordinates␣less␣than␣{0}␣in␣absolute␣
        value".format(viewsize)))
```

The basic reason this even makes sense is that we can use the Euclidean algorithm here. First, let's use the *same definition of norm* as we used earlier for the points, so that $N(x + iy) = x^2 + y^2$.

Example 14.1.5. The norm of $3 + 2i$ is $3^2 + 2^2 = 13$ while the norm of $13 = 13 + 0i$ is 169.

The difference is that instead of saying simply that $a = bq + r$ for $r < b$, we will need to compare the *norms* of r and b. Namely, you can write two Gaussian integers a and b as $a = bq + r$, where $0 \leq N(r) < N(b)$. Continue this process just as in Euclidean algorithm, and it ends by the Well-Ordering Principle to define $\gcd(a, b)$. In this case ± 1 and $\pm i$ are all possible stopping points if a and b don't share a factor.

Further, if g and h are "relatively prime" Gaussian integers ($\gcd(g, h) = \pm 1$ or $\pm i$), then there are other such integers x and y such that $gx + hy = 1$. So we have a Bezout identity as well to play with.

Computing with Gaussian integers this way is possible in Sage.

```
ZZI.<I> = GaussianIntegers()
(1+I).is_prime()
```

Crucially, I am skipping whether we actually have **unique factorization** in $\mathbb{Z}[i]$; this is true, and used below in Fact 14.1.6, but properly belongs in an algebra course.

14.1.3 A different approach to sums of squares

This allows a quite different approach to the fact primes of the form $4n + 1$ can be written as a sum of squares. We *could* use complex numbers instead. Unfortunately, it requires us to take an algebraic fact on faith instead of the fact we proved using geometry; there are no shortcuts. Still, it's worth looking at.

Fact 14.1.6. *If $p \equiv 1 \pmod 4$, then p can be written as a sum of two squares. (This is Theorem 13.4.3.)*

Proof. We already know, from the proof of Lemma 13.3.3 that

$$r = \left(\frac{p-1}{2}\right)!$$

is a square root of -1 modulo p. But now, instead of doing geometry, let's look at what that means.

By definition of

$$r^2 \equiv -1 \pmod p$$

we know that $p \mid r^2 + 1$. Since $r^2 + 1$ is $r^2 - i^2$, let's factor:

$$r^2 + 1 = (r + i)(r - i)$$

Clearly p does not divide either of those as something of the form $a + bi$. So (crucially!), *if we assume the Fundamental Theorem of Arithmetic still holds for Gaussian integers*, then p factors in $\mathbb{Z}[i]$, and has a prime divisor $a + bi$ (in the sense of Subsection 14.1.2).

It's not hard to show that then $a - bi$ also must divide p. We'll skip this.

To finish up, we see that

$$(a + bi)(a - bi) \mid p^2 \Rightarrow a^2 + b^2 \mid p^2$$

and the factor $a^2 + b^2$ is a nontrivial divisor, since $a + bi$ was a proper divisor of p. So the only possibility is

$$a^2 + b^2 = p\ .$$

□

14.2 More Sums of Squares and Beyond

There are many interesting questions one can ask about sums of squares we have not even touched upon. Each of these is very worthy of independent study by undergraduates, and also ideal for computer exploration.

14.2.1 Summing more squares

Fact 14.2.1 (Sums of three squares)**.** *A positive integer may be written as a sum of* three *squares if and only if it is has the form of a product of an* even *power of two times an* odd *number which is congruent to seven modulo eight.*

Proof. We will skip the proof, but see Exercise 14.4.4 and Exercise 14.4.5. □

One might think that every type of square sum is not always possible, but we have this result (see also Exercise 14.4.6), first conjectured by our old friend Bachet:

Fact 14.2.2 (Lagrange's four square theorem)**.** *Any nonnegative integer may be written as a sum of four squares.*

Proof. There are both algebraic proofs using facts similar to Fact 14.1.6 and geometric proofs using Minkowskian ideas from Subsection 13.4.4. These are both interesting, because on the one hand it can use the extension of the complex numbers called the **quaternions**, and on the other hand it shows that geometric ideas can still work in more than two dimensions. □

One can generalize in many ways.

Example 14.2.3. For example, one can ask how *many* ways one can write a number as a sum of three, four, etc. squares. In Exercise 13.7.7 we defined $r_2(n)$ as giving the number of ways to write n as a sum of two squares; the equivalent functions here would be $r_k(n)$ for $n \geq 1$. In that case, Lagrange's four square theorem above could be more succinctly stated as

$$r_4(n) \geq 1 \text{ for all } n \geq 0$$

But in general one may want to be able to compute this, or to give bounds for it as a function of n.

14.2.2 Beyond squares

There are other directions one can generalize our questions. For instance:

Question 14.2.4.

- What numbers can be written as a sum of two *cubes*?
- Three cubes?
- k cubes?

It turns out that *any* number can be written as a sum of at most nine cubes. In the first half of the twentieth century, American mathematician L. E. Dickson proved this, and with the assistance of very substantial tables generated by hand by some of his assistants (*before* the advent of the digital computer!) he showed that every number except 23 and 239 can be represented by eight or fewer cubes!

Alternately, one could keep the number of powers the same, but change the powers.

Question 14.2.5.

- What numbers can be written as a sum of two cubes?
- Two fourth powers?
- Two nth powers?

The reader should feel free to explore this in Exercise 14.4.7. Note that the answers for odd powers will be very different if one allows negative numbers!

Now it is time to recall our discussions in Section 3.4, alluded to in Remark 13.1.4. In that situation, we essentially were looking for integer solutions to

$$x^2 + y^2 = z^2$$

In fact, we characterized such triples x, y, z in Theorem 3.4.5.

But we can reinterpret this as a question in this context – when is a *perfect square* is a sum of two squares? In that case, the previous question can be further specialized:

Question 14.2.6.

- What perfect cubes can be written as a sum of two cubes?
- Fourth powers as a sum of two fourth powers?
- What about nth powers? What (integer) solutions are there to this?

$$x^n + y^n = z^n$$

Ordinarily, as author I would now send the reader to explore some of these questions in Exercise 14.4.8. However, Fermat already proved that other than trivial solutions (such as writing $0^4 + (-1)^4 = 1^4$) there were no solutions in the case $n = 4$; this is Fact 14.2.7. Euler *nearly* proved the same statement for $n = 3$, but made the same hidden assumption as in Fact 15.3.4 (and see [C.3.13] again for the proof).

There is a huge field (algebraic number theory) which developed from this, but we will not digress further upon it. If you recall the discussion in Subsection 11.6.4, it turns out Germain originally investigated n in the case where it is one of the numbers now known as Germain primes. In 1995 Andrew Wiles, along with his former student Richard Taylor, proved the following result via a very deep investigation of (among other things) elliptic curves (recall the brief mention in Section 3.5).

Fact 14.2.7 (Fermat's Last Theorem)**.** *For $n > 2$, there are no three positive integers x, y, z such that*

$$x^n + y^n = z^n$$

Proof. Hanc marginis exiguitas non caperet. □

14.2.3 Waring's problem

The English mathematician Edward Waring asked for an outrageous generalization of these questions of sums of powers, which is still an active area of research called **Waring's Problem**. The most important result is truly spectacular.

Fact 14.2.8 (Hilbert-Waring Theorem)**.** *For each positive integer power m, there is a number $g(m)$ such that* every *nonnegative integer can be written as a sum of $g(m)$ mth powers.*

There is even a potential formula that

$$g(m) = 2^m + \left\lfloor \frac{3}{2} \right\rfloor - 2$$

This has been verified for m out to many millions, and is conjectured to always be true. The aforementioned Dickson notes that this formula was first conjectured by Euler's son, Johann Albrecht.

On the other hand, the question of finding the *smallest* integer $G(m)$ (for a given m) such that every *sufficiently large* number can be written as a sum of that many mth powers is still wide open. Perhaps you will explore it? (See e.g. Exercise 14.4.9.)

14.3 Related Questions About Sums

There is yet another generalization that will serve better as a lead-in to the next chapters. Think about the following two problems.

- What numbers can be written as $x^2 + 2y^2$? (Think of it as $x^2 + y^2 + y^2$.)
- What numbers can be written as $x^2 + 3y^2$?

These are very natural generalizations to the "two squares" question. How could we approach them? Here's one type of idea.

Fact 14.3.1. *No number*

$$n \equiv 5 \text{ or } n \equiv 7 \pmod 8$$

can be written as $x^2 + 2y^2$.

Proof. Try all numbers modulo 8 and see what is possible! (See Exercise 14.4.3.) □

Already Fermat (unsurprisingly) claimed a partial converse to Fact 14.3.1. He stated that any *prime* number p which satisfies $p \equiv 1$ or $p \equiv 3 \pmod 8$ could be written as a sum of a square and twice a square.

This time, Euler wasn't the one who proved it! But you could almost imagine that by factoring

$$x^2 + 2y^2 = (x - \sqrt{2}iy)(x + \sqrt{2}iy)$$

you could start proving such things. When might a square root of two exist modulo p …

Here are some numbers which can be written in this form.

```
@interact
def _(n=10):
    pretty_print(html("Using␣$a$␣and␣$b$␣up␣to␣$%s$:"%n))
    L=[a^2+2*b^2 for a in [0..n] for b in [0..n]]
    L.sort(); print L
```

In Exercise 14.4.10, you will try to discover a similar pattern for $x^2 + 3y^2$. See also Section 15.4.

14.4 Exercises

1. Look up the concepts of 'Gaussian moat', 'Gaussian zoo', and/or 'Gaussian prime spiral' and tell what you think!

2. Look up 'Eisenstein integers'. Can you find any interesting theorems along these lines which they prove? What would Eisenstein *primes* look like? What about "Eisenstein triples"? (See [C.6.17] and Exercise 3.6.12.)

3. Finish proving Fact 14.3.1.

4. Find numbers writeable in two different ways as a sum of three squares.

5. Show that an odd number which is congruent to seven modulo eight may not be written as a sum of three squares.

6. Research Lagrange's four-square theorem and write an essay about it; which proof do you prefer?

7. Write a program in Sage (or another language) to explore which numbers may be written as a sum of two cubes, two fourth powers, and so forth.

8. Write a program in Sage (or another language) to verify Fermat's Last Theorem for some small x, y, z and n.

9. Write a program in Sage (or another language) to compute $g(n)$ in the Hilbert-Waring Theorem for small n.

10. Look for a pattern, similar to the one we found for sums of squares, for which primes can be written in the form $x^2 + 3y^2$. Prove that the primes *not* of this form are impossible.

Chapter 15

Points on Curves

We have already seen a lot of the geometric viewpoint of number theory; think about Section 13.4, for instance.

The goal of the next several chapters is to examine what other questions can one ask of a purely geometric nature – or how far geometry can go in answering other questions.

This chapter returns to the notion of finding specific types of points on graphs of number-theoretic equations. But instead of looking at lines as we did before, there are a variety of *curves* we can consider.

For instance, our previous discussion about the sum of two squares was essentially interpreted as asking when the curve $x^2 + y^2 = n$ has an (integer) lattice point on it or not. We have completely answered this question.

But if we were considering $x^2 + y^2 = n$ to be about a circle of radius $\sqrt{n}$, then $x^2 + 2y^2 = n$ must be about an ellipse! Here is a visualization of points on these ellipses.

```
var('x,y')
@interact
def _(n=3):
    plot1=implicit_plot(x^2+2*y^2-n, (x,-n,n), (y,-n,n),
        plot_points=100)
    grid_pts = [[i,j] for i in [-n..n] for j in [-n..n]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (2*coords[1]^2+coords[0]^2)==n]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1 + plot_grid_pts + plot_lattice_pts,
        figsize=[5,5],aspect_ratio=1)
    pretty_print(html("The␣ellipse␣$x^2+2y^2=%s"%n))
```

Questions like this are at the heart of modern number theory – plus, there are such nice pictures! It turns out this investigation will have surprising connections to calculus and group theory too.

With that in view, you may want to try to find integer points on the following curves. Each exemplifies a type we will discuss in this chapter.

1. $y^3 = x^2 + 2$
2. $x^2 + 2y^2 = 9$
3. $x^2 - 2y^2 = 1$

What we will do is to slowly try to make our way to finding *integer* solutions to some more difficult Diophantine equations, using an idea about *rationals* which simplifies Pythagorean triple geometry. We'll then return to the integer setup once we've gotten this background.

15.1 Rational Points on Conics

15.1.1 Rational points on the circle

Remember that in Section 3.4 we thought of Pythagorean triples as solutions to

$$x^2 + y^2 = z^2 \, .$$

Now, let's divide the whole Pythagorean thing by z^2:

$$\frac{x^2}{z^2} + \frac{y^2}{z^2} = 1 \Rightarrow \left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2 = 1 \, .$$

Since we can always get any two rational numbers to have a common denominator, what that means is the Pythagorean problem is the same as finding all *rational* solutions to the simpler formula

$$a^2 + b^2 = 1 \, ,$$

which *seems* to be a very different problem. Let's investigate this.

```
var('x,y')
@interact
def _(slope=-2/3):
    plot1=implicit_plot(x^2+y^2-1, (x,-1.5,1.5),
        (y,-1.5,1.5), plot_points=100)
    plot2=plot(slope*(x-1),x,-1.5,1.5)
    plot3=point(((slope^2-1)/(slope^2+1),
        -2*slope/(slope^2+1)), rgbcolor=(1,0,1),
        pointsize=20)
    show(plot1+plot2+plot3 + point((1,0),
        rgbcolor=(0,0,0), pointsize=20), figsize=[5,5],
        aspect_ratio=1)
```

In the interact above, the blue line intersects the circle $x^2 + y^2 = 1$ in the point $(1, 0)$ and has rational slope denoted by `slope`. If you change the variable `slope`, then the line will change.

It is not a hard exercise to see that the line through two rational points on a curve will have rational slope, nor what its formula is, so that *every* rational point on the circle is gotten by intersecting $(1, 0)$ with a line with rational slope.

It is a little harder to show that intersecting such a line with the circle always gives a rational point, but this is also true! It is also far more useful, as it gives us a technique to find *all* rational points and hence *all* Pythagorean triples.

Fact 15.1.1. *All lines with rational slope through* $(1, 0)$ *intersect the unit circle in a second rational point.*

Proof. In fact, we can do even better than prove this; we can get a formula for the points.

First, any line with slope t has formula $y = t(x-1)$. We can then obtain all intersections with the circle $x^2+y^2=1$ by plugging in y, so:

$$x^2 + (t(x-1))^2 = 1 \Rightarrow x^2 + t^2x^2 - 2xt^2 + t^2 = 1$$

We will skip showing (see Exercise 15.7.1)that the quadratic formula yields the two answers $\frac{t^2\pm1}{t^2+1}$. Note that $\frac{t^2+1}{t^2+1} = 1$ gives the point $(1,0)$ which we already knew.

The other, new, point is $\frac{t^2-1}{t^2+1} = x$; plugging this in gives $y = t\left(\frac{t^2-1}{t^2+1} - 1\right) = \frac{-2t}{t^2+1}$.

In summary, every rational slope t gives us the point $\left(\frac{t^2-1}{t^2+1}, \frac{-2t}{t^2+1}\right)$. □

Even the inputs $t = 0$ and $t = \infty$ have an appropriate interpretation in this framework. Such a description of the (rational) points of the circle is called a **parametrization**. Plug in various t and see what you get!

Remark 15.1.2. You could start the whole process with $(-1,0)$ or $(0,1)$, use all lines through it with rational slopes, and get a different parametrization.

15.1.2 Parametrization in general

But will this always work? Here is an amazing fact we will not prove.

Fact 15.1.3. *If you have a quadratic equation with rational coefficients with* at least one *rational point, then you can get* all *other rational points by intersecting all lines with rational slope through that point on the curve.*

Example 15.1.4. Here's an example with $x^2+3y^2=1$.

```
var('x,y')
@interact
def _(slope=-1/2,f=x^2+3*y^2-1):
    f(x,y)=f
    plot1=implicit_plot(f, (x,-1.5,1.5), (y,-1.5,1.5),
        plot_points=100)
    plot2=plot(slope*(x-1),x,-1.5,1.5)
    show(plot1+plot2+point((1,0), rgbcolor=(0,0,0),
        pointsize=20), figsize=[5,5], aspect_ratio=1)
```

As in the proof of Fact 15.1.1, the line going through $(1,0)$ has equation $y = t(x-1)$. Here, the ellipse has equation $x^2+3y^2=1$, so that we must solve the equation

$$x^2 + 3t^2(x-1)^2 = 1 \Rightarrow x^2 + 3t^2x^2 - 6t^2x + 3t^2 - 1 = 0$$

for x to find a parametrization of x in terms of t.

This seems daunting. Here are two strategies (see Exercise 15.7.2 to try them).

- We *already know* that there is a solution $x = 1$, so that $x-1$ must be a factor of the expression! So we could factor it out if we wished.
- Alternately, we could use the quadratic formula and discard the solution $x = 1$.

In either case you should get

$$x = \frac{3t^2-1}{3t^2+1}, y = \frac{-2t}{3t^2+1}$$

Now you can find all kinds of interesting solutions like $\left(\frac{11}{13}, \frac{-4}{13}\right)$.

Where does this go? These solutions lead us to *integer* solutions of three-variable equations. In this case, it gives solutions to ones like $x^2+3y^2=z^2$.

```
var('x,y')
@interact
def _(viewsize=15):
    plot1=plot3d(sqrt(x^2+3*y^2), (x,-viewsize,viewsize),
        (y,-viewsize/2,viewsize/2))
    grid_pts = [[i,j,k] for i in [-viewsize..viewsize] for
        j in [-viewsize..viewsize] for k in [0..viewsize]]
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]^2+3*coords[1]^2==coords[2]^2)]
    plot_lattice_pts = point3d(lattice_pts, rgbcolor =
        (1,0,0),pointsize=40)
    show(plot1+plot_lattice_pts)
```

Since x and y have a common denominator, we can just multiply through by the square of that denominator to get a solution to this. E.g.

$$11^2+3(-4)^2=13^2$$

which is a rather non-obvious solution, to say the least, and only one of many that this method can help us find.

15.1.3 When curves don't have rational points

However, this method does *not* always work. Namely, you need *at least one* rational point to start off with. And what if there isn't one that exists? It turns out that Diophantus already knew of some such curves.

Fact 15.1.5. *The circle $x^2+y^2=15$ has no rational points.*

Proof. First, note this is a much stronger statement than what we already know, which is that this curve has no *integer* points (see Fact 13.1.1).

The way to prove this is to correspond this to *integer points* on $x^2+y^2=15z^2$.

Every rational point on the first curve looks like $(p/q, r/q)$ for some $p, r, q \in \mathbb{Z}$, so multiplying through by the common denominator gives us integer points on the second surface.

But now consider the whole thing modulo 4. The reader should definitely check that there are *no* legitimate possibilities! (See Exercise 15.7.4) □

```
var('x,y')
@interact
def _(viewsize=15):
    plot1=plot3d(sqrt(x^2+y^2)/sqrt(15), (x,0,viewsize),
        (y,0,viewsize))
    grid_pts = [[i,j,k] for i in [0..viewsize] for j in
        [0..viewsize] for k in [0..3*viewsize]]
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]^2+coords[1]^2==15*coords[2]^2)]
```

```
plot_lattice_pts = point3d(lattice_pts, rgbcolor =
    (1,0,0),pointsize=40)
show(plot1+plot_lattice_pts)
```

As we can see, there are *no* rational points on a circle of radius $\sqrt{15}$ because there are no *integer* points on the corresponding surface other than ones with $x, y = 0$ – and those correspond to $z = 0$, which would give a zero denominator on the circle. Here is a place where rational points are helped by integer points instead of vice versa.

Let's do another example.

Example 15.1.6. Try to find rational points on the ellipse $2x^2 + 3y^2 = 1$.

Solution. A rational point would correspond to $2x^2 + 3y^2 = z^2$. You can try looking at it modulo four, but that goes nowhere. Instead, given the three as a coefficient, look at it modulo 3!

In this case it reduces to

$$2 \equiv (zx^{-1})^2 \pmod 3$$

This is impossible since $[0], [1], [2]$ all square to $[0]$ or $[1]$ in $\mathbb{Z}_3$.

The point is that, at least sometimes, modular arithmetic and going back and forth between integer and rational points helps us both *find* points and prove there *are no such* points.

15.2 A tempting cubic interlude

It is interesting that our investigation of rational points, initially motivated by integer points like Pythagorean triples, inevitably led back to integer points. Soon we will look at some remarkable properties that sets of integer points on certain curves have, and whether any such points even exist.

But before moving on, it is worth looking at some interesting tidbits relating to another type of equation, $x^3 + ay^3 = b$.

For the first example, consider that sometimes mathematicians like to explore hard questions for their own sake. Sometimes proofs are very challenging, indeed. Then again, sometimes a very easy proof is missed.

One example of this is the equation $x^3 - 117y^3 = 5$. At one point a well-known number theorist specializing in Diophantine equations asserted this was known to have few solutions. A few years later, using field theory, this was proved.

Two years later, a note was published in an obscure Romanian journal that if one reduces the original equation modulo nine, a simple congruence is obtained which one can show has no solutions just by trying all possibilities by hand (you can try it in Exercise 15.7.5). (See this MathOverflow question for background.)

Another interesting story related to this is that of Henry Dudeney's "Puzzle of the Doctor of Physic", related by Andrew Bremner of Arizona State University in [C.6.15]. Dudeney was on of the most famous puzzle constructors of a century ago, and this puzzle is a doozy.

Question 15.2.1. Find the (rational) diameters of two spheres whose combined volume is that of two spheres of diameters one foot and two feet.

This is equivalent to finding rational points on the curve $x^3 + y^3 = 9$.

The puzzle itself gives the points $(1, 2)$ and $(2, 1)$, so the question is whether one can find any *other* such points. Bremner takes the reader through a geometric tour of trying to intersect this curve with various lines with rational slope in the hope of finding a proper solution to this problem. Here is a potential first step, using the *tangent* line to the curve at $(2, 1)$.

```
var('y')
implicit_plot(x^3+y^3==9, (x,-3,3), (y,-3,3)) +
    plot(-4*x+9,(x,1,3)) + point((2,1),size=10)
```

It turns out that this point is not acceptable as a solution (why?). In fact, it takes several more steps of connecting points to arrive at a solution, namely

$$\left(\frac{415280564497}{348671682660}, \frac{676702467503}{348671682660}\right)$$

which does seem a bit excessive but is sure fun.

We are now ready to begin our discussion of more integer points on curves.

As mentioned before, we'll try to find integer points on the following types of curves:

- $y^3 = x^2 + 2$ (sometimes called the *Bachet* equation)
- $x^2 + 2y^2 = 9$ (a well-known friend, the ellipse)
- $x^2 - 2y^2 = 1$ (a hyperbola with surprising connections to $\sqrt{2}$)

15.3 Bachet and Mordell Curves

Let's start by talking about $y^3 = x^2 + 2$ as a type of curve. Recall from Section 3.5 that Bachet de Méziriac first asserted this had one positive integer solution in 1621, very early in the development of modern number theory.

Example 15.3.1. What is that solution?

Fermat, Wallis, and Euler also studied this and gave various discussions and proofs of this fact. As we saw earlier, this equation is actually one of a more general class of equations called the *Mordell* equation:

$$y^3 = x^2 + k\ , \quad k \in \mathbb{Z}$$

Louis Mordell, an American-born British mathematician, proved some remarkable theorems about this class of equations.

Notice that Mordell's set of curves are not quadratic/conic but rather cubic, which makes them more mysterious (and, as it happens, more useful for cryptography). There is a theorem that they can only have *finitely many* integer points (in fact, there are even useful bounds for how many that depend only on the prime factorization of k). At the same time, they are apparently "simple" enough that they can still have infinitely many *rational* points; Gerd Faltings won a Fields Medal for proving that higher-degree curves cannot.

```
var('x,y')
@interact
def _(k=(2,[-15..15]),viewsize=10):
    g(x,y)=y^3-x^2
```

```
    plot1 = implicit_plot(g-k, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 100)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[1]^3-coords[0]^2==k)]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
        [5,5], xmin = -viewsize, xmax = viewsize, ymin =
        -viewsize, ymax = viewsize)
    pretty_print(html("Integer␣points␣on␣the␣Mordell␣
        equation␣$y^3=x^2+%s$␣in␣this␣window"%k))
```

15.3.1 Verifying points don't exist

Proving things about Mordell's equation is quite tricky, but once in a while there is something you can do. For instance, we can verify something we can see in the interact above.

Fact 15.3.2. *There are no integer solutions to* $x^3 = y^2 - 7$.

Proof. Look at the equation modulo 4, which gives

$$x^3 \equiv y^2 - 3 \pmod 4$$

Two cases are easily dealt with.

- If x is even, this reduces to $y^2 \equiv 3 \pmod 4$, which we already know is not possible.
- Likewise, if $x \equiv 3 \pmod 4$, then $x^3 \equiv 3$ as well, so that $y^2 \equiv 6 \equiv 2 \pmod 4$, also impossible.

So it remains to look at the possibility that $x \equiv 1 \pmod 4$.

Now we do a cute trick. Since 7 is one less than a perfect cube, we rewrite the original equation as

$$x^3 + 8 = y^2 + 1 \Rightarrow (x+2)(x^2 - 2x + 4) = y^2 + 1$$

We rewrite this as

$$(x+2) \mid y^2 + 1$$

which implies any prime divisor of $x+2$ must also divide y^2+1.

Since we are assuming $x \equiv 1 \pmod 4$, we have that

$$x + 2 \equiv 3 \pmod 4 ,$$

so at least one of those prime divisors p of $x+2$ must also be $p \equiv 3 \pmod 4$. This implies

$$p \mid (y^2 + 1) \Rightarrow y^2 \equiv -1 \pmod p .$$

This is not possible, because Fact 13.3.2 implies there are no square roots modulo p for this type of p. □

This is a simple version of a far more general statement.

Theorem 15.3.3. *If the following hold:*

- $M \equiv 2 \pmod 4$,
- $N \equiv 1 \pmod 2$, *and*
- *all prime divisors p of N are of the form $4k+1$.*

Then there is no solution to

$$y^2 = x^3 + (M^3 - N^2)$$

Proof. The proof basically follows the same outline as Fact 15.3.2; see Exercise 15.7.7. □

There are lots of similar statements one can prove too. But there is a larger point, based on the very specific conditions on M and N. Namely, if we want to prove anything about such equations with methods we *currently* have access to in this text, we have no hope of getting any general results.

15.3.2 More on Mordell

Let's see what I mean by "no hope" here by returning to Bachet's original equation, $y^3 = x^2 + 2$. What are some naive things we can say?

- It should be clear that x and y must have the same parity.
- If they are both even then y^3 is divisible by 4 but $x^2 + 2 \equiv 2 \pmod 4$, which is impossible.
- So x and y are both odd.
- That doesn't really narrow things down too much, really.

Now, Euler *nearly* proves the following fact.

Fact 15.3.4. *The only positive solution to the Bachet equation is $x = 5, y = 3$.*

Proof. Proving this is already a little sophisticated, and is closely connected to the use of complex numbers in Section 14.1. Here we will give the idea behind Euler's 'proof'.

In examining $a^2 + b^2$, we factored it as $(a + bi)(a - bi)$ for a square root of negative 1. Just as there, we would like to factor the $x^2 + 2$. But it can't be done in $\mathbb{Z}[i]$.

Instead, we could try to use the square root of -2, and define

$$\mathbb{Z}[\sqrt{-2}] = \{a + b\sqrt{-2} \mid a, b \in \mathbb{Z}\}$$

Then

$$y^3 = (x - \sqrt{-2})(x + \sqrt{-2})$$

We haven't done anything with cubes yet …

Here is the tricky bit. In the integers, if $y^3 = pq$ and $\gcd(p, q) = 1$, then p and q must both be perfect (integer) cubes. So Euler *assumes this works* in $\mathbb{Z}[\sqrt{-2}]$ as well, and that the factors of $x^2 + 2$ are "coprime" (whatever that means in this new number system). (A very nice discussion of this is in [C.3.13], including a full proof in its appendix.)

Then some basic algebraic manipulation of

$$x - \sqrt{-2} = \left(a + b\sqrt{-2}\right)^3$$

and and divisibility considerations end up showing that $b \mid 1$ and $a = \pm b$, which ends up showing $x = \pm 5$ and $y = 3$. (We will not take this further; see Exercise 15.7.8.) □

Where's the problem? It turns out you *can* say that a product which is a cube is a product of cubes in this situation, but it requires some (geometrically motivated) proof, just like with $\mathbb{Z}[i]$. In his 1765 "Vollständige Anleitung zur Algebra", sections 187-188 and 191, Euler explicitly says that this just works – in *any* number system with $\mathbb{Z}[\sqrt{c}]$. He solves this one in section 193, and solves $x^2 + 4 = y^3$ using the same technique in section 192, without realizing the problem.

But we shouldn't be too hard on Euler! He was one of the first people to even consider some essentially random new number system of this type. And, in 1738, he gives a correct *and full* proof of the observation that 8 and 9 is the only time a perfect square is preceded by a perfect cube, which is Mordell's equation for $k = -1$. (See also Question 3.5.1.)

If you are interested in more information about how to prove cases of Mordell's equation, there are many good resources, including a nice one on Keith Conrad's website.

In case you are wondering, even finding a bound on the *size* of the set of solutions to Mordell's equation for a given k is tricky.

- Mordell, Siegel, and Thue all had a part after World War I in showing there are finitely many solutions for a given k, but said nothing about how big x and y might be.
- An early bound was that

  $$|x| < e^{10^{10}|k|^{10^4}}$$

  which is of course ridiculously huge.
- More recent conjectures are that x has absolute value less than $e^C|d|^{2+\epsilon}$, where ϵ is as small as you want and C seems to pretty close to one, probably less than two.

Finally, we have to mention a very famous result. Recall that these curves can have infinitely many *rational* points, even if they have finitely many (or zero) integer points. The following is a bit of a surprise, then; the rational points can still be *described* finitely.

Theorem 15.3.5 (Mordell's Theorem)**.** *Essentially, the set of points on a Mordell curve is a combination of finitely many "cyclic" groups (in a very specific way I will not describe), and so it can be described using finitely many of the rational points.*

If you like, the rational points might be infinite, but not *too* infinite.

15.4 Points on Quadratic Curves

On the other hand, finding lattice points on a *quadratic* curve is much more tractable. This is because we understand conic sections so well, after having worked with them for two thousand years!

```
var('x,y')
@interact
def _(a=1,b=2,c=9):
```

```
viewsize=math.sqrt(c)+1
g(x,y)=a*x^2+b*y^2
plot1 = implicit_plot(g-c, (-viewsize,viewsize),
    (-viewsize,viewsize), plot_points = 200)
grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
    in [-viewsize..viewsize]]
plot_grid_pts =
    points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
lattice_pts = [coords for coords in grid_pts if
    (a*coords[0]^2+b*coords[1]^2==c)]
plot_lattice_pts = points(lattice_pts, rgbcolor =
    (0,0,1),pointsize=20)
show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
    [5,5], xmin = -viewsize, xmax = viewsize, ymin =
    -viewsize, ymax = viewsize, aspect_ratio=1)
```

Here we see our second prototype, $x^2 + 2y^2 = 9$. You can see that, in addition to the obvious solution where $y = 0$, there is the (nearly as obvious, because the numbers are small, but still interesting) solution $x = 1, y = 2$.

In general, for our purposes an ellipse is special because there are only finitely many lattice points to check. So much for the computational problem – just get a fast computer! However, I just want to mention where a general theory for such things might come from. After all, it gets harder to check with "industrial strength" ellipses, and we want theorems.

15.4.1 Transforming conic sections

Although it's being removed from the curriculum nowadays, there is something that often happens in high school mathematics or first-year college calculus where you learn how to transform one conic section to another one of the same type with a matrix.

Example 15.4.1. We can get from the circle $x^2 + y^2 = 9$ to $x^2 + 2y^2 = 9$ by multiplying the vector (x, y) by the matrix $\begin{pmatrix} 1 & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix}$; that would not stretch the x-axis, but shrinks in the y axis by the appropriate amount.

However, one can also think of *both* conics in such a transformation as coming from matrices. Compare these:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 + y^2$$

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 + 2y^2 \, .$$

Gauss was interested in extending Fermat's question; namely, what numbers are representable in these ways, as opposed to just a sum of squares? It turns out that many such **quadratic forms** represent the same sets of integers (recall Section 14.3).

The Sage reference manual even uses our example to demonstrate this:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 + 2y^2 \text{ and } \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 + 2xy + 3y^2$$

Both of these should fulfill Fermat's result about primes modulo 8 in the discussion around Fact 14.3.1; as an example, both should represent 11. Clearly $11 = 3^2 + 2 \cdot 1^2$ works, but what about the other version?

```
var('x,y')
@interact(layout=[['a','b'],['c','d'],['output']])
def _(a=1,b=1,c=1,d=3,output=11):
    viewsize=ceil(math.sqrt(output)+1)
    g(x,y)=a*x^2+(b+c)*x*y+d*y^2
    plot1 = implicit_plot(g-output,
        (x,-viewsize,viewsize), (y,-viewsize,viewsize),
        plot_points = 200)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (a*coords[0]^2 + (b+c)*coords[0]*coords[1] +
        d*coords[1]^2 == output)]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
        [5,5], xmin = -viewsize, xmax = viewsize, ymin =
        -viewsize, ymax = viewsize, aspect_ratio=1)
    pretty_print(html("Integer␣lattice␣points␣on␣
        $%sx^2+%sxy+%sy^2=%s$"%(a,b+c,d,output)))
```

Looks like $x = 2, y = 1$ will do it. The real reason behind this is that

$$x^2 + 2xy + 3y^2 = (x + y)^2 + 2y^2$$

is a coordinate transformation.)

There is some very deep theory there, which is another place where lie the beginnings of algebraic number theory, just like with the Gaussian integers. But we'll let it rest there.

15.4.2 More conic sections

Instead, we will continue looking for integer points on a given specific curve. Assuming that ellipses are doable by simply counting, what is next?

The parabola comes to mind. A general parabola would look like $ny = mx^2$; this can be thought of in your usual terms as $y = ax^2$ and $a = m/n$.

Then I can just check all $x \in \mathbb{Z}$ such that $n \mid mx^2$. Since $\gcd(m, n) = 1$ for this (lowest terms), we would just need in fact that $n \mid x^2$ (so if n is prime, $n \mid x$ suffices)!

So if $y = mx^2$ for integer m, any x will do. That makes sense; integer input had better give integer output, which would be a lattice point!

Example 15.4.2. If $2y = x^2$, we just look at it as $2 \mid x$, so that requiring x even will give lattice points.

And so on.

```
var('x,y')
@interact
def _(m=1,n=2):
    viewsize=3*n^2
    f(x,y)=(m/n)*x^2
    plot1 = plot(f,-viewsize,viewsize)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [0..viewsize^2]]
```

```
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (m*coords[0]^2==n*coords[1])]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
        [5,5], xmin = -viewsize, xmax = viewsize, ymin =
        -1, ymax = (m/n)*viewsize^2)
```

One might think this is all there is to say about points on the parabola. But before we go on, I want to point out something very interesting. Look at the following two setups in interacts. In one I create the line through two integer points on the conic, in the other I create the tangent line through one integer point.

```
a=1
b=2
var('x,y')
viewsize=10
R.<x,y> = ZZ[]
f(x)=(a/b)*x^2
plot1 = plot(f,-viewsize,viewsize)
grid_pts = [[i,j] for i in [-viewsize..viewsize] for j in
    [0..viewsize^2]]
plot_grid_pts =
    points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
lattice_pts = [coords for coords in grid_pts if
    (a*coords[0]^2==b*coords[1])]
plot_lattice_pts = points(lattice_pts, rgbcolor =
    (0,0,1),pointsize=20)
line1 = plot((f(2)-f(4))/(2-4)*(x-2)+f(2),
    -viewsize,viewsize, color='red')
line2 = plot((f(2)-f(4))/(2-4)*x, -viewsize,viewsize,
    color='red',linestyle='--')
line3 = plot((f(2)-f(4))/(2-4)*(x+2)+f(-2),
    -viewsize,viewsize, color='red',linestyle='--')
show(plot1+plot_grid_pts + plot_lattice_pts +
    line1+line2+line3, xmin=-5,ymin=-1,ymax=viewsize^2/2)
```

```
a=1
b=2
var('x,y')
viewsize=10
R.<x,y> = ZZ[]
f(x)=(a/b)*x^2
plot1 = plot(f,-viewsize,viewsize)
grid_pts = [[i,j] for i in [-viewsize..viewsize] for j in
    [0..viewsize^2]]
plot_grid_pts = points(grid_pts,
    rgbcolor=(0,0,0),pointsize=2)
lattice_pts = [coords for coords in grid_pts if
    (a*coords[0]^2==b*coords[1])]
plot_lattice_pts = points(lattice_pts, rgbcolor =
    (0,0,1),pointsize=20)
line1 = plot(2*(a/b)*4*(x-4)+f(4), -viewsize,viewsize,
    color='red')
```

```
line2 = plot(2*(a/b)*4*x,-viewsize,viewsize,
    color='red',linestyle='--')
line3 = plot(2*(a/b)*4*(x+2)+f(-2), -viewsize,viewsize,
    color='red',linestyle='--')
show(plot1+plot_grid_pts + plot_lattice_pts +
    line1+line2+line3, xmin=-5,ymin=-1,ymax=viewsize^2/2)
```

In both cases you get another integer point! Could this be coincidence?

15.5 Making More and More and More Points

Based on our previous work, we know that we should be able to do these things to get new *rational* points.

Algorithm 15.5.1 (Getting New Rational Points)**.** *Two ways to obtain new rational points on a conic from rational points you already have are:*

- *Connect two points with a line, and then make a line with the same slope but through another (rational) point. We call this* ***adding*** *points.*
- *Find the* tangent *line through a point, and then make a line with the same slope but through another point. We call this* ***doubling*** *a point.*

Fact 15.5.2. *The set of rational points on a conic section is an Abelian group. Assuming you have a point selected as an identity element, the group operation on two points P and Q is given by the first, "adding points", operation Algorithm 15.5.1. That is, you connect P and Q by a secant line of slope m, and then connect the identity to a fourth point $P+Q$ with a line of slope m. Adding a point P to itself uses the slope of the tangent line at P, the second, "doubling points", operation in Algorithm 15.5.1.*

15.5.1 Toward integer points

More germane to our investigation, our limited experience in the previous section suggests these processes may often give you integer points. This is not a coincidence; in general, we should try to add or double points to get (new) integer points.

As we are only guaranteed *rational* points, this doesn't always work. Below, I try this on the ellipse from the beginning of Section 15.4.

```
a=1
b=2
c=9
var('x,y')
viewsize=ceil(math.sqrt(c)+1)
f=a*x^2+b*y^2
g(x,y)=f
plot1 = implicit_plot(g-c, (-viewsize,viewsize),
    (-viewsize,viewsize), plot_points = 200)
grid_pts = [[i,j] for i in [-viewsize..viewsize] for j in
    [-viewsize..viewsize]]
plot_grid_pts = points(grid_pts,
    rgbcolor=(0,0,0),pointsize=2)
lattice_pts = [coords for coords in grid_pts if
    (a*coords[0]^2+b*coords[1]^2==c)]
plot_lattice_pts = points(lattice_pts, rgbcolor =
    (0,0,1),pointsize=20)
```

```
line1 = plot(x+3, -viewsize,viewsize, color='red')
line2 = plot(x+1-2, -viewsize,viewsize,
    color='red',linestyle='--')
line3 = plot(x-1-2, -viewsize,viewsize,
    color='red',linestyle='--')
show(plot1+plot_grid_pts+plot_lattice_pts+line1+line2+line3,
    figsize = [5,5], xmin = -viewsize, xmax = viewsize,
    ymin = -viewsize, ymax = viewsize, aspect_ratio=1)
```

Rotten luck. But in some circumstances, this strategy works very well indeed. The following hyperbola is simple, just $x^2 - dy^2 = 1$.

```
var('x,y')
@interact
def _(viewsize=slider(10,20,1),d=2):
    f(x,y)=x^2-d*y^2
    plot1 = implicit_plot(f-1, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 200)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts = points(grid_pts, rgbcolor=(0,0,0),
        pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]^2-d*coords[1]^2==1)]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
        [5,5], xmin = -viewsize, xmax = viewsize, ymin =
        -viewsize, ymax = viewsize, aspect_ratio=1)
    pretty_print(html("The␣hyperbola␣$x^2-%sy^2=1$"%d))
```

So let's try it. What happens when we take the tangent line to the point $(3, 2)$ as the solution to $x^2 - 2y^2 = 1$?

```
d=2
var('x,y')
@interact
def _(x_0=3,y_0=2,lattice=False,auto_update=False):
    g(x,y)=x^2-d*y^2
    x_1,y_1=x_0^2+2*y_0^2,2*x_0*y_0
    plot1 = implicit_plot(g-1,(x_0-4,x_1+4),(x_0-4,x_1+4),
        plot_points = 200)
    grid_pts = [[i,j] for i in [x_0-4..x_1+4] for j in
        [x_0-4..x_1+4]]
    plot_grid_pts = points(grid_pts, rgbcolor=(0,0,0),
        pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]^2-d*coords[1]^2==1)]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    line1 = plot((x_0/(2*y_0))*(x-x_0)+y_0,x_0-4,x_1+4,
        color='red')
    line2 = plot((x_0/(2*y_0))*(x-1),x_0-4,x_1+4,
        color='red', linestyle='--')
    if lattice:
        show(plot1 + plot_grid_pts + plot_lattice_pts +
            line1 + line2, figsize = [5,5], xmin = x_0-4,
            xmax = x_1+4, ymin = y_0-4, ymax = y_1+4,
            aspect_ratio=1)
```

```
    else:
        show(plot1+plot_lattice_pts+line1+line2, figsize =
            [5,5], xmin = x_0-4, xmax = x_1+4, ymin =
            y_0-4, ymax = y_1+4, aspect_ratio=1)
    pretty_print(html("The␣new␣points␣are␣$x_1=%s$␣and␣
        $y_1=%s$"%(x_1,y_1)))
```

A new point, amazing! And if we plug that one in, another one. Hmm …

As it turns out, this is quite an old idea. Finding integer solutions to this hyperbola is called solving **Pell's equation**, and has been studied in this form since the seventeenth century. But a process very similar to this was already rigorously discussed by Brahmagupta centuries before *that*!

Remark 15.5.3. In the event, Pell did not have anything to do with them; it was all based on a misunderstanding. But names stick. In mathematics this phenomenon of not naming things after the actual discoverer is sometimes called Boyer's law, more generally Stigler's law of eponymy (which are themselves self-referential).

15.5.2 A surprising application

The particular equation $x^2 - 2y^2 = 1$ was studied by Greeks such as Theon of Smyrna to shed light on $\sqrt{2}$, though not in the generality we are. Why would this help?

Well, imagine that (x, y) fulfill this equation. Then divide and rearrange the original equation to get

$$\frac{x^2}{y^2} = 2 + \frac{1}{y^2}$$

If you can find a solution to this equation with a big y, then $\frac{x^2}{y^2}$ should be pretty close to 2, which means x/y itself is pretty close to $\sqrt{2}$.

Let's see this in action. We already tried to find integer points on this curve.

```
var('x,y')
@interact
def _(viewsize=slider(10,20,1),d=2):
    f(x,y)=x^2-d*y^2
    plot1 = implicit_plot(f-1, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 200)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    plot_grid_pts =
        points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]^2-d*coords[1]^2==1)]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(plot1+plot_grid_pts+plot_lattice_pts, figsize =
        [5,5], xmin = -viewsize, xmax = viewsize, ymin =
        -viewsize, ymax = viewsize, aspect_ratio=1)
    pretty_print(html("Points␣on␣the␣curve␣
        $x^2-%sy^2=1$"%d))
```

The easy one for $d = 2$ was $(3, 2)$. And after all, $\frac{3}{2} = 1.5$ isn't too far from $\sqrt{2} \approx 1.414$. There seems to be another point if we zoom out, but that would be a tedious way to compute them …

Example 15.5.4. What if we double the point and take the tangent at $(3, 2)$? (See Algorithm 15.5.1.) Then we take that slope, and make a new line through the "base" point (in this case, $(1, 0)$).

Then the next point we get is $(17, 12)$. (See Exercise 15.7.12.) Indeed, $17^2 - 2 \cdot 12^2 = 1$ and $17/12 \approx 1.417$, already correct to three significant digits. Those Greeks!

15.6 The Algebraic Story

15.6.1 Computing the hyperbola

What is going on algebraically here? The algebra is not hard, but a little dense; follow this proof closely.

Proposition 15.6.1. *Doubling integer points on the hyperbola $x^2 - 2y^2 = 1$ yields more integer points.*

Proof. Algebraically, if $x^2 - 2y^2 = 1$, then the tangent line at any point (x_0, y_0) is given by implicit differentiation to be $y' = \frac{x_0}{2y_0}$. So we start there.

What is the line through $(1, 0)$ with that same slope? It's

$$y = \frac{x_0}{2y_0}(x - 1) \,,$$

of course. Let's check where else this intersects the hyperbola, if at all.

Start off with plugging the line into the hyperbola:

$$x^2 - 2y^2 - 1 = x^2 - 2\left(\frac{x_0}{2y_0}(x-1)\right)^2 - 1 =$$

$$\left(1 - \frac{x_0^2}{2y_0^2}\right)x^2 + \left(\frac{x_0^2}{y_0^2}\right)x + \left(-1 - \left(\frac{x_0^2}{2y_0^2}\right)\right) = 0 \,.$$

This can be simplified and then *solve*, unbelievably (via the quadratic formula or factoring out $x - 1$)

$$(2y_0^2 - x_0^2)x^2 + 2x_0^2 x + (-2y_0^2 - x_0^2) = 0$$

$$x = \frac{-2x_0^2 - 4y_0^2}{-2x_0^2 + 4y_0^2} = \frac{x_0^2 + 2y_0^2}{x_0^2 - 2y_0^2} = x_0^2 + 2y_0^2$$

Finally, do a slick substitution of the original point:

$$y = \frac{x_0}{2y_0}(x-1) = \frac{x_0}{2y_0}(x_0^2 + 2y_0^2 - (x_0^2 - 2y_0^2)) = 2x_0y_0 \,.$$

□

Now let's try this with actual points in Sage!

```
@interact
def _(x_0=17,y_0=12):
    x_1=x_0^2+2*y_0^2
    y_1=2*x_0*y_0
    pretty_print(html("Initial␣point␣was␣$(%s,%s)$;␣new␣
        point␣is␣$(%s,%s)$."%(x_0,y_0,x_1,y_1)))
    pretty_print(html("And␣indeed␣$%s^2-2\cdot%s^2$␣equals␣
        $%s$"%(x_1,y_1,x_1^2-2*y_1^2)))
```

```
d=2
var('x,y')
@interact
def _(x_0=3,y_0=2,lattice=False,auto_update=False):
    g(x,y)=x^2-d*y^2
    x_1,y_1=x_0^2+2*y_0^2,2*x_0*y_0
    plot1 = implicit_plot(g-1, (x_0-4,x_1+4),
        (x_0-4,x_1+4),plot_points = 200)
    grid_pts = [[i,j] for i in [x_0-4..x_1+4] for j in
        [x_0-4..x_1+4]]
    plot_grid_pts = points(grid_pts, rgbcolor=(0,0,0),
        pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]^2-d*coords[1]^2==1)]
    plot_lattice_pts = points(lattice_pts, rgbcolor =
        (0,0,1), pointsize=20)
    line1 = plot((x_0/(2*y_0))*(x-x_0)+y_0,x_0-4,x_1+4,
        color='red')
    line2 = plot((x_0/(2*y_0))*(x-1),x_0-4,x_1+4,
        color='red', linestyle='--')
    if lattice:
        show(plot1 + plot_grid_pts + plot_lattice_pts +
            line1 + line2, figsize = [5,5], xmin = x_0-4,
            xmax = x_1+4, ymin = y_0-4, ymax = y_1+4,
            aspect_ratio=1)
    else:
        show(plot1+plot_lattice_pts+line1+line2, figsize =
            [5,5], xmin = x_0-4, xmax = x_1+4, ymin =
            y_0-4, ymax = y_1+4, aspect_ratio=1)
    pretty_print(html("The␣new␣points␣are␣$x_1=%s$␣and␣
        $y_1=%s$"%(x_1,y_1)))
```

Awesome!

15.6.2 Yet more number systems

Brahmagupta knew how to do this, though of course he did it both without our geometric interpretation (which was only made possible by Descartes and Fermat's introduction of coordinate systems) and also without the benefit of symbolically representing $\sqrt{2}$, which provides this alternate description of what we did.

Fact 15.6.2. *If (x_0, y_0) is a solution to $x^2 - 2y^2 = 1$, then so is (x_1, y_1) where*

$$(x_0 + \sqrt{2}y_0)^2 = x_1 + \sqrt{2}y_1 \,.$$

If you were to do the algebra out here, you'd get exactly the same answer as we did above (Exercise 15.7.14).

Notice that once again we seem to have created a new number system, though this time with a square root of a positive, not negative number! (And yes, it turns out that finding solutions to things like this is related to $\mathbb{Z}[\sqrt{2}]\cdots$)

This *precisely* corresponds to multiplying a group element by 2, e.g.

$$[5] + [5] \equiv 3 \pmod 7 \text{ is the same type of thing as } (3,2) + (3,2) = (17,12) \,.$$

It turns out that there is a more general formula that corresponds to taking the line through two points and then moving it so that it goes through the original point $(1, 0)$:

Example 15.6.3. If (x_1, y_1) and (x_2, y_2) are both solutions of $x^2 - 2y^2 = 1$, then so is

$$(x_1x_2 + 2y_1y_2, x_1y_2 + y_1x_2)$$

If you apply this to two points opposite each other like $(3, 2)$ and $(3, -2)$, you will get

$$(3 \cdot 3 + 2 \cdot 2 \cdot (-2), 3 \cdot (-2) + 3 \cdot 2) = (1, 0)\,,$$

which is then the group identity.

```
@interact(layout=[['x_0','y_0'],['x_1','y_1'],
    ['auto_update']])
def _(x_0=3,y_0=2,x_1=17,y_1=12,auto_update=False):
    if x_0 != x_1:
        x_3,y_3=x_1*x_0+2*y_1*y_0,x_1*y_0+y_1*x_0
        pretty_print(html("Initial␣points␣were␣$(%s,%s)$␣
            and␣$(%s,%s)$;␣new␣point␣is␣
            $(%s,%s)$."%(x_0,y_0,x_1,y_1,x_3,y_3)))
        pretty_print(html("And␣indeed␣$%s^2-2\cdot%s^2$␣
            equals␣$%s$"%(x_3,y_3,x_3^2-2*y_3^2)))
    elif y_0==y_1:
        x_3,y_3=x_0^2+2*y_0^2,2*x_0*y_0
        pretty_print(html("Initial␣points␣were␣$(%s,%s)$␣
            and␣$(%s,%s)$;␣new␣point␣is␣
            $(%s,%s)$."%(x_0,y_0,x_1,y_1,x_3,y_3)))
        pretty_print(html("And␣indeed␣$%s^2-2\cdot%s^2$␣
            equals␣$%s$"%(x_3,y_3,x_3^2-2*y_3^2)))
    else:
        print "Input␣correct␣numbers!"
```

This ends up working for any n. Just change all the 2s above to n. Let's see this "by hand" for $n = 3$, where we solve $x^2 - 3y^2 = 1$ with

$$2^2 - 3 \cdot 1^2 = 1\,.$$

That is, I use

$$x' = x^2 + 3y^2 \text{ and } y' = 2xy$$

```
@interact
def _(x=2,y=1,auto_update=False):
    x,y=x*x+3*y*y,x*y+y*x
    pretty_print(html("$%s^2-3\cdot%s^2=%s$"%(x, y,
        x^2-3*y^2)))
    pretty_print(html("New␣point␣is␣$(%s,%s)$"%(x, y)))
```

15.6.3 The general solution (any n)

The general solution, given two points (x_1, y_1) and (x_2, y_2), would be, for $n > 0$ and *not* a perfect square,

$$x' = x_1x_2 + ny_1y_2 \text{ and } y' = x_1y_2 + x_2y_1\,.$$

Even more generally, the same formula works for combining solutions of *two* different equations like the Pell.

Fact 15.6.4.

$$\text{If } x_0^2 - ny_0^2 = k \text{ and } x_1^2 - ny_1^2 = \ell$$

$$\text{then } x = x_0x_1 + ny_0y_1,\ y = x_0y_1 + y_0x_1 \text{ solves } x^2 - ny^2 = k\ell\ .$$

Proof. See Exercise 15.7.15. □

This is particularly nice if $k = \ell = -1$, because getting a solution for that would then give a solution to the Pell equation!

Brahmagupta used analogous techniques for his time (and more sophisticated things) to solve very hard ones, as did the later English mathematicians who answered some challenges of Fermat.

Question 15.6.5.

- Find a nontrivial solution to $x^2 - 61y^2 = 1$.
- Find a nontrivial solution to $x^2 - 109y^2 = 1$.

Solution. The smallest solution to the second one is

$$x = 158070671986249, y = 15140424455100\ ,$$

which we can check below.

```
158070671986249^2-109*15140424455100^2
```

Considering that Brahmagupta says that finding the solution $x = 1151, y = 120$ to the equation $x^2 - 92y^2 = 1$ within a year proved the person "was a mathematician", we can be very thankful for computers!

15.7 Exercises

1. Do the algebra which we skipped in Exercise 15.7.1.

2. Do the algebra which we skipped in Example 15.1.4.

3. Find a parametrization (similar to Fact 15.1.1) for rational points on the following curves:

- The ellipse $x^2 + 3y^2 = 4$
- The hyperbola $x^2 - 2y^2 = 1$

4. Finish proving (Fact 15.1.5) that $x^2 + y^2 = 15$ cannot have any rational points.

5. Finish the proof that $x^3 - 117y^3 = 5$ has no integer solutions, looking modulo nine.

6. Show that the equation $x^3 = y^2 - 999$ has no integer solutions.

7. Fill in some (or all) of the details of Theorem 15.3.3.

8. Fill in the details of divisibility to finish Euler's 'proof' of Fact 15.3.4.

9. Look up the current best known bound on the number of integer points on a Mordell equation curve.

10. Get the tangent line to the Dudeney curve (see Question 15.2.1) and find the point of intersection; why can it not give an answer to the original problem?

11. Research Boyer's or Stigler's laws. What is the most egregious example of this, in your opinion?

12. Fill in the details of Example 15.5.4, and find an integer point with even bigger values.

13. Show that the Pell equation with $d = 1$ ($x^2 - y^2 = 1$) has only two solutions. Generalize this to when d happens to be a perfect square.

14. Show that the algebra in Fact 15.6.2 yields the same formulas as Proposition 15.6.1.

15. Verify that if

$$x_0^2 - ny_0^2 = k \text{ and } x_1^2 - ny_1^2 = \ell$$

then

$$x = x_0x_1 + ny_0y_1,\ y = x_0y_1 + y_0x_1 \text{ solves } x^2 - ny^2 = k\ell\ .$$

16. Explain why the previous problem reduces to the method from Section 15.5 where we were trying to use a tangent line to find more integer solutions.

17. Find a non-trivial integer solution to $x^2 - 17y^2 = -1$, and use it to get a nontrivial solution to $x^2 - 17y^2 = 1$.

18. Recreate the geometric constructions in Section 15.5 using tangent lines on the hyperbola with $x^2 - 5y^2 = 1$, and use it find three (positive) integer points on this curve with at least two digits for both x and y. Yes, you will have to find a non-trivial solution on your own; it's not hard, there is one with single digits.

Chapter 16

Solving Quadratic Congruences

We have been doing a lot of stuff now with squares. It is almost time to see one of the great theorems of numbers, which gives us great insight into the nature of squares in the integer world – and whose easiest proof involves lattice points!

This theorem, in the next chapter, will come from our trying to find the solution to a useful general problem, which I like to think of as the last piece of translating high school algebra to the modular world. That is the task of solving **quadratic congruences**, the modular equivalent to the well-known quadratic equations.

A quadratic congruence is just something of the form

$$ax^2 + bx + c \equiv 0 \pmod{n}$$

In algebra, we would use the quadratic formula. This chapter will see how far we can extend this to the modular world.

16.1 Square Roots

16.1.1 Recalling existing answers

To use the quadratic formula, one needs square roots in the real numbers. So too, our first task for modular arithmetic will be finding such square roots. Given our work in Chapter 7, e.g. Fact 7.2.2, it should be sufficient to solve

$$x^2 \equiv n \pmod{p},$$

finding square roots modulo p a prime.

We have already done some of this! We restate here a fact in the proof of Theorem 12.3.1 and the combination of Fact 13.3.2 and Lemma 13.3.3.

Fact 16.1.1. *The congruence $x^2 \equiv 1$ (mod p) always has two solutions, $x \equiv \pm 1$. So 1 has two square roots modulo p (or one, if $p = 2$).*

Fact 16.1.2. *The congruence $x^2 \equiv -1$ (mod p) does* not *always have solutions. It* does *when $p = 2$ or when $p \equiv 1$ (mod 4), and when it does there are two solutions, namely*

$$x \equiv \pm \left(\frac{p-1}{2}\right)!.$$

16.1.2 Finding more answers

We know the full answer (any modulus) for square roots of $+1$ from Fact 7.3.1. What about finding out when -1 has a square root for *non*-prime moduli? We can ask Sage about this:

```
var('x')
@interact
def _(n=50):
    for i in [2..n]:
        sols = [sol[0] for sol in solve_mod([x^2==-1],i)]
        l = len(sols)
        if l!=0:
            pretty_print(html("$x^2=-1\\text{ (mod }%s)$ 
                has $%s$ solutions, $%s$"%(i,l,sols)))
```

But recall that we actually can get a complete answer to this and similar questions by using Hensel's Lemma and the Chinese Remainder Theorem.

Algorithm 16.1.3. *To solve a polynomial modulo a given modulus, the following information suffices.*

- *If we can solve, for a given prime p,*
$$f(x) \equiv 0 \ (mod\ p)\ ,$$
then up to the technical condition about $\gcd(p, f'(p^{e-1})) = 1$ *we can solve*
$$f(x) \equiv 0 \ (mod\ p^e)\ .$$
- *If we can solve, for coprime integers p and q, $f(x) \equiv 0 \ (mod\ p)$ and (q), then we can solve*
$$f(x) \equiv 0 \ (mod\ pq)\ .$$

Example 16.1.4 (Prime power modulus)**.** For instance, let's go from p to p^2 by trying a bit of Example 7.2.4 from earlier. Here, $f(x) = x^2 + 1$ is what we want a solution for. If we are looking (mod 25), then we already know that (mod 5) we have $x \equiv 2$ as a solution. Then a solution (mod 25) will look like $2 + k(5)$ (review earlier where we did this), and, remembering that $f'(x) = 2x$, in fact it will satisfy

$$\frac{2^2 + 1}{5} + k(2 \cdot 2) \equiv 0 \ (\text{mod } 5)$$

which is $1 + 4k \equiv 0$, which has solution $k \equiv 1$; hence a solution (mod 25) should be $2 + 1(5) \equiv 7$, and the computations above verify this!

(Notice that $5 \nmid f'(2) = 4$, so the technical condition is granted; otherwise we'd have to solve $1 \equiv 0$!)

Example 16.1.5 (Composite moduli)**.** Similarly, if I want solutions to $x^2 \equiv -1$ (mod 14) I should immediately note that although $x^2 \equiv -1$ (mod 2) has a solution, $x^2 \equiv -1$ (mod 7) does *not* (it's a prime of the form $4k + 3$) so I can't use the CRT.

But if I am looking (mod 65), since $65 = 5 \cdot 13$ and $x^2 \equiv -1$ has solutions both (mod 5) and (mod 13), I can use the CRT to combine them:

- $x \equiv 2$ (mod 5)
- $x \equiv 5$ (mod 13)

So $x \equiv 2 \cdot 13 \cdot (13^{-1} \ (\text{mod } 5)) + 5 \cdot 5 \cdot (5^{-1} \ (\text{mod } 13)) \equiv 26 \cdot 2 + 25 \cdot 8 \equiv 252 \equiv 57$ (mod 65) And that also is consistent with the computations above!

16.2 General Quadratic Congruences

From the preceding section, it should be clear that, as far as just determining *whether* a solution exists, all we need to examine is prime moduli. Everything else is taken care of by previous work.

But it's not like $x^2 + k$ is the only quadratic game in town. What about other quadratics? It turns out that we can use something you are already familiar with to reduce the whole game to the following.

Question 16.2.1. For what primes p is there a solution to $x^2 \equiv k \pmod{p}$?

Let's confirm this with a look at general quadratic congruences.

First let's try computing. As an example, take $x^2 - 2x + 3 \pmod 9$. The Sage function `solve_mod` works, if a little naively.

```
solve_mod([x^2-2*x+3==0],9)
```

Sage note 16.2.2 (Commands of more sophistication)**.** Notice that the `solve_mod` command is more complicated than `divmod`. `solve_mod` returns a list of tuples, where a tuple of length one has a comma to indicate it's a tuple. (If you tried to solve a multivariate congruence you would find it returns a longer tuple.)

The result shows that $x^2 - 2x + 3 \equiv 0 \pmod 9$ has two solutions. But how might I solve a general quadratic congruence?

16.2.1 Completing the square solves our woes

The key is completing the square! First let's do an example.

Example 16.2.3. Completing the square for $x^2 - 2x + 3$ is done by

$$x^2 - 2x + 3 = \left(x^2 - 2x + \left(\frac{2}{2}\right)^2\right) + 3 - \left(\frac{2}{2}\right)^2 = (x-1)^2 + 2\,,$$

so solving the original congruence reduces to solving

$$(x-1)^2 \equiv -2 \pmod{n}$$

Then assuming I have a square root s of $-2 \pmod n$, I just compute $s + 1$ and I'm done! Go ahead and try this for a few different n, including of course $n = 9$, with Sage.

```
solve_mod([x^2==-2],9)
```

Should you not particularly enjoy completing the square, here is the basic idea.

Algorithm 16.2.4 (Completing the square modulo n)**.** *To complete the square for $ax^2 + bx + c \equiv 0$, the key thing to keep in mind is that* we do not actually divide *by $2a$, but instead multiply by $(2a)^{-1}$. Here are the steps.*

- *Multiply by four:* $4a^2x^2 + 4abx + 4ac \equiv 0$
- *Factor the square:* $(2ax + b)^2 - b^2 + 4ac \equiv 0$

- *Isolate the square:* $(2ax + b)^2 \equiv b^2 - 4ac$

So to solve, we'll need that $2a$ *is a unit (more or less requiring that* n *is odd), and then to find all square roots of* $b^2 - 4ac$ *in* $\mathbb{Z}_n$*.*

Fact 16.2.5. *The full solution to*

$$ax^2 + bx + c \equiv 0 \pmod{n}$$

is the same as the set of solutions to

$$x \equiv (2a)^{-1}(s - b) \pmod{n}, \text{ where } s^2 \equiv b^2 - 4ac \pmod{n}$$

Note that this means $\gcd(2a, n) = 1$ *must be true and that* $s^2 \equiv b^2 - 4ac$ *must have a solution.*

Example 16.2.6. Let's do all this with $x^2 + 3x + 5 \equiv 0 \pmod{n}$. Notice that $b^2 - 4ac = 9 - 20 = -11$, so this equation does *not* have a solution over the integers, or indeed over the real numbers. Does it have a solution in $\mathbb{Z}_n$ for some n, though?

```
L = [(n,solve_mod([x^2==-11],n)) for n in
    prime_range(3,100)]
for l in L:
    L1 = [m[0] for m in l[1]]
    modulus = l[0]
    pretty_print(html("Modulo␣$%s$,␣$x^2\equiv␣-11$␣has␣
        the␣solutions:␣%s"%(modulus,L1)))
    if L1 != []:
        try:
            LS = [mod(2*1,modulus)^(-1)*(m-3) for m in L1]
            pretty_print(html("For␣each␣of␣these,␣$x\equiv␣
                (2\cdot␣1)^{-1}(s-3)$:␣%s"%(LS)))
            LS = [ls^2+3*ls+5 for ls in LS]
            pretty_print(html("And␣$x^2+3x+5$␣gives␣for␣
                each␣of␣these:␣%s\n\n"%(LS)))
        except ZeroDivisionError:
            pretty_print(html("Since␣2␣doesn't␣have␣an␣
                inverse␣modulo␣$%s$,␣we␣can't␣use␣
                this.\n\n"%modulus))
```

16.3 Quadratic Residues

The previous section should really resolve that examining square roots suffices to a complete solution, so that is what not only the remainder of this chapter, but the next chapter, will focus on.

16.3.1 Some definitions

We now introduce two definitions, a little more formal in nature.

Definition 16.3.1. Assume that $a \not\equiv 0 \pmod{p}$, for p a prime.

- If there *is* a solution of $x^2 \equiv a \pmod{p}$ we say that a is a **quadratic residue** of p (or a **QR**).

- If there *is not* a solution of $x^2 \equiv a \pmod{p}$ we say that a is a **quadratic nonresidue** of p.

Note that this is the same thing as saying that a does or does not have a square root modulo p, but the focus changes to a instead of the square root itself.

It is not so easy at all to determine even *when* something is a QR, much less to compute the square roots, so we will take some significant time on this.

Remark 16.3.2. By the way, the terminology is explained by the fact (recall Section 4.4) that the equivalence classes $[a]$ are called residues, so one which is a perfect square is justly called quadratic.

Sage note 16.3.3 (Quadratic residues)**.** Sage can calculate these for us, of course.

```
quadratic_residues(17)
```

Notice that Sage counts zero as a quadratic residue (since $0^2 = 0$ always); there are technical reasons not to consider it as one in our theoretical treatment, as will be seen soon.

```
least_quadratic_nonresidue(17)
```

This last function gives the smallest nonresidue, in case you need it.

16.3.2 First try for square roots of two

To get more of a flavor for this, let's explore for which p it is true that $x^2 \equiv 2 \pmod{p}$ has a solution. Or, to put it another way, when does two have a square root modulo p?

First do some by hand; for what primes up to 20 does 2 have a square root?

Once you've done this, *then* evaluate the next Sage cell to look at more data.

```
@interact
def _(odd_primes_up_to=50):
    for p in prime_range(3,odd_primes_up_to):
        solutions=solve_mod([x^2==2],p)
        if len(solutions)!=0:
            pretty_print(html("$x^2\equiv␣2\\text{␣(mod␣
                }%s)$␣has␣solutions␣$%s$␣and␣
                $%s$"%(p,solutions[0][0],solutions[1][0])))
        else:
            pretty_print(html("No␣solutions␣modulo␣
                $%s$"%p))
```

Question 16.3.4. What do you think? Do you see any patterns?

As it turns out, it is *quite* hard to prove any such patterns you may find without the benefit of powerful theoretical machinery. Which means it is hard to even know *whether* there is a solution to a given congruence without such machinery!

16.3.3 Some history

In fact, it is even hard to conjecture such things for harder cases unless you are quite clever. Euler was one of the first to do so. In a very unusual paper, he included nary a proof, just closely related conjectures to this question. We list here three links for the paper. Note that if you read it carefully, you will have hints to the question in the previous subsection, with regard to numbers of the form $a^2 + 2b^2$!

- Euler archive listing
- Euler archive translation
- Euler's work explained by Ed Sandifer

Next, look at a table made by the great Italian-French mathematician Lagrange, courtesy of the French National Library and its online repository, Gallica.

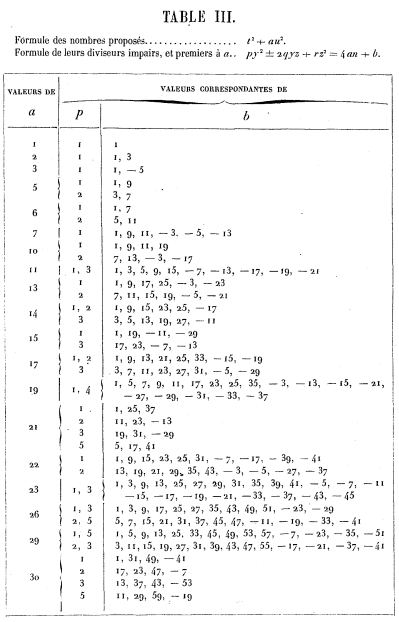

TABLE III.

Formule des nombres proposés..... $t^2 + au^2$.
Formule de leurs diviseurs impairs, et premiers à a.. $py^2 \pm 2qyz + rz^2 = 4an + b$.

VALEURS DE a	VALEURS CORRESPONDANTES DE p	VALEURS CORRESPONDANTES DE b
1	1	1
2	1	1, 3
3	1	1, −5
5	1	1, 9
	2	3, 7
6	1	1, 7
	2	5, 11
7	1	1, 9, 11, −3, −5, −13
10	1	1, 9, 11, 19
	2	7, 13, −3, −17
11	1, 3	1, 3, 5, 9, 15, −7, −13, −17, −19, −21
13	1	1, 9, 17, 25, −3, −23
	2	7, 11, 15, 19, −5, −21
14	1, 2	1, 9, 15, 23, 25, −17
	3	3, 5, 13, 19, 27, −11
15	1	1, 19, −11, −29
	3	17, 23, −7, −13
17	1, 2	1, 9, 13, 21, 25, 33, −15, −19
	3	3, 7, 11, 23, 27, 31, −5, −29
19	1, 4	1, 5, 7, 9, 11, 17, 23, 25, 35, −3, −13, −15, −21, −27, −29, −31, −33, −37
21	1	1, 25, 37
	2	11, 23, −13
	3	19, 31, −29
	5	5, 17, 41
22	1	1, 9, 15, 23, 25, 31, −7, −17, −39, −41
	2	13, 19, 21, 29, 35, 43, −3, −5, −27, −37
23	1, 3	1, 3, 9, 13, 25, 27, 29, 31, 35, 39, 41, −5, −7, −11 −15, −17, −19, −21, −33, −37, −43, −45
26	1, 3	1, 3, 9, 17, 25, 27, 35, 43, 49, 51, −23, −29
	2, 5	5, 7, 15, 21, 31, 37, 45, 47, −11, −19, −33, −41
29	1, 5	1, 5, 9, 13, 25, 33, 45, 49, 53, 57, −7, −23, −35, −51
	2, 3	3, 11, 15, 19, 27, 31, 39, 43, 47, 55, −17, −21, −37, −41
30	1	1, 31, 49, −41
	2	17, 23, 47, −7
	3	13, 37, 43, −53
	5	11, 29, 59, −19

In this table, Lagrange is referring to integers of the form $t^2 + au^2$, and then what form their divisors can have. That this corresponds to what we have seen is clear in that $a = 1$ just means that primes can divide a sum of squares if they are themselves of the form $y^2 + z^2$ when they are of the form $4n + 1$. (See the discussion around Theorem 13.5.5.) And if you were diligent in your pattern searching in the previous subsection, you will have found the significance for $8n + 1$ and $8n + 3$ with respect to $t^2 + 2u^2$. (For more on this and its history, see the excellent book *Mathematical Masterpieces* [C.4.7].)

Figure 16.3.5: Lagrange's Table III from "Recherches d'arithmétique"

Originally from what is now Italy, Lagrange was Euler's successor in Berlin after he went back to Russia under the stable (if despotic) regime of Catherine the Great. One interesting point to make about him is that Lagrange gave *proofs* of many of the patterns in quadratic forms (what numbers look like $a^2 + b^2$, $a^2 + 2b^2$, etc.) that Fermat and Euler talked about. Although he isn't always mentioned as highly as other contemporaries like Euler or Gauss, note that we've already seen two of his theorems (7.4.1 and 8.3.11). Later he moved to France and was quite influential there. And if you ever read any science fiction or space stuff that talks about stable places to orbit being called Lagrange points – that's him too!

16.4 Send in the Groups

What made things work out best in the end was a couple innovations of Lagrange's successor in Paris, Adrien-Marie Legendre. These were innovations Gauss made great use of.

One can approach this subject from many vantage points, including the historical one. However, we have the advantage of having developed the basics of groups and primitive roots, which will simplify much of our exposition.

16.4.1 Quadratic residues form a group

Definition 16.4.1. Consider the set of all non-zero quadratic residues modulo some prime p. We call this the **group of quadratic residues** Q_p.

This terminology suggests I had better have a proof in my pocket for the following theorem.

Theorem 16.4.2. *The set of non-zero quadratic residues Q_p modulo a prime p really is a group, and is even a subgroup of the group of units U_p.*

Proof. We will proceed by showing the group axioms hold under multiplication. Since this is a subset of U_p essentially by definition, that will imply it is a subgroup of it as well. Let's look at the three main axioms.

- It is clear that $1 \in Q_p$, since $1 \equiv 1^2$. So there is an identity.
- Next, if a and b are both in Q_p (with $a \equiv s^2$ and $b \equiv t^2$), then ab is also a quadratic residue (since $(st)^2 \equiv s^2 t^2 \equiv ab$).
- All that remains is to check that this set has inverses under multiplication.

To show this last, assume that $a \equiv s^2 \in Q_p$. Then note that

$$\left(s^{-1}\right)^2 a \equiv \left(s^{-1}\right)^2 s^2 \equiv \left(s \cdot s^{-1}\right)^2 \equiv 1$$

So *by definition* of inverses

$$\left(s^{-1}\right)^2 = a^{-1} \,,$$

which means that if $a \in Q_p$ then $a^{-1} \in Q_p$ as well. □

Remark 16.4.3. (For those with some additional algebraic background, it turns out Q_p is in fact a **quotient group** of U_p as well, but we will not delve further into this here.)

16.4.2 Quadratic residues connect to primitive roots

You might be wondering how this piece of U_p connects to the most important thing we've seen so far about U_p. Recall that U_p was *cyclic*, that it had a generator whose powers gave us all units modulo p. We called such an element a **primitive root** of p (recall Chapter 10).

```
g=mod(primitive_root(19),19);g
```

So let's compare the primitive root's powers and the quadratic residues. Shouldn't be too hard … if you aren't computing this with Sage, just try it with an even smaller modulus, like seven.

```
g=mod(primitive_root(19),19)
L=[g^n for n in range(1,19)]
print L
print quadratic_residues(19)
```

Note the pattern! This exemplifies a major fact.

Fact 16.4.4. *For odd prime modulus p, the quadratic residues are precisely the* even *powers of a primitive root g.*

Proof. Certainly $g^{2n} = (g^n)^2$, so the even powers of g are QRs.

Now we need the other set inclusion. Suppose that $a \in Q_p$ and $a = s^2$. Then first note that s and a are themselves still powers of g (by definition of g). So let $s = g^n$ and $a = g^m$ for some n, m. Then we have the implication

$$a = g^m \equiv g^{2n} \pmod{p} \implies m \equiv 2n \pmod{p-1} .$$

This is only possible if m is even since $p-1$ and $2n$ are even. □

This fact will turn out to be a fantastically useful *theoretical* way to find Q_p. It will show up in lots of proofy settings.

16.5 Euler's Criterion

However, Fact 16.4.4 is a *terrible* way to actually find quadratic residues for a given p, since it involves finding a primitive root and listing lots of powers! We need both theory and practice.

Here is a much easier way. Recall our observation in Theorem 12.3.1:

$$a^{(p-1)/2} \equiv \pm 1 \text{ for all } a \text{ not divisible by } p .$$

We visualized it as the middle column in this graphic.

```
@interact
def _(p=(13,prime_range(50))):
    P=matrix_plot(matrix(p-1,[mod(a,p)^b for a in
        range(1,p) for b in srange(p)]),cmap='jet')
    show(P,figsize=6)
```

But as so often in mathematics, solving one question leads to another; after all, we didn't say *when* we got plus or minus 1, just that these are the only possibilities. Let's extend this as follows.

Theorem 16.5.1 (Euler's Criterion)**.** *If p is an odd prime, then for all integers a not a multiple of p, the sign of the following expression determines whether a is a QR.*

$$a^{(p-1)/2} \equiv \pm 1 \pmod{p}$$

We obtain $+1$ if a is a QR, otherwise -1.

Proof. Let g be a primitive root of p, so that $a \equiv g^i$ for some i. Then if we let $h = g^{(p-1)/2}$, Fermat's Little Theorem shows that

$$h^2 = g^{p-1} \equiv 1 \pmod{p} .$$

Since g is primitive, $h \equiv 1$ is impossible, so $h \equiv -1$. But then

$$a^{(p-1)/2} \equiv \left(g^i\right)^{(p-1)/2} \equiv \left(g^{(p-1)/2}\right)^i \equiv h^i \equiv (-1)^i .$$

This is $+1$ when i is even and -1 when i is odd. According to Fact 16.4.4, this is precisely when a is a quadratic residue and nonresidue, respectively. □

Example 16.5.2. This immediately gives the result in Fact 16.1.2 that -1 has a square root modulo an odd prime p precisely when $p \equiv 1 \pmod 4$, because $(-1)^{(p-1)/2} = +1$ precisely when $(p-1)/2$ is even, or $p \equiv 1 \pmod 4$. That is *much* easier than our previous ad-hoc way of doing it!

Using this same idea, we can prove a very nice result about when a composite number is a QR.

Proposition 16.5.3. *If $n = ab$ is a factorization (not necessarily nontrivial) of n, then n is a QR of p precisely when either both a and b are QRs of p or both a and b are* not *QRs of p.*

Proof. Modulo p, write $a \equiv g^i$ and $b \equiv g^j$ for some i, j. Then $n = ab \equiv g^{i+j}$, and $i + j$ is even when i and j have the same parity. Because of Fact 16.4.4, this is exactly the same thing as the conclusion of the proposition. □

Hence if one of a, b is and the other isn't, neither is $n = ab$.

Example 16.5.4. Now I can immediately decide that $-2 \equiv 21$ is not a QR (mod 23). Why? First, $-2 \equiv 21$; then recall that -1 is not a QR but 2 is, which we knew before. Likewise, I can immediately decide that $-2 \equiv 9$ *is* a QR (mod 11) by the fact that neither -1 nor 2 is a QR.

```
quadratic_residues(23)
```

```
quadratic_residues(11)
```

16.6 The Legendre Symbol

We now introduce a new notation for this entire discussion. Why? Think on this; Gauss revolutionized number theory, and much of what made it possible (and accessible to others) was the notion of congruence, along with the symbol $\equiv$.

We alluded to a great innovation of Legendre's earlier, and it is likewise a concept *and* symbol. Earlier in his research into these questions, he discovered that certain powers were always either ± 1, omitting multiples of what we would today call the modulus. Some of what he found was essentially Theorem 16.5.1.

Like many mathematicians of the eighteenth century, Legendre worked in many areas, including function theory and mathematical physics. Notably, as increased professionalization of studies of higher mathematics came about in post-Revolutionary French engineering studies (a development Judith Grabiner argues led to rigorization of calculus), he wrote a widely used geometry textbook.

16.6.1 Introducing the Legendre symbol

What of the plus or minus 1? To quote an article [C.6.5] on this subject, if one had a symbol for it, it becomes

> … more than a notational convenience … Legendre reifies this concept, and makes it into an object of independent study.
>
> —Steven H. Weintraub

In our modern terms, he takes advantage of the fact that $a = g^i$ is an even power exactly when a is a QR, and $(-1)^i = 1$ precisely when i is even (and hence precisely when a is a QR). This is the so-called **Legendre symbol**. (However, he did not use the term QR, just the symbol.)

Definition 16.6.1. We write $\left(\frac{a}{p}\right)$ for the Legendre symbol.

$$\left(\frac{a}{p}\right) = 1 \text{ if } a \text{ is a QR modulo } p \text{ and } \left(\frac{a}{p}\right) = -1 \text{ if it's not.}$$

We define the Legendre symbol of a modulo p to be zero if $p \mid a$.

Example 16.6.2. We can restate Fact 16.1.2: We have that $\left(\frac{-1}{p}\right) = 1$ for p prime if and only if $p = 2$ or $p \equiv 1 \pmod 4$.

The command in Sage is pretty straightforward.

```
legendre_symbol(-2,11)
```

```
@interact
def _(p=(17,prime_range(50))):
    for n in [q for q in quadratic_residues(p) if q!=0]:
        pretty_print(html("$%s$ is a QR of $%s$ and
            $\left(\\frac{%s}{%s}\\right)=%s$"%(n, p, n,p,
            legendre_symbol(n,p))))
```

Remark 16.6.3. A brief note is in order regarding the special status of zero in Definition 16.3.1, especially since Sage includes zero as a QR.

On the one hand, this recognizes the special case that $0^2 = 0$, while $1 = 1^2 = (-1)^2$ (and everything else) usually have two square roots modulo a prime.

On the other hand, this also conveniently ignores the only integer from 0 to $p-1$ which is *not* in U_p, so that in fact you can think of $x \to \left(\frac{x}{p}\right)$ to be a function from U_p to $\{1, -1\}$ of the kind we call a **group homomorphism**. (Indeed, it gets us from a cyclic group of order $p-1$ to a cyclic group of order 2, with "kernel" a cyclic group of order $(p-1)/2$.)

16.6.2 Primitive roots and quadratic residues

Let's use our usual example to visualize the tension and connection between primitive roots and quadratic residues.

Question 16.6.4. Before looking at the next graphic, do you see why a quadratic residue automatically can't be a primitive root? (This follows from results earlier in this chapter; see Exercise 16.8.7.)

There is another way to view the tension between primitive roots and quadratic residues. Try the following graphic yet again.

```
@interact
def _(p=(13,prime_range(50))):
    P=matrix_plot(matrix(p-1,[mod(a,p)^b for a in
        range(1,p) for b in srange(p)]),cmap='jet')
    show(P,figsize=6)
```

All the residues are the second column (labeled 1), and all the quadratic residues are the colors in the third column (labeled 2 as they are squares). See how that column is symmetric about the middle of the rows, with two of each of the colors appearing. Note that these are the same colors as *every other* column in a row with a primitive root (the rows with every color represented), though not necessarily in that order.

Also notice that the color of the middle column determines whether the color beginning that row shows up in the second column. This seems weird – but is precisely the content of Section 16.5.

Here's a final fun thing. What is the sum of all Legendre symbols for a given prime? (As usual, you can do this by hand for small primes if you aren't computing.)

```
@interact
def _(p=(19,prime_range(100))):
    L = [legendre_symbol(a,p) for a in [0..p-1]]
    pretty_print(html("All␣Legendre␣symbols␣
        $\left(\\frac{a}{%s}\\right)$␣can␣be␣listed:"%p))
    print L
    pretty_print(html("And␣they␣sum␣up␣to␣$%s$"%sum(L)))
```

This is cool, and a nice example of the kind of fun one can have experimenting. What do you think? Do you think we can prove it? Try in Exercise 16.8.6.

16.7 Our First Full Computation

We will now completely calculate $\left(\frac{2}{p}\right)$ using Euler's Criterion to prove that our explorations in Section 16.3. (There are many proofs of this fact; a nice one using only the existence of a primitive root is [C.6.16].)

Theorem 16.7.1 (Quadratic Residues of Two)**.** *The quadratic residues of two modulo a prime p is as follows.*

- $\left(\frac{2}{p}\right) = 1$ *if* $p \equiv \pm 1$ *(mod* 8*)*
- $\left(\frac{2}{p}\right) = -1$ *if* $p \equiv \pm 3$ *(mod* 8*)*

Proof. We will try to show this by writing $(p-1)!$ in two different ways.

This is easiest to approach with an example first. We'll indicate different ideas to pay attention to with bullets.

- For instance, take $p = 11$. Then we can write

$$\begin{aligned} 10! &= 1(2)(3)(4)(5)(6)(7)(8)(9)(10) \\ &= (2 \cdot 4 \cdot 6 \cdot 8 \cdot 10) \cdot (1 \cdot 3 \cdot 5 \cdot 7 \cdot 9) \\ &= 2^5(1 \cdot 2 \cdot 3 \cdot 4 \cdot 5) \cdot (1 \cdot 3 \cdot 5 \cdot 7 \cdot 9)\,. \end{aligned}$$

  Notice that $1, 3, 5$ repeat; these are all the odd numbers less than or equal to $\frac{11-1}{2} = 5$.

- Now we will try to create 10! again from what is left. The only ones we need to change would be $1, 3, 5$, as I just said.

- But $-1, -3, -5 \equiv 10, 8, 6$, which are exactly the missing pieces to 10!. So I factor out -1 from those three, thus:

$$10! = 2^5(1{\cdot}2{\cdot}3{\cdot}4{\cdot}5){\cdot}(1{\cdot}3{\cdot}5{\cdot}7{\cdot}9) = 2^5(1{\cdot}2{\cdot}3{\cdot}4{\cdot}5){\cdot}(-1)^3{\cdot}(-1{\cdot}-3{\cdot}-5){\cdot}(7{\cdot}9)$$

$$\equiv 2^5(-1)^3(1 \cdot 2 \cdot 3 \cdot 4 \cdot 5)(10 \cdot 8 \cdot 6)(7 \cdot 9) \equiv (-1)^3 2^5 (10)!$$

- Finally, cancel 10! from both sides, and we get

$$1 \equiv 2^5(-1)^3 \Longrightarrow 2^{(11-1)/2} \equiv 2^5 \equiv (-1)^3 \equiv -1$$

  and so 2 is not a QR of 11.

Proving the general case basically follows this to its natural conclusion; there was nothing special in the above argument about $p = 11$.

After writing $(p-1)!$ and factoring out $2^{(p-1)/2}$, the "repeated" numbers will be the odd numbers between 1 and $(p-1)/2$. Clearly the only "missing" numbers are even ones between $(p-1)/2$ and p, which are just the negatives of these odd numbers, so the same argument as above with $(p-1)!$ will work.

- If $p \equiv 3 \pmod 4$, like $p = 11$, then $(p-1)/2$ is odd so there will be

$$\left(\frac{p-1}{2} - 1\right)\frac{1}{2} + 1 = \frac{p+1}{4}$$

  repeated factors, as $1, 3, 5$ above.

- If $p \equiv 1 \pmod 4$ (like $p = 13$), on the other hand, then $(p-1)/2$ is even and there are exactly

$$\left(\frac{p-1}{2}\right)\frac{1}{2} = \frac{p-1}{4}$$

  repeated factors (in that case, still $1, 3, 5$).

In either case, whether the number of repeated factors ($(p+1)/4$ or $(p-1)/4$, respectively) is even or odd determines whether 2 is a quadratic residue!

So in general:

- If $p \equiv 1 \pmod 4$ and $\frac{p-1}{4}$ is even, it works, so $p \equiv 1 \pmod 8$ *does* have 2 as a QR.

- If $p \equiv 1 \pmod 4$ and $\frac{p-1}{4}$ is odd, it does not work, so $p \equiv 5 \pmod 8$ *does not* have 2 as a QR.

- If $p \equiv 3 \pmod 4$ and $\frac{p+1}{4}$ is even, it works, so $p \equiv 7 \pmod 8$ *does* have 2 as a QR.

- If $p \equiv 3 \pmod 4$ and $\frac{p+1}{4}$ is odd, it does not work, so $p \equiv 3 \pmod 8$ *does not* have 2 as a QR.

□

The following Sage cell shows Theorem 16.7.1 off.

```
@interact
def _(p = (17,prime_range(3,100))):
    l = legendre_symbol(2,p)
    r = p%8
    pretty_print(html("The␣prime␣$%s\equiv␣%s\\text{␣(mod␣
        }8)$␣and␣
        $\left(\\frac{2}{%s}\\right)=%s$."%(p,r,p,l)))
```

16.8 Exercises

1. Prove that if $e > 1$, then there is no solution to

$$x^2 \equiv -1 \pmod{2^e}\,.$$

Use our knowledge of squares modulo 4.

2. For what n does -1 have a square root modulo n? (Hint: use prime factorization and the previous problem along with results earlier in the chapter.)

3. Clearly 4 has a square root modulo 7. Find all square roots of 4 modulo 7^3 without using Sage or trying all 343 possibilities.

4. Solve $x^2 + 3x + 5 \pmod{15}$ using completion of squares and trial and error for square roots.

5. Solve the following congruences without using a computer:

- $x^2 + 6x + 5 \pmod{17}$
- $5x^2 + 3x + 1 \pmod{17}$

6. Prove that if p is an odd prime

$$\sum_{a=1}^{p-1} \left(\frac{a}{p}\right) = 0\,.$$

7. Show that a quadratic residue can't be a primitive root if $p > 2$.

8. Use Euler's Criterion to find all quadratic residues of 13.

9. Use Euler's Criterion to prove that 2 has a square root modulo p if $p \equiv 1 \pmod 8$.

10. Evaluate all Legendre symbols for $p = 7$ using Euler's Criterion.

11. Explore for a pattern for when -5 is a quadratic residue. Try *not* to use any fancy criteria, but just to seek a pattern based on the number.

12. Explore for a pattern for, given p, how many pairs of *consecutive* residues are both actually quadratic residues. Then connect this idea to the following formula, which you should evaluate for the same values of p:

$$\sum_{a=1}^{p-2} \left(\frac{a}{p}\right)\left(\frac{a+1}{p}\right)$$

(A harder problem is to prove your evaluation works for all p.)

Chapter 17

Quadratic Reciprocity

So far, we have determined at least when *some* quadratic congruences have solutions, but at the pace set thus far, most cases should seem beyond reach. We certainly won't want to use Theorem 16.5.1 directly for every single one.

Yet one might think we must, and that our task of finding out when numbers have square roots (mod p) is hopeless. But quite the opposite is true! In fact, we will derive results that took quite a bit of work almost effortlessly using the great theorem that is the title of this chapter.

17.1 More Legendre Symbols

To begin with, let's get some more intuition by trying to calculate some more Legendre symbols. Remember, we have several interesting basic properties, not just Euler's criterion. In the previous chapter we proved the following as Proposition 16.5.3:

Proposition 17.1.1. *If $n = ab$ is a factorization (not necessarily nontrivial) of n, then n is a QR of p precisely when either both a and b are QRs of p or both a and b are* not *QRs of p. (Hence if one of a, b is and the other isn't, neither is n.)*

In terms of Legendre symbols, it means

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$$

There is another useful computational fact that comes from the observation that $x^2 \equiv a \pmod{n}$ if and only if $x^2 \equiv a + n \pmod{n}$.

Proposition 17.1.2.

$$\left(\frac{a+p}{p}\right) = \left(\frac{a}{p}\right)$$

So we can use these ideas to calculate a lot more Legendre symbols!

Example 17.1.3. Let's compute a few this way.

What is $\left(\frac{62}{19}\right)$? On the one hand,

$$\left(\frac{62}{19}\right) = \left(\frac{62-19-19-19}{19}\right) = \left(\frac{5}{19}\right)$$

but we don't know this yet either. On the other hand,

$$\left(\frac{62}{19}\right) = \left(\frac{62+19}{19}\right) = \left(\frac{81}{19}\right)$$

Since 81 is a perfect square regardless, this equals 1.

Let's try $\left(\frac{8}{19}\right)$. Here, factoring will help;

$$\left(\frac{8}{19}\right) = \left(\frac{4}{19}\right) \cdot \left(\frac{2}{19}\right)$$

Since 4 is a perfect square, its symbol is one, and by Theorem 16.7.1 we know that two is not a QR modulo 19, so we get $1 \cdot -1 = -1$ and eight doesn't have a square root there either.

Before continuing, alternately try each of these strategies until you either get to a perfect square or a number you already know is (or isn't) a residue. (See also Exercise 17.7.3.)

- $\left(\frac{55}{17}\right)$
- $\left(\frac{83}{17}\right)$
- $\left(\frac{45}{17}\right)$
- $\left(\frac{41}{31}\right)$
- $\left(\frac{27}{31}\right)$
- $\left(\frac{22}{31}\right)$

Sage note 17.1.4 (Check your work)**.** You can always check your work, if you wish, using Sage.

It turns out you can resolve theoretical questions this way too.

Fact 17.1.5. *There are always consecutive quadratic residues for $p > 5$.*

Proof. First, we know that $1, 4, 9$ are all quadratic residues. Thus, if at least one of $2, 5, 10$ was *also* a QR, then we could guarantee that there were *always* consecutive quadratic residues somewhere!

As it turns out, if $p = 5$ this doesn't work, because the only (nonzero) QRs are ± 1 for that prime. But if $p = 7$, then $a = 1$ and $a = 9 \equiv 2$ are consecutive.

Now suppose $p > 7$ is prime. Then at least *one* of $2, 5, 10$ *must* be a QR, since one of these things must be true:

- 2 could be a QR
- 5 could be a QR
- If 2 and 5 both aren't, then

  $$\left(\frac{10}{p}\right) = \left(\frac{2}{p}\right)\left(\frac{5}{p}\right) = (-1)(-1) = 1$$

  means 10 is!

□

Thus we see that calculation and theory must go hand in hand; they are not separate.

17.2 Another Criterion

Now, we might want to do something more general than just try to compute Legendre symbols one by one. Notice that what we did in using the Euler's Criterion to find $\left(\frac{2}{p}\right)$ was to look at numbers like $2x$. So one might ask whether something like this *calculation* could work with general a and numbers like ax to find a better *theoretical* result.

It turns out that this is true. We are going to follow the steps of Gauss' protege Eisenstein here to find a way to evaluate $\left(\frac{a}{p}\right)$ for p an odd prime and $\gcd(a,p)=1$. It will be slow, and we won't see the payoff until we prove Theorem 17.4.1, but it *will* give us good practice in thinking about the numbers themselves.

Remark 17.2.1. Eisenstein was yet another brilliant young mathematician who came out of nowhere but died young because he couldn't find a job which could help his chronic illness. (I say "yet another" because this is similar to the story of Abel (after whom Abelian groups are named), and quite likely would have been the story of Galois if he hadn't been killed in a duel first).

17.2.1 Laying the foundation

First, let's introduce a new set and look at a couple properties. I *strongly* advise following along with a prime like $p=11$ or $p=13$.

Definition 17.2.2. Fix an odd prime p. Let E be the set of even numbers less than p. That is,

$$E = \{2, 4, 6, \ldots, p-1\}$$

Then call the set of multiples of a by even numbers aE (reduced modulo p) so that

$$aE = \{2a, 4a, 6a, \ldots, (p-1)a\}$$

Claim 17.2.3. *The set of numbers*

$$\{(-1)^x x \mid x \in aE\}$$

is the same as the set E, considered as (least nonnegative) residue classes modulo p.

Proof. First, both sets are all evens.

- If x is even, then $(-1)^x x$ is just x.
- If x is odd, then $(-1)^x x$ is equivalent to $p-x$, which (as the difference of two odds) is also even.

Second, the elements of the set in question are all different. Why?

- If any two such numbers were the same, then for some even numbers e and e' we have
$$(-1)^{ae} ae \equiv (-1)^{ae'} ae'$$
- Since $\gcd(a,p)=1$ we can remove the a from the previous congruence and get that
$$e \equiv \pm e'$$
- Finally, if e and e' are different then $e \equiv -e'$ so $e+e' \equiv 0$. But $e+e'=2p$ is not possible since they are smaller than p, and $e+e'=p$ is not possible since p is odd. The only remaining choice is that $e=e'$.

□

Example 17.2.4. For instance, with $p = 11$ and $a = 3$ we get $E = \{2, 4, 6, 8, 10\}$ and the set in the claim as

$$\{(-1)^6 6, (-1)^1 1, (-1)^7 7, (-1)^2 2, (-1)^8 8\} \equiv \{6, 10, 4, 2, 8\}$$

17.2.2 Getting the new criterion

Now we will try to use this set to arrive at something similar to Euler's criterion. Our goal would be to actually have that in it, but with something different to compute, so we would need to arrive at $a^{(p-1)/2}$ in the end. Let's follow some steps that might lead us in that direction.

- Crucially, notice first that the exponent is exactly the number of elements in E. So to get this, we could *multiply* all of them:
$$\prod_{e \in E} ae = a^{(p-1)/2} \prod_{e \in E} e$$
- With that in mind, let's give a name to the least positive residues modulo p of each ae for $e \in E$; call them r_e. Then the previous statement, modulo p, is
$$\prod_{e \in E} r_e \equiv a^{(p-1)/2} \prod_{e \in E} e$$
(Note the change from $=$ to $\equiv$.)
- If we focus temporarily just on the product of es, then using Claim 17.2.3 and adding all the powers of (-1) in question, we can write
$$\prod_{e \in E} e \equiv \prod_{e \in E} (-1)^{r_e} r_e \equiv (-1)^{\sum_{e \in E} r_e} \prod_{e \in E} r_e$$
- Now we substitute everything from the previous points together and move minus signs via multiplication. This gets us that
$$\prod_{e \in E} r_e \equiv a^{(p-1)/2} \prod_{e \in E} e \equiv a^{(p-1)/2} (-1)^{\sum_{e \in E} r_e} \prod_{e \in E} r_e$$
which means, since dividing and multiplying by powers of (-1) is the same thing,
$$a^{(p-1)/2} \equiv (-1)^{\sum_{e \in E} r_e}.$$

Example 17.2.5. For instance, with $p = 11$ and $a = 3$ we get

$$6 \cdot 12 \cdot 18 \cdot 24 \cdot 30 \equiv 6 \cdot 1 \cdot 7 \cdot 2 \cdot 8 \equiv 2^5 (-1)^{6+1+7+2+8} 6 \cdot 1 \cdot 7 \cdot 2 \cdot 8$$

Checking, we see that $6+1+7+2+8$ is even. So by Euler's criterion a should be a QR modulo p, and $11 + 11 + 3 = 25 = 5^2$ so in this case it is.

More generally, we have the following resulting fact.

Fact 17.2.6.

$$\left(\frac{a}{p}\right) = (-1)^{\sum_{e \in E} r_e}$$

Proof. Use Euler's Criterion and the above steps. □

What have we done? We have reduced evaluating the Legendre symbol (and hence deciding whether things have square roots modulo p) to calculating the *parity* of a certain sum. Hopefully that's an improvement on taking powers.

Remark 17.2.7. Transforming such computations to a simple parity (or other) check is very common in algebra and number theory.

17.2.3 The final form

Of course, Fact 17.2.6 is still somewhat unwieldy, so there is a final simplification.

First, where do these r_e come from anyway? Well, for $e \in E$, they are the least positive residues, and that means they are precisely what we get from the Division Algorithm:

$$ae = p\left\lfloor \frac{ae}{p} \right\rfloor + r_e$$

So if we add them all up, we get

$$\sum_{e\in E} r_e = \sum_{e\in E} ae - p\sum_{e\in E}\left\lfloor \frac{ae}{p} \right\rfloor$$

But we only care about the *parity* of this sum! So we can remove the whole piece with e in it, as that's all even, and we can replace the $-p$ by 1, since they are the same modulo 2.

Theorem 17.2.8 (Eisenstein's Criterion for the Legendre Symbol)**.**

$$\left(\frac{a}{p}\right) = (-1)^{\sum_{e\in E}\left\lfloor \frac{ae}{p} \right\rfloor}.$$

Remark 17.2.9. The name of the criterion is long to avoid confusion with *another* famous criterion Eisenstein came up with. (See David Cox's excellent 2011 Monthly article [C.6.4], which won the Lester R. Ford award, on whether Theodor Schönemann deserves the credit for *that* criterion too.)

Example 17.2.10. To continue the example with $p = 11$ and $a = 3$, let's compute this exponent.

$$\left\lfloor \frac{6}{11} \right\rfloor + \left\lfloor \frac{12}{11} \right\rfloor + \left\lfloor \frac{18}{11} \right\rfloor + \left\lfloor \frac{24}{11} \right\rfloor + \left\lfloor \frac{20}{11} \right\rfloor = 0+1+1+2+2 = 6$$

Once again, this is even so 3 is confirmed to be a QR modulo 11.

This very abstruse-seeming criterion will actually be the key to proving the soon-to-come Theorem 17.4.1. See Laubenbacher and Pengelley's article [C.6.8] for an excellent exposition which is elaborated on significantly above.

17.3 Using Eisenstein's Criterion

Let's calculate for a bit using this criterion. It says that we can tell whether a number a has a square root modulo p simply by checking whether $\sum_{\text{even } e,\ 0<e<p}\left\lfloor \frac{ae}{p} \right\rfloor$ is even or odd. So let's apply it to evaluating $\left(\frac{3}{p}\right)$ for odd primes p. Equivalently, we can answer this question:

Question 17.3.1. When does 3 have a square root modulo p?

If you liked some of the integer-point counting arguments earlier, you will like this.

For this case, we care about

$$\sum_{\text{even } e,\ 0<e<p} \left\lfloor \frac{3e}{p} \right\rfloor .$$

Said another way, we are adding the integer parts of $\frac{y}{p}$ for y a multiple of six less than $3(p-1)$.

Example 17.3.2. Let's try with $p = 7$: We have

$$\left\lfloor \frac{6}{7} \right\rfloor + \left\lfloor \frac{12}{7} \right\rfloor + \left\lfloor \frac{18}{7} \right\rfloor = 0 + 1 + 2 = 3$$

so $7 \not\equiv s^2$.

What about with $p = 11$? The criterion would be for

$$\left\lfloor \frac{6}{11} \right\rfloor + \left\lfloor \frac{12}{11} \right\rfloor + \left\lfloor \frac{18}{11} \right\rfloor + \left\lfloor \frac{24}{11} \right\rfloor + \left\lfloor \frac{30}{11} \right\rfloor = 0 + 1 + 1 + 2 + 2 = 6$$

This is even, and we already saw several times in the previous section that this correctly implies 3 is a QR.

What will a fact like this look like in general? All we care about is the *parity* of this sum. So, we can really ignore the terms in the sum that are 0 or 2, as they won't *change* the parity! That means we are really only looking at $\left\lfloor \frac{3e}{p} \right\rfloor$ for $3e$ that are between p and $2p$, since ones less than p are 0 and there can't be any number bigger than $3p$ if we only let e go up to $e = p - 1$.

This means we are considering precisely even e such that $p < 3e < 2p$, or all integers y such that the multiples of 6 give

$$p < 6y < 2p \Rightarrow \frac{p}{6} < y < \frac{p}{3} .$$

We've reduced to the *parity* of the *cardinality* of this small set of integers.

It should be clear that when p moves above or below a multiple of six, this might change how many are in between. So it seems reasonable to look at primes of the form $p = 6k + r$ when examining this. That gives

$$\frac{p}{6} < y < \frac{p}{3} \Rightarrow \frac{6k+r}{6} < y < \frac{6k+r}{3} \Rightarrow k + \frac{r}{6} < y < 2k + \frac{r}{3}$$

$$\Rightarrow \frac{r}{6} < y < k + \frac{r}{3} .$$

(This works because the cardinality of the sets will be the same if we subtract an integer.)

Claim 17.3.3. *Both of the numbers we are adding to get the parity we are after can be easily computed:*

- *The parity of k.*
- *The parity of the size of the set of integers y such that $\frac{r}{6} < y < \frac{r}{3}$.*

The sum of these two parities should be the parity of the set between $\frac{r}{6}$ and $k + \frac{r}{3}$.

Proof. These can be easily dealt with.

- The parity of k has two options.
  - If k is even, then $k = 2\ell$ and $p = 6k + r = 12\ell + r$.
  - If not, then $k = 2\ell + 1$ and $p = 6k + r = 12\ell + 6 + r$.
- There are two possible residues r modulo 6 for prime p. Either $r = 1$ or $r = 5$.
  - If $r = 1$, we are looking for y such that $\frac{1}{6} < y < \frac{1}{3}$, of which there are none.
  - If $r = 5$, we are looking for y such that $\frac{5}{6} < y < \frac{5}{3}$, of which there is one.

□

Proposition 17.3.4. *Three is a quadratic residue (or not) in the following circumstances.*

- $\left(\frac{3}{p}\right) = 1$ *if* $p \equiv \pm 1$ *(mod* 12*)*
- $\left(\frac{3}{p}\right) = -1$ *if* $p \equiv \pm 5$ *(mod* 12*)*

Proof. Combine the facts in Claim 17.3.3. We see that

- If $p = 12\ell + 1$ we add two even numbers, so 3 *is* a QR.
- If $p = 12\ell + 5$, we add an even number and 1, so 3 is *not* a QR.
- If $p = 12\ell + 6 + 1 = 12\ell + 7$, we add an odd and zero, so 3 is *not* a QR.
- If $p = 12\ell + 6 + 5 = 12\ell + 11$, we add an odd and 1, which is even, so 3 *is* a QR.

□

Try it!

```
for p in prime_range(5,50):
    pretty_print(html("$%s\equiv␣%s\\text{␣(mod␣}12)$␣and␣
        $\left(\\frac{3}{%s}\\right)=%s$"%(p,p%12,p,
        legendre_symbol(3,p))))
```

17.4 Quadratic Reciprocity

Now, if we had to do this prime by prime, it would still be horrible.

Instead, we will end up computing *all* Legendre symbols $\left(\frac{a}{p}\right)$ with $a \neq -1, 2$ by reducing them to $\left(\frac{-1}{p}\right)$ or $\left(\frac{2}{p}\right)$ using techniques from Section 17.1 and the following theorem.

As we've already alluded more than once, this theorem is venerable. Parts of it were conjectured and proved by Euler, and all of it was conjectured by Legendre in terms of remainders (some commentators say he proved it as well). Carl Friedrich Gauss provided no fewer than eight proofs over the course of his lifetime. See Subsection 17.6.3 for a few more comments.

17.4.1 The theorem

Theorem 17.4.1 (Quadratic Reciprocity)**.** *If p and q are odd primes not equal to each other, then*

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\left(\frac{p-1}{2}\right)\left(\frac{q-1}{2}\right)}.$$

Proof. See Section 17.6. □

Remark 17.4.2. We can multiply the fractions to rewrite it in a way some authors prefer:

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}$$

Example 17.4.3 (Computing with QR)**.** We immediately apply this to vastly simplify the calculations in Section 17.3. Let $q = 3$ and $p > 3$.

Let's write the theorem out for this case. Since $(3-1)/2 = 1$, we have

$$\left(\frac{3}{p}\right)\left(\frac{p}{3}\right) = (-1)^{(p-1)/2}, \text{ or } \left(\frac{3}{p}\right) = (-1)^{(p-1)/2}\left(\frac{p}{3}\right).$$

There are two parts to this:

- Since $1 \in Q_3$ and $2 \notin Q_3$, the Legendre symbol on the right is:

$$\left(\frac{p}{3}\right) = 1 \text{ if } p \equiv 1 \pmod 3 \text{ and } \left(\frac{p}{3}\right) = -1 \text{ if } p \equiv 2 \pmod 3.$$

- We can also compute the power of -1:

$$(-1)^{(p-1)/2} = 1 \text{ if } p \equiv 1 \pmod 4 \text{ and } (-1)^{(p-1)/2} = -1 \text{ if } p \equiv 3 \pmod 4.$$

Combine these together and we get that $\left(\frac{3}{p}\right) = 1$ exactly when one of these two cases occurs:

- $p \equiv 1 \pmod 3$ and mod (4)
- $p \equiv 3 \pmod 4$ and $\equiv 2 \pmod 3$

This is precisely $p \equiv 1, 11 \equiv \pm 1 \pmod{12}$ as in Proposition 17.3.4!

It's amazing that this can work so easily.

17.4.2 Why is this theorem different from all other theorems?

17.4.2.1 What does it mean?

What does the term "quadratic reciprocity" even mean?

It means that there is a *reciprocating* relationship between Legendre symbols, and hence between whether there is a square root of two primes modulo each other.

For instance, it says that *usually* the following matrix is symmetric – and that we have a simple formula for when it isn't.

```
ls=prime_range(3,50)
M=matrix(len(ls),[legendre_symbol(a,b) for a in ls for b
    in ls])
show(block_matrix(2,[0, matrix(1,len(ls),ls),
    matrix(len(ls),1,ls), M]))
```

Here is another way to say it. For odd primes p and q,

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$$

except when $p \equiv q \equiv 3 \pmod 4$.

17.4.2.2 What does it do?

What does quadratic reciprocity do?

It makes computation of Legendre symbols very, very easy if you have a prime factorization of p (and all the intermediate steps). You just need to use the following facts we already proved, in addition to quadratic reciprocity.

- $\left(\frac{-1}{p}\right) = 1 \iff p \equiv 1 \pmod 4$
- $\left(\frac{2}{p}\right) = 1 \iff p \equiv \pm 1 \pmod 8$

Algorithm 17.4.4. *Any Legendre symbol can be computed using the following steps, not necessarily in this order and often multiple times:*

- *Factor the top and use Proposition 16.5.3, then computing each one separately.*
- *Reduce modulo the bottom and/or use Proposition 17.1.2 to get convenient tops (especially perfect squares).*
- *When you get to an odd prime on the top and bottom, use Theorem 17.4.1.*
- *When the top is -1 or 2, use Example 16.6.2or Theorem 16.7.1 to finish your computation.*

Proof. Read the chapter up to this point, plus the proof of Theorem 17.4.1. □

Example 17.4.5. Let's calculate $\left(\frac{99}{167}\right)$.

- Since they are coprime factors, $\left(\frac{99}{167}\right) = \left(\frac{9}{167}\right) \cdot \left(\frac{11}{167}\right)$.
- Since both 11 and 167 are prime and congruent to 3 modulo four, $\left(\frac{9}{167}\right) \cdot \left(\frac{11}{167}\right) = \left(\frac{3^2}{167}\right) \cdot - \left(\frac{167}{11}\right)$
- Reducing, we get $\left(\frac{3^2}{167}\right) \cdot - \left(\frac{167}{11}\right) = -1 \cdot \left(\frac{2}{11}\right)$
- Finally, we use Theorem 16.7.1 and note that $11 \equiv 3 \pmod 8$ to get $-1 \cdot \left(\frac{2}{11}\right) = -1 \cdot -1 = 1$ and we see that ninety-nine is a QR modulo one hundred sixty-seven.

Example 17.4.6. In a classroom experience, try these. (Else, see Exercise 17.7.15.)

- $\left(\frac{83}{103}\right)$
- $\left(\frac{219}{383}\right)$

And we can check them, of course.

```
legendre_symbol(83,103),legendre_symbol(219,383)
```

We can also come up with congruence criteria like above for other primes. See the exercises, such as 17.7.5 and 17.7.17.

17.4.2.3 The Jacobi symbol

What else does quadratic reciprocity do? Indirectly, it allows us to compute Legendre symbols $\left(\frac{a}{p}\right)$ *without* factoring a.

Definition 17.4.7. Let n be an *odd* number which factors as

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \ .$$

Then the **Jacobi symbol** , $\left(\frac{a}{n}\right)$, is just the product of the relevant Legendre symbols:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

Amazingly, the Jacobi symbol has all the same properties the Legendre symbol has – even quadratic reciprocity and the values for $a = -1, 2$. And if

$$\left(\frac{a}{n}\right) = -1$$

then a is not a QR of n. The only thing *not* the same is:

Fact 17.4.8. *If n is not prime, then $\left(\frac{a}{n}\right) = 1$ does not necessarily imply a is a QR of n.*

Sage note 17.4.9 (Names of functions may vary)**.** In Sage, this is named after yet another generalization called the Kronecker symbol.

```
kronecker_symbol(8,15), quadratic_residues(15)
```

The goal of introducing this is not to use the definition of the Jacobi symbol to do anything. That would be pointless.

Instead, the idea is that you can use the Jacobi symbol to do your calculation of Legendre symbols! After all, they follow *almost* all the same rules. You'd only need to factor here in order to make sure you don't have an even number in the denominator of the symbol.

It turns out that, unlike factoring (as far as we know), this leads to an algorithm which needs only about the square of the number of digits of p steps to evaluate the symbol, which is much better than one would need if one had to factor first.

Some examples would be just as fast doing it either way, like $\left(\frac{83}{103}\right)$. But others would be much slower, because you'd have to factor a few different times. Here's an example; note that 943 is not prime.

Example 17.4.10.

$$\left(\frac{943}{997}\right) = \left(\frac{997}{943}\right) \text{ since } 997 \equiv 1 \pmod 4$$

$$= \left(\frac{54}{943}\right) = \left(\frac{2}{943}\right)\left(\frac{27}{943}\right) = (+1)\left(\frac{27}{943}\right) \text{ since } 943 \equiv -1 \pmod 8$$

$$= -\left(\frac{943}{27}\right) \text{ since both are } \equiv 3 \pmod 4$$

$$= -\left(\frac{25}{27}\right) = -1 \text{ because } 25 = 5^2$$

And we can check this out with Sage:

```
kronecker_symbol(943,997)
```

Compare this example with having to first factor 943 and then still do the whole reciprocity dance, and this is much easier and more automatic for a computer to do. (By the way, factoring $943 = 23 \cdot 41$ is itself not a gimme 'by hand'.)

Before we go one, if you haven't tried to compute lots of things with quadratic reciprocity, don't go on until you do. You won't appreciate the power and usefulness of the proof until then. It's just the way these things are.

17.5 Some Surprising Applications of QR

What else can quadratic reciprocity do? The answer is, a *lot*.

17.5.1 Factoring, briefly

As an example, it can help us with factoring large integers n; Gauss did this. Describing the process itself is just a little too long to allow its inclusion, but it's important to get the flavor.

The essential idea is that if a is a QR of n, then a is a QR of any prime $p \mid n$. So since QRs often have congruence conditions associated with them, n must obey *all* of the congruence conditions for $\left(\frac{a}{p}\right)$ for all the p which divide it. Which might be a lot of conditions.

Then we can use a variant on the Fermat factoring method to check for possible a for which a prime divisor p of n definitely is or definitely is not a QR (which quadratic reciprocity can help with), and then one can compute Legendre/Jacobi symbols of possible $p \mid n$ to reduce to just having to check a very few bigger possible prime factors.

17.5.2 Primality testing

Another application is that it can help us check primality.

One specific place where it is helpful is with the so-called Fermat numbers. Recall (Subsection 12.1.1) that Euler blasted the following conjecture of Fermat's out of the water:

$$F_n = 2^{2^n} + 1 \text{ is always prime for } n \geq 0\,.$$

But what about bigger ones; surely they are inaccessible to the usual factoring techniques?

Just like with Mersenne numbers (Subsection 12.1.3), for which the Lucas-Lehmer test can check for primality (remember GIMPS?), there is a test called Pépin's test which can check for primality of Fermat numbers. (Pépin did this work in the late 1800s.) It turns out that no bigger Fermat numbers have turned out to be prime (all the way through $n = 31$); see the relevant member of the excellent Prime Pages for more information.

Here is the test in Sage:

```
@interact
def _(n=(1,[1..6])):
    F=2^(2^n)+1
```

```
    pretty_print(html("The␣$%s$th␣Fermat␣number␣is␣
        $%s$"%(n,F)))
    test = mod(3,F)^((F-1)/2)
    if test == -1:
        pretty_print(html("Since␣$3^{(%s-1)/2}\equiv␣%s$,␣
            this␣Fermat␣number␣is␣prime"%(F,test)))
    else:
        pretty_print(html("Since␣$3^{(%s-1)/2}\equiv␣%s$,␣
            this␣Fermat␣number␣is␣not␣prime"%(F,test)))
```

You can already see from this code that it is checking Euler's criterion mod (F_n), and looking for a negative answer. Why would this test primality? Let's formally state and prove the criterion.

Fact 17.5.1. *For $n > 0$, $F_n = 2^{2^n} + 1$ is prime exactly when*

$$3^{2^{2^n-1}} \equiv -1 \ (mod\ 2^{2^n} + 1)$$

Proof. First, let's assume F_n is prime. Since F_n is one more than a multiple of four, it is clearly

$$F_n \equiv 1, 5, \text{ or } 9 \pmod{12} .$$

- If $F_n \equiv 1 \pmod{12}$, then $3 \mid 2^{2^n} = F_n - 1$, which cannot be true.
- If $F_n \equiv 9 \pmod{12}$, then F_n is a number greater than three which is divisible by three – but it's prime, so that's not possible.
- So $F_n \equiv 5 \pmod{12}$.

Since F_n is prime, that means by Proposition 17.3.4 we know $3 \notin Q_p$.

For the converse, let's assume that Euler's criterion gives this answer of -1. Then square both sides to get

$$3^{F_n - 1} \equiv 1 \pmod{p}$$

for all primes p dividing F_n. Now, what order does 3 have here?

- Since $F_n - 1 = 2^{2^n}$, that means 3 has order some power of 2 (in U_p).
- But 3 can't have order 2^{2^n-1} (or less), because it isn't the identity when taken to that power.
- So it must have order 2^{2^n}.

The only way 3 can have *that* big of an order is if p is at least $2^{2^n} + 1 = F_n$! So since $p \mid F_n$, they must be equal. □

17.5.3 Solving equations

There is even more! As one example, quadratic reciprocity (or at least the Legendre symbol) helps us solve Mordell equations (e.g. Subsection 15.3.2). The easiest cases use $\left(\frac{2}{p}\right)$ and multiplicativity. But more advanced ones need to compute more complicated square roots. Here are two examples, without proof; there are many others.

- The equation $x^3 = y^2 + 16$ has no integer solutions. (Uses $\left(\frac{-8}{p}\right)$.)
- The equation $x^3 = y^2 - 46$ has no integer solutions. (Uses $\left(\frac{-18}{p}\right)$.)

17.5.4 Artin's conjecture

Let's return to the test for F_n's primality in Fact 17.5.1. A careful look at the proof shows that 3 is a primitive root for F_n, if F_n is prime. Thus, if we had infinitely many Fermat primes (and not just five of them), we'd have an integer which is a primitive root of infinitely many primes.

Such would provide a proof of at least one *explicit* case for the following long-standing question.

Conjecture 17.5.2 (Artin's Conjecture)**.** *Every nonsquare integer except* -1 *is a primitive root for infinitely many primes.*

This conjecture is interesting for several reasons.

- Although it is mostly believed to be true, currently there are *no* integers known to be a primitive root for infinitely primes.
- Weirder, it is known that at least one of 3, 5, and 7 does in fact satisfy this (that is, at least one of 3, 5, or 7 is a primitive root for infinitely many primes) but we don't know which one!
- Weirdest, it has been proved that there are *at most two* exceptions to this conjecture, yet we also know of no integers which do not satisfy it!

  That is, there are at most two nonsquare integers which are not a primitive root for infinitely many primes, yet we do not have a single specific integer which we can prove that for.

There is some historical connection as well. Gauss spent some time investigating the patterns of repetitions in simple decimal expansions of fractions, like $\frac{1}{3} = .333\ldots$ or $\frac{2}{7} = .285714285714\ldots$. It turns out that this is directly connected to whether 10 is a primitive root for a given prime (see Exercise 17.7.19). Likewise, the 'Great Theorem' in William Dunham's book *Journey Through Genius* where Euler found that $F_5 = 4294967297$ was composite (recall Subsection 12.1.1) would have been helped along quite a bit, as Euler's proof looked directly at factors of powers of 2 (plus one) and their possible form, not powers of 3.

```
@interact
def _(n=(1,[1..6])):
    F = 2^(2^n)+1
    a = mod(3,F)
    if a.multiplicative_order()==F-1:
        pretty_print(html("$3$ is a primitive root of
            $F_{%s}=%s$"%(n,F)))
    else:
        pretty_print(html("Not prime, no primitive root!"))
```

We can use these ideas to find another possible way to attack Artin's Conjecture. It's not directly related to reciprocity per se, but still connects all our theoretical ideas of the last several sections.

Example 17.5.3. We put this in the form of several steps. Verifying several facts in these steps is left to Exercise Group 17.7.8–17.7.11.

Recall from the very end of Section 11.6 that if q and $p = 2q + 1$ are both odd primes, then we call q a Germain prime. Every residue of p for which primitive root makes sense (i.e. not $a = -1$ or $a = 0$) is either a primitive root or a QR.

Such a prime p must be of the form $p \equiv 3 \pmod 4$, because $q = 2k+1$ so that

$$p = 2(2k+1)+1 = 4k+3$$

In this case, not only are all residues either a primitive root or a QR, but a is one of these things precisely when $p-a$ is the other kind. We know that

$$1^2, 2^2, 3^2, \ldots, q^2$$

are all different modulo p, and of course all of these are QRs (and so not primitive roots).

Here is the key; that means that the additive inverses of perfect suqares, $p-k^2$, for $2 \leq k \leq q$, must all be primitive roots. The smallest of these, $p-4$, must thus be a primitive root for any such prime $p = 2q+1$!

So if there were infinitely many such Germain primes, we would also have an explicit example of Artin's conjecture … but, so far, no such luck.

The largest currently known (as of this writing, discovered in early 2016) Germain prime, due to James Scott Brown, is

$$2618163402417 \cdot 2^{1290000} - 1$$

which is a number with close to four hundred thousand digits. (The previous record had about half as many, so this is a huge advance.)

```
@interact
def _(q=(11,[r for r in prime_range(3,100) if
    is_prime(2*r+1)])):
    p = 2*q+1
    a=mod(p-4,p)
    if a.multiplicative_order()==p-1:
        pretty_print(html("$-4$ is a primitive root of
            $%s$"%p))
    else:
        pretty_print(html("Mistake!"))
```

17.6 A Proof of Quadratic Reciprocity

You are most likely now exhausted by the many applications and uses of quadratic reciprocity. Now we must prove it.

Recall the statement: For odd primes p and q, we have that

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$$

That is to say, the Legendre symbols are the same unless p and q are both of the form $4k+3$.

Before beginning, let's recall the tools we will need on our jouney. First, recall that p and q are odd primes in the context of this proof. Also, we will use the criterion of Eisenstein's 17.2.8 we've used throughout. So we'll let

$$R = \sum_{\text{even } e,\ 0<e<p} \left\lfloor \frac{qe}{p} \right\rfloor$$

be the exponent in question, so that

$$\left(\frac{q}{p}\right) = (-1)^R .$$

17.6.1 Re-enter geometry

The key to our proof will be *geometrically* interpreting $\left\lfloor \frac{qe}{p} \right\rfloor$. We can think of it as being the biggest integer less than $\frac{qe}{p}$, which means we can think of it as an *integer* height.

The following features are present in the next graphic, which should clarify how we'll think of it geometrically. Each type of object is highlighted with a different color.

- The line through the origin with slope q/p (dotted blue).
- All the grid points in the box of width p and height q (box red, points black).
- Points with *even* x-coordinate corresponding to the highest that one can get while staying under the line of slope q/p (points blue).
- The box of width $\frac{p-1}{2}$ and height $\frac{q-1}{2}$ (green), which we'll need in a moment.

```
var('x,y')
@interact
def _(p=(11,prime_range(3,100)),q=(7,prime_range(3,100))):
    E = [2,4..p-1]
    plot4 = plot((q/p)*x,(x,0,p),linestyle='--')
    plot3 = line([[0,0],[p,0],[p,q],[0,q],[0,0]],
        rgbcolor=(1,0,0))
    plot2 = line([[0,0], [(p-1)/2,0], [(p-1)/2,(q-1)/2],
        [0,(q-1)/2], [0,0]], color='green')
    grid_pts_1 = [[i,j] for i in [1..p] for j in [1..q]]
    grid_pts_2 = [[i,j] for i in [1..(p-1)/2] for j in
        [1..(q-1)/2]]
    plot_grid_pts =
        points(grid_pts_1,rgbcolor=(0,0,0),pointsize=10)
    lattice_pts1 = [coords for coords in grid_pts_1 if
        (coords[0]*q-coords[1]*p>0 and coords[0]<p and
        coords[0] in E)]
    lattice_pts2 = [coords for coords in grid_pts_2 if
        (coords[0]*q-coords[1]*p>0 and coords[0]>p/2)]
    num1, num2 = len(lattice_pts1), len(lattice_pts2)
    if len(lattice_pts1)!=0:
        plot_lattice_pts1 = points(lattice_pts1, rgbcolor
            = (0,0,1),pointsize=20)
    else:
        plot_lattice_pts1 = Graphics()
    if len(lattice_pts2)!=0:
        plot_lattice_pts2 = points(lattice_pts2, rgbcolor
            = (0,.5,0),pointsize=20)
    else:
        plot_lattice_pts2 = Graphics()
    show(plot2+plot3+plot4 + plot_grid_pts +
        plot_lattice_pts1, xmax=p,ymax=q,ymin=0)
    forms = '$'+'+'.join(['\left\lfloor\\frac{%s\cdot␣
        %s}{%s}\\right\\rfloor'%(q,e,p) for e in E])+'$'
    pretty_print(html("The␣blue␣dots␣represent␣"+forms))
    forms2 = '$'+'+'.join(['\left\lfloor\\frac{%s}{%s}␣
        \\right\\rfloor'%(q*e,p) for e in E])
```

```
    forms3 = '+'.join(['%s'%(floor(q*e/p)) for e in
        E])+'=%s\equiv%s\\text{ (mod
        }2)$'%(sum([floor(q*e/p) for e in
        E]),sum([floor(q*e/p) for e in E])%2)
    pretty_print(html("This simplifies to
        "+forms2+'='+forms3))
```

It should be clear that each blue stack has the same height as $\left\lfloor \frac{qe}{p} \right\rfloor$ for some even e. The core *geometric* point of the proof is to convince ourselves of this:

Claim 17.6.1. *The number of blue points (which is R) has the same parity as the* total *number of (positive) points in and on the green box which are under the dotted line.*

Claim 17.6.2. *Suppose that we have proved Claim 17.6.1. Then we can quickly prove Quadratic Reciprocity.*

Proof. The following steps are all we need. Essentially, we take the previous claim and use it for both Legendre symbols, add, and get the result.

- First, to get $\left(\frac{q}{p}\right)$, we can ignore R and just focus on the number (indeed, parity) of positive lattice points in that more-or-less triangle.
- The same argument applies to $\left(\frac{p}{q}\right)$; we could ignore the exponent

$$R' = \sum_{\text{even } e',\ 0<e'<q} \left\lfloor \frac{pe'}{q} \right\rfloor$$

  and instead focus on the number of positive lattice points in and on the green box *to the left of* the dotted line. (Think about this; it switches the role of the vertical and horizontal axes.)
- So the total exponent of -1 we expect from $\left(\frac{q}{p}\right)\left(\frac{p}{q}\right)$ is just the sum of those two amounts. That is, the exponent is the number of points in and on the green box. (There is no overlap, because q and p are coprime, so there are no lattice points *on* the dotted line until we get to (p, q), which is well outside the green box.)
- You'll note that this box has dimensions $\frac{p-1}{2}$ and $\frac{q-1}{2}$, so that would mean

$$\frac{p-1}{2} \cdot \frac{q-1}{2} \equiv \sum_{\text{even } e,\ 0<e<p} \left\lfloor \frac{qe}{p} \right\rfloor + \sum_{\text{even } e',\ 0<e'<q} \left\lfloor \frac{pe'}{q} \right\rfloor \pmod 2 ,$$

  so that

$$\left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{R+R'} = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}} .$$

□

17.6.2 Proving proper parity

So to finish the proof via Claim 17.6.1, we must show that the number of blue points (points under the line with even x-coordinate) is the same as the number of positive points in the green box under the line. Along with Eisenstein, we call this second number μ.

In the next graphic, there is a lot going on, all of which we will use for the proof (note especially the new, green, points). We will clarify each of the pieces below.

```
var('x,y')
@interact
def _(p=(11,prime_range(3,100)),q=(7,prime_range(3,100))):
    E = [2,4..p-1]
    plot4 = plot((q/p)*x,(x,0,p),linestyle='--')
    plot3 = line([[0,0],[p,0],[p,q],[0,q],[0,0]],
        rgbcolor=(1,0,0))
    plot2 = line([[0,0], [(p-1)/2,0], [(p-1)/2,(q-1)/2],
        [0,(q-1)/2], [0,0]], color='green')
    grid_pts_1 = [[i,j] for i in [1..p] for j in [1..q]]
    grid_pts_2 = [[i,j] for i in [1..(p-1)/2] for j in
        [1..(q-1)/2]]
    plot_grid_pts =
        points(grid_pts_1,rgbcolor=(0,0,0),pointsize=10)
    lattice_pts1 = [coords for coords in grid_pts_1 if
        (coords[0]*q-coords[1]*p>0 and coords[0]<p and
        coords[0] in E)]
    lattice_pts2 = [coords for coords in grid_pts_1 if
        (coords[0]*q-coords[1]*p<0 and coords[0]>(p-1)/2
        and coords[1]<q and coords[0] in E)]
    lattice_pts3 = [coords for coords in grid_pts_1 if
        (coords[0]*q-coords[1]*p>0 and coords[0]<=(p-1)/2
        and coords[0] not in E)]
    num1, num2 = len(lattice_pts1), len(lattice_pts2)
    if len(lattice_pts1)!=0:
        plot_lattice_pts1 = points(lattice_pts1, rgbcolor
            = (0,0,1),pointsize=20)
    else:
        plot_lattice_pts1 = Graphics()
    if len(lattice_pts2)!=0:
        plot_lattice_pts2 = points(lattice_pts2, rgbcolor
            = (0,.5,0),pointsize=20)
    else:
        plot_lattice_pts2 = Graphics()
    if len(lattice_pts3)!=0:
        plot_lattice_pts3 = points(lattice_pts3, rgbcolor
            = (0,.5,0),pointsize=20)
    else:
        plot_lattice_pts3 = Graphics()
    show(plot2+plot3+plot4 +
        plot_grid_pts+plot_lattice_pts1 +
        plot_lattice_pts2+plot_lattice_pts3,
        xmax=p,ymax=q,ymin=0)
    forms = '$\mu='+'+'.join(['\left\lfloor\\frac{%s\cdot␣
        %s}{%s}\\right\\rfloor'%(q,e,p) for e in
        [1..(p-1)/2]])+'$'
    pretty_print(html("The␣blue␣and␣green␣dots␣in␣the␣
        small␣triangle␣represent"))
    pretty_print(html("the␣sum␣"+forms))
```

Let's look at the two sets of green dots.

- One set is on top, the lattice points with *even* x-coordinates *greater* than $\frac{p-1}{2}$ which have y-coordinate *less* than q which are *above* the dotted line.
- The other set is similar, but on the bottom, with *odd* x-coordinates *less* than $\frac{p-1}{2}$ which have y-coordinate greater than 0 and are below the line.

You can think of the first set as filling in the even columns greater than $\frac{p-1}{2}$, while the latter set fills in the the triangle for odd columns less than $\frac{p-1}{2}$. To further understand this, in the interactive form of the text you may wish to try q relatively large compared to p to see this more clearly. Try several different values!

The key observation is that these two sets of green dots *are symmetric images* – they are simply rotated around the point

$$\left(\frac{p}{2}, \frac{q}{2}\right) .$$

This makes sense, since with p and q odd, this would change odd to even and vice versa.

So in order to say that μ has the same parity as R (which is our goal to finish the proof), we just have to show that either set of green points has the same parity as that of the set of blue points outside the green box. Again, refer to the interactive graphic and try it with different primes for best understanding.

Claim 17.6.3. *Either set of green points has the same parity as the set of blue points outside the green box.*

Proof. There are $q-1$ points in each column of points outside the green box. In particular, there an *even* number of points in each such column.

So whether the number of blue points in a given column is even or odd, it is guaranteed that the parity of the green points in that same column is *also* even or odd, respectively. So the parity of the green points outside the green box is the same as the parity of the blue points outside the green box.

This means the parity of the points inside the triangle (μ) is the same as that of the blue points (R), which is what we wanted to prove! □

17.6.3 Postlude

It's really quite amazing how we needed to understand congruence, parity, some geometry, and of course the idea of a quadratic residue in the first place to prove this. As of right now, there is a list of well over two hundred proofs of this theorem. The very shortest might be one by G. Rousseau , and there is a nice list online of "favorite proofs" by various mathematicians.

So this is one proof where it is appropriate to say Q.E.D.

17.7 Exercises

1. Evaluate the Legendre symbols for $p = 11$ and $a = 2, 3, 5$ using Eisenstein's Criterion for the Legendre Symbol.

2. Use the previous problem, your knowledge of $\left(\frac{-1}{11}\right)$ and of perfect squares to evaluate the other Legendre symbols for $p = 11$.

3. Do any Legendre symbols after Example 17.1.3 which you didn't already do.

4. Make up several hard-looking Legendre symbols $\left(\frac{a}{29}\right)$ (modulo $p = 29$) that are easy to solve by adding p or by factoring a.

5. Use the multiplicative property of the Legendre symbol to give a congruence condition for when $\left(\frac{-2}{p}\right) = \pm 1$.

6. For $0 < a, b < p$, prove that at least one of a, b, and ab is a quadratic residue of p.

7. In Exercise 16.8.6 we proved, for an odd prime p,

$$\sum_{a=1}^{p-1} \left(\frac{a}{p}\right) = 0\,.$$

Conjecture (and, if you can, prove) a similar result for

$$\sum_{a \in Q_p} a\,.$$

In Example 17.5.3 there are a number of small issues which need proof; here, you have the opportunity to finish them off.

8. Let p be a prime of the form $p = 2q + 1$, where q is prime (recall that q is called a Germain prime in this case). Show that *every* residue from 1 to $p - 2$ is either a primitive root of p or a quadratic residue. (Hint: Use Euler's Criterion, and ask yourself how many possible orders an element of U_p can have.)

9. Prove: if $p \equiv 3 \pmod 4$, and if $a \not\equiv \pm 1, 0$, then a is a QR modulo p if and only if $p - a$ is not a QR.

10. Prove that for any prime p, if $1 < i, j < \frac{p}{2}$ and $i \neq j$, then $i^2 \not\equiv j^2 \pmod p$. (Hint: factor!)

11. Verify the previous exercise for $p = 23$.

12. Prove that if $\left(\frac{2}{n}\right)$ is the Jacobi symbol instead of the Legendre symbol, it is still true that $\left(\frac{2}{n}\right) = 1$ precisely when $n \equiv \pm 1 \pmod 8$. (Remember, n has to be odd by definition.)

13. Compute some Legendre symbols that seem pretty hard by using the Jacobi symbol instead.

14. Show that if p is an odd prime, then there are exactly $\frac{p-1}{2} - \phi(p-1)$ residues which are neither QRs nor primitive roots. (Hint: don't think too hard – just do the obvious counting up.)

15. If you didn't do them already, do the exercises in Example 17.4.6.

16. Evaluate five non-obvious Legendre symbols $(\frac{a}{p})$ for $p = 47$ using quadratic reciprocity.

17. Find congruence criteria for p for when $a \in Q_p$ for $a = -3, 6$, and 9. (Hint: Don't do any extra work – use what you know!)

18. Use quadratic reciprocity to prove the surprising statement that -5 is a quadratic residue for exactly those primes for whom the sum of the ones and tens digit is odd. (Did you conjecture this when you completed Exercise 16.8.11? See [C.6.10] about a story behind this unusual result.)

19. Use Sage to explore why repetition in the decimal expansion of $\frac{a}{p}$ is related to whether 10 is a primitive root modulo p.

Chapter 18

An Introduction to Functions

The further one goes into number theory, the more one needs to think about the functions involved *as functions*, and not just as handy computational shorthand.

Question 18.0.1. What properties do number-theoretic functions (such as $\phi(n)$) have? What can we do with them?

Most of the remainder of the text deals with such questions. This short chapter introduces some of the questions we will ask through the lens of one function we have done a fair amount with, and then through the eyes of one we have examined in less detail.

The Euler function, like many we have seen and will see, is an example of an **arithmetic function**. An arithmetic function is a function with the natural numbers as its domain, usually going to integer, real, or complex values.

Remark 18.0.2. We pronounce this word with the stress on the *third* syllable in number theory when used as an adjective, but (as usual) on the second syllable when used as a noun.

> A-rith-*me*-tic functions show up when studying the higher a-*rith*-me-tic.

There are three types of questions we'll spend a lot of time with regarding arithmetic functions. For any given function, we wish to find or examine the following.

- We want to have as explicit of formulas as possible for our functions, which are often defined implicitly or in terms of counting.
- We wish to find relational formulas, either between our function and other functions, or especially among different values of the function itself.
- We desire to see what the long-term or aggregate behavior of the functions is; in practice this usually involves summation of various kinds.

In this chapter, we will start the process, but it will recur throughout the remainder of the text.

18.1 Three Questions for Euler phi

It's easier to say useful things about some functions than others! To begin, let's go back and remind ourselves of some of the nice properties of one particular function we *did* study a fair amount. In the next chapter, we'll start exploring some that we have not yet encountered.

That function is, naturally, the Euler ϕ function. Recall that $\phi(n)$ gives the size of the set

$$\{k \mid 0 < k \leq n,\ \gcd(k, n) = 1\}$$

of residues modulo n which are coprime to n.

Example 18.1.1. We can use Sage to calculate.

```
euler_phi(25)
```

18.1.1 Formulas

Of course, such small values can be calculated by hand. But what about more general ones? Surely we don't want to have to check every number up to n just to compute this.

And indeed, in Exercise 9.6.9 you should have gotten a formula. Do you remember it? The following Sage cell is a hint.

```
print factor(275)
print euler_phi(275)
print 275*(1-1/5)*(1-1/11)
```

Fact 18.1.2. *If n is the product of prime powers $n = \prod_{i=1}^{k} p_i^{e_i}$ then we have the formula*

$$\phi(n) = n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

Proof. Do Exercise 9.6.9! □

If you are in a classroom setting, you may want to discuss whether it seems likely that arbitrary arithmetic functions have formulas.

18.1.2 Relations

One piece of getting a formula for ϕ is the rather interesting property ϕ has (Fact 9.5.2) that if m, n are coprime then $\phi(m)\phi(n) = \phi(mn)$. This is a general property an arithmetic function can have.

Definition 18.1.3. We say that $f(n)$ is **multiplicative** if

$$f(m)f(n) = f(mn) \text{ when } m, n \text{ are coprime.}$$

The terminology is kind of bad, because of course the function only 'multiplies' for coprime integer inputs, but since relative primality is such a fundamental concept this seems okay nonetheless. We can test this here.

```
@interact
def _(a=25,b=11):
    pretty_print(html("$\phi(%s)=%s\\text{ and }\phi(%s)=%s$"%(a, euler_phi(a), b, euler_phi(b))))
    if gcd(a,b)==1:
        pretty_print(html("And $\phi(%s\cdot %s)=%s\cdot %s=%s$, their product!"%(a, b, euler_phi(a), euler_phi(b), euler_phi(a*b))))
    else:
        pretty_print(html("But $%s$ and $%s$ aren't coprime, so $\phi(%s\cdot %s)=%s\\neq %s\cdot %s$"%(a, b, a, b, euler_phi(a*b), euler_phi(a), euler_phi(b))))
```

So ϕ is multiplicative. Do you think this is an unusual property to have?

Again, in a class setting you may wish to discuss whether it seems likely that arithmetic functions might have some property along these lines.

18.1.3 Summation (and limits)

One thing that might be useful to look at in a function is its behavior in the long term. In calculus, we certainly talk a lot about things like asymptotes, even asymptotes other than horizontal and vertical ones. Unfortunately, arithmetic functions don't often look that great in this way.

Example 18.1.4. For instance, let's look at the plot of ϕ.

```
plot(euler_phi,1,1000)
```

This doesn't look like it's "going" anywhere.

That said, we could look at the highest or lowest points, at least. Certainly prime numbers p will always have the formula $\phi(p) = p - 1$, and that is a nice graph; the lower limit seems reasonably regular as well. Try to think about how one might encapsulate such observations in terms of limits.

One strategy that is sometimes used to "smooth" such behavior in places like analyzing stock prices is trying to calculate "averages" – that is, sum it up and divide. We are not ready for this with ϕ (see Section 20.5).

However, there was a different interesting property about summation of $\phi(n)$, namely Fact 9.5.4. To recall, what was the sum of $\phi(d)$ over the set of divisors d of n?

```
@interact
def _(n=275):
    pretty_print(html("$%s$ factors as $%s$"%(n,latex(factor(n)))))
    pretty_print(html("Its divisors are $%s$"%latex(divisors(n))))
    pretty_print(html("The sum of $\phi$ of the divisors is $%s$"%sum([euler_phi(d) for d in divisors(n)])))
```

Ah yes, it was just that $\sum_{d|n} \phi(d) = n$. Even if we can't say something about limiting behavior yet, this kind of summation must be getting us closer!

As a final classroom discussion point, what kind of behavior do you think could happen when summation of arithmetic functions is considered? What about limits? Could you get anything you can get in calculus, or should some things not be possible?

18.2 Three Questions, Again

Hopefully your appetite is whetted a bit by the previous section, and especially the discussion opportunities about what you think might be possible.

So let's start exploring these questions with *new* functions.

Example 18.2.1. Let $r(n)$ be the number of (all!) ways to write n as a sum of (two) squares. (This was called $r_2(n)$ when first encountered in Exercise 13.7.7, but we will not consider other r_k.)

For instance, $r(25) = 12$. Why?

Because you can write it using the pairs

$$(\pm 3, \pm 4), \quad (\pm 4, \pm 3), \quad (\pm 5, 0) \text{ and } (0, \pm 5) \, .$$

Remember, we count *all* solutions, positive or negative, and in any particular order possible, in determining the value of $r(n)$.

18.2.1 Formulas

In Exercise 13.7.7, we saw that $r(2^m) = 4$. But we didn't discuss it enough to question whether there might be a formula that was easier to compute than the process of counting all possible sums!

As an encouragement to our search for answers to our three questions, I will give you a (totally unmotivated!) formula. To see what it looks like, we use an extension of the Fundamental Theorem of Arithmetic.

Fact 18.2.2. *Write the prime factorization of n as*

$$n = 2^d p_1^{e_1} \cdots p_k^{e_k} q_1^{f_1} \cdots q_\ell^{f_\ell}$$

where we write primes of the form $4k+1$ as p, and primes of the form $4k+3$ as q. Then

$$r(n) = \begin{cases} 0 & \text{if any } f_j \text{ is odd} \\ 4 \prod_{i=1}^{k} (e_i + 1) & \text{otherwise} \end{cases}$$

Proof. Unfortunately, it turns out that every single proof of this is not very elementary. They all either go into some detail regarding factorization of Gaussian integers (recall our allusion to this in Fact 14.1.6), or they do some lengthy divisibility and congruence analysis. So we will skip the proof. □

To use this, notice that the empty product (no primes of the form $4k+1$) is 1, just like a sum over zero elements is zero. To prove Exercise 13.7.7, we note that if $r(2^m)$ then all e_i and f_j are zero, then we are in the second case and we just get $4 \cdot 1$ for the product.

Sage note 18.2.3 (Review quiz)**.** You can use various tools we've already seen to compute this with Sage, such as factoring and multiplication. Try it!

18.2.2 Relations

We just saw an impressive relation among values of $\phi(n)$. As an example of it, $\phi(5)\phi(3) = \phi(15)$, since the inputs are coprime. Similarly, there are some relations with multiplying for r, though it certainly isn't multiplicative.

Example 18.2.4. Indeed, now that we have a formula, we can compute this.

- For instance,
$$r(3)r(5) = r(15)$$
because both sides are zero!
- For the same reason, $r(8)r(7) = r(56)$.
- On the other hand,
$$r(25)r(13) = 12 \cdot 8 = 96 \neq 24 = r(325)$$
- Similarly, $r(25)r(4) = 12 \cdot 4 = 48 \neq 12 = r(100)$.

In these examples, the inputs are relatively prime but it doesn't multiply. What might still be true? See Exercise Group 18.3.1–18.3.2.

Sage note 18.2.5 (Explore here)**.** Feel free to explore here!

18.2.3 Limits (and summation)

In Subsection 18.1.3 we saw that (for ϕ) even though we couldn't yet address long term behavior, we could at least see some patterns, and could say something about summing values. In this subsection, we will try to directly address *long-term, average* behavior for $r(n)$.

To be precise, we will talk about *limits* with functions. Yes, limits in number theory!

Observe the following graphic. It has as its basic content the circle with radius $\sqrt{n}$ and blue lattice points representing all pairs (x, y) such that $x^2 + y^2 \leq n$. There is a little box of area one around each such lattice point.

```
@interact
def _(n=(5,[1..100])):
    viewsize=ceil(math.sqrt(n))+2
    a=(math.sqrt(n)+1/math.sqrt(2))^2
    b=(math.sqrt(n)-1/math.sqrt(2))^2
    g(x,y) = x^2+y^2
    P=Graphics()
    P += implicit_plot(g-n, (-viewsize,viewsize),
        (-viewsize,viewsize), plot_points = 200)
    P += implicit_plot(g-a, (-viewsize,viewsize),
        (-viewsize,viewsize), linestyle='--',plot_points =
        200)
    P += implicit_plot(g-b, (-viewsize,viewsize),
        (-viewsize,viewsize), linestyle='--',plot_points =
        300)
    grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
        in [-viewsize..viewsize]]
    P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[1]^2+coords[0]^2<=n)]
    P += points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    squares=[line([[k-1/2,l-1/2],
        [k+1/2,l-1/2],[k+1/2,l+1/2],
        [k-1/2,l+1/2],[k-1/2,l-1/2]], rgbcolor=(1,0,0)) for
        [k,l] in lattice_pts]
    for object in squares:
```

```
        P += object
    show(P, figsize = [5,5], xmin = -viewsize, xmax =
        viewsize, ymin = -viewsize, ymax = viewsize,
        aspect_ratio=1)
    pretty_print(html("There␣are␣$%s$␣boxes␣with␣a␣circle␣
        of␣radius␣$%s$"%(len(squares),math.sqrt(n))))
    pretty_print(html("The␣ratio␣of␣the␣area␣of␣boxes␣to␣
        the␣square␣of␣the␣radius␣is␣
        $\\approx%s$"%(len(squares)/(math.sqrt(n)^2))))
```

As you might expect, the boxes *roughly* cover the circle, but certainly not exactly. So what does this have to do with $r(n)$?

Each unit box around each lattice point can be thought of as standing in for a representation (as a sum of squares) of a given integer less than or equal to n. Adding up *all* the areas would thus give a number, as a summation:

$$\sum_{k=0}^{n} r(k)$$

So the area of the boxes can give us information about r.

Fact 18.2.6. *Observe that the boxes neither cover nor are covered by the circle in question. However, we can say two things about them.*

- *These boxes will entirely cover a disk of radius $\sqrt{n}$ minus half the diagonal length of the boxes, namely $\frac{1}{\sqrt{2}}$, which is the inner circle above.*
- *Likewise, they are completely contained in a disk of radius $\sqrt{n}$ plus half the diagonal length of the boxes.*

Proof. Geometry. □

Let's use this fact to create a double inequality in terms of the area covered by two circles and the squares.

$$\pi\left(\sqrt{n}-\frac{1}{\sqrt{2}}\right)^2 \leq \sum_{k=0}^{n} r(k) \leq \pi\left(\sqrt{n}+\frac{1}{\sqrt{2}}\right)^2,$$

If we divide by n and simplify a bit, then factor, we obtain

$$\pi\frac{n-\sqrt{2n}+1/2}{n} \leq \frac{1}{n}\sum_{k=0}^{n} r(k) \leq \pi\frac{n+\sqrt{2n}+1/2}{n},$$

$$\pi\left(1-\sqrt{\frac{2}{n}}+\frac{1}{2n}\right) \leq \frac{1}{n}\sum_{k=0}^{n} r(k) \leq \pi\left(1+\sqrt{\frac{2}{n}}+\frac{1}{2n}\right)$$

We're almost at something interesting.

- First, the limit as n goes to ∞ of the lower *and* upper bounds with each of these inequalities exists. In fact, the limit of the bounds in both cases is π.
- Then, the beloved squeeze theorem from calculus implies that

$$\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n} r(k) = \pi\,.$$

- Finally, note that $r(0) = 1$, so its presence or absence will not affect the average in the limit at all.

We can interpret this line of thought as proving and saying:

Fact 18.2.7. *The average number of representations of a positive integer as a sum of squares is π.*

WHAT?!

But it's true. And there's more to come.

18.3 Exercises

We see in Subsection 18.2.2 that r is not multiplicative. But could some things still be true?

1. Look at the cases where zero is involved. State the broadest possible multiplicativity result you can for this case.

2. Look at the second two examples in Subsection 18.2.2. There seems to be a specific *sort* of relationship in the precise way in which these examples are *not* multiplicative. What is that relationship? Can you prove it? (Hint: first compare the results, only then the individual inputs.)

3. For a fixed $p(x)$, let $Z_{p(x)}(n)$ be the number of solutions of the polynomial congruence $p(x) \equiv 0 \pmod{n}$. Use facts from earlier in the text to show that this function is multiplicative. Connect this to the question of whether $-1 \in Q_n$.

4. Let the function g be given by

$$\begin{cases} 0 & n \text{ is even} \\ 1 & n \equiv 1 \bmod(4) \\ -1 & n \equiv 3 \pmod 4 \end{cases}$$

Show that the function $g(n)$ is multiplicative.

Chapter 19

Counting and Summing Divisors

Among all the possible arithmetic functions one could discuss, there is one family which is both truly ancient and part of cutting-edge research. We'll let ourselves be inspired by the summations in the previous chapter, by summing the simplest functions of all and seeing what we get.

19.1 Exploration: A New Sequence of Functions

Definition 19.1.1. Let $\sigma_k(n)$ be defined as the sum of the kth power of the (positive) divisors of n, thus:

$$\sigma_k(n) = \sum_{d|n} d^k \,.$$

Before doing *any* computing, think about what special information about a number σ_1 and σ_0 might encode.

Remark 19.1.2. Incidentally, very (very) often one will see $\sigma_0(n)$ written as $\tau(n)$, sometimes also as $d(n)$. Usually $\sigma_1(n)$ is written simply $\sigma(n)$, though Euler apparently used $\int n$ in his writings (can you think why?).

Hopefully, you realized σ_1 is adding all the divisors of n (including n itself), and that σ_0 is the *number* of (positive) divisors of n. Now, get ready to explore! Try to figure out as much as you can about these functions. If you're in a group in a class, you can certainly save time by dividing up the initial computations among yourselves, then sharing that information so you have a bigger data set to look at.

Question 19.1.3. Can you find some or all of the following for these functions?

- A formula, at least for some input types.
- See if at least a limited form of multiplicativity (recall Definition 18.1.3) holds.

You might also want to look at questions like these.

- Can two different n yield the same σ_k (for a given k)? If so, when – or when not? Can they be consecutive?

- Is it possible to say anything about when one of these functions yields even results – or ones divisible by three, four, … ?

- Clearly the size of these functions somehow is related to the size of n – for instance, it is obvious that $\sigma_0(n) = \tau(n)$ can't possible be bigger than n itself! So how big *can* these functions get, relative to n? How small?

- Can anything be said about congruence values of these functions? (This is a little harder.)

If you come up with a new idea, why not challenge someone else to prove it? See Exercise Group 19.6.2–19.6.4.

19.2 Conjectures and Proofs

Remark 19.2.1. Don't read this section until you have tried some of the exploration in the previous section!

In the last section we defined some new functions, and asked some questions about them. You can try them by hand, or use computation to explore them further.

Sage note 19.2.2 (Syntax for sigma)**.** Here is the syntax for doing this in Sage. This was a case where it was better to try it out by hand first, though!

```
sigma(12,1),sigma(12,0),sigma(12)
```

What were some of your conjectures? It is quite likely that you (or others, if in a class setting) discovered some of these:

- $\sigma_1(p) = p + 1$ if p is prime.

- $\sigma_0(p^e) = e + 1$ if p^e is a prime power.

- σ_i is in fact multiplicative for $i = 0, 1$.

If you dug a little deeper, or had a little more time to spend, your conjectures may have also included ones *like* these:

- $\sigma_1(p^e) = 1 + p + p^2 + \cdots + p^e$ for p^e a prime power.

- $\sigma_1(2^e) = 2^{e+1} - 1$.

- $\sigma_0(n)$ is odd precisely if n is a perfect square.

Let's prove the most important of these things, as well as mention a few other useful formulas.

19.2.1 Prime powers

Again, usually one will have discovered various formulas that are special cases of the following, among others. It's surprisingly easy to find the patterns!

Fact 19.2.3. *If p^e is a perfect prime power, then*

$$\sigma_0(p^e) = e + 1 \text{ and } \sigma_1(p^e) = 1 + p + p^2 + \cdots + p^e = \frac{p^{e+1} - 1}{p - 1} .$$

Proof. There isn't much to prove here, once discovered. Both formulas come from the same fundamental observation.

- All possible divisors of a prime power must have only that prime as divisors, by the Fundamental Theorem of Arithmetic. So, these divisors are just other (smaller) powers of that prime.
- There are exactly $e + 1$ of these divisors, and these divisors are the ones summed up in the σ_1 formula.

  The step giving the fraction formula is just the usual geometric summation formula familiar from precalculus and calculus.

□

19.2.2 Multiplicativity

It's a bit harder to prove the following.

Fact 19.2.4. *For any i, $\sigma_i(n)$ is multiplicative (again, see Definition 18.1.3). That is,*

$$\sigma_i(mn) = \sigma_i(m)\sigma_i(n) \text{ when } \gcd(m, n) = 1 .$$

This automatically leads to many facts, in particular to this one.

Theorem 19.2.5. *If we factor $n > 0$ as*

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

then we have formulas

$$\sigma_0(n) = \prod_{i=1}^{k} \left(e^i + 1\right) \text{ and } \sigma_1(n) = \prod_{i=1}^{k} \left(\frac{p_i^{e_i+1} - 1}{p_i - 1} \right) .$$

We will *not* prove this fact directly! It is possible, and might make a good challenge exercise. But it is not *efficient*.

Instead, we will prove below a theorem that exemplifies a general principle:

Principle 19.2.6. *In the long run, it is better to prove general results for sums of arithmetic functions than to do each one by itself.*

Otherwise we do an endless line of proofs like the ones we did for ϕ (recall Fact 9.5.2), but for every arithmetic function.

19.2.3 A very powerful lemma

Let $\sum_{d|n}$ denote the sum over all positive divisors (including 1 and n) of n. Then we have the following, the proof of which will be easier than for Euler's function.

Theorem 19.2.7. *If g is multiplicative and $f(n)$ is defined as*

$$f(n) = \sum_{d|n} g(d)$$

then f is also multiplicative.

Proof. We follow here [C.1.1]. Let m and n be coprime; we are interested in $f(mn)$.

Basically, this all boils down to asking what the divisors of mn look like. Any divisor of mn must be the product of some divisor a of m and some divisor b of n.

The previous observation is just about multiplication and divisibility, not even coprimeness. But that guarantees that a and b are coprime as well, given that m and n are. So each divisor $d \mid mn$ gives us a (unique) pair of (coprime) divisors a and b of m and n.

Instead of summing over all divisors of mn, we can instead sum over each divisor of n for each divisor of m. In symbols,

$$f(mn) = \sum_{a|m} \sum_{b|n} g(ab) \ .$$

Now we can use all the facts we have at hand (coprimeness, multiplicativity, etc.) to finish it off.

$$f(mn) = \sum_{a|m} \sum_{b|n} g(ab) = \sum_{a|m} \sum_{b|n} g(a)g(b)$$

$$= \left(\sum_{a|m} g(a) \right) \left(\sum_{b|n} g(b) \right) = f(m)f(n) \ .$$

□

Corollary 19.2.8. *Since $g(n) = n^i$ is clearly multiplicative, it is true that*

$$\sum_{d|n} g(d) = \sum_{d|n} d^i = \sigma_i(n)$$

is also multiplicative.

In the special cases $i = 0$ and $i = 1$ of the corollary we see that $\sigma_0 = \tau$ and $\sigma_1 = \sigma$ are multiplicative. Since we will use them later, we properly define the special cases of n^i here.

Definition 19.2.9. Let us set the following two arithmetic functions:

- $u(n) = 1$ to be the *unit* function
- $N(n) = n$ to be the *identity* function

19.3 The Size of the Sum of Divisors Function

For the rest of this chapter, we will focus on $\sigma_1 = \sigma$ itself, since the sum of divisors function has a deep richness of its own. We could ask questions about evenness, other patterns, and so forth.

This short section asks a particularly interesting question. Try the following interactive cell.

```
@interact
def _(n=range_slider(1,150,1,(1,20))):
    top = n[1]
    bottom = n[0]
    cols = ((top-bottom)//10)+1
    T = [cols*['$n$','$\sigma(n)$','$\sigma(n)/n$']]
    list = [[i,sigma(i),(sigma(i)/i).n(digits=3)] for i in
        range(bottom,top+1)]
    list.extend((10-(len(list)%10))*['',''])
    for k in range(10):
        t = [item for j in range(cols) for item in
            list[k+10*j]]
        T.append(t)
    pretty_print(html(table(T,header_row = True, frame =
        True)))
```

This table helps you see possibilities for the *relative* size of $\sigma(n)$ with respect to n itself. Alternately, we have the following.

Question 19.3.1. For any given n, what is the constant C_n such that $\sigma(n) = C_n \cdot n$? How big can this get?

The spread of these ratios, for n under one hundred fifty, certainly goes both above and below 2. If you look carefully, you will see that only one of the numbers above has a sum of divisors without 1 or 2 as the integer part. What is it?

Instead of simply trying larger and larger input numbers, we might use a little theory to get a higher ratio. To wit, if a number has lots of small prime divisors, we might think it has lots of factors. So taking big powers of these would have even more small prime divisors and might get us big ratios.

```
@interact
def _(n=[1..15]):
    pretty_print(html("Try␣
        $2^{%s}\cdot3^{%s}\cdot5^{%s}=%s$"%(n, n, n,
        2^n*3^n*5^n)))
    pretty_print(html("Then␣$\sigma(%s)=%s=%s\cdot␣
        %s\\approx␣%s\cdot␣%s$"%(2^n*3^n*5^n,
        sigma(2^n*3^n*5^n),
        sigma(2^n*3^n*5^n)/(2^n*3^n*5^n), 2^n*3^n*5^n,
        (sigma(2^n*3^n*5^n)/(2^n*3^n*5^n)).n(digits=3),
        2^n*3^n*5^n)))
```

You'll notice that although we quickly get a ratio above 3 (so that $\sigma(n) >$), we don't seem to get much further. Why?

A helpful thing to think about with this is the following rewrite, using the

formula for $\sigma(n)$ with the usual writing of $n = \prod_{i=1}^{k} p_i^{e_i}$:

$$\frac{\sigma(n)}{n} = \frac{\prod_{i=1}^{k} \left(\frac{p_i^{e_i+1}-1}{p_i-1} \right)}{\prod_{i=1}^{k} p_i^{e_i}} = \prod_{i=1}^{k} \frac{p_i - (1/p_i^{e_i})}{p_i - 1} \approx \prod_{i=1}^{k} \frac{p_i}{p_i - 1}$$

Based on this, we should expect this approximation to be very close when e_i are all quite large. Then for large numbers, since $\frac{p}{p-1} > 1$, if we multiply by enough of these we will get very large numbers and so $\sigma(n)/n$ will be greater than any given C, and then $\sigma(n) > Cn$.

Of course, $p = 2$ is the best for this since $\frac{2}{2-1} = 2$, but the other primes will hopefully be useful for this as well. For instance, $n = 2^{10}3^{10}$ will have

$$\sigma(n)/n = \frac{2 - 1/2^{10}}{2-1} \frac{3 - 1/3^{10}}{3-1} \approx \frac{2}{2-1} \frac{3}{3-1} = 3$$

so certainly $\sigma(6^{10})$ will be nearly $3 \cdot 6^{10}$.

If we multiply it by 5 as well that should do it, and that gives the results we saw in the previous cell:

$$\frac{2 - 1/2^{10}}{2-1} \frac{3 - 1/3^{10}}{3-1} \frac{5 - 1/5}{5-1} \approx \frac{2}{2-1} \frac{3}{3-1} \frac{5}{5-1} = 2 \cdot \frac{3}{2} \cdot \frac{5}{4} = \frac{15}{4} = 3.75$$

We can check out some of these ideas, and how much bigger we can get.

```
print (sigma(6^10)/(6^10)).n()
print (sigma(5*6^10)/(5*6^10)).n()
```

```
print (sigma(2^4*3^4*5^4*7)/(2^4*3^4*5^4*7)).n(digits=3)
```

```
N = prod([p^4 for p in primes_first_n(100)])
print (sigma(N)/N).n(digits=3)
```

Continuing this for more primes suggests the following.

Fact 19.3.2. *For any positive C, there is a positive integer n such that*

$$\sigma(n) > Cn \, .$$

The argument outlined above is not completely rigorous, but is good enough for now. Trying to prove it this way could bring the distribution of primes to the table, so doing so might not be trivial.

19.4 Perfect Numbers

19.4.1 A perfect definition and theorem

Definition 19.4.1. When the ratio $\frac{\sigma(n)}{n}$ is exactly 2, we say n is a **perfect number**.

This is a big definition, and it goes back at least to Euclid. Euclid defines the notion at the beginning of the number-theoretic books of the Elements, and only mentions it again over one hundred propositions later, where he proves that certain numbers are, in fact, perfect. (A careful reader will notice that the primes in question are, in fact, the Mersenne primes!) Such a conclusion is a fitting end, as William Dunham says in his book, *Journey through Genius* [C.4.5].

Theorem 19.4.2. *If n is a number such that $2^n - 1$ is prime, then the (even) number $2^{n-1}(2^n - 1)$ is perfect.*

Proof. Euclid's proof (in the link) of this is worth looking at. □

Many centuries later, Euler proved the converse; we will prove them together. (See also Chapter 1 of Dunham's *Euler: The Master of Us All* [C.4.6].)

Theorem 19.4.3 (Characterization of Even Perfect Numbers)**.** *If n is an even number, it is perfect if and only if it is the product of a power 2^{n-1} and a prime of the form $2^n - 1$.*

Proof. First, assume that $2^n - 1$ is prime. Then the factors of $2^{n-1}(2^n - 1)$ are coprime, so

$$\sigma\left(2^{n-1}(2^n-1)\right) = \sigma\left(2^{n-1}\right)\sigma(2^n-1) = (2^n-1)(2^n-1+1)$$

The steps are because of multiplicativity and the formulas we had earlier (see Theorem 19.2.5) for σ of powers of two and primes. But then

$$(2^n-1)(2^n-1+1) = 2^n(2^n-1) = 2\left[2^{n-1}(2^n-1)\right]$$

so that the sum of divisors is exactly twice the original number.

Now for the converse, which is somewhat longer. Let us start with an even perfect number, which is perforce divisible by some power of two.

Looking ahead, call this power the $(n-1)$th power! Then our even perfect number may be written as $2^{n-1}q$, where q is the (odd) quotient.

Let's divide the rest of the proof into several pieces. First, two facts.

- We know that this number is perfect, so
$$\sigma\left(2^{n-1}q\right) = 2 \cdot 2^{n-1}q = 2^n q$$
- We also know how to compute σ, so
$$\sigma\left(2^{n-1}q\right) = \sigma\left(2^{n-1}\right)\sigma(q) = (2^n-1)\sigma(q)$$

We can combine these observations to see that

$$2^n q = (2^n-1)\sigma(q)$$

Note that this means $2^n - 1 \mid q$, since q is the only odd part of the left-hand side (implicitly using some of Theorem 6.3.2). Let's write

$$(2^n-1)m = q\,.$$

Substituting, we have

$$2^n(2^n-1)m = (2^n-1)\sigma(q) \Rightarrow 2^n m = \sigma(q)$$

Since m and q both divide q, by the definition of σ we have

$$\sigma(q) \geq q + m = (2^n-1)m + m = 2^n m$$

Since these two divisors (q and m) alone add up to $\sigma(q)$, it must be true that q has exactly these two divisors, so it is prime. *That* means$m = 1$, and $q = 2^n - 1$, and so the perfect number is $2^{n-1}(2^n - 1)$. Great! □

We will leave the question about whether there are *odd* perfect numbers to Section 19.5.

19.4.2 Speculation and more terminology

There are many things people have claimed about numbers of this type. A Hellenistic Roman in the first century in Gerasa[1] named Nichomachus claimed that the nth perfect number had n decimal digits.

Nicomachus was more concerned with mystical claims about perfect numbers (which many repeated), but this mathematical assertion continued to be made for over a thousand years. However, knowing what we do about Mersenne primes, we see that the fifth such prime is 13, so that the next perfect number,

$$\left(2^{13}-1\right)\cdot 2^{12},$$

was very large and so lay mysterious for a long time. It was apparently discovered in the fifteenth century.

```
(2^13-1)*2^12
```

Until the early modern period, such numbers were basically inaccessible.

Number theorists (often of the amateur variety, but certainly not always) have come up with all kinds of other names for various concepts related to $\sigma(n)/n$.

Definition 19.4.4. Recall that if $\sigma(n) = 2n$, then n is perfect.

- If $\sigma(n) = kn$ for some integer k, then we say that n is **k-perfect**.
- Or, if $\sigma(n) > 2n$, then n is **abundant.**
- If $\sigma(n) < 2n$, we say n is **deficient**.

As it will turn out, these things are not really good characterizations of what it means to have "too many" or "too few" divisors, but in recognition of the Greeks' contributions we keep this allusive and fairly standard terminology. As examples, Exercise 19.6.7 asks for a 3-perfect number, if one exists, and Exercise 19.6.17 asks for a 4-perfect number.

Definition 19.4.5. Here are some less well-known, but nonetheless interesting, terms.

- A number is **pseudoperfect** if it is the sum of *some* of its divisors (other than itself).
- A number n is **superabundant** if the ratio $\sigma(n)/n$ for n is bigger than the value of the ratio for all smaller $m < n$.
- A number is **weird** if it is abundant but not pseudoperfect. (There is a famous paper of Erdős on this topic.)

There are many questions one can ask about these and other definitions; see Exercise Group 19.6.15–19.6.21 One cheeky such question is this.

Question 19.4.6. Is a perfect number pseudoperfect?

One other interesting idea is that of **amicable** numbers, which are pairs m, n of numbers such that $\sigma(n) = \sigma(m) = m + n$. Clearly any perfect number is amicable with itself. The smallest pair of unequal amicable numbers is $(220, 284)$; this was known to the ancient Greeks, cherished by some medieval

[1] Interestingly, this is the same place as one setting of the Biblical story of the demons called "Legion" who went into swine.

Muslims, and apparently was *not* improved upon until the modern number-theoretic era.

Fermat, Descartes, and Euler all worked with this and found large examples, but it turns out that the next smallest pair was found by a sixteen-year old Italian boy in 1860!

```
sigma(1184),sigma(1210),1184+1210
```

Apparently he came up with this by trial and error, though no one knows for sure. The internet can provide some of the most current data on these pairs.

There is a way to get as many amicable pairs as you like, discovered by Ibn Qurra and (later) Fermat, finally used by Euler.

Algorithm 19.4.7 (Get Amicable Numbers)**.** *Here is one way to get amicable numbers.*

- *Make a list of numbers of the form* $p_n = 3 \cdot 2^n - 1$ *and* $q_n = 9 \cdot 2^{2n-1} - 1$.
- *Then check if* p_{n-1}, p_n, *and* q_n *are all prime.*
- *If so, then* $2^n p_{n-1} p_n$ *and* $2^n q_n$ *are an amicable pair.*

Proof. Since only primes and powers of two are involved, it's easy to calculate σ in this case, so proving it is left as an exercise (see Exercise 19.6.21). □

```
@interact
def _(n=[2..20]):
    pretty_print(html("We have $p_{%s}=%s$ and
        $p_{%s}=%s$"%(n,3*2^(n-1)-1,n,3*2^(n)-1)))
    pretty_print(html("And $q_{%s}=%s$ as
        well."%(n,9*2^(2*n-1)-1)))
    if is_prime(3*2^n-1) and is_prime(3*2^(n-1)-1) and
        is_prime(9*2^(2*n-1)-1):
        pretty_print(html("Then the pair $%s$ and $%s$ is
            amicable!"%(2^n*(3*2^(n-1)-1)*(3*2^(n)-1),
            2^n*(9*2^(2*n-1)-1))))
    else:
        pretty_print(html("Doesn't give an amicable pair"))
```

19.4.3 The abundancy index

It's time to give a name to the mysterious ratio at the core of this section.

Definition 19.4.8. The ratio $\frac{\sigma(n)}{n}$ may be called the **abundancy index** of n.

A beautiful thing is that once you name a concept, you can ask questions about it. Here's another largely open question which seems like it should be easy...

Question 19.4.9. Rather than asking which integers can be gotten, which *rational* numbers can be gotten as $\frac{\sigma(n)}{n}$?

```
@interact
def _(n=(20,[1..200])):
    cols = ceil(n/10)
    T = [cols*['$n$','$\sigma(n)/n$']]
    list = [[i,(sigma(i)/i)] for i in range(1,n+1)]
    list.extend((10-(len(list)%10))*['',''])
    for k in range(10):
        t = [item for j in range(cols) for item in
            list[k+10*j]]
        T.append(t)
    pretty_print(html(table(T,header_row = True, frame =
        True)))
```

There are some interesting theorems about this already known. For one thing, the abundancy index is the same thing as $\sigma_{-1}(n)$.

Fact 19.4.10.

$$\sigma_{-1}(n) = \frac{\sigma(n)}{n}$$

Proof. We have that

$$\sigma(n)/n = \left(\sum_{d|n} d\right)/n$$

Now note that for every $d \mid n$, the quotient is also an integer divisor d' of n. So

$$\sigma(n)/n = \sum_{d|n} \frac{1}{d'}$$

This is the same list as the original divisor list, so reordering gives

$$\sigma(n)/n = \sum_{d|n} \frac{1}{d} = \sigma_{-1}(n)$$

□

Fact 19.4.11. *Clearly all such numbers are in the interval $[1, \infty)$! Here are some more known facts about the abundancy index.*

- *If $m \mid n$, then $\sigma_{-1}(n) \geq \sigma_{-1}(m)$.*
- *If $\sigma_{-1}(n) = \frac{a}{b}$ in lowest terms, then $b \mid n$.*
- *If r is "caught" between $\sigma(n)$ and n (such that $n < r < \sigma(n)$) and is relatively prime to n, then r/n is not an abundancy index.*

Proof. We skip the proof, but proving the first two facts is left as Exercise 19.6.22. □

Holdener and Stanton picturesquely call rational numbers which are not abundancies **abundancy outlaws**. The end of this hyper-linked paper [C.6.11] has a nice list of which numbers thus far have been found, and which have not.

19.5 *Odd* Perfect Numbers

19.5.1 Are there odd perfect numbers?

We will return to the abundancy index momentarily. First, we return to a question alluded to above – one whose answer is still unknown, and open after two and a half millennia:

Question 19.5.1. Does there exist an odd perfect number?

Yikes!

We do know some things about the question. First, recall from Section 19.3 that

$$\frac{\sigma(n)}{n} = \prod_{i=1}^{k} \frac{p_i - 1/p_i^{e_i}}{p_i - 1} < \prod_{i=1}^{k} \frac{p_i}{p_i - 1}$$

when n is a product of the prime powers $p_i^{e_i}$. This leads to the following first information.

Theorem 19.5.2 (Odd perfect numbers aren't simple)**.** *Here are simple forms of numbers that can't be perfect.*

- *An odd perfect number cannot be a prime power.*
- *An odd perfect number cannot be a product of exactly* two *prime powers.*
- *An odd perfect number cannot be a product of exactly* three *prime powers unless the first two are* 3^e *and* 5^f*.*

Proof. We leave many details to Exercise 19.6.24.

- An odd perfect number cannot be a prime power. This is easy; $2 = \frac{\sigma(n)}{n} < \frac{p}{p-1}$ isn't possible, even for $p = 2$; since we are looking for an *odd* perfect number, it definitely won't be possible!
- An odd perfect number cannot be a product of exactly *two* prime powers. Use the same idea, but now with the biggest possible values for odd primes.
- An odd perfect number cannot be a product of exactly *three* prime powers unless the first two are 3^e and 5^f. This proof is slightly longer.
  - Suppose that 3 is not the smallest prime involved. Then the biggest that
    $$\frac{p_1}{p_1 - 1}\frac{p_2}{p_2 - 1}\frac{p_3}{p_3 - 1}$$
    can be is
    $$\frac{5}{4}\frac{7}{6}\frac{11}{10} = \frac{77}{48} < 2\,.$$
  - Suppose that 5 is not the second-smallest prime involved (assuming 3 is the smallest). We again get a contradiction.

□

19.5.2 The abundancy index and odd perfect numbers

What is particularly interesting about this is the connection to something we have tacitly avoided until now. This is the question whether there are *odd* perfect numbers! The connection below is due to P. Weiner in [C.6.14]

We begin with a useful lemma, which answers questions very closely related to Exercises 19.6.11 and 19.6.12.

Lemma 19.5.3. *If n and $\sigma(n)$ are both odd, then n is a perfect square.*

Proof. If n is odd, it is a product of odd prime powers. Let's look at σ as applied to each piece, thanks to multiplicativity.

If $\sigma(n)$ is odd, then each factor $1 + p + p^2 + \cdots + p^e$ is odd. Such a factor of $\sigma(n)$ is a sum of odd numbers, which is only odd if there is an odd number of them.

Since there are $e+1$ summands, e must be even for every primes p dividing $n.$, which finishes proving the lemma. □

Theorem 19.5.4. *If $\frac{5}{3}$ is the abundancy index of N, then $5N$ is an odd perfect number.*

Proof. Assume this works for some N. Then $3\sigma(N) = 5N$.

Let's look at divisors. First, $3 \mid N$. So if N is even, then $6 \mid N$, so by Fact 19.4.11,

$$\sigma_{-1}(N) \geq \sigma_{-1}(6) = 2 > \frac{5}{3}\,,$$

which is impossible. If N is not even, then N is odd, so $3\sigma(N) = 5N$ is odd, which implies $\sigma(N)$ itself is odd.

Since $3 \mid N$ and using Lemma 19.5.3, we see that we must have that $3^2 \mid N$.

Let's return to the divisors. We know that $5 \nmid N$, because otherwise

$$\sigma_{-1}(N) \geq \sigma_{-1}\left(3^2 \cdot 5\right) = \frac{26}{15} > \frac{5}{3}$$

which is again impossible.

Now we can compute directly that

$$\sigma_{-1}(5N) = \sigma_{-1}(5)\sigma_{-1}(N) = \frac{6}{5}\frac{5}{3} = 2\,!$$

□

19.5.3 Even more about odd perfect numbers, if they exist

Naturally, all of this is somewhat elementary; there are many more criteria. They keep on getting more complicated, so I can't list them all, but here is a selection, including information from two big computer-assisted searches going on right now.

Fact 19.5.5. *An odd perfect number must (as of 2016):*

- *Be greater than 10^{1500}. (The most recent announcement says researchers have 'pushed the computation to 10^{2000}':*
- *Have at least 101 prime factors (not necessarily distinct).*
- *Have at least 10* distinct *prime factors. (This is new and relies on heavy computation by Pace Nielsen in* Odd perfect numbers, Diophantine equations, and upper bounds *in* Mathematics of Computation.*)*

- *Have a largest prime factor at least* 10^8.
- *Have a second largest prime exceeding* 10000.
- *Have the sum of the reciprocals of the* prime divisors *of the number between* about 0.6 *and* 0.7.
- *Have the sum of the reciprocals be finite (since the sum of the reciprocals of* all *perfect numbers is finite!). In fact, the sum of the reciprocals must be less than* 2×10^{-150} *(see [C.6.6]), and that of all perfects is less than about* 0.0205.
- *Obey the rule that if* n *is an odd perfect number, then* $n \equiv 1 \mod 12$ *or* $n \equiv 9 \mod 36$.

Finally, as an appropriate way to finish up this at times overwhelming overview, since he finished the characterization of *even* perfect numbers, let us present Euler's own criterion – see also the linked article [C.6.19] by Euler expert Ed Sandifer.

Proposition 19.5.6. *An odd perfect number must be of the form* $p^e m^2$, *where* m *is odd,* p *is prime, and* p and e *are both* $\equiv 1 \mod 4$.

19.6 Exercises

1. Review the proof of Fact 9.5.2 that $\phi(n)$ is multiplicative. Can you think of a way to modify it *directly* to prove that σ or σ_0 are multiplicative?

My students discovered various facts about the functions in this chapter on their own; why not you?

2. Conjecture and prove a formula for the difference between $\sigma_k(p)$ and $\sigma_k(p^2)$. (Thanks to Becca Brule and Olivia Gray.)

3. Conjecture and prove a necessary (or even sufficient) criterion for when $5 \mid \sigma_2(2k)$. (Thanks to Andrew Kwiatkowski and Daniel Brito.)

4. Come up with some new (to you) conjecture about one of these functions you observed from the data, and which isn't mentioned in this book. Tell what led you to this conjecture.

5. Read Euclid's original proof that certain even numbers are perfect and write it down in modern notation.

6. Do you think these numbers should be called perfect, and why? Establish a connection to GIMPS.

7. Can you find a number such that $\sigma(n) = 3n$?

8. Could there be a function $g(n)$ which is multiplicative, where $g(2n) = 0$, $g(n) = a_1 = 1$ if $n \equiv 1 \pmod 8$, $g(n) = a_2$ if $n \equiv 3 \pmod 8$, $g(n) = a_3$ if $n \equiv 5 \pmod 8$, and $g(n) = a_4$ if $n \equiv 7 \pmod 8$?

9. Let $\tau_o(n)$ and $\sigma_o(n)$ be the same as τ and σ but where only *odd* divisors of n are considered; let τ_e and σ_e be similar for even divisors of n. Evaluate these functions for $n = 1$ to 12, and decide whether each of them is multiplicative or not (either proving it, or showing not by counterexample).

10. Use the estimate toward the end of Section 19.3 for σ to find numbers for which $\sigma(n) > 5n$ and $\sigma(n) > 6n$. (Possibly long.)

11. Discover and prove conditions for which $\tau(n)$ and $\sigma(n)$ are even and odd numbers.

12. Show that if n is odd then $\tau(n)$ and $\sigma(n)$ have the same parity.

13. For which types of n is $\tau(n) = 4$?

14. Prove that if $n \equiv 7 \pmod 8$, then $8 \mid \sigma(n)$.

Here are facts about various definitions beyond perfect numbers in Subsection 19.4.2.

15. Show that every prime power is deficient.

16. Show that a multiple of an abundant number is abundant.

17. Find a 4-perfect number.

18. Compute "by hand" σ_{-1} for the numbers up to 30. Come up with and prove a criterion for when $\sigma_{-1} = 2$.

19. Find three pseudoperfect numbers less than 100.

20. Find a weird number less than 100.

21. In the proof of Algorithm 19.4.7, confirm that if p_n, p_{n-1}, and q_n are prime, then the numbers in question are amicable.

22. Prove the first and second facts about the abundancy index in Fact 19.4.11.

23. Find five numbers that must be abundancy outlaws based on the facts (don't just copy from the list).

24. Fill in the details in the proof of Theorem 19.5.2 (that odd perfect numbers need at least three prime divisors, and that 3 and 5 would need to be the first two if there were exactly three).

25. Read the article linked right after Fact 19.5.5 about Euler and *odd* perfect numbers, and restate and reprove his criterion in modern notation.

26. There are always more connections. Here are a few exercises about a formula one would have likely never guessed.

$$\left(\sum_{d|n} \tau(d)\right)^2 = \sum_{d|n} \tau(d)^3$$

First, test it out by hand with $n = 6$ and $n = 8$. Then try it with bigger numbers below:

```
@interact
def _(n = 24):
    divs = divisors(n)
    pretty_print(html("The divisors of $%s$ are
        $%s$"%(n,divs)))
    pretty_print(html("And $\\tau$ of each of them is
        $%s$"%([sigma(div,0) for div in divs])))
    pretty_print(html("The sums of the cubes and the
        square of the sum are $%s$ and $%s$,
        respectively!"%(sum([sigma(div,0)^3 for div in
        divs]),sum([sigma(div,0) for div in divs])^2)))
```

Start a proof by noting that it's clearly true for a prime power $n = p^e$, for which $\tau(p^f) = f + 1$, and all divisors of n look like such a power of p.
Continue the proof by examining the proof that σ_i is multiplicative for what can be said about the divisors of mn, and how a sum over divisors $d \mid mn$ can be a product of two different sums over divisors of m and n.

Chapter 20

Long-Term Function Behavior

We will now move on to think of these same functions in a different way from the previous chapter. We will examine different *limits* in number theory, and how integrals and calculus are inextricably bound up with this sort of question.

If, after this chapter, you are interested in more of this kind of material, definitely check out[1] Stopple's excellent [C.3.5], to which I am indebted for many of the ideas here, or the more challenging book [C.3.6] by Apostol.

Finally, note that some proficiency in calculus is helpful in understanding the results in this chapter, though a proper course is not necessarily a prerequisite.

20.1 Sums of Squares, Once More

Our motivational example will be the one we discussed in Section 18.1. Recall that $r(n)$ denotes the (total) number of ways to represent n as a sum of squares, so that $r(3) = 0$ but $r(9) = 4$ and $r(5) = 8$. Then we saw in Fact 18.2.7, more or less rigorously, that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} r(k) = \pi \,.$$

20.1.1 Errors, not just limits

As it happens, we can say something far more specific than just this limit. Recall one of the intermediate steps in our proof.

$$\pi\left(1 - \sqrt{\frac{2}{n}} + \frac{1}{2n}\right) \leq \frac{1}{n}\sum_{k=0}^{n} r(k) \leq \pi\left(1 + \sqrt{\frac{2}{n}} + \frac{1}{2n}\right)$$

Notice that if I subtract the limit, π, from the bounds, I can think of this in terms of an *error*. Using absolute values, we get, for large enough n,

$$\left|\frac{1}{n}\sum_{k=0}^{n} r(k) - \pi\right| \leq \pi\left(\frac{\sqrt{2}}{\sqrt{n}} + \frac{1}{2n}\right) \leq Cn^{-1/2}$$

[1] Two other books with useful presentations are the terse one in [C.1.9] and the more intuitive, if shorter, one in [C.1.11].

where the value of C is not just $\pi\sqrt{2}$, but something a little bigger because of the $\frac{1}{2n}$ term.

In the next two cells we set up some functions and then plot the actual number of representations compared with the upper and lower bound implied by this analysis.

```
def r2(n):
    n = prime_to_m_part(n,2)
    F = factor(n)
    ret = 4
    for a,b in F:
        if a%4==3:
            if b%2==1:
                return 0
            else:
                n = prime_to_m_part(n,a)
        else:
            ret = ret * (b+1)
    return ret

def L(n):
    ls = []
    out = 0
    for i in range(1,n+1):
        out += r2(i)
        ls.append((i,out/i))
    return ls
```

```
@interact
def _(n=100):
    P = line(L(n))
    P += plot(pi+pi*sqrt(2)/sqrt(x),x,3,n,color='red')
    P += plot(pi-pi*sqrt(2)/sqrt(x),x,3,n,color='red')
    P += plot(pi,x,3,n,color='red',linestyle='--')
    show(P)
```

Note that the actual number is well within the bounding curves given by the red lines. This shows a general rule that, as often happens, the constant we *proved* is a lot bigger than necessary. Often new research is about improving such bounds.

20.1.2 Landau notation

It turns out there is a nice notation for how 'big' an error is.

Definition 20.1.1 (Big Oh)**.** We say that $f(x)$ is $O(g(x))$ ("eff of eks is Big Oh of gee of eks") if there is some constant C for which

$$|f(x)| \leq Cg(x) \text{ for all large enough } x\,.$$

This is known as **Landau notation**.

See Exercise Group 20.6.1–20.6.5 for some practice with this.

Example 20.1.2. The average number of representations of an integer as a sum of squares is π, and if you do the average up to N, then the *error* will

be no worse than some constant times $1/\sqrt{N}$. So the sum's error is Big Oh of $1/\sqrt{N}$.

It is unknown in this case just how small the error term really is. In 1906 it was shown that it is $O(x^{-2/3})$; see the following graphic.

```
@interact
def _(n=100,C=pi):
    P = line(L(n))
    P += plot(pi+C/x^(2/3),x,3,n,color='red')
    P += plot(pi-C/x^(2/3),x,3,n,color='red')
    P += plot(pi,x,3,n,color='red',linestyle='--')
    show(P)
```

It is also known that the error is *not* $O(x^{-3/4})$.

Now let's apply these ideas to the τ and σ functions.

Question 20.1.3. What is the "average" number of divisors of a positive integer? What is the "average" sum of divisors of a positive integer?

It turns out that clever combinations of many ideas from the course as well as calculus ideas will help us solve these questions! And they will motivate us to ask the (much harder) similar questions one can ask about prime numbers, starting in Chapter 21.

20.2 Average of Tau

20.2.1 Beginnings

Let's begin by observing the following graphic of the average for τ.

Sage note 20.2.1 (Try to be efficient)**.** The first cell computes values of τ as a list, so that we don't recalculate the entire sum each time. Try being efficient in your programming!

```
def L(n):
    ls = []
    out = 0
    for i in range(1,n+1):
        out += sigma(i,0)
        ls.append((i,out/i))
    return ls
```

```
@interact
def _(n=100):
    P = line(L(n))
    show(P)
```

The graphic shows how the average value of τ up to n changes as we let n get bigger. This isn't enough data to tell whether there is a limiting value for the average value of $\tau(n)$, even if you look out to the first 1000 integers, but it's suggestive. Part of the unpredictability is from primes; every prime number contributes just 2 to the total (and so reduces the average value)!

Nonetheless, thinking about this might lead us to look a little deeper. For example, the 'trend' is concave down. So let's look at comparing it with various concave down functions. (The following interact supports multiplied constants with them as well.)

```
@interact
def _(n=100,C=.5,f=[x^(1/2), x, x^(1/3), x^(1/4), log(x),
    log(log(x)), x^(1.5), x^2]):
    f(x) = f
    P = line(L(n),legend_label='average␣of␣$\\tau$')
    P += plot(C*f,(x,1,n), color='black', linestyle='--',
        legend_label='$%s%s$'%(RDF(C),latex(f(x))))
    show(P)
```

At the very least I can tell that the average value is Big Oh of a certain function. But how does it go on?

```
line(L(1000000))
```

Here, out to one million, is once again our graph of averages of $\tau(n)$ versus n. Certainly this looks sort of like some kind of fractional exponent function, though a very slowly growing one – probably slower than $\sqrt{x}$, our initial estimate in the interact.

20.2.2 Heuristics for tau

We'll start with a heuristic, going right back to the sieve of Eratosthenes.

In that algorithm (6.2.2), we proved that in order to test whether n is prime, you just have to check all numbers up through $\sqrt{n}$. This is because any divisor $\sqrt{n} < d < n$ implies the existence of a divisor $\frac{n}{d}$ such that

$$1 = \frac{n}{n} < \frac{n}{d} < \frac{n}{\sqrt{n}} = \sqrt{n}\,.$$

So the absolute most number of divisors possible (for a given n) is if *every* number d less than $\sqrt{n}$ was a divisor, and then all the $\frac{n}{d} > \sqrt{n}$ you get were also divisors. That is a silly idea beyond very small n, but let's go with it.

Even if all the divisors were there, you would have $\tau(n) \leq 2\sqrt{n}$ so that $\tau(n)$ is $O(\sqrt{n})$.

That estimate is very important! It means we can get a sense of a first bound on the average value of τ. At the very least we have that

$$\frac{1}{n}\sum_{k=1}^{n}\tau(k) \leq \frac{1}{n}\sum_{k=1}^{n}2\sqrt{k} = \sum_{k=1}^{n}\frac{1}{n}2\sqrt{n(k/n)}$$

20.2.3 Using sums to get closer

Let's rewrite this inequality in a more suggestive form.

$$\frac{1}{n}\sum_{k=1}^{n}\tau(k) \leq \sum_{k=1}^{n}\frac{1}{n}2\sqrt{n(k/n)}$$

This form looks an awful lot like a Riemann sum with $\Delta x = \frac{1}{n}$. To review, recall writing a Riemann sum for $\int_0^1 x^2\,dx$ in the form

$$\frac{1}{n}\left(\frac{1}{n}\right)^2 + \frac{1}{n}\left(\frac{2}{n}\right)^2 + \cdots + \frac{1}{n}\left(\frac{n}{n}\right)^2 .$$

(If you need a calculus refresher, there are several great free calculus texts in the American Institute of Mathematics list of approved textbooks.)

Doing the same type of summation for the function $2\sqrt{nx}$ would give

$$\sum_{k=1}^{n} \frac{1}{n} 2\sqrt{n(k/n)} \approx \int_0^1 2\sqrt{nx}dx = 2\sqrt{n}\int_0^1 \sqrt{x}\,dx = \frac{4}{3}\sqrt{n}\,.$$

That certainly suggests that the average of τ might be $O(\sqrt{n})$ with $C = 4/3$.

To make this rigorous, we will need to make a slight change of point of view in order to ensure it will be viewed as a left-hand sum of an increasing function (and hence the Riemann sum is less than the actual value of the integral).

Namely, consider that

$$\frac{1}{n}\sum_{k=1}^{n} 2\sqrt{k} = \sum_{k=0}^{n-1}\left(\frac{1}{n}\right)2\sqrt{k+1} = \sum_{k=0}^{n-1}\left(\frac{1}{n}\right)2\sqrt{n(k/n)+1} \le \int_0^1 2\sqrt{nx+1}\,dx$$

This integral evaluates to

$$\frac{4}{3}\sqrt{n}\left[\left(1+\frac{1}{n}\right)^{3/2} - \left(\frac{1}{n}\right)^{3/2}\right] .$$

The big extra factor on the right can be shown to be decreasing (using derivatives), and is always less than 2 for integers, so the entire expression will always be less than $\frac{8}{3}\sqrt{n}$.

Thus one can write

$$\frac{1}{n}\sum_{k=1}^{n}\tau(k) \le \frac{1}{n}\sum_{k=1}^{n} 2\sqrt{k} \le \frac{8}{3}\sqrt{n}$$

so that the average value is bounded by a constant times $\sqrt{n}$ and is hence $O(\sqrt{n})$. This implies, *perhaps*, that the average number of divisors goes steadily up! (If so, it guarantees it's concave down.)

20.2.4 But Big-Oh isn't enough

However, we might also want to know what the average value of τ *is*. The preceding subsections only tell us what it's less than! Here, it seems that it's hard to find the "right" value of C so that the average value would be the same order as $\sqrt{n}$.

```
def L(n):
    ls = []
    out = 0
    for i in range(1,n+1):
        out += sigma(i,0)
        ls.append((i,out/i))
    return ls

P = line(L(1000000))
```

```
@interact
def _(a=.02,n=2):
    show(P + plot(a*x^(1/n), (x,1,10^6),
        color='red',linestyle='--'))
    pretty_print(html("Blue␣is␣the␣average␣value␣of␣
        $\\tau$$"))
    pretty_print(html("Red␣is␣$%sx^{1/%s}$"%(a,n)))
```

Try $x^{1/3}$ in the interact; it doesn't seem to make matters any better.

In fact, one can show that $\tau(n) = O(\sqrt[3]{n})$ as well. Here are the steps one might take. We make fleshing out the details Exercise 20.6.8 (adapted from [C.3.5]):

- First, note that τ is multiplicative.
- For a given prime p, note that $\tau(p^x) = x + 1$ grows much more slowly than $(p^x)^{1/3} = p^{x/3}$, which is exponential in x.
    - What value do each of these have at $x = 0$?
    - Take derivatives of both functions at $x = 0$ to show that the growth statement is definitely true for $p \geq 23$.
    - Show that for each prime p less than 23 there is an x_p such that the growth statement is true after x_p.
- Put these pieces of information together to show that τ is $O\left(x^{1/3}\right)$.

20.3 Digging Deeper and Finding Limits

So where *does* the number of divisors function go? To answer this, we will look at a very different graph!

The fundamental observation that makes this graphic possible is that $\tau(n)$ is precisely the same as the number of positive integers (x, y) such that $xy = n$. Before going on, spend some time convincing yourself of this.

Then, if we translate $xy = n$ to a graph of $y = n/x$ and (x, y) to a lattice point, we get the following.

```
n=10
viewsize=n+1
var('x,y')
g(x)=1/x
P=Graphics()
P += contour_plot(x*y,(x,0,viewsize),(y,0,viewsize),
    cmap=['black'],fill=False,
    contours=[2,3,8],labels=True,
    label_inline=True,label_inline_spacing=5,
    label_fmt="$n=%d$")
P += plot(n*g,(x,0,n+1))
grid_pts = [[i,j] for i in [1..viewsize] for j in
    [1..viewsize]]
P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
lattice_pts = [coords for coords in grid_pts if
    (coords[0]*coords[1]<=n)]
P += points(lattice_pts, rgbcolor = (0,0,1),pointsize=20)
show(P,ymax=viewsize,aspect_ratio=1)
```

20.3.1 Moving toward a proof

To be more in line with our previous notation, we will say that $\tau(n)$ is exactly given by the number of positive integer points $\left(d, \frac{n}{d}\right)$ with the property that $d\frac{n}{d} = n$. Now we can interpert $\sum_{k=1}^{n} \tau(k)$ as the number of lattice points *on or under* the hyperbola $y = n/x$.

This is a *completely* different way of thinking of the divisor function! We can see it for various sizes below.

```
@interact
def _(n=(15,range(2,50))):
    viewsize=n+1
    g(x)=1/x
    P=Graphics()
    P += plot(n*g,(x,0,n+1))
    P += plot(2*g,(x,0,n+1), linestyle="--")
    if n>7:
        P += plot((n-5)*g,(x,0,n+1),linestyle="--")
    grid_pts = [[i,j] for i in [1..viewsize] for j in
        [1..viewsize]]
    P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]*coords[1]<=n)]
    P += points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    show(P,ymax=viewsize,aspect_ratio=1)
```

So what we will do is try to look at the lattice points as approximating an area! Just like with the sum of squares function (recall Subsection 18.2.3 and Section 20.1), we will exploit the geometry. For each lattice point involved in $\sum_{k=1}^{n} \tau(k)$, we put a unit square to the lower right.

```
@interact
def _(n=(8,range(2,25))):
    viewsize=n+1
    g(x)=1/x
    P=Graphics()
    P += plot(n*g,(x,0,n+1))
    P += plot(n*g,(x,1,n),fill=True,fillalpha=.3)
    grid_pts = [[i,j] for i in [1..viewsize] for j in
        [1..viewsize]]
    P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]*coords[1]<=n)]
    P += points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    squares=[line([[k,l], [k+1,l],[k+1,l-1],
        [k,l-1],[k,l]], rgbcolor=(1,0,0)) for [k,l] in
        lattice_pts]
    for object in squares:
        P += object
    show(P,ymax=viewsize,aspect_ratio=1)
```

In examining this graph, we will interpret the lattice points as two different sums.

- We can think of it as $\sum_{k=1}^{n} \tau(k)$ – adding up the lattice points along each *hyperbola*.

- We can think of it as $\sum_{j=1}^{n} \left\lfloor \frac{n}{k} \right\rfloor$, or adding up the lattice points in each vertical column.

The area of the squares can then be thought of as another Riemann-type sum, similar to our summation of τ.

It should be clear that the area, an estimate for the sum, is "about"

$$\int_1^n \frac{n}{x} dx = n\log(x)\Big|_1^n = n\log(n) - n\log(1) = n\log(n)$$

where the logarithm is the 'natural' one. Why is this integral actually a *good* estimate, though? The answer is in the *error*!

```
@interact
def _(n=(8,range(2,25))):
    viewsize=n+1
    g(x)=1/x
    P=Graphics()
    P += plot(n*g,(x,1,n))
    P += plot(piecewise([[(j,j+1),floor(n/j)] for j in
        [1..n-1]]), (x,1,n), fill=n/x,fillalpha=.3,
        linestyle='') + plot(1,(x,n,n+1),fill=True,
        fillalpha=.3,linestyle='')
    grid_pts = [[i,j] for i in [1..viewsize] for j in
        [1..viewsize]]
    P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]*coords[1]<=n)]
    P += points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    squares=[line([[k,l],[k+1,l],[k+1,l-1],[k,l-1],[k,l]],
        rgbcolor=(1,0,0)) for [k,l] in lattice_pts]
    for object in squares:
        P += object
    show(P,ymax=viewsize,aspect_ratio=1)
    pretty_print(html("Error␣between␣$\\tau(%s)$␣and␣
        $%s\log(%s)$"%(n,n,n)))
```

Look at the shaded difference between the area under the curve (which is $n\log(n)$) and the area of the red squares (which is the sum of all the τ values).

- All the areas where the red squares are above the hyperbola add up to less than n, because they are all 1 in width or less, and do not intersect vertically (they stack, as it were).
- Similarly, all the areas where the hyperbola is higher add up to less than n, because they are all 1 in height or less, and are horizontally non-intersecting.

(Actually, we would expect they would cancel quite a bit … and they do, as we will see. We don't need that yet.)

We can summarize this in the following three implications.

Fact 20.3.1.

- *The error* $\sum_{k=1}^{n} \tau(k) - n\log(n)$ *is a positive real number less than* n *minus a (different positive real) number less than* n*.*
- *So the error is* certainly $O(n)$ *(less than some multiple of* n *as* n *gets huge).*

- *So, the error in the* average *is less than some* constant *as n gets huge! i.e.,*

$$\frac{1}{n}\sum_{k=1}^{n}\tau(k) - \log(n) = O(1)$$

(Again, throughout we use $\log(n)$ *to mean the natural logarithm of base e.)*

We can verify this graphically by plotting the average value against $\log(n)$.

```
def L(n):
    ls = []
    out = 0
    for i in range(1,n+1):
        out += sigma(i,0)
        ls.append((i,out/i))
    return ls

line(L(10000)) + plot(log(x),(x,1,10000),
    color='red',linestyle='--')
```

Lookin' good! There does seem to be some predictable error. What might it be?

```
@interact
def _(pts=range_slider(0,5000,50,(0,200))):
    show(point([(a,b-log(a)) for a,b in
        L(pts[1])[pts[0]:]],pointsize=3,rgbcolor=(0,0,0)))
```

Keeping $x = 0$ in view, it seems to be somewhat less than 0.2, although the error clearly bounces around. By zooming in, we see the error bouncing around roughly between 0.15 and 0.16, more or less, as x gets large. So will this give us something more precise?

20.3.2 Getting a handle on error

To answer this, we will try one more geometric trick.

```
@interact
def _(n=(8,range(2,25))):
    viewsize=n+1
    g(x)=1/x
    P=Graphics()
    P += plot(n*g,(x,0,n+1))
    P += plot(2*g,(x,0,n+1),linestyle="--")
    if n>7:
        P += plot((n-5)*g,(x,0,n+1),linestyle="--")
    grid_pts = [[i,j] for i in [1..viewsize] for j in
        [1..viewsize]]
    P += points(grid_pts, rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]*coords[1]<=n)]
    P += points(lattice_pts, rgbcolor =
        (0,0,1),pointsize=20)
    P += plot(x,(x,0,viewsize),
        linestyle="--",rgbcolor=(0,0,0))
    show(P,ymax=viewsize,aspect_ratio=1)
```

Notice we have now divided the lattice points up into three parts, two of which are 'the same':

- The ones on the line $y = x$.
- The lattice points above the line and below the hyperbola.
- The lattice points to the right of the line and below the hyperbola.

Let's count how many there are of each.

First, there are exactly $\lfloor\sqrt{n}\rfloor \leq \sqrt{n}$ points on the line. At each integer y-value d up to $y = \sqrt{n}$, there are are $\lfloor n/d\rfloor - d$ above the line and below the hyperbola. Analogously, at each integer x-value d up to $x = \sqrt{n}$, there are are $\lfloor n/d\rfloor - d$ points to the right of the line and below the hyperbola.

Combining these computations as sums over the divisors d less than n, and noting the floor is less than the number by at most one for each d,

$$\sum_{k=1}^{n}\tau(k) = \sum_{d\leq\sqrt{n}}(\lfloor n/d\rfloor - d) + \sum_{d\leq\sqrt{n}}(\lfloor n/d\rfloor - d) + \lfloor\sqrt{n}\rfloor \leq 2\sum_{d\leq\sqrt{n}}(n/d - d) + \sqrt{n}$$

so the total error gained by this approximation is at most $2\sqrt{n} + 1 = O(\sqrt{n})$.

Next we rewrite this using the formula for the sum of the first ℓ integers:

$$\sum_{k=1}^{n}\tau(k) = 2n\sum_{d\leq\sqrt{n}}\frac{1}{d} - 2\sum_{d\leq\sqrt{n}} d + O(\sqrt{n})$$

$$= 2n\sum_{d\leq\sqrt{n}}\frac{1}{d} - 2\left(\frac{\lfloor\sqrt{n}\rfloor(\lfloor\sqrt{n}\rfloor+1)}{2}\right) + O(\sqrt{n})\,.$$

The difference between $\left(\frac{\lfloor\sqrt{n}\rfloor(\lfloor\sqrt{n}\rfloor+1)}{2}\right)$ and $\frac{n}{2}$ is once again far less than $O(\sqrt{n})$ (and negative to boot), so using some of the work in Exercise Group 20.6.1–20.6.5 finally get that

$$\sum_{k=1}^{n}\tau(k) = 2n\sum_{d\leq\sqrt{n}}\frac{1}{d} - n + O(\sqrt{n}) \Rightarrow \frac{1}{n}\sum_{k=1}^{n}\tau(k) = 2\sum_{d\leq\sqrt{n}}\frac{1}{d} - 1 + O(1/\sqrt{n})\,.$$

20.3.3 The end of the story

We're almost at the end of the story! It's been a while since we explored the long-term average of τ in Subsection 20.2.1; at that point, you likely convinced yourself that $\log(n)$ is close to the average value of τ.

So now we just need to relate the sum $2\sum_{d\leq\sqrt{n}}\frac{1}{d} - 1$ to $\log(n)$. I wish to emphasize just how small the error term $O(1/\sqrt{n})$ is!

```
@interact
def _(n=(8,range(2,25))):
    viewsize=n+1
    P=Graphics()
    P += plot(1/x,(x,1,n))
    P += plot(piecewise([[(j,j+1),1/j] for j in
        [1..n-1]]), (x,1,n), fill=1/x,linestyle='')
    show(P)
```

This graphic shows the *exact* difference between $\sum_{k=1}^{m-1} \frac{1}{k}$ and $\log(m)$. Clearly, even as $m \to \infty$, the total area is simply the sum of a bunch of nearly-triangles with width exactly one and no intersection of height (again this idea), with total height less than 1. So the difference between $\sum_{k=1}^{m-1} \frac{1}{k}$ and $\log(m)$ will be finite as $m \to \infty$.

This number is very important! First of all, it clearly is related to the archetypal *divergent* series from calculus, the **harmonic series**

$$\sum_{k=1}^{\infty} \frac{1}{k}$$

However, this constant has taken on a life of its own.

Definition 20.3.2. The number γ, or the Euler-Mascheroni constant, is defined by

$$\gamma = \lim_{m\to\infty} \left(\sum_{k=1}^{m-1} \frac{1}{k} - \log(m) \right)$$

You have almost certainly never heard of this number, but it is very important. There is even an entire book, by Julian Havil [C.3.14] about this number. It's a pretty good book, in fact!

Remark 20.3.3. Among other crazy properties, it is the derivative of a generalization of the factorial function, called Gamma (Γ). I am not making this up.

Consider the area corresponding to *gamma* compared to its finite approximations. Notice that the "missing" part of the area (since we can't actually view all the way out to infinity) must be less than $1/m$, since it will be the part lower than all the pieces we can see in the graphic for any given m. So γ is within $O(1/n)$ of any given amount *finite* $\sum_{k=1}^{m-1} \frac{1}{k} - \log(m)$.

Now we put it all together! We know from above that

$$\frac{1}{n} \sum_{k=1}^{n} \tau(k) = 2 \sum_{d \le \sqrt{n}} \frac{1}{d} - 1 + O(1/\sqrt{n}) \,.$$

Further, we can now substitute in the following for $\sum_{d\le\sqrt{n}} \frac{1}{d}$;

$$\sum_{d \le \sqrt{n}} \frac{1}{d} = \log(\sqrt{n}) + \gamma + O(1/\sqrt{n}) \,.$$

Once we do that, and take advantage of the log fact $2\log(z) = \log\left(z^2\right)$, we get

$$\frac{1}{n} \sum_{k=1}^{n} \tau(k) = \log(n) + (2\gamma - 1) + O(1/\sqrt{n}) \,.$$

That is exactly the asymptote and type of error that I have depicted below!

```
@interact
def _(pts=range_slider(0,5000,50,(0,200))):
    show(point([(a,b-log(a)) for a,b in
        L(pts[1])[pts[0]:]],pointsize=3,rgbcolor=(0,0,0)) +
        plot(2*euler_gamma-1,(x,0,pts[1])) +
        plot(2*euler_gamma-1+.5/sqrt(x),(x,0,pts[1]),
        color='red',linestyle='--'),
        xmin=pts[0],xmax=pts[1],
        ymax=2*euler_gamma-1+.5/sqrt(pts[0]+1))
```

It's not hard to prove that τ grows at least as fast as $\log(n)$, so this is a fairly sharp result. (It's even possible to show that the error in the average is $O(1/\sqrt[3]{x})$, but is *not* $O(1/\sqrt[4]{x})$.)

20.4 Heuristics for the Sum of Divisors

20.4.1 Numbers instead of points

Could this type of argument conceivably be used for $\sigma = \sigma_1$?

The answer is ***yes***! Consider the following rewrite of the sum of sigmas, which are themselves the sum of divisors:

$$\sum_{n\le x}\sigma(n) = \sum_{n\le x}\sum_{q|n} q = \sum_{q,d \text{ such that } qd\le x} q = \sum_{d\le x}\sum_{q\le \frac{x}{d}} q\,.$$

We have changed from a sum of sums of *divisors* (which might not be consecutive, and makes σ annoying to compute) to a sum of sums *of consecutive integers*.

We can think about this graphically again. Instead of comparing *points* on a hyperbola with *points* in columns or rows, though, we will compare *numbers* at points on a hyperbola with *numbers* at points in rows. We can think of it as summing up a *weighted* set of points. The picture below tells it all.

```
@interact
def _(n=(6,range(2,50))):
    viewsize=n+1
    g(x)=1/x
    P=Graphics()
    P += plot(n*g,(x,0,n+1))
    grid_pts = [[i,j] for i in [1..viewsize] for j in
        [1..viewsize]]
    P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]*coords[1]<=n)]
    for thing in lattice_pts:
        P += text(thing[0],thing,rgbcolor=(0,0,0))
    show(P,ymax=viewsize,aspect_ratio=1)
```

Example 20.4.1. In the first example that shows up in the interact, we see that

$$\sum_{k=1}^{6}\sigma(k) = 1 + (1+2) + (1+3) + (1+2+4) + (1+5) + (1+2+3+6) =$$

$$(1+2+3+4+5+6) + (1+2+3) + (1+2) + 1\,,$$

which means we can think of it as a sum of sums from 1 to the length of each row.

Now let's note three things about the general case.

- Each row is, of course, $\left\lfloor \frac{n}{k} \right\rfloor$ in length, as with τ.
- Adding up the first j integers from one to j is of course

  $$\frac{j(j+1)}{2} = \frac{j^2}{2} + \frac{j}{2}\,,$$

  which we used above.

- The most wrong $\frac{\lfloor x \rfloor (\lfloor x \rfloor + 1)}{2}$ can be from $\frac{x(x+1)}{2}$ is $j + 1 = O(j)$ (this is simple algebra).

So if we combine the information above with the formula, we get

$$\sum_{n \leq x} \sigma(n) = \sum_{d \leq x} \sum_{q \leq \frac{x}{d}} q = \sum_{d \leq x} \left[\frac{1}{2} \left\lfloor \frac{x}{d} \right\rfloor^2 + \frac{1}{2} \left\lfloor \frac{x}{d} \right\rfloor \right] = \sum_{d \leq x} \left[\frac{1}{2} \left(\frac{x}{d} \right)^2 + \frac{1}{2} \left(\frac{x}{d} \right) + O\left(\frac{x}{d} \right) \right] .$$

20.4.2 Order calculations and more

But this is actually possible to analyze! First, some order calculations.

We already saw that $\sum_{d \leq x} \frac{1}{d} = \log(x) + O(1)$, so

$$\sum_{d \leq x} \frac{1}{2} \left(\frac{x}{d} \right) = \frac{1}{2} O(x \log(x)) = O(x \log(x)) .$$

(See 20.6.13.)Also, $\sum_{d \leq x} O\left(\frac{x}{d}\right)$ must be

$$O\left(x \sum_{d \leq x} \frac{1}{d} \right) = O(x \log(x)) .$$

Next, let's get more information about $\sum_{d \leq x} \left[\frac{1}{2} \left(\frac{x}{d} \right)^2 \right]$. Recall that the (convergent) improper integral $\int_x^\infty \frac{dy}{y^2}$ approximates $\sum_{d > x} \frac{1}{d^2}$.

Since both converge, and by the same pictures as above, the error is certainly $O(1/x^2)$. Then I can rewrite things as

$$\sum_{d \leq x} \frac{1}{d^2} = \sum_{d=1}^{\infty} \frac{1}{d^2} - \sum_{d > x} \frac{1}{d^2} = \sum_{d=1}^{\infty} \frac{1}{d^2} - \int_x^\infty \frac{1}{y^2} dy + O(1/x^2) = \sum_{d=1}^{\infty} \left(\frac{1}{d^2} \right) - \frac{1}{x} + O(1/x^2) .$$

Thus the whole crazy double sum can be approximated as follows, quite accurately:

$$\sum_{n \leq x} \sigma(n) = \frac{x^2}{2} \sum_{d \leq x} \left(\frac{1}{d^2} \right) + \frac{x}{2} \sum_{d \leq x} \frac{1}{d} + O(x \log(x))$$

$$= \frac{x^2}{2} \left(\sum_{d=1}^{\infty} \left(\frac{1}{d^2} \right) - \frac{1}{x} + O(1/x^2) \right) + O(x \log(x)) = \frac{x^2}{2} \sum_{d=1}^{\infty} \left(\frac{1}{d^2} \right) - \frac{x}{2} + O(x \log(x)) .$$

And the average value of σ must be this divided by x, namely

$$\frac{1}{x} \sum_{n \leq x} \sigma(n) \text{ is } \frac{x}{2} \sum_{d=1}^{\infty} \frac{1}{d^2} + O(\log(x)) .$$

Since we know that the series converges, this means the average value of σ increases quite linearly, with an error (at most) increasing logarithmically! This might be a shock – that one could actually get something fairly accurate like this relatively easily using calculus ideas like improper integrals and (implicitly) the integral test for infinite series. But check out the data!

```
def M(n):
    ls = []
    out = 0
```

```
    for i in range(1,n+1):
        out += sigma(i)
        ls.append((i,out/i))
    return ls

@interact
def _(j = [10,100,1000,10000]):
    show(line(M(j)))
```

Of course, one might ask what the slope of this line is! It would have to be $m = \frac{1}{2}\sum_{k=1}^{\infty} \frac{1}{d^2}$. Have you seen this constant before? (In a calculus class, you should have proved that it does converge.)

```
@interact
def _(j = [10,100,1000,10000]):
    show(line(M(j)) + plot(x*zeta(2)/2,(x,0,j),
        color='black',linestyle="--"))
```

Finding a summation of this was the so-called Basel problem, which Euler solved and showed is $\frac{\pi^2}{6}$. So the slope is $\frac{\pi^2}{12}$. Amazing! (See also Section 24.4.)

20.5 Looking Ahead

Let's recap.

- The average value of $\tau(n)$ was $\log(n) + 2\gamma - 1$.
- The average value of $\sigma(n)$ was $\left(\frac{1}{2}\sum_{d=1}^{\infty} \frac{1}{d^2}\right)\ n$.
    - Because of Euler's amazing solution to the Basel problem, we know that
    $$\sum_{d=1}^{\infty} \frac{1}{d^2} = \frac{\pi^2}{6}$$
    so the constant in question is $\frac{\pi^2}{12}$.

We end with the question of yet another average value. What might happen with the ϕ function? You can try out various ideas below; `a` is the coefficient and `n` is the power of a model ax^n.

```
def L(n):
    ls = []
    out = 0
    for i in range(1,n+1):
        out += euler_phi(i)
        ls.append((i,out/i))
    return ls

LS = L(1000)
P = line(LS)
@interact
def _(a=.01,n=2,view=(50,[25,50,..500])):
    show(P+plot(a*x^n,0,view,
        color='black',linestyle="--"), xmin=1,xmax=view,
        ymax=LS[view][1])
```

```
    pretty_print(html("Blue is the average value of
        $\phi$"))
    pretty_print(html("Red is $%s x^{%s}$"%(latex(a),n)))
```

Hopefully you started finding something interesting. However, we aren't ready to prove anything about that quite yet.

20.6 Exercises

We start with some exercises testing understanding of Landau notation.

1. Show that $\sigma(n)$ is $O(n^2)$ (compare to the sum of all integers).
2. Use the formula for the sum of the first n perfect squares (often encountered in a Transition to Proof course or when first doing definite integrals in Calculus) and the previous exercise to show that the average value of $\sigma(n)$ is Big Oh of n^2. (This can be loosey-goosey.)
3. Show that if g and h are both $O(f)$ for some f, then $g+h$ is also $O(f)$.
4. Show that if g is $O(f)$ for some f, then if $c > 0$ we have that g is $O(cf)$ and cg is $O(f)$.
5. Show that if g is $O(f)$ for some f and if $f(x) \leq h(x)$ for x large enough, then g is also $O(h)$.

6. Find a formula for the average value of the u and N functions (up through n), where $u(n) = 1$ for all n and $N(n) = n$ for all n (recall Definition 19.2.9).

7. Finish off all calculus details in the argument in 20.2.3.

8. Finish the details of the proof that τ is $O(\sqrt[3]{x})$

9. Show that $\tau(n)$ is *not* $O(1)$. (Hint: that means there is no constant C such that $\tau(n) \leq C$ always.)

10. Why would it not contradict our theorem above that $\frac{1}{n}\sum_{k=1}^{n} \tau(k) = O(\log(n))$ to say that $\tau(n)$ is not $O(\log(n))$?

11. Show that $\tau(n)$ is not $O(\log(n))$. (Hint: look at numbers of the form 6^k, and compare τ of these to any given multiple of the natural logarithm using calculus.)

12. Finish all calculus details of the proof of σ's average size in 20.4.

13. Finish the details of the first computation of Big Oh in 20.4.2.

14. Find absolute bounds for $\phi(n)$ (simple polynomial or log formulas in terms of n).

15. Use data, graphs, whatever to conjecture what type of growth the average value of ϕ has up to n. Is it logarithmic, linear, quadratic, exponential, something else? Bonus if you find a coefficient for the growth!

Chapter 21

The Prime Counting Function

Up to now, our examples of arithmetic functions $f(n)$ have been clearly based on some property of the number n itself, such as its divisors, the numbers coprime to it, and so forth.

However, there is one function of prime importance which, as far as we yet know, bears no particular obvious relation to the input – yet *in the aggregate* bears amazing relations to the input! It is the most mysterious of all these functions.

Definition 21.0.1. The **prime counting function** $\pi(x)$ is defined, for all positive numbers x, as the number of primes less than or equal to xk denoted

$$\pi(x) = \#\{p \leq x \mid p \text{ is prime } \}.$$

21.1 First Steps

It might seem at first there is very little we can say about this function; after all, thus far we've seen no particular pattern in the primes themselves (other than that they are nearly all odd). You may wish to see what the function looks like to confirm this sense. It is a not particularly smoothly increasing function with no upper bound (recall Theorem 6.2.1).

```
plot(prime_pi,2,100,color='black',legend_label="$\pi(x)$")
```

Sage note 21.1.1 (Syntax for counting primes)**.** The syntax for this function is `prime_pi(n)`.

21.1.1 A funky formula

Given the skepticism of the paragraphs so far this chapter, you may be surprised to learn there are *exact* formulas for this function, as well as for the nth prime. The following formula (for $n > 3$) is one of my favorites (see the Appendix of the exhaustive Hardy and Wright, [C.1.2], and also Exercise 21.5.1):

$$\pi(n) = -1 + \sum_{j=3}^{n} \left((j-2)! - j \left\lfloor \frac{(j-2)!}{j} \right\rfloor \right).$$

Can you see why this is not useful in practice? So there is plenty left for us to discuss.

On the other hand, it works! We can confirm this by using the following code.

```
def primeish(n):
    if n==1:
        return 0
    elif n==2:
        return 1
    elif n==3:
        return 2
    else:
        result = -1
        fact = 1
        for j in range(3,n+1):
            fact = fact*(j-2)
            result += (fact - j*floor(fact/j))
        return result

import math
def plotprimeish(n):
    n = int(math.floor(n))
    return primeish(n)

pretty_print(html("The␣number␣of␣primes␣up␣to␣20000␣this␣
    formula␣gives␣is␣$%s$"%primeish(20000)))
pretty_print(html("The␣real␣function␣in␣Sage␣gives␣
    $%s$"%prime_pi(20000)))
pretty_print(html("And␣let's␣compare␣plots:"))
plot(lambda x:plotprimeish(x), (x,2,100)) +
    plot(prime_pi,2,100,color='black')
```

Sage note 21.1.2 (Cython)**.** It's possible to significantly speed up many such computations by converting to Cython, a way to take Python/Sage and turn it into the much-faster compiled language C. For a project, try to speed this function up using Cython!

Sage note 21.1.3 (Not all algorithms are equal)**.** Don't forget that just because an algorithm works, doesn't guarantee it will be useful in practice! However, it's often useful to get something correct first, and only then try to optimize.

21.1.2 A very low bound

On a more computationally feasible note, one can find a very rudimentary (lower) bound on this function. Recall that unadorned logarithms are the natural log.

Fact 21.1.4. *There are at least*

$$\frac{\log(\log(x)/\log(2))}{\log(2)} + 1 = \log_2(\log_2(x)) + 1$$

primes less than or equal to x.

Proof. In Saidak's proof of the infinitude of the primes, he constructs the sequence

$$2,\ (2+1),\ (2(2+1)+1),\ (2(2+1)(2(2+1)+1))+1\ldots$$

Then he shows, similarly to Euclid's proof, that there is at least one *new* prime divisor in each element of the sequence (even if not necessarily a larger one). So the nth prime can be no bigger than the nth element of this sequence.

By induction, we see that this element is less than or equal to $2^{2^{n-1}}$.

- The case $n = 1$ is clear.
- The nth element is the previous elements multiplied together, plus 1, which is less than

  $$2^{2^0}2^{2^1}\cdots 2^{2^{n-2}}+1 = 2^{1+2+4+\cdots+2^{n-2}}+1 = 2^{2^{n-1}-1}+1 \le 2^{2^{n-1}}$$

  (this uses the same type of technique as in Subsection 4.5.2).

So the number of primes less than $2^{2^{n-1}}$ can't be less than n. Take two logs of this to get

$$\log(\log(2^{2^{n-1}})) = \log(2^{n-1}\log(2)) = (n-1)\log(2)+\log(\log(2))$$

This yields the given statement. □

As you can see below, this is not a very useful bound, considering there are actually 25 primes less than 100, not 3!

```
plot(log(log(x)/log(2))/log(2)+1,(x,2,100)) +
    plot(prime_pi,2,100,color='black')
```

21.1.3 Knowledge from nowhere

Finally, although it may not seem evident, you should know that it is not necessary to actually find all the first n primes (even of a particular type) to compute how many there are, at least not always.

Definition 21.1.5. Let $\phi(n,a)$ to be the number of positive integers less than n which are not divisible by any of the first a primes

Now it is possible to develop the recursive formula

$$\phi(n,a) = \phi(n,a-1) - \phi\left(\left\lfloor\frac{n}{p_a}\right\rfloor, a-1\right),$$

which allows use a type of inductive argument to compute $\phi(n,a)$ without having to use many computational resources.

It is then not too hard to use a counting argument to prove that

$$\pi(n) = \pi(\sqrt{n}) + \phi(n,\pi(\sqrt{n})) - 1$$

This is the typical way to count π without actually counting primes, and with some speedups it can be quite efficient.

Interestingly, this is also how one finds the nth prime. You use an approximation to the nth prime like $n\log(n)$ and then check values of $\pi(n)$ near that point to see where the value changes, which should lead you exactly to the prime you seek. (Recall Sage note 4.2.1 about `%time` when using the following cell.)

```
%time nth_prime(10^7)
```

21.2 Some History

Somewhat remarkably, given how long humans have been studying primes, the first people we know of compiling substantial data about them are Gauss and Legendre, around 1800.

Legendre first tried to estimate $\pi(x)$. He said that $\pi(x) \approx \frac{x}{\log(x)-A}$, where he fudges the constant $A \approx 1.08366$. More precisely, he claimed that $\pi(x)$ is *asymptotic* to this function.

Definition 21.2.1. We say that two functions $f(x)$ and $g(x)$ are **asymptotic** to each other when

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1$$

Essentially, in the long run these functions get as close to each other as you like, on a percentage basis.

Here is another way to think about this. Think of the average chance of a number of size x being prime; Legendre guessed this was of the form $\frac{1}{\log(x)-A}$. This general notion was based on a lot of data he had collected, and the constant A he finally settled on seemed to give the best match to the data.

Not long after this, Gauss came up with a solution that was more elegant – and despite not being 'fitted' to the data in the same way, was correct. *And he didn't tell anyone for over fifty years!* Gauss' conjecture was that

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1$$

Or, using our new term, $\pi(x)$ is asymptotic to $\frac{x}{\log(x)}$.

21.2.1 The first really accurate estimate and errors

In fact, Gauss makes this estimate even more precise. Here is the general idea.

First, reinterpret the proportion as suggesting that $1/\log(x)$ integers near x are prime. If we do that, then we can think of $1/\log(x)$ as a **probability density function**. What do we do with such functions? We *integrate* the function to get the cumulative amount!

That is, we should expect that $\pi(x) \approx \int_2^x \frac{dt}{\log(t)}$ or equivalently

$$\lim_{x\to\infty} \frac{\pi(x)}{\int_2^x \frac{dt}{\log(t)}} = 1 \ .$$

Definition 21.2.2. We give the name **logarithmic integral**[1] to the (convergent) integral $Li(x) = \int_2^x \frac{dt}{\log(t)}$.

That a function as rigid as π would be close to an integral function should sound like it has a 100% probability of being crazy! But Gauss was no fool, and the accuracy is astounding.

```
@interact
def _(n=100):
```

[1] There is also a definition for this integral $\int_0^x \frac{dt}{\log(t)}$, which has a properly defined value (beyond the level of this course) despite the integrand going to negative infinity. The form used for the prime counting function is traditionally this one, for reasons clear in the rest of this text, and there are no divergence issues at stake.

```
    show(plot(prime_pi,3,n,color='black',
        legend_label='$\pi(x)$') +
        plot(x/log(x),3,n,color='red',
        legend_label='$x/\log(x)$') + plot(Li,3,n,
        color='green', legend_label='$Li(x)$'))
```

Notice how much closer $Li(x)$ is to the actual value of $\pi(x)$ than the $x/\log(x)$ estimate. It's usually closer by several orders of magnitude.

```
@interact
def _(n=[100,1000,1000000,1000000000]):
    P = prime_pi(n)
    pretty_print(html("$\pi(%s)=%s$"%(n,prime_pi(n))))
    pretty_print(html("The␣error␣with␣$%s/\log(%s)$␣is␣
        $\\approx␣%s$"%(n,n,P-(n/log(n)).n())))
    pretty_print(html("The␣error␣with␣$Li(%s)$␣is␣
        $\\approx␣%s$"%(n,(P-Li(n)).n())))
```

21.2.2 Exploring Li

Can we try for some more analysis? Since we saw that $x/\log(x)$ didn't seem to be as good an approximation, we'll leave it out for now. This graphic follows one along a roughly 1000-wide stretch at a time.

```
@interact
def _(n=100):
   P = plot(prime_pi,3,n,
       color='black',legend_label='$\pi(x)$')
   P += plot(Li,3,n, color='green',legend_label='$Li(x)$')
   show(P, xmin=max(n-1000,0),
       ymin=prime_pi(max(n-1000,0)))
```

Based on this evidence, it seems clear that $Li(x)$, even if it's a good approximation, should not ever be less than the actual count of primes. And yet, the English mathematician Littlewood proved the following result.

Fact 21.2.3. *For any number x, there is an $x' > x$ such that*

$$Li(x') < \pi(x') \ .$$

As remarkable as this seems, his student Skewes proved the following even more amazing fact.

Fact 21.2.4. *The first time this happens is no higher than*

$$10^{10^{10^{10^{1000}}}} \ .$$

In the original paper, this bound had a 34 instead of 1000 in the last exponent, but that result relied upon a special assumption (the so-called **Riemann Hypothesis**, see Chapter 25).

Today we know that the first time this "switch" happens is no higher than 1.4×10^{316}. There is still no explicit number known for which this is true, however, and we haven't even gotten remotely near those bounds with computers.

This sounds terrible, but actually is *good* news. After all, if π beats Li once in a while, then Li must be a great approximation indeed! So, just how great is it?

21.3 The Prime Number Theorem

It turns out $Li(x)$ is a pretty good approximation indeed.

21.3.1 Stating the theorem

Theorem 21.3.1 (Prime Number Theorem)**.** *If $\pi(x)$ is the number of primes $p \leq x$, then*

$$\lim_{x\to\infty} \frac{\pi(x)}{Li(x)} = 1\,.$$

In fact, the first bound also has this property (see Exercise 21.5.4):

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1\,.$$

This result, conjectured by Riemann, was proved about 100 years after the initial investigations of Gauss by the French and Belgian mathematicians Jacques Hadamard and Charles-Jean de la Vallée-Poussin. They made good use of the analytic methods we are slowly approaching.

Any proof is this is well beyond the bounds of this text. One of several modern versions is in the analytic number theory text [C.3.6] by Apostol; see also [C.1.9]. Additionally, as a series of exercises (!) in that book, one can also explore a proof due to Selberg and Erdős that is "elementary", in the sense of not using complex-valued integrals. There is a well-known exposition of a very similar proof in [C.1.2], and another in [C.3.4].

Later, we'll see that many better approximations to $\pi(x)$ exist which come out of this sort of thinking. Notice how the approximations in the next cell take the logarithmic integral and subtract various correction factors in the attempt to get closer.

```
@interact
def _(n=100):
    P = plot(prime_pi,3,n,
        color='black',legend_label='$\pi(x)$')
    P += plot(Li,3,n, color='green',legend_label='$Li(x)$')
    P += plot(lambda x: Li(x) - sqrt(prime_pi(x)),3,n,
        color='orange', legend_label='$Li(x)-\sqrt{\pi(x)}$')
    P += plot(lambda x: Li(x) - .5*Li(sqrt(x)),3,n,
        color='red',
        legend_label='$Li(x)-\\frac{1}{2}Li(\sqrt{x})$')
    P += plot(lambda x: Li(x) - sqrt(x)/log(x),3,n,
        color='purple',
        legend_label='$Li(x)-\sqrt{x}/\log(x)$')
    show(P, xmin=max(n-1000,0),
        ymin=prime_pi(max(n-1000,0)))
```

21.3.2 Chebyshev's contributions

Although we cannot explore the theorem itself in depth, we can understand some of the steps one must take on the way there. It is a good place to highlight the number-theoretic contributions of the great Russian mathematician Chebyshev ( ), who made fundamental advances in this type of number theory as well as in statistics.

He was the first person to prove a conjecture known (even today!) as Bertrand's Postulate, after the French mathematician who first proposed it.

Theorem 21.3.2 (Bertrand's Postulate)**.** *For any integer $n \geq 2$, there is a prime between n and $2n$.*

Proof. It is actually quite possible to prove this at the level we have reached, but any proof is long enough to take us a little far afield. □

Try testing it yourself below!

```
@interact
def _(n=25):
    pretty_print(html("$%s$ is a prime between $%s$ and
        $%s$"%(next_prime(n),n,2*n)))
```

On a related note, although this proves you can't have too long of stretches without prime numbers, you can certainly have arbitrary stretches of composite numbers. Paul Nahin, in [C.6.13], describes the following cute result of Louis A. Graham.

Fact 21.3.3. *Multiply all the primes p from 2 to $n+1$ to get $N = \prod_{2 \leq p \leq n+1} p$. Then we have n consecutive composite integers from $N-(n+1)$ to $N-2$.*

Proof. We know that N is a multiple of a prime factor[1] of each number x from 2 to $n+1$. For each such x and prime factor p_x, Proposition 1.2.6 guarantees that $N - x$ is also a multiple of p_x. □

Try testing it yourself below!

```
@interact
def _(n=5):
    N = prod(prime_range(n+2))
    pretty_print(html("The numbers between $%s$ and $%s$
        are all composite"%(N-(n+1),N-2)))
    L = [N-(n+1)..N-2]
    print [N-(n+1)..N-2]
    pretty_print(html("have factors"))
    print [l.divisors()[1] for l in L]
    pretty_print(html("and there are $%s$ of
        them"%(len(L))))
```

More immediately germane to our task of looking at $\pi(x)$ and its value, Chebyshev proved the first substantial result on the way to the Prime Number Theorem, validating Legendre's intuition.

Theorem 21.3.4 (Big Oh of Prime Pi)**.** *It is true both that:*

- $\pi(x)$ *is* $O\left(\frac{x}{\log(x)}\right)$ and
- $\frac{x}{\log(x)}$ *is* $O(\pi(x))$.

Interestingly, this is *not* the same as the Prime Number Theorem; see Exercise 21.5.6.

What we will show here is the gist of a smaller piece of this theorem.

Proposition 21.3.5. *For big enough x, $\pi(x) < 2\frac{x}{\log(x)}$.*

[1] In fact, all such factors.

Proof. We follow Stopple's presentation in Section 5.2 of [C.3.5] closely in sketching out most of a proof of this below; see also [C.1.11]. It is a little longer than some of our other proofs. It uses some very basic combinatorial ideas and calculus facts, however, so it is a great example of several parts of mathematics coming together.

First, it's not hard to verify this for $x < 1000$.

```
plot(prime_pi,1,1000)+plot(2*x/log(x),1,1000,color='black')
```

Now we'll proceed by induction, in an unusual way. We'll assume it is true for n, and prove it is true for $2n$. This needs a little massaging for odd numbers, but *is* a legitimate induction method.

With this in mind, we first assume that $\pi(n) < 2\frac{n}{\log(n)}$. Now what?

Below, in Lemma 21.3.6 we look at the *product* of all the primes (if any) between n and $2n$, which we write as

$$P = \prod_{n<p<2n} p\,.$$

In that result some combinatorial thinking leads to the following estimate:

$$n^{\pi(2n)-\pi(n)} < P \leq \frac{(2n)!}{n!n!} < 2^{2n}$$

These bounds show that P is between a certain power of n and a certain power of 2.

Now we will manipulate this to get the final result. Begin by taking log of both ends to get

$$(\pi(2n) - \pi(n))\log(n) < 2n\log(2)$$

Now divide out and isolate to get

$$\pi(2n) < \frac{2n\log(2)}{\log(n)} + \pi(n) < \frac{2n\log(2)}{\log(n)} + 2\frac{n}{\log(n)} = (\log(2)+1)\frac{2n}{\log(n)}\,.$$

In Exercise 21.5.8 you will show that, as long as $n > 1000$, we have the inequality

$$\frac{\log(2)+1}{\log(n)} < \frac{2}{\log(2)+\log(n)} = \frac{2}{\log(2n)}$$

Now we can put it all together to see that

$$\pi(2n) < (\log(2)+1)\frac{2n}{\log(n)} < 2\frac{2n}{\log(2n)}\,,$$

which is exactly what the proposition would predict.

To rescue this for $2n+1$, we need another calculus comparison. First, from above we have

$$\pi(2n+1) \leq \pi(2n) + 1 < \frac{2n\log(2)}{\log(n)} + \pi(n) + 1$$

$$< \frac{2n\log(2)}{\log(n)} + 2\frac{n}{\log(n)} + 1$$

Since $\frac{2n+1}{\log(2n+1)} > \frac{2n}{\log(2n+1)}$, it will suffice then to show

$$(2+2\log(2))\frac{n}{\log(n)} + 1 < \frac{2n}{\log(2n+1)}\,.$$

Since $n > 1000$,

$$(2 + 2\log(2))\frac{n}{\log(n)} + 1 < 3.386\frac{n}{\log(n)} + 1 < 3.394\frac{n}{\log(n)}$$

so it suffices to show

$$3.394\frac{n}{\log(n)} < \frac{2n}{\log(2n+1)} .$$

Showing this is Exercise 21.5.9. □

Lemma 21.3.6. *Let the product of all the primes between n and $2n$ be written*

$$P = \prod_{n<p<2n} p$$

Then we can bound it as

$$n^{\pi(2n)-\pi(n)} < P \leq \frac{(2n)!}{n!n!} < 2^{2n}$$

Proof. Think of all the primes in question. On the one hand, each of these primes p is greater than n, and there are $\pi(2n) - \pi(n)$ of them. So

$$n^{\pi(2n)-\pi(n)} < P .$$

On the other hand, each of these primes is greater than n *but* they are all in the list of numbers from n to $2n$, so their product divides

$$\frac{(2n)\cdot(2n-1)\cdot(2n-2)\cdots(n+1)}{n\cdot(n-1)\cdot(n-2)\cdots 1}$$

That is to say P is a factor of a binomial coefficient

$$P \mid \frac{(2n)\cdot(2n-1)\cdot(2n-2)\cdots(n+1)}{n\cdot(n-1)\cdot(n-2)\cdots 1} = \frac{(2n)!}{n!n!}$$

and in particular,

$$P \leq \frac{(2n)!}{n!n!}$$

Now here is the conceptual key of the proof. We reinterpret this factorial fraction as the number of ways to choose n things from a collection of $2n$ things! And the number of ways to choose n things is certainly less than the number of ways to pick *any* old collection out of $2n$ things, which is 2^{2n} (because you either pick it or you don't).

Since we showed both bounds, this concludes the proof. □

21.4 A Slice of the Prime Number Theorem

We end this chapter with a substantial piece of a real proof in the direction of the Prime Number Theorem, courtesy of a function also first introduced by Chebyshev. The argument is dense, but requires nothing beyond calculus and a willingness to allow a lot of algebraic and integral manipulation for the purposes of estimation.

21.4.1 Functions to know

First, we'll review the main function. Think of the prime counting function π as a so-called **step function**, where every time you hit a new prime you add 1.

```
@interact
def _(n=100):
    show(plot(prime_pi,1,n))
```

Let's define a new function. Instead of adding 1 each time x hits a prime, we will add $\log(p)$ (recall that this is the natural logarithm) each time we hit a prime p. Of course, this value we add will get bigger as p gets bigger.

```
def theta(x): return sum(math.log(p) for p in
    prime_range(1,floor(x)+1))
@interact
def _(n=100):
    show(plot(theta,1,n))
```

Definition 21.4.1. We call the function given by this formula Chebyshev's theta function:

$$\Theta(x) = \sum_{p \leq x} \log(p)\,.$$

Earlier in this chapter we noted that the Prime Number Theorem is logically equivalent to the limit $\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1$. There are actually many such logical equivalences. One of them involves Θ:

$$\lim_{x\to\infty} \frac{\Theta(x)}{x} = 1$$

This is certainly numerically plausible. Here is a plot of both limits, along with the constant function 1.

```
def theta(x): return sum(math.log(p) for p in
    prime_range(1,floor(x)+1))
def pnt(n): return prime_pi(n)*log(n)/n
def thox(n): return theta(n)/n
@interact
def _(end=100000):
    show(plot(1,(1,end),color='black') +
        plot(pnt,(1,end),color='red',legend_label='Prime␣
        Number␣Theorem') +
        plot(thox,(1,end),legend_label='Chebyshev␣Theta'))
```

As usual, proving such things completely is beyond the level of this course, but we *can* prove the following partial implication.

Proposition 21.4.2. *If the Prime Number Theorem is true, then it is also true that* $\Theta(x)/x$ *approaches* 1.

Proof. The rest of this section is the proof. □

21.4.2 Getting a formula with sleights of hand

In order to prove this implication, we will first need a formula telling us more about $\Theta(x)$. Our strategy will be to first turn $\Theta(x)$ into an even more hopelessly complicated sum, but then use calculus to trickily get something usable by summing up *integrals*.

In order to do this, we need two subsidiary functions. First recall the notation $m = \lfloor x \rfloor$ for the greatest integer less than x. Secondly:

Definition 21.4.3. We let $a(n)$ be the prime number indicator function defined by

$$a(n) = \begin{cases} 1 & \text{if } n \text{ is prime} \\ 0 & \text{otherwise} \end{cases}$$

. Another way to say this is

$$a(n) = \pi(n) - \pi(n-1)\,.$$

```
@interact
def _(end=10):
    show(plot(prime_pi,1,end,color='black')+plot(lambda
        x:prime_pi(x)-prime_pi(x-1),1,end))
```

Then we can rewrite these step functions as weighted sums of $a(n)$:

$$\pi(x) = \sum_{n=1}^{m} a(n) \text{ and } \Theta(x) = \sum_{n=1}^{m} a(n)\log(n)\,.$$

Our goal is to rearrange Θ to be a sum of something involved π. First we turn it into a difference of sums by rearranging (and using $\log(1) = 0$):

$$\Theta(x) = \sum_{1 \le n \le x} a(n)\log(n) = \sum_{n=1}^{m} a(n)\log(n) =$$

$$\sum_{n=2}^{m} [\pi(n) - \pi(n-1)]\log(n) = \sum_{n=2}^{m} \pi(n)\log(n) - \sum_{n=1}^{m-1} \pi(n)\log(n+1)$$

This difference of sums can be combined into a single sum, with just two left over terms, the second of which is equal to 0.

$$\Theta(x) = \sum_{n=2}^{m-1} \pi(n)[\log(n) - \log(n+1)] + \pi(m)\log(m) - \pi(1)\log(1)\,.$$

To continue, we will rewrite this as an integral. We use a few key facts:

- The difference which appears in the last Θ formula is an integral, $\log(n+1) - \log(n) = \int_n^{n+1} \frac{dt}{t}$.
- We have that $\pi(x) = \pi(m)$ is constant on $[m, x]$, so it may be factored out of any integral of a unit distance.
- We can rearrange and add sums and integrals as usual.

This yields the following rewrite.

$$\Theta(x) = -\sum_{n=2}^{m-1}\Big[\pi(n)\int_n^{n+1}\frac{dt}{t}\Big] + \pi(m)\log(m)$$

$$= -\sum_{n=2}^{m-1}\Big[\pi(n)\int_n^{n+1}\frac{dt}{t}\Big] + \pi(m)\log(m) - \pi(x)\log(x) + \pi(x)\log(x)$$

$$= -\int_2^m \frac{\pi(t)dt}{t} + \pi(x)\log(x) - \int_m^x \frac{\pi(t)dt}{t} = \pi(x)\log(x) - \int_2^x \frac{\pi(t)dt}{t}\,.$$

Now we have a formula for Θ which will allow us to prove something.

21.4.3 Finish the proof

We can divide the formula $\Theta(x) = \pi(x)\log(x) - \int_2^x \frac{\pi(t)dt}{t}$ by x:

$$\frac{\Theta(x)}{x} = \frac{\pi(x)\log(x)}{x} - \frac{\int_2^x \frac{\pi(t)}{t}dt}{x}\,,$$

Given that the Prime Number Theorem says that $\lim_{x\to\infty}$ of the fraction with $\pi(x)$ in it is 1, proving that $\lim_{x\to\infty}$ of $\frac{\Theta(x)}{x}$ is also 1 is equivalent to proving

$$\lim_{x\to\infty}\frac{1}{x}\int_2^x \frac{\pi(t)}{t}dt = 0\,.$$

Now, the Prime Number Theorem also implies that $\frac{\pi(t)}{t}$ and $\frac{1}{\log(t)}$ are asymptotic, so that their integrals also are,

$$\frac{1}{x}\int_2^x \frac{\pi(t)}{t}dt \text{ and } \frac{1}{x}\int_2^x \frac{dt}{\log(t)}\,.$$

This reduces our proof to showing that the average value of $1/\log(t)$ tends to zero. Since integral have a graphical interpretation, we now use the following graph of the integral limit to finish the proof!

Consider that one possible upper sum for the integral of $1/\log(t)$ between 2 and 9 is the area of the two rectangles shown below, one with area $\frac{1}{\log(2)}(\sqrt{9}-2)$ and the other with area $\frac{1}{\log(\sqrt{9})}(9-\sqrt{9})$. (Of course $\sqrt{9}=3$ but this form is more useful here.)

```
@interact
def _(top=(16,[n^2 for n in [2..10]])):
    f(x)=1/log(x)
    P=plot(f,1,top+1)
    P += line([(2,0),(2,f(2)),
        (math.sqrt(top),f(2)),(math.sqrt(top),0)],
        rgbcolor='black')
    P += line([(math.sqrt(top), f(math.sqrt(top))),
        (top,f(math.sqrt(top))),(top,0)], rgbcolor='black')
    P.show(ymax=2)
```

In general, the same argument should hold, so a possible *over*estimate of $\int_2^x dt/\log(t)$ is

$$\frac{1}{\log(2)}(\sqrt{x}-2) + \frac{1}{\log(\sqrt{x})}(x-\sqrt{x})$$

and we want the limit as $x \to \infty$ of $\frac{1}{x}$ times that quantity.

Now is the time to recklessly use logarithmic identities:

$$\frac{1}{x}\left(\frac{1}{\log(2)}(\sqrt{x}-2)+\frac{1}{\log(\sqrt{x})}(x-\sqrt{x})\right) = \frac{1}{\log(2)x^{1/2}}-\frac{2}{x\log(2)}+\frac{1}{\log(\sqrt{x})}-\frac{1}{\log(\sqrt{x})x^{1/2}}$$

$$= \frac{1}{\log(2)x^{1/2}} - \frac{2}{x\log(2)} + \frac{2}{\log(x)} - \frac{2}{\log(x)x^{1/2}}$$

This last expression has positive powers of x and their logs in the denominators, so it pretty clearly goes to zero as $x \to \infty$.

If the algebra doesn't convince you, perhaps the graphs will. Below, black is the overestimate to the integral and red is $1/x$ times the integral.

```
@interact
def _(top=(16,[n^2 for n in [2..10]])):
    f(x)=1/log(x)
    P=plot(f,1,top+1)
    P += line([(2,0),(2,f(2)),(math.sqrt(top),f(2)),
        (math.sqrt(top),0)],
rgbcolor='black')
    P += line([(math.sqrt(top),f(math.sqrt(top))),
        (top,f(math.sqrt(top))),(top,0)], rgbcolor='black')
    P +=
        line([(2,0),(2,f(2)),(2+(math.sqrt(top)-2)/top,f(2)),
        (2+(math.sqrt(top)-2)/top,0)], rgbcolor='red')
    P += line([(math.sqrt(top),f(math.sqrt(top))),
        (math.sqrt(top)+(top-math.sqrt(top))/top,
        f(math.sqrt(top))), (math.sqrt(top) +
        (top-math.sqrt(top))/top,0)], rgbcolor='red')
    P.show(ymax=2)
```

The picture confirms our analytic proof that the limit of $\frac{\theta(x)}{x}$ is the same as that of $\frac{\pi(x)}{x/\log(x)}$, which is what we desired!

21.5 Exercises

1. Consider Wilson's Theorem and consider what will happen to $(j-2)!$ modulo primes and composites (this is Exercise 7.7.6). Use this to prove the bizarre formula in Section 21.1.

2. Come up with two functions $f(x)$ and $g(x)$ that both go to infinity as $x \to \infty$, such that $f(x)$ is always ahead of $g(x)$, but f and g are asymptotic (to each other).

3. Come up with two functions $f(x)$ and $g(x)$ that both go to infinity as $x \to \infty$, but that switch the lead infinitely often and f and g are asymptotic.

4. Show that the two limits in the Prime Number Theorem are really equivalent. That is, show that if $\lim \pi(x)/Li(x) = 1$, then the other limit is 1, and vice versa.

5. Find an arbitrarily long sequence of consecutive composite numbers using factorials.

6. Come up with two functions $f(x)$ and $g(x)$ such that $f(x)$ is $O(g(x))$ and $g(x)$ is $O(f(x))$, but are *not* asymptotic.

7. Use Proposition 21.3.5 to show that $\lim_{x\to\infty} \pi(x)/x = 0$.

8. Show that if $n > 1000$ then

$$\frac{\log(2)+1}{\log(n)} < \frac{2}{\log(2)+\log(n)} = \frac{2}{\log(2n)}$$

To do this, you should compare $2\log(n)$ and $\log(2)(\log(2)+\log(n))$ and *their derivatives* for $n = 1000$ and up, then divide the two expressions appropriately. You will need to show that if $f(x_0) > g(x_0)$ and $f' > g'$ for $x \geq x_0$, then $f > g$ as well.

9. Verify that $3.394\frac{n}{\log(n)} < \frac{2n}{\log(2n+1)}$ for $n > 1000$. You will need to verify that the derivative of $\frac{\log(n)}{\log(2n+1)}$ is positive there.

Chapter 22

More on Prime Numbers

This chapter serves two purposes. First, there are all kinds of interesting facts ahout prime numbers, and this chapter collates some of the ones I personally find amazing. What are your favorites?

Secondly, exploring the wonderful world of primes will start us heading back toward other arithmetic functions, especially toward developing the language we'll need to explore $\pi(x)$ more rigorously.

There are lots of resources beyond this for exploring primes! One interesting resource is Numberphile's YouTube channel for prime videos. Paulo Ribenboim has several well-known books about them, such as *The Little Book of Bigger Primes* [C.3.16].

But for usability, I have to mention one of the best web sites about primes. This is the Prime Pages, hosted at the University of Tennessee, Martin. It's just amazingly full of useful information, but also quite user-friendly and usable for a large variety of backgrounds. In particular, the top twenty page has links to the top twenty of just about every prime type you can imagine, a cornucopia of information. My personal favorite is the prediction of when the first billion digit prime will surface.

22.1 Prime Races

One of Chebyshev's more interesting observations was that our familiar categories of primes – the classes $4k+1$ and $4k+3$ – don't always seem to have the 'same size'. Before moving on, try solving the next question by hand.

Question 22.1.1. How many primes of each type there are up to $n = 10$, $n = 20$, and $n = 50$? Try making a table.

We can, as always, use computational power to try to see more.

```
@interact
def _(n=7):
    L = map(None,[p for p in prime_range(n+1) if
        p%4==1],[p for p in prime_range(n+1) if p%4==3])
    L = [['',l[1]] if l[0] is None else l for l in L]
    T = [['$p\equiv␣1\\text{␣(mod␣}4)$','$p\equiv␣3\\text{␣
        (mod␣}4)$']]
    pretty_print(html(table(T+L,header_row=True,
        frame=True)))
```

```
@interact
def _(k=100):
    p1 = 0
    p3 = 0
    for i in prime_range(k):
        if i%4==1:
            p1 += 1
        if i%4==3:
            p3 += 1
    pretty_print(html("Up to $k=%s$, there are"%k))
    pretty_print(html("%s primes $p\equiv 1\\text{ (mod
        }4)$ and "%p1))
    pretty_print(html("%s primes $p\equiv 3\\text{ (mod
        }4)$."%p3))
```

Question 22.1.2. Do you detect the bias Chebyshev did? Do you think it will persist?

22.1.1 Infinitude of types of primes

Of course, for this question to make sense, we need to make sure this 'prime race' won't suddenly run out of gas. We know there are infinitely many primes, but what about each *type* of prime?

Fact 22.1.3. *There are infinitely many primes congruent to 3 modulo 4* and *there are infinitely many primes congruent to 1 modulo 4.*

Proof. See the following two Propositions 22.1.4 and 22.1.5. □

It turns out that proving the first part of the proposition is nearly as easy as proving the Infinitude of Primes. But the second part seems to requires something equivalent to the idea of a square root of -1 existing modulo some primes but not modulo others (recall Fact 16.1.2).

Proposition 22.1.4 (Infinitude of primes 3 mod 4)**.** *There is no largest prime congruent to 3 modulo 4.*

Proof. We'll prove this by contradiction. Let $p_1, p_2, \ldots, p_k$ be the (finite) set of primes congruent to 3 modulo 4.

Further define the product of all these primes with four, then subtracting one:

$$m = 4p_1p_2 \cdots p_k - 1$$

What are the prime divisors of this number?

- Clearly none of the p_i can be a prime divisor, since m is congruent to -1 modulo all the p_i.
- Yet m is not even, so it's not just a power of 2.
- But if m is a product only of primes congruent to 1 modulo 4, then it would have to be 1 modulo 4 itself (since any product of 1s is 1).
- This is false, so there must be another prime congruent to 3 modulo 4 which divides it.

This contradicts our assumption of having the full set of such primes, so that assumption must have been wrong. □

Proposition 22.1.5 (Infinitude of primes 1 mod 4)**.** *There is no largest prime congruent to 1 modulo 4.*

Proof. As usual, suppose there are finitely many primes p_i which are congruent to 1 modulo 4. Let's form the modified product

$$m = (2p_1p_2 \dots p_k)^2 + 1 \ .$$

What are its prime divisors? It is again clear that m is odd and that it is not divisible by any of the p_i, for the same reasons as above in the proof of Proposition 22.1.4.

It would be nice to directly use the same argument to show that one of the primes p which divides m is 1 modulo 4. Unfortunately, both 3^2 and 1^2 are congruent to 1 modulo 4, so this doesn't tell us anything about m.

However, we can use an indirect argument. For any prime divisor p of m and for $x = 2p_1p_2 \dots p_k \dots$, $m = x^2 + 1 \equiv 0 \pmod{p}$. So by definition -1 is a quadratic residue modulo p! Because of Fact 13.3.2, this can only happen if $p \equiv 1 \pmod{4}$.

Since that wouldn't be one of the p_i, this contradicts that we already had all such primes. □

22.1.2 Back to bias

Now, from what we've seen it looks like the $4k+3$ ones will always stay ahead. But that's not quite right. Here's one place where they fall behind.

```
def prime_race_up_to_n(n):
    p1 = 0
    p3 = 0
    for i in prime_range(n):
        if i%4==1:
            p1 += 1
        if i%4==3:
            p3 += 1
    pretty_print(html("Up␣to␣$n=%s$,␣there␣are:<ul><li>%s␣
        primes␣$p\equiv␣1\\text{␣(mod␣}4)$</li><li>%s␣
        primes␣$p\equiv␣3\\text{␣(mod␣
        }4)$.</li></ul>"%(n,p1,p3)))

prime_race_up_to_n(26860); prime_race_up_to_n(26862);
    prime_race_up_to_n(26864); prime_race_up_to_n(26880)
```

There are other n for which we have such an 'inversion' as well, and it can be fun to look for them. The next such time is over six hundred thousand, for a little while; after that, you have to look at n over twelve million. Indeed, there is a theorem that there are *infinitely* many times where this will happen, and that the 'wrong' team will get ahead by at least a specified amount.

Fact 22.1.6. *No matter how far out you go, there exists an n where the $4k+1$ team is ahead at x by*

$$\frac{1}{2}\frac{\sqrt{n}}{\log(n)}\log(\log(\log(n))) \ .$$

You may not be surprised to learn that this result is due to Littlewood, who was also one of the first contributors in studying the race between π and Li. That his result is highly nontrivial is seen in the following interact.

```
@interact
def _(n=26862):
    L = []
    p1 = 0
    p3 = 0
    for i in prime_range(n):
        if i%4==1:
            p1 += 1
            L.append([i,p1-p3])
        if i%4==3:
            p3 += 1
            L.append([i,p1-p3])
    P = plot(1/2*sqrt(x)/log(x)*log(log(log(x))),
        (x,10,n+10))
    P += plot_step_function(L)
    show(P)
```

Even though we can see the difference surge to become positive a few times, it seems hopeless to ever reach even the extremely slow $\log(\log(\log(x)))$. But it does.

22.1.3 Other prime races

There are many races we can check out, and mathematicians have. (Indeed, this section is indebted to the excellent expository article [C.6.3], which has a host of recent references.) What is the pattern here, for modulus eight?

```
@interact
def _(n=100):
    p1,p3,p5,p7=0,0,0,0
    L1 = []
    L3 = []
    L5 = []
    L7 = []
    for i in prime_range(n):
        if i%8==1:
            p1 += 1
            L1.append([i,p1])
        elif i%8==3:
            p3 += 1
            L3.append([i,p3])
        elif i%8==5:
            p5 += 1
            L5.append([i,p5])
        elif i%8==7:
            p7 += 1
            L7.append([i,p7])
    L1.append([n,p1])
    L3.append([n,p3])
    L5.append([n,p5])
    L7.append([n,p7])
    P = Graphics()
    P += plot_step_function(L1,color='red',legend_label='1␣
        (mod␣8)')
    P +=
        plot_step_function(L3,color='green',legend_label='3␣
        (mod␣8)')
```

```
P +=
    plot_step_function(L5,color='blue',legend_label='5␣
    (mod␣8)')
P +=
    plot_step_function(L7,color='orange',legend_label='7␣
    (mod␣8)')
show(P,xmin=max(0,n-1000),ymin=max(0,L1[-1][1]-100))
```

It turns out there are several types of theorems/conjectures one can make about such races. The key observation (which we will not explain here) is that the 'slow' teams are the residue classes $[a]$ such that $nk + a$ can be a perfect square (see Exercise 22.4.2). In our cases, only $4k + 1$ and $8k + 1$, respectively, are possible perfect (odd) squares. See also Exercise 22.4.3.

Nonetheless, for any a, b coprime to each other and to n,

$$\lim_{x\to\infty} \frac{\text{Number of } p \equiv a \pmod{n} \text{ less than } x}{\text{Number of } p \equiv b \pmod{n} \text{ less than } x} = 1$$

so the teams can't get too far away from each other, at least not on a percentage basis. The more specific result that the numerator and denominator are both asymptotic to $\frac{Li(x)}{\phi(n)}$ is often called the prime number theorem for arithmetic progressions, and it was also proved by Vallée-Poussin. (See the next section as well.)

With such a close connection to Chapter 21, at this point you won't be surprised to learn that, even though some teams are usually ahead, that just like with π and Li, each team does get ahead in the race infinitely often. But if you "count right" (and assume some other technical but important hypotheses), the proportion of the time the 'wrong' teams are ahead in the race is very small. (See the [C.6.3] for more details.)

22.2 Sequences and Primes

22.2.1 Primes in sequences

There is an interesting question implicit in the prime races. To legitimize doing the first prime race, we proved that there are infinitely many primes of the forms $4k + 1$ and $4k + 3$. However, we then proceeded to do prime races for several other such forms. Is it legitimate to do so?

The answer is yes, as proved in this *major* theorem that introduced limiting and calculus methods to the study of number theory.

Theorem 22.2.1 (Dirichlet's Theorem on Primes in an Arithmetic Progression)**.** *If* $\gcd(a, b) = 1$*, then there are infinitely many primes of the form* $ax + b$ *for* x *an integer.*

Proof. The proof of this theorem is far beyond the level of this text, but [C.3.6] is a standard resource for this. □

That is, $ax + b$ defines a progression of numbers separated always by a, and this theorem says there are infinitely many primes in any such progression that makes sense in terms of relative primeness. It is a weak version of a prime race; it just says that it makes sense to do them, though (as we saw) there is much more information one can glean from them.

```
@interact
def _(a=8,b=7,n=100):
    if gcd(a,b)!=1:
        pretty_print(html("Oops!␣␣The␣progression␣won't␣
            have␣many␣primes␣if"))
        pretty_print(html("$a$␣and␣$b$␣share␣a␣common␣
            factor!"))
    else:
        pretty_print(html("Primes␣of␣the␣form␣$%sx+%s$␣up␣
            to␣$%s$:"%(a,b,n)))
        for x in prime_range(n):
            if x%a==b:
                print x
```

We have already proved this for $a = 4$. It is easy to prove for $a = 2$! (See Exercise 22.4.4.)

It is also possible to prove the theorem for $b = 1$, or $b = -1$, without developing much bigger tools. In the article [C.6.1] a lot of factoring and expanding is used, and a much more recent article by Xianzu Lin [C.6.7] is similarly elementary. One can even prove Dirichlet's theorem without Dirichlet's methods for any b such that $b^2 \equiv 1 (\text{mod } a)$, but doing so involves some high-level details about polynomial factorization (see Murty and Thain's paper for details).

22.2.2 Sequences in primes

We can also look at the opposite question. Instead of considering whether primes exist in a given arithmetic progression, are there arithmetic progressions made of solely of primes?

Question 22.2.2. Can you get a (finite) sequence of the form

$$ak + b, k = 0, 1, 2, 3, \ldots n$$

where all entries are prime?

It's easy to find short arithmetic progressions in the primes. We say such a progression has length $n + 1$ in the above notation.

- 3, 5, 7 is an arithmetic progression of length 3, where $a = 2$.
- 41, 47, 53, and 59 is an arithmetic progression of length 4, where $a = 6$.

Longer ones get harder to find. Can you find a progression of length 5? (This is Exercise 22.4.5; there is a small one where the differences and starting number are both less than 10. See also Exercise 22.4.6.)

```
@interact
def _(p = prime_range(200), n=110):
    L = [p,p+n,..p+4*n]
    for z in L:
        if is_prime(z):
            print z
        else:
            print factor(z)
            break
```

Fact 22.2.3. *There is such a sequence of length 10 starting at 199, with differences of 210.*

Question 22.2.4. Can find *arbitrarily long* such sequences in the primes?

The answer is yes! This is a theorem of Ben Green and Terry Tao, which was a significant piece of Tao's 2006 Fields Medal (though he probably would have won it even without this, remarkable as it may seem). How might one *prove* this? That might seem mysterious, so we give the gist of the approach to it.

Remember how there seem to be fewer primes the further out we go, even in an arithmetic subsequence (e.g. prime mod 4 or mod 8)? That isn't a coincidence. There is a technical way to measure this:

$$\lim_{n\to\infty} \frac{\pi(n)}{n} = 0 \, .$$

This follows from Chebyshev's estimate in Theorem 21.3.4, and is called having **zero density.** We can try this for π with specific numbers:

- $\pi(100)/100 = 1/4 = 0.25$
- $\pi(200)/200 = 0.23$
- $\pi(1000)/1000 = 0.168$, or under 17%.
- $\pi(1000000)/1000000 \approx 0.0785$, or under 8%.

Now, if you have a collection of numbers which has **positive density** (i.e. the limit is positive, not zero), it is a theorem from 1974 (by Endre Szemerédi) that you can get arithmetic progressions of arbitrary length in such sets. Sadly, even our data suggests the primes are indeed approaching zero density.

But Green and Tao managed to show this type of method still works for the primes! You can't get arithmetic progressions in any old set with zero density; but somehow, although there are not many primes, there are just enough for things to work.

If you are interested in the current status of really long sequences, see the primerecords.dk website. The following example, the first of length 26, was found quite recently, on April 10, 2010.

```
difference=23681770*2*3*5*7*11*13*17*19*23
start=43142746595714191
for n in [0..25]:
    print start+n*difference,is_prime(start+n*difference)
```

There are currently only six known 26-length sequences, as of this writing (including one found just days before). Currently, there are no known 27-length sequences (though they must exist, by the Green/Tao theorem). They must even obey the following ridiculous bound.

Fact 22.2.5. *A sequence of length k must occur before*

$$2^{2^{2^{2^{2^{100k}}}}}$$

How do people find such lists? For that, we need a new notation.

Definition 22.2.6. For a prime p, we call the **primorial** the number

$$p\# = \prod_{q \leq p,\ q \text{ prime}} q$$

where the "p sharp" or "p hash"[1] denotes p primorial.

Armed with primorials, one usually finds such lists by the following method.

- First, for some fixed p, compute a large set of primes of the form $a \cdot p\# + 1$, keeping track of the a values in question.
- Next, find arithmetic progressions among the values of a from your list (not the values of $a \cdot p\# + 1$).
- If you find a bunch of a values in a progression of the form $k + \ell \cdot n$, then you've also found a progression of primes of the form $(k \cdot q\# + 1) + (\ell \cdot q\#)n$.

If you want to, you can even sign up to find a length 27 sequence at the PrimeGrid distributed search!

22.3 Types of Primes

There are many types of primes we have encountered up to this point. For instance:

- Germain (Subsection 11.6.4)
- Mersenne (Subsection 12.1.3)
- repunit (Exercise 6.6.1)

Notice that for ***many*** of these types, we don't know if there are finitely many or not! Are there any conjectures for how often certain types of primes might appear?

22.3.1 Twin primes

Consider primes in an arithmetic progression $ax + b$. Can one say anything about the constants involved in these progressions? Since b is pretty arbitrary, we would focus on a. Here are some natural questions to consider for small values of a.

Question 22.3.1.

- Find some primes that look like $2x + b$ for some b and several consecutive x. How many x in a row can you do?
- How about for $3x + b$?
- What about $4x + b$?
- Are the primes you get in these cases ever *consecutive*?

Hopefully it's pretty clear that you can't do every possible combination of b and a, nor can every such progression go on indefinitely! Why?

Thinking about this and the Sieve of Eratosthenes led the French mathematician Alphonse de Polignac to the following.

[1] Officially, this should be called an octothorp(e).

Conjecture 22.3.2 (Polignac's Conjecture)**.** *Every even number is the difference between* consecutive *primes in infinitely many ways.*

We have no proof of this. In fact, even the most basic case of Polignac's conjecture is one of the most celebrated open questions in number theory – celebrated enough that well-known comedian Stephen Colbert interviewed Fields medalist Tao about it.

Conjecture 22.3.3 (Twin prime conjecture)**.** *There are infinitely many consecutive odd prime numbers.*

Definition 22.3.4. Pairs of primes p and q such that $p + 2 = q$ are called **twin primes**.

There are lots of twin primes. The following cell computes twin prime pairs, numbered by which twin prime pair it is. The pair 17 and 19 is the fourth pair, for example.

```
def twin_primes_upto(n):
    v = prime_range(n+1)
    L = []
    counter = 0
    for i in range(len(v)-1):
        if v[i+1]-v[i]==2:
            counter += 1
            L.append((v[i],v[i+1],counter))
    return L

twin_primes_upto(100)
```

We can use similar searching to try to see whether there are enough that there are infinitely many

```
def twin_primes_upto(n):
    v = prime_range(n+1)
    L = []
    counter = 0
    for i in range(len(v)-1):
        if v[i+1]-v[i]==2:
            counter += 1
            L.append((v[i+1],counter))
    return L

var('t')
plot_step_function(twin_primes_upto(1000000),
    legend_label='twin␣prime') +
    plot(2*twinprime*x/log(x)^2,(x,1,1000000),
    color='black',legend_label='$C_2␣x/log(x)$') +
    plot(lambda t:
    2*twinprime*numerical_integral(1/log(x)^2,2,t)[0],
    (t,1,1000000), color='red',legend_label='$C_2␣Li_2(x)$')
```

You can see in the preceding graphic that it's certainly possible to approximate the twin prime counting function in a similar way to how we approximated the prime counting function π. There is a mysterious constant I've used; it will be explained below.

22.3.2 Heuristics for twin primes

To explain how to get to twin primes, there is a nice little rule of thumb; see e.g. [C.3.5] for what follows. Even though we definitely do *not* have a proof, we can still give you a good idea of how these ideas come about.

First, one might want to estimate how many primes there are up to a certain point to start. The problem is we should use a different idea than just looking at tables! What can we say that is a little smarter?

- About half the numbers less than n are not divisible by 2.
- About $2/3$ the numbers less than n are not divisible by 3.
- About $4/5$ the numbers less than n are not divisible by 5.
- Etc. for each prime less than $\sqrt{n}\ldots$

If we take this thinking to its logical extreme, you might even expect that

$$\prod_{p<\sqrt{x}} \left(1 - \frac{1}{p}\right)$$

is a good approximation of the probability that a given number x is prime. Unfortunately, it isn't. In fact, this product turns out to be asymptotic to $2e^{-\gamma}/\log(x)$ (recall that γ from Definition 20.3.2).

Still, this kind of thinking is still helpful, and might help us make ideas for how many *twin* primes there are – especially if we keep in mind this isn't really a probability. After all, if $p > 2$ is prime, then with one hundred percent probability the next number is not prime! And for p and $p+2$ to be both prime, they must also both be odd; so if p is odd, then $p+2$ is much more likely than a random number to be prime.

So we do the following analysis instead. (See Exercises 22.4.11 and 22.4.12.)

- Although one would expect for $1/4$ of all pairs separated by two to both be odd, $n+2$ has the same parity as n so we should expect $1/2$ the pairs to both be odd.
- The chances that n and $n+2$ are both not divisible by three is $1/3$.
- The chances that n and $n+2$ are both not divisible by five is $3/5$.
- And so forth.

So, having gotten a little more sophisticated, we might expect that

$$\frac{1}{2} \prod_{p<\sqrt{x}, p>2} \left(1 - \frac{2}{p}\right)$$

is a decent approximation of the probability that a given pair of consecutive odd numbers are both prime.

This doesn't look so recognizable yet, but we can do some algebra to turn this into something that looks better and has logarithms, just like in the prime number theorem. If we substitute

$$\left(1 - \frac{2}{p}\right) = \left(1 - \frac{1}{(p-1)^2}\right)\left(1 - \frac{1}{p}\right)^2$$

then the approximation of the number of twin primes less than x looks more like this:

$$\frac{1}{2}\prod_{p<\sqrt{x},p>2}\left(1-\frac{1}{(p-1)^2}\right)\prod_{p \text{ prime}}\left(1-\frac{1}{p}\right)^2$$

Finally, if we now use the earlier suggestion about the right-hand side being more or less the square of the number of primes, we come up with a reasonable suggestion that looks more familiar.

$$\frac{1}{2}\prod_{p<\sqrt{x},p>2}\left(1-\frac{1}{(p-1)^2}\right)\left(\frac{x}{\log(x)}\right)^2$$

Remark 22.3.5. The constant part of this formula is finite, and known as the **twin prime constant** :

$$C_2 = 2\prod_{p>2}\left(1-\frac{1}{p-1}^2\right).$$

The graphs in Subsection 22.3.1 use this constant (which is built-in in Sage) as well as a logarithmic integral version of the preceding analysis.

There is some inconsistency in the literature about whether the 2 in front of the formula for C_2 is part of the twin prime constant or not.

This also leads to a conjecture of Hardy and Littlewood.

Conjecture 22.3.6. *The number of ways to write an even number $2k$ as a sum of primes is also asymptotic to $\frac{1}{2}\prod_{p<\sqrt{x},p>2}\left(1-\frac{1}{(p-1)^2}\right)\left(\frac{x}{\log(x)}\right)^2$.*

This would provide a very overwhelming proof of the following old suggestion, going back to correspondence between Euler and Goldbach.

Conjecture 22.3.7 (Goldbach Conjecture)**.** *Any even number can be written in at least* one *way as a sum of two primes.*

In fact, there are two such conjectures, with the other one suggesting that *any* positive integer may be written as a sum of *three* primes.

Returning to the twin prime constant, computing it (as in the Sage cell below) led to a very interesting real-life application.

```
2*twinprime.n()
```

Computing this constant to arbitrary precision led to the discovery of the infamous Pentium chip bug, where some floating-point calculations would be incorrect in high decimal places. This is a quite surprising 'application' of number theory! (It turns out manufacturers do use number-theoretic computations to stress-test their products.)

It is still unknown whether there are infinitely many twin prime pairs. In a 2013 result that shocked the mathematics world, (then) unknown mathematician Yitang Zhang proved that there exists some N less than seventy million such that there are infinitely many pairs of primes separated by exactly N. This was a huge improvement over previous results, and further work of an unusually collaborative nature have now reduced this bound to $N \leq 246$.

As we finish this subsection, we must mention another constant affiliated with twin primes. Although there may really be infinitely many pairs, the sum of their reciprocals

$$\sum_{p,p+2 \text{ both prime}}\frac{1}{p}+\frac{1}{p+2}$$

is still a finite constant. At the very least means twin primes must be pretty rare. This (possibly infinite) sum is called **Brun's constant**, and is also in Sage.

```
brun.n(digits=5)
```

22.3.3 Other types of primes

In the quest toward Polignac's Conjecture, researchers have dubbed primes (not necessarily consecutive) with spacing $N = 4$ **cousin primes** and those $N = 6$ apart **sexy primes**. In another result of similar vintage to Zhang's (and also collaborative like its refinement), we know (conditional upon the so-called "generalized Elliott-Halberstam conjecture", which is closely related to our investigations in Subsection 22.2.2) that at least one of the classes of twin, cousin, or sexy primes is infinite[1]. This is a very special case of exploring something called **prime constellations**; see Exercise 22.4.13.

In addition, there are many other heuristics like the ones above. Here is a sampling of those we don't have space or expertise in this text to dig further into.

- As one example, consider the chance that n and $2n + 1$ are both not divisible by a given prime p. Probabilistically, this is basically the same chance as that n and $n + 2$ are both not divisible by p, so it turns out that Germain primes might also be distributed in the same fashion as twin primes.
- Using similar ideas, one can get a heuristic that Mersenne primes are distributed as
$$e^{\gamma} \log(\log(x))/\log(2)$$
This is known as Wagstaff's conjecture.
- Bizarrely, one can use the same idea to get a heuristic for **factorial primes**. These are primes of the form $n! \pm 1$, like 5, 7, 23, and 719. It's conjectured that there are $e^{\gamma} \log(n)$ such primes less than n.
- These rules of thumb even seem to apply to the so-called **primorial primes** – primes of the form $p\# \pm 1$, like 3, 5, 7, 29, 31, 211, etc. It's truly weird, yet also cool.

There is so much to explore! There is never a lack of questions for mathematicians to explore when it comes to prime numbers.

22.4 Exercises

1. Explain why, to show that any number can be written as a sum of three primes, it suffices to prove Conjecture 22.3.7.

2. In Subsection 22.1.3 a statement is made about residue classes $[a]$ such that $nk + a$ can be a perfect square. What is another name for such a?
Also, the claim is made that, "In our case, only $4k + 1$ and $8k + 1$ are possible perfect (odd) squares." Either prove this claim or find the reference for when that is proved in the book.

[1] Go to the video of Tao's interview with Colbert again to see his reaction to this; it's quite amusing.

3. What 'teams' would you expect to be in the lead long-term for a modulo ten prime race? Why? Compute a value where the 'wrong' team is in the lead, if you can!

4. Prove Dirichlet's Theorem on Primes in an Arithmetic Progression for the case $a = 2$.

5. Find an arithmetic progression of primes of length five with less than ten between primes.

6. Find an arithmetic progression of primes of length six or seven, starting at a number less than ten.

7. Prove that there can be only one set of "triple primes" – that is, three consecutive odd primes.

8. Find the value of $23\#$.

9. Compute some twin primes greater than one thousand.

10. Show that $\left(1 - \frac{2}{p}\right) = \left(1 - \frac{1}{(p-1)^2}\right)\left(1 - \frac{1}{p}\right)^2$.

11. What form must n have for n and $n+2$ to both *not* be divisible by three?

12. Which residues modulo five must n avoid for n and $n+2$ to both *not* be divisible by five?

13. Search a few resources to learn about "prime constellations" and write a report. The Prime Pages or Tomás Oliveira e Silva's very nice graphs of "admissible" constellations are a good place to start.

14. Let $D(N) = \prod_{p<N}\left(1 - \frac{1}{p}\right)$. Compute $D(N)$ by hand for all N between 10 and 20, *without* adding the fractions (just "FOIL" it out). What patterns do you notice in the denominators? The numerators?

15. Search a good book (see the general C.1 or specialized C.3 references) or the internet for an amazing fact about primes. Describe it in a way your classmates (or peers, if you're not in a course) will understand.

Chapter 23

New Functions from Old

We are heading toward the end of the text. There are even more interesting functions out there; just as important, there are more interesting ways to start connecting these functions to calculus.

In the previous section's exercises, we introduced an interesting function. Letting p be running just over primes, we let

$$D(N) = \prod_{p<N} \left(1 - \frac{1}{p}\right)$$

As an example,

$$D(3) = (1 - 1/2)(1 - 1/3) = \left(1 - \frac{1}{2} - \frac{1}{3} + \frac{1}{6}\right)$$

Before starting this chapter, try expanding the expression D for bigger and bigger N (as above, without adding the fractions). What patterns do you find?

- What denominators show up?
- Which ones don't?
- For the ones that do, what are the values of the numerator?
- Can you predict the value of the numerator for some types of denominators? (E.g., primes, perfect squares, prime powers, etc.)

The function unveiled by this is quite important in expanding our roster of arithmetic functions and unlocking their secrets, as well as in connecting to calculus.

23.1 The Moebius Function

23.1.1 Möbius mu

Let's define the function which gives the numerator associated with denominator n in the products above.

Definition 23.1.1 (Moebius mu)**.** Let $N = 2 \cdot 3 \cdot 5 \cdots q$ be the product of the first few primes, up to q. Then we define $\mu(n)$ as follows:

$$\prod_{p|N} \left(1 - \frac{1}{p}\right) = \sum_{d|N} \frac{\mu(d)}{d}$$

The product is over *prime* factors of N but the sum is over *all* factors of N.

Yes, this is the same Moebius (or Möbius) as the Moebius strip.

Example 23.1.2. Using the example in the chapter introduction,

$$D(3) = (1 - 1/2)(1 - 1/3) = \left(1 - \frac{1}{2} - \frac{1}{3} + \frac{1}{6}\right)$$

implies that $\mu(2) = -1$ while $\mu(6) = 1$.

There is no product of $(1 - 1/p)$ that will yield a four in the denominator, since $(1-1/2)$ only occurs once in such a product. So $\mu(4) = 0$, as the example above already implies.

23.1.2 A formula

Before describing this function further, let's think more about the product $\prod_{p<N}\left(1 - \frac{1}{p}\right)$.

- First, as the comment at the end of the last subsection points out, it seems to create denominators with each prime factor to just the first power. We couldn't get a square or cube of any given p in the denominator.
- Similarly, the numerators really can only be products of 1 and -1. For a moment, think about why there are no other numerators available.
- Finally, the number of prime factors in the denominator should be the same as the number of times -1 is part of the product in the numerator.

This essentially proves the following proposition.

Proposition 23.1.3. *If $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ then a nice formula for $\mu(n)$ is*

$$\mu(n) = \begin{cases} 0 & \text{any } e_i > 1 \\ (-1)^k & \text{otherwise} \end{cases}$$

Proof. See above. □

23.1.3 Another definition

The μ function is so important that we will want several more approaches as well. It is the mark of an important concept that there are ways to define it from many directions.

One important way that μ is *often* defined is via a recurrence relation. That is, one defines

$$\mu(1) = 1, \text{ and } \sum_{d|n} \mu(d) = 0 \, .$$

Now, we haven't proved this identity yet, and probably the reader hasn't even noticed it. But if we can prove the identity works for μ, then since $\mu(1) = 1$ is true, this would give an alternate definition.

Proposition 23.1.4 (Recursive definition of μ)**.**

$$\sum_{d|n} \mu(d) = 0$$

Proof. Let's rewrite the sum $\sum_{d|n} \mu(d) = 0$ by trying to omit the ones that are zero anyway. If we do this, the sum reduces to the long, but correct,

$$\sum_{d|n} \mu(d) = \sum_{\substack{\text{all divisors } d \text{ with just one or zero} \\ \text{of each prime factor } p_i \text{ of } n}} (-1)^{\text{the number of primes dividing } d}.$$

Now let's set up a little notation. First, let k be the total number of (distinct) prime divisors of n. Next, let's borrow from Definition 23.3.3 the notation $\omega(d)$ for the number of distinct prime divisors of a divisor d of n.

Then the crazy sum $\sum_{d|n} \mu(d)$ becomes easier to write:

$$\sum_{\substack{\text{all divisors } d \text{ with just one or zero} \\ \text{of each prime factor } p_i \text{ of } n}} (-1)^{\omega(d)} = \sum_{d \text{ that work}} (1)^{k-\omega(d)}(-1)^{\omega(d)}$$

You should be asking yourself why I bothered introducing k. You may want to think about that briefly, noting that $(k - \omega(d)) + \omega(d) = k$.

The rationale is that we can think of each of the divisors d that have no square factors (the ones in question) as having $\omega(d)$ of the prime factors of n picked, and the other $k - \omega(d)$ factors omitted. So, in some sense, for each d we are really picking a subset of the primes dividing n, of size $\omega(d)$, and then multiplying by 1 for each prime picked and -1 for each one not picked.

But if instead we consider just picking a subset of $\{1, 2, \ldots, k\}$ and assigning ± 1, that would be the same thing, with the difference that we know this is the same as the result of expanding

$$(1 + (-1))^k = (1 + (-1))(1 + (-1)) \cdots (1 + (-1)) \text{ (k times)}$$

So the sum is

$$\sum_{d \text{ that work}} (-1)^{\omega(d)} = (1 + (-1))^k = 0.$$

This finishes the proof. □

Sage note 23.1.5 (Check your work again)**.** We can always check things like this by computing.

```
moebius(30) + moebius(15) + moebius(10) + moebius(6) +
    moebius(5) + moebius(3) + moebius(2) + moebius(1)
```

Fact 23.1.6. *The function μ is multiplicative.*

Proof. We will postpone a formal proof of this to a much bigger theorem, from which this result (Corollary 23.4.13) will fall "for free". □

Let's check it:

```
print gcd(111,41)
print moebius(111)*moebius(41)==moebius(41*111)
```

23.2 Inverting Functions

The main point of the Moebius function is the following famous theorem.

Theorem 23.2.1 (Möbius Inversion Formula)**.** *If* $f(n) = \sum_{d|n} g(d)$*, then*

$$g(n) = \sum_{d|n} \mu(d) f\left(\frac{n}{d}\right) .$$

Proof. The proof is delayed to Subsection 23.2.2. □

We can interpret this result briefly as follows. Suppose you sum an arithmetic function over the set of its (positive) divisors to create a new function. Then summing *that* function over divisors, along with μ, gives you back the original function.

The reason we care about this is that we are able to *use* the μ function to get *new*, useful, arithmetic functions via this theorem. In particular, we can "invert" all of our usual arithmetic functions, and this will lead to some very powerful applications.

23.2.1 Some useful notation

In order to better understand what this theorem is saying, let's introduce some notation.

Definition 23.2.2 (Dirichlet product)**.** Let f and g be arithmetic functions. Then we define the new function $f \star g$, the **Dirichlet product**, via the formula

$$(f \star g)(n) = \sum_{de=n} f(d)g(e) = \sum_{d|n} f(d) g\left(\frac{n}{d}\right) .$$

Example 23.2.3. For example, if $u(n) = 1$ and $N(n) = n$, then

$$(\phi \star u)(n) = \sum_{d|n} \phi(d) u\left(\frac{n}{d}\right) = \sum_{d|n} \phi(d) = n = N(n) .$$

We saw this originally in Fact 9.5.4, but now we can write it concisely as $\phi \star u = N$ and see it is part of a bigger context. (See also Fact 23.3.2.)

This notation, like all the best notation, practically demands that we restate the inversion theorem in a very insightful way:

$$\text{If } f = g \star u, \text{ then } g = f \star \mu .$$

23.2.2 Proof of Moebius inversion

Now we are ready to prove the Möbius Inversion Formula, following [C.1.1].

Let's expand the formula for $g(n)$ the theorem would give, in terms of g itself.

$$\sum_{d|n} \mu(d) f\left(\frac{n}{d}\right) = \sum_{d|n} \mu(d) \left[\sum_{e|\frac{n}{d}} g(e)\right] .$$

Each time $g(e)$ appears in this sum, it has a coefficient of $\mu(d)$. How often does this happen, and what is d anyway?

If $e \mid \frac{n}{d}$, then $e \mid n$, which means $\frac{n}{e}$ is an integer. However, this integer must have at least a factor of d "left" in it (after division by e). Why? Since e divides $\frac{n}{d}$, we have $ed \mid n$, in which case certainly $d \mid \frac{n}{e}$.

So $g(e)$ shows up once for each $d \mid \frac{n}{e}$, with coefficient $\mu(d)$. Thus,

$$\sum_{d|n} \mu(d) f\left(\frac{n}{d}\right) = \sum_{e|n} \left(\sum_{d|\frac{n}{e}} \mu(d) \right) g(e) \,.$$

Here comes the final step. Unless $\frac{n}{e} = 1$, we have $\sum \mu(d) = 0$. So the only subsum in this double sum that sticks around is the term for $\frac{n}{e} = 1$, or when $e = n$.

Thus the whole sum collapses to $g(n)$, as desired!

23.3 Making New Functions

23.3.1 First new functions

In order to see what good this does, let's see what happens when we mess around and make Dirichlet products with functions we know. We already know two of these functions, and I give you a third.

Definition 23.3.1. We define a new simple arithmetic function to go along with those from Definition 19.2.9.

- $u(n) = 1$ for all n
- $N(n) = n$ for all n
- $I(n) = \begin{cases} 1 & n = 1 \\ 0 & n > 1 \end{cases}$

In the next computational cell, we define these, as well as a Dirichlet product function. Now let's see what we get!

```
def u(n): return 1
def N(n): return n
def I(n): return floor(1/n)
def DirichletProduct(f,g,n): return sum(f(d)*g(n/d) for d
    in divisors(n))
```

For instance, what happens if we look for the inverse of N?

```
@interact
def _(n=10):
    H = [['$i$','$(N\star_\mu)(i)$']]
    T = [(i,DirichletProduct(N,moebius,i)) for i in [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

Maybe this is a surprise! But this makes sense, if you remember Example 23.2.3 just previously about $N = \phi \star u$. Let's confirm that fact numerically as well.

```
@interact
def _(n=10):
    H = [['$i$','$(\\phi\star u)(i)$']]
    T = [(i,DirichletProduct(u,euler_phi,i)) for i in
        [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

We summarize these explanations as follows.

Fact 23.3.2. *We may identify the following Dirichlet products as known functions.*

- $\phi \star u = N$
- $N \star \mu = \phi$

The second part of Fact 23.3.2 gives an alternate proof for our formula for ϕ from Exercise 9.6.9.

$$\phi(n) = N \star \mu(n) = \sum_{d|n} N\left(\frac{n}{d}\right)\mu(d) = n\sum_{d|n}\frac{\mu(d)}{d} = n\prod_{p|n}\left(1-\frac{1}{p}\right)$$

The last step follows from our initial definition of μ in Definition 23.1.1.

23.3.2 More new functions

Next, please try computing the Moebius inversions of our old friends, σ and τ, *by hand* for several values. (Hint: try primes and perfect powers first, as they don't have many divisors!)

You can try something out here in Sage as well.

```
```

Here one can try this interactively. (You'll need to evaluate the earlier cell after Definition 23.3.1 if you get an error.)

```
@interact
def _(n=10):
    H = [['$i$','$(\\tau\star\mu)(i)$']]
    T = [(i,DirichletProduct(lambda y:
        sigma(y,0),moebius,i)) for i in [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

```
@interact
def _(n=10):
    H = [['$i$','$(\sigma\star\mu)(i)$']]
    T = [(i,DirichletProduct(sigma,moebius,i)) for i in
        [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

There is loads of fun to be had here. We could try to see what $\mu \star \mu$ is, or $u \star u$. Could there be a formula for $|\mu|$, or could we calculate $|\mu| \star u$?

```
@interact
def _(n=10):
    H = [['$i$','$(\mu\star\mu)(i)$']]
    T = [(i,DirichletProduct(moebius,moebius,i)) for i in
        [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

```
@interact
def _(n=10):
    H = [['$i$','$(u\star u)(i)$']]
    T = [(i,DirichletProduct(u,u,i)) for i in [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

It turns out there are all kinds of other functions you can define. We already saw the first of these informally in our discussion of the Moebius function in Proposition 23.1.4.

Definition 23.3.3. If

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

then we call give the name $\omega(n) = k$. This is the number of unique prime divisors of an integer. (This is sometimes called $\nu(n)$ in the literature.)

Definition 23.3.4. If $n = \prod_{i=1}^{k} p_i^{e_i}$, we summarize the parity of the total powers of primes dividing a number as

$$\lambda(n) = (-1)^{e_1+e_2+\cdots+e_k}.$$

This is called **Liouville's function**.

In both cases, you might want to try a few values to see what these functions look like. See Exercise 23.5.1, or pursue these ideas:

- What is the value for primes?
- What is the $\star$ product of this with something – say, u?

Finally, although we provide some Sage cells to try things out, you should try them not just with Sage, but also *by hand*; this is part of the allure of number theory. The sky's the limit. Enjoy!

```
def u(n): return 1
def N(n): return n
def I(n): return floor(1/n)
def omega(n): return len(n.prime_divisors())
def liouville(n): return (-1)^sum([z[1] for z in
    n.factor()])
def DirichletProduct(f,g,n): return
    sum(f(d)*g(Integer(n/d)) for d in divisors(n))
```

```
@interact
def _(n=10,f=[liouville,u,N,moebius,omega,I],
    g=[liouville,u,N,moebius,omega,I]):
    H = [['$i$','$(%s\star %s)(i)$'%(f,g)]]
    T = [(i,DirichletProduct(f,g,i)) for i in [1..n]]
    pretty_print(html(table(H+T, header_row=True,
        frame=True )))
```

23.4 Generalizing Moebius

There is a more serious side to the panoply of new functions, though. This is our key to arithmetic functions. We will now turn to algebra again, with a goal of generalizing the Moebius result.

23.4.1 The monoid of arithmetic functions

Definition 23.4.1. A **commutative monoid** is a set with multiplication (an operation) that has an identity, is associative and commutative.

You can think of a commutative monoid as an Abelian group without requiring inverses. (That means it's not necessarily a group, though it could be.)

Theorem 23.4.2. *Let A be the set of all arithmetic functions. Then $\star$ turns the set A into a* commutative monoid.

Proof. The function $I(n)$, which is equal to zero except when $n = 1$, plays the role of identity. Then one would need to prove the following three statements.

- $f \star g = g \star f$
- $(f \star g) \star h = f \star (g \star h)$
- $f \star I = f = I \star f$

We include one of the proofs. The others are similar – see Exercise 23.5.2. Note that for the second one, one can use the fact that $dc = n, ab = d$ implies $abc = n$.

Proof of commutativity:

$$f \star g = \sum_{d|n} f(d)g\left(\frac{n}{d}\right) = \sum_{de=n} f(d)g(e)$$

$$= \sum_{de=n} g(e)f(d) = \sum_{e|n} g(e)f\left(\frac{n}{e}\right) = g \star f$$

□

Can you think of other commutative monoids? What sets have an operation and an identity, but no inverse?

23.4.2 Bringing in group structure

Let's get deeper in the algebraic structure behind the $\star$ operation. Remember, $f \star g$ is defined by

$$(f \star g)(n) = \sum_{de=n} f(d)g(e) \ .$$

This structure is so neat is because it actually allows us to generalize the idea behind the Moebius function!

Theorem 23.4.3. *If f is an arithmetic function and $f(1) \neq 0$, then f has an inverse in the set A under the operation $\star$. We call this inverse f^{-1}. It is given by the following recursive definition:*

$$\begin{cases} f^{-1}(1) = \frac{1}{f(1)} & n = 1 \\ \sum_{d|n} f^{-1}(d) f\left(\frac{n}{d}\right) = \sum_{de=n} f^{-1}(d)f(e) = 0 & n > 1 \end{cases}$$

Proof. As in all the best theorems, there is really nothing to prove. We can always get the next value of $f^{-1}(n)$ by knowledge of $f^{-1}(d)$ for $d \mid n$, and that is enough for an induction proof, since we do have a formula given for $f^{-1}(1)$. (See Exercise 23.5.9) □

Corollary 23.4.4. *This says* exactly *that the Moebius function μ is $\mu = u^{-1}$.*

This is a good time to try to figure out what the inverse of N or ϕ is *with paper and pencil*. See Exercises Exercise 23.5.4 and Exercise 23.5.5.

In general, we can also say that

$$f \star f^{-1} = I = f^{-1} \star f$$

There is another, more theoretical, implication too.

Corollary 23.4.5. *The subset of A which consists of all arithmetic functions with $f(1) \neq 0$ is actually a* group.

23.4.3 More dividends from structure

This new way of looking at things yields an immediate slew of information about arithmetic functions. The following results will yield dividends about number theory and analysis/calculus (no, we haven't forgotten that!) in the next chapter on Infinite Sums and Products.

Fact 23.4.6. *The Moebius inversion formula that if $f = g \star u$ then $g = f \star \mu$ can be proved concisely by*

$$g = g \star I = g \star u \star \mu = f \star \mu$$

(We need no parentheses, since $\star$ is associative).

Fact 23.4.7. *Conversely, if $g = f \star \mu$, then*

$$f = f \star I = f \star \mu \star u = g \star u$$

so the inversion formula is true in both directions.

Proposition 23.4.8. *If g and h are multiplicative, then $f = g \star h$ is also multiplicative.*

Proof. See Exercise 23.5.8. □

The next result has a long proof, but most of it is following the definitions and keeping careful track of indices.

Proposition 23.4.9. *If f is multiplicative and $f(1) \neq 0$, then f^{-1} is also multiplicative.*

Proof. This basically can be done by induction, but each step is somewhat involved so we will break this into several lemmata. Throughout, recall that the inverse is defined by

$$f^{-1}(1) = \frac{1}{f(1)}$$

and, for $n > 1$, the condition

$$\sum_{d|n} f^{-1}(d) f\left(\frac{n}{d}\right) = \sum_{de=n} f^{-1}(d) f(e) = 0\,.$$

First, in Lemma 23.4.10 we will show that $f^{-1}(1)$ behaves well.

Then, assuming as an inductive hypothesis that f^{-1} is multiplicative for inputs less than mn, with $\gcd(m,n) = 1$, we will show in Lemma 23.4.11 that

$$f^{-1}(mn) = - \sum_{(ac)(bd)=(m)(n),\ ab<mn} f^{-1}(a) f^{-1}(b) f(c) f(d)$$

Finally, in Lemma 23.4.12 we will show how to rewrite this as

$$f^{-1}(mn) = f^{-1}(m) f^{-1}(n)$$

which finishes the induction argument. □

Lemma 23.4.10. *We know that both $f^{-1}(1) = \frac{1}{f(1)}$ and $f(1) = 1 = f^{-1}(1)$*

Proof. Left to the reader in Exercise 23.5.10; use everything you know about f. □

Lemma 23.4.11. *Assume as above that f^{-1} is multiplicative for inputs less than mn, with $\gcd(m,n) = 1$. Then*

$$f^{-1}(mn) = - \sum_{(ac)(bd)=(m)(n),\ ab<mn} f^{-1}(a) f^{-1}(b) f(c) f(d)$$

Proof. Assume that $m, n > 1$ and coprime. By the definition of inverse, we have

$$0 = (f^{-1} \star f)(mn) = \left[\sum_{x<mn,\ xy=mn} \left(f^{-1}(x) f(y)\right) \right] + f^{-1}(mn) f(1)\,.$$

By assumption, every function in this expression (both f and f^{-1}) is multiplicative on the values in question, with the possible exception of $f^{-1}(mn)$. Further, each summand is over xy, each of which is itself, by coprimeness of m and n, itself a product of things that are coprime.

So let $x = ab$ and $y = cd$, where $a, c \mid m$ and $b, d \mid n$. Then, as everything is multiplicative, $f^{-1}(x) f(y) = f^{-1}(a) f^{-1}(b) f(c) f(d)$.

Since by the previous lemma $f(1) = 1$, we can subtract the summation from both sides of the equation whose left-hand side is zero at the beginning of this lemma's proof, yielding

$$f^{-1}(mn) = - \sum_{(ac)(bd)=(m)(n),\ ab<mn} f^{-1}(a) f^{-1}(b) f(c) f(d)$$

□

Lemma 23.4.12. *Under the same hypotheses as before,* $f^{-1}(mn) = f^{-1}(m)f^{-1}(n)$.

Proof. We now write all this in terms of things we already can evaluate.

If the sum in question were summed over every $ab \leq mn$ instead of $ab < mn$, it would easily simplify as a product:

$$\sum_{(ac)(bd)=(m)(n)} f^{-1}(a)f^{-1}(b)f(c)f(d) = \sum_{ac=m} f^{-1}(a)f(c) \sum_{bd=n} f^{-1}(b)f(d)$$

The sum in Lemma 23.4.11 only lacks the term with $a = m, b = n$, in fact. So

$$\sum_{(ac)(bd)=(m)(n),\ ab<mn} f^{-1}(a)f^{-1}(b)f(c)f(d) =$$

$$\left[\sum_{ac=m} f^{-1}(a)f(c) \sum_{bd=n} f^{-1}(b)f(d)\right] - \left(f^{-1}(m)f^{-1}(n)f(1)f(1)\right)$$

Now we can plug this back into the previous characterization of $f^{-1}(mn)$:

$$f^{-1}(mn) = -\left[\sum_{ac=m} f^{-1}(a)f(c) \sum_{bd=n} f^{-1}(b)f(d) - f^{-1}(m)f^{-1}(n)f(1)f(1)\right]$$

Since $m, n > 1$, the individual sums may be rewritten as

$$(f^{-1} \star f)(m) = I(m) = 0 = I(n) = (f^{-1} \star f)(n)$$

That means we achieve the desired result

$$f^{-1}(mn) = f^{-1}(m)f^{-1}(n)f(1)f(1) = f^{-1}(m)f^{-1}(n)$$

□

Finally, we get the following promised corollary from the beginning of the chapter, Fact 23.1.6.

Corollary 23.4.13. *The function* μ *is multiplicative.*

Proof. This follows since u is multiplicative (trivially) and $\mu = u^{-1}$. □

23.5 Exercises

1. Factoring by hand, compute the first 24 values of λ and ω.

2. Finish the proof that the set of arithmetic functions is a commutative monoid in Theorem 23.4.2.

3. Show that if $f = g \star u$ (equivalently, if $g = f \star \mu$), then f and g are either both multiplicative or both not. Strategy hint: Use Proposition 23.4.9.

4. Do enough calculations the old-fashioned way to discover a name for the inverse of N.

5. Do enough calculations the old-fashioned way to discover a name for the inverse of ϕ.

6. Show that the inverse of $\lambda(n)$ from Definition 23.3.4 is a variant of another of our new functions.

7. Can you identify $\omega \star \mu$ as anything familiar? (Recall Definition 23.3.3.) If yes, then try to prove it; if not, explain why you think it is new to us.

8. Prove Proposition 23.4.8 that using the Dirichlet product on two multiplicative functions stays multiplicative.

9. Complete all details of the proof of Theorem 23.4.3 defining inverses under the $\star$ product.

10. Prove Lemma 23.4.10.

11. Come up with another good exercise for this chapter and have a friend try it!

Chapter 24

Infinite Sums and Products

We are almost at the very frontiers of serious number theory research now. In order to start to understand this, we will need to introduce two final concepts:

- **Euler products**
- **Dirichlet series**

These concepts both deeply involve infinitely applied operations, and are what this chapter is about. If you wish, think of this chapter as the 'infinite' version of the previous chapter on new functions.

24.1 Products and Sums

In order to motivate bringing infinite processes to this part of number theory, let's step back a bit. Many functions we have already seen may be thought of in two ways – *either* as a product *or* as a sum.

24.1.1 Products

Let $p \mid n$ as an indexing tool denote the set of primes which divide n. Then we have the following *product* representation of two familiar arithmetic functions. (Recall Theorem 19.2.5 and Fact 18.1.2.)

$$\sigma(n) = \prod_{p|n} \left(\frac{p^{e+1} - 1}{p - 1} \right) = \prod_{p|n} \left(1 + p + p^2 + \cdots + p^e\right)$$

$$\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Both of these functions therefore may be thought of as (finite) products.

As a related example, we explicitly wrote out the product for the abundancy index in Section 19.3.

$$\frac{\sigma(n)}{n} = \frac{\prod_{p|n} \left(\frac{p^{e+1}-1}{p-1}\right)}{\prod_{p|n} p^e} = \prod_{p|n} \frac{p - (1/p^e)}{p - 1}$$

Alternately, to avoid fractions:

$$\frac{\sigma(n)}{n} = \frac{\prod_{p|n} \left(1 + p + p^2 + \cdots + p^e\right)}{\prod_{p|n} p^e} = \prod_{p|n} \left(1 + p^{-1} + p^{-2} + \cdots + p^{-e}\right)$$

Note that $\frac{\phi(n)}{n} = \prod_{p|n} \left(1 - \frac{1}{p}\right)$.

24.1.2 Products that are sums

On the other hand, these *products over primes* are also *sums over divisors*; this is true either by definition or by theorem, depending on how you look at it.

It's clear with σ that this is the case, since we defined (in Definition 19.1.1)

$$\sigma(n) = \sum_{d|n} d$$

We can even cleverly add up the divisors in the opposite order to get the slightly more felicitous

$$\sigma(n) = \sum_{d|n} \frac{n}{d} = n \sum_{d|n} \frac{1}{d}$$

This leads us directly to writing $\frac{\sigma(n)}{n} = \sum_{d|n} \frac{1}{d}$.

With ϕ we have something to prove to make this connection, but not much. In Fact 23.3.2 we saw that $\phi \star u = N \Rightarrow \phi = N \star \mu$. Equivalently, we have Möbius-inverted Fact 9.5.4 to obtain, from $\sum_{d|n} \phi(d) = n$, the formula

$$\sum_{d|n} d\mu\left(\frac{n}{d}\right) = \phi(n)$$

By adding the divisors in the opposite order (alternately, by noting $\star$ is commutative) we can write

$$\phi(n) = \mu \star N = \sum_{d|n} \mu(d)\left(\frac{n}{d}\right) = n \sum_{d|n} \frac{\mu(d)}{d},$$

which allows us to also write the fraction as

$$\frac{\phi(n)}{n} = \sum_{d|n} \frac{\mu(d)}{d}$$

Now, in some sense we already knew all this. Great, some arithmetic functions can be represented either as a sum over divisors or as a product over primes, depending on what you need from them. So what?

The genius of Euler was to *directly connect* these ideas.

Fact 24.1.1. *We can equate sums over divisors and products over primes for special formulas.*

$$\frac{\phi(n)}{n} = \sum_{d|n} \frac{\mu(d)}{d} = \prod_{p|n}\left(1 - \frac{1}{p}\right)$$

$$\frac{\sigma(n)}{n} = \prod_{p|n}\left(1 + \frac{1}{p} + \frac{1}{p^2} + \cdots + \frac{1}{p^e}\right) = \sum_{d|n} \frac{1}{d}$$

Well, almost; his *real* genius was to take them to the limit!

One can't *really* take these expressions to infinity as they stand – one would get massive divergence. So what can we do? To analyze this, we will define new, related functions which preserve the summation, but allow for convergence.

24.2 The Riemann Zeta Function

24.2.1 A fundamental function

The most important such infinite process is the following fundamental function. It is one of the most studied, yet most mysterious functions in all of mathematics.

Definition 24.2.1 (Riemann zeta function)**.** We define the **zeta function** (denoted ζ) as the sum of the infinite series

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = 1 + \frac{1}{2^s} + \frac{1}{3^s} + \cdots$$

as a function of s.

For now we'll keep the domain of ζ to be only the s where this series converges. Later, in Subsection 25.3.1, we'll see that it will be useful to think about what ζ might mean for other values of s.

Here we plot the function for a few positive values of s.

```
plot(zeta,0,4,ymin=-1,ymax=10)
```

Riemann, the quietly devout son of a Lutheran pastor, made groundbreaking contributions in nearly every area of mathematics. He did it in analysis (Riemann sums), in geometry (Riemannian metrics, later used by Einstein), in function theory (Riemann surfaces) – and in one paper that changed the course of number theory. He died quite young (around 40).

24.2.2 Motivating the Zeta function

The motivation for this definition comes from this function with the case $s = 1$.

We begin with the second formula in Fact 24.1.1:

$$\prod_{p|n} \left(1 + \frac{1}{p} + \frac{1}{p^2} + \cdots + \frac{1}{p^e}\right) = \sum_{d|n} \frac{1}{d}$$

Try computing both sides of this and seeing how they come together for a few fairly composite n, like 12, 16, 18, 20, or 30.

```
@interact
def _(n=[30,20,18,24,12,16]):
    str = '$$'+'␣+␣'.join(['\\frac{1}{%s}'%d for d in
        divisors(n)])+'=%s$$'%sum([1/d for d in
        divisors(n)])
    str2 = '$$' +
        ''.join(['\left('+'+'.join(['\\frac{1}{%s^{%s}}'%(p,
        k) for k in [0..e]])+'\\right)' for (p,e) in
        factor(n)]) + '=%s$$'%prod([sum([p^(-k) for k in
        [0..e]]) for (p,e) in factor(n)])
    pretty_print(html(str))
    pretty_print(html("compare␣to␣"+str2))
```

Notice how *every* integer d formable by a product of the prime powers dividing n shows up *precisely* once (as a reciprocal) in the sum. This gives us a way into introducing limits.

What would happen if we introduced infinity in each term of the product, for instance?

$$\left(1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots\right)\left(1 + \frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^3} + \cdots\right)$$

By analogy, we *should* get a sum with exactly one copy of the reciprocal of each number divisible by only 2 and 3, e.g.

$$\sum_{2|n \text{ or } 3|n} \frac{1}{n}.$$

```
@interact
def _(e=(1,[0..6]),f=(2,[0..6])):
    n = 2^e*3^f
    pretty_print(html("You picked
        $%s=2^{%s}3^{%s}$"%(n,e,f)))
    str = '$$'+' + '.join(['\\frac{1}{%s}'%d for d in
        divisors(n)])+'=%s$$'%sum([1/d for d in
        divisors(n)])
    str2 = '$$' + ''.join(['\left('+'+
        '.join(['\\frac{1}{%s^{%s}}'%(p,k) for k in
        [0..e]])+'\\right)' for (p,e) in factor(n)]) +
        '=%s$$'%prod([sum([p^(-k) for k in [0..e]]) for
        (p,e) in factor(n)])
    pretty_print(html(str))
    pretty_print(html("compare to "+str2))
```

There is no reason this wouldn't continue to work for many prime factors.

Because every integer is *uniquely represented* as a product of prime powers (Fundamental Theorem of Arithmetic), this implies that we might multiply out the left-hand side of an *infinite* product of *infinite* sums to get

$$\prod_p \left(1 + \frac{1}{p} + \frac{1}{p^2} + \frac{1}{p^3} + \cdots\right) = \sum_{n=1}^{\infty} \frac{1}{n}.$$

Since each of the multiplied terms on the left is a geometric series, we can simplify the product slighlty to write

$$\prod_p \left(\frac{1}{1-1/p}\right) = \sum_{n=1}^{\infty} \frac{1}{n}.$$

24.2.3 Being careful

So much for Euler's contribution, a very impressive one. The only problem with all this is that both of these things clearly diverge!

Thus we cannot use a simple equality (=) for this discussion. Nonetheless, Euler's intuition is spot on, and we *will* be able to fix this issue quite satisfactorily. For now, we can say is that, in some sense, the harmonic series is also an infinite product:

$$\zeta(1) = \sum_{n=1}^{\infty} \frac{1}{n} \quad \text{“=”} \prod_p \left(\frac{1}{1-1/p}\right) = \prod_p \left(\frac{1}{1-p^{-1}}\right).$$

To make this rigorous, we should start talking about convergence. Recall this informal version of the integral test for series.

Proposition 24.2.2 (Integral test for series convergence)**.** *Assume f is a positive decreasing function going to zero as $x \to \infty$. Then the series $\sum_{i=1}^{n} f(i)$ converges if and only if the integral $\int_1^{\infty} f(x)dx$ converges.*

How does this apply to our situation? The improper integral in this case is

$$\int_1^{\infty} x^{-s}\, dx$$

which evaluates to

$$\left.\frac{-x^{-s+1}}{1-s}\right|_1^\infty = \frac{1}{1-s}\left(1 - \lim_{x\to\infty}\frac{1}{x^{s-1}}\right).$$

For s a real number, this converges precisely when $s > 1$ (since that keeps x in the denominator).

Fact 24.2.3. *The infinite sum $\zeta(s)$ converges for all $s > 1$.*

But why is the (infinite) product equal to this infinite sum too? Is this product even *meaningful*? After all, it is *not* true in general that if a partial product equals a partial sum, then the 'full' sum is the 'full' product.

One has to carefully set up the convergence. If we can show that the product converges to the sum, then *both* will converge. Then it will make sense to say that

$$\zeta(s) = \sum_{n=1}^{\infty}\frac{1}{n^s} = \prod_p\left(\frac{1}{1-p^{-s}}\right)$$

24.3 From Riemann to Dirichlet and Euler

In order to see this (the convergence of the infinite product), let's instead observe our other main example of a sum over divisors equalling a product over primes working. When we compared them for ϕ above, we got

$$\sum_{d|n}\frac{\mu(d)}{d} = \prod_{p|n}\left(1-\frac{1}{p}\right)$$

```
@interact
def _(e=(1,[0..3]),f=(2,[0..3]),g=(0,[0..3])):
    n = 2^e*3^f*5^g
    pretty_print(html("You␣picked␣
        $%s=2^{%s}3^{%s}5^{%s}$"%(n,e,f,g)))
    str = '$$'+'+'.join(['\\frac{%s}{%s}'%(moebius(d),d)
        for d in divisors(n)])+'=%s$$'%sum([moebius(d)/d
        for d in divisors(n)])
    str2 = '$$'+''.join(['\left(1-\\frac{1}{%s}\\right)'%p
        for (p,e) in factor(n)])+'=%s$$'%prod([1-1/p for
        (p,e) in factor(n)])
    pretty_print(html(str))
    pretty_print(html("compare␣to␣"+str2))
```

We could make the powers far higher, or include more primes, and it would still work. Going to both limits, this would lead to the series

$$\sum_{n=1}^{\infty}\frac{\mu(n)}{n^s}.$$

24.3.1 Dirichlet series

We give such series a name. The following definition is *formally*, considered irrespective of things like convergence.

Definition 24.3.1. In general, for an arithmetic function $f(n)$, its **Dirichlet series** is

$$F(s) = \sum_{n=1}^{\infty} \frac{f(n)}{n^s}.$$

Answer the following three questions to see if you understand this definition. (See Exercise 24.7.1.)

- For what arithmetic function is the Riemann zeta function the Dirichlet series?
- What would the Dirichlet series of N be?
- What about the Dirichlet series of I?

Note that this already indicates some level of connection between arithmetic functions. These are connections which may not have been evident otherwise.

24.3.2 Euler products

For our purposes, the very important thing to note about such series is that they often can be expanded as infinite products.

Definition 24.3.2. In general, for an arithmetic function $f(n)$, its Dirichlet series has an **Euler product** the series can be written as an infinite product in the following manner.

$$\sum_{n=1}^{\infty} \frac{f(n)}{n^s} = \prod_p (\text{ a formula involving } f(p) \text{ and } p^s)$$

Example 24.3.3 (Euler product for Riemann zeta function)**.** We have already suggested one for the zeta function:

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_p \left(\frac{1}{1-p^{-s}} \right)$$

Based on the logic of this section, we have a potential new Euler product for the Dirichlet series of the Moebius function:

$$\sum_{n=1}^{\infty} \frac{\mu(n)}{n^s} = \prod_p \left(1 - \frac{1}{p^s} \right) = \prod_p (1-p^{-s})$$

At least, we can consider this wherever it makes sense.

In the next section, we justify more of this discussion, and connect our wonderful results about Dirichlet products of finite arithmetic functions to deep properties of their Dirichlet series.

24.4 Multiplication

24.4.1 Some coincidences

One surprising thing about end of the previous subsection is that the Euler products for the Riemann ζ function and the Dirichlet series of the Möbius function are multiplicative inverses of each other. That is,

$$\prod_p \frac{1}{1-p^{-s}} = 1/\left(\prod_p 1-p^{-s} \right).$$

We can check this numerically as well; in the following examples, we use $s = 2$.

```
sum([moebius(n)/n^2 for n in [1..10000]]).n()
```

```
1/zeta(2).n()
```

They agree up to quite a few digits when we approximate it, so that is a start at reasonability!

Remark 24.4.1. Zeta has interesting values at integers. Recall from our exploration of the average value of σ in Section 20.4 that $\zeta(2) = \frac{\pi^2}{6}$ (though there we just used this as a sum, and didn't call it $\zeta(2)$). Compare this computation.

```
1/(pi^2/6).n()
```

Euler calculated many even values of ζ, which all look like π^{2n} times a rational number (see any description of the so-called Bernoulli numbers). However, it was only in 1978 that $\zeta(3)$ was shown to be *ir*rational. It was then named Apéry's constant after the man who proved this, Roger Apéry.

To compare with the situation for even n, as of this writing it is still only known that *at least one* of the next four odd values $(\zeta(5), \zeta(7), \zeta(9), \zeta(11))$ is irrational[1] . See Wadim Zudilin's website for many links, though this page hasn't been updated for some time.

24.4.2 Multiplication of both kinds

Let's reinterpret this just a little bit. Assuming we can prove that all this makes sense (which we haven't, yet), we have the following two analogous facts.

Fact 24.4.2. *The arithmetic functions u and μ are inverses as arithmetic functions; that is, $u \star \mu = I$.*

The Dirichlet series of these functions are also inverses, as ordinary functions:

$$\prod_p \frac{1}{1-p^{-s}} = 1/\left(\prod_p 1 - p^{-s}\right)$$

Alternately, $\sum_{n=1}^{\infty} \frac{\mu(n)}{n^s} = 1/\zeta(s)$

This analogy is not a coincidence.

Theorem 24.4.3. *Use the following notation:*

- *Take $f(n)$ and $g(n)$ to be two arithmetic functions.*
- *Let $h = f \star g$ be their Dirichlet product.*
- *Let F, G, H be the corresponding Dirichlet series (in the variable s).*

Then if the series F and G converge absolutely *for any particular s, then H converges and $H = FG$ for that s as well.*

[1] And various other similar facts.

Proof. First, we need there is a key fact you may or may not have seen in calculus. Roughly speaking, when series converge absolutely, you can mess around with them with a lot with impunity. (See, for instance, Mertens' Theorem on convergence of Cauchy products; even [C.3.6] doesn't say any more than this in discussing this proof.) Since F and G converge absolutely, we will mess around a lot with the product

$$F(s)G(s) = \sum_{n=1}^{\infty} \frac{f(n)}{n^s} \sum_{m=1}^{\infty} \frac{g(m)}{m^s}$$

In particular, we can group the products by the terms $\frac{f(n)g(m)}{n^s m^s}$ (the same way we did in proving things about $\star$ in Subsection 23.4.3), without loss of equality.

We can further group by when n and m are complementary divisors of the same number. This gives

$$F(s)G(s) = \sum_{d=1}^{\infty} \sum_{nm=d} \frac{f(m)g(n)}{d^s} .$$

Notice that the inner sum is precisely the Dirichlet $\star$ product (except divided by d^s). So we may rewrite this as

$$F(s)G(s) = \sum_{d=1}^{\infty} \frac{(f \star g)(d)}{d^s} .$$

The numerators are the definition of h, so this is just $H(s)$, as desired. □

This is a quite remarkable and deep connection between the discrete/algebraic point of view and the analytic/calculus point of view. It is a shame that this is not exploited more in the standard calculus curriculum, though see [C.5.8] for a very good resource for those who wish to do so.

24.5 Multiplication and Inverses

24.5.1 A series for Euler phi

We can now feel confident applying these amazing facts to calculate the Dirichlet series of ϕ in terms of the Riemann ζ function. We'll see a few facts along the way which could serve as templates for many such investigations.

Fact 24.5.1. *Call P the Dirichlet series for ϕ; it converges for $s > 2$.*

Proof. From Fact 23.3.2, we recall that $\phi \star u = N$. Also, we know from earlier in this chapter that ζ is absolutely convergent for $s > 1$.

Then the Dirichlet series of ϕ is absolutely convergent as well, as

$$0 < \sum_{n=1}^{\infty} \frac{\phi(n)}{n^s} \leq \sum_{n=1}^{\infty} \frac{n}{n^s} = \sum_{n=1}^{\infty} \frac{1}{n^{s-1}}$$

which converges by the integral test if $s > 2$.

Apply Theorem 24.4.3 to the series for ϕ and u; then they multiply to the series for N and it all converges. □

Remark 24.5.2. The series for N may also be written as $\zeta(s-1)$.

We can do even better than this, though, to get a single formula for the series P.

Fact 24.5.3. *The series for ϕ, $P(s)$, evaluates as*

$$P(s) = \sum_{n=1}^{\infty} \frac{\phi(n)}{n^s} = \frac{\zeta(s-1)}{\zeta(s)}$$

Proof. Recall that the Riemann zeta function is just the Dirichlet series for u, and that the series for N (see the previous remark) is $\zeta(s-1)$.

Then the previous proof can be expanded to state $P(s)\zeta(s) = \zeta(s-1)$ for $s > 2$, which suffices to prove the fact. □

We can check this with Sage at any particular point if we wish.

```
sum([euler_phi(n)/n^3 for n in [1..10000]]).n()
```

```
(zeta(2)/zeta(3)).n()
```

24.5.2 A general theorem

It turns out that such Euler products (and hence nice computations like this) show up quite frequently.

Theorem 24.5.4. *If $\sum_{n=1}^{\infty} \frac{f(n)}{n^s}$ converges absolutely and f is multiplicative, then*

$$\sum_{n=1}^{\infty} \frac{f(n)}{n^s} = \prod_p \left(1 + \frac{f(p)}{p^s} + \frac{f(p^2)}{p^{2s}} + \cdots\right).$$

Proof. Doing this is Exercise 24.7.2. We have a proof that Moebius μ's Dirichlet series converges to its Euler product in the next subsection; the proof of this is very similar, just more general. □

24.5.3 A missing step: convergence of Dirichlet series

Before we start using this in the next section, we have to acknowledge there is a missing step thus far. Namely, we haven't demonstrated much about convergence of these series *or* products, much less that they converge to *each other*. Although it is fun to play around, and numerical experimentation will convince you they are very likely, we need more to really use these tools with abandon.

Our goal in this subsection is to prove for the Moebius μ function that its Dirichlet series converges to the Euler product. Proofs for most other such functions (such as the Riemann zeta function) are similar enough to leave more general proofs to a graduate course.

Fact 24.5.5. *For $s > 1$ we have*

$$\sum_{n=1}^{\infty} \frac{\mu(n)}{n^s} = \prod_p \left(1 - \frac{1}{p^s}\right) = \prod_p (1 - p^{-s})$$

Proof. This proof follows [C.1.1] closely. First we will come up with a way to write a partial *product* as a specific sum. Then we will use this to get a precise error between partial products and the infinite sum, and finally bound said error by something going to zero, the final step of which we separate out as an independent claim.

We will begin with the identity we already know as defining μ in Definition 23.1.1:

$$\sum_{d|n} \frac{\mu(d)}{d} = \prod_{p|n} (1 - p^{-1}) .$$

Assuming we multiply these products out through the kth prime, we get

$$\prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right) =$$

$$1 - \frac{1}{p_1} - \frac{1}{p_2} - \cdots + \frac{1}{p_1 p_2} + \frac{1}{p_1 p_3} + \cdots - \frac{1}{p_1 p_2 p_3} - \frac{1}{p_1 p_2 p_4} - \cdots =$$

$$\sum_{n \text{ squarefree and only } p_i | n, 1 \leq i \leq k} \frac{\mu(n)}{n} .$$

This certainly suggests the entire fact is true.

Next, let's introduce the set

$$A_k = \{n \mid n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}, e_i \geq 0\}$$

This is the set of all integers built out of the first k primes. Since $\mu(n) = 0$ unless it has no higher prime powers, then in this notation the big right hand side sum is equal to

$$\prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right) = \sum_{n \text{ squarefree and only } p_i | n, 1 \leq i \leq k} \frac{\mu(n)}{n} = \sum_{n \in A_k} \frac{\mu(n)}{n} .$$

Since the Fundamental Theorem of Arithmetic gives all these relations, I can replace p_i with p_i^s with no harm and write

$$\prod_{i=1}^{k} (1 - p_i^{-s}) = \sum_{n \in A_k} \frac{\mu(n)}{n^s} .$$

Our next step is to get a bound on the *difference* between the infinite product and infinite series,

$$\prod_{i=1}^{k} (1 - p_i^{-s}) - \sum_{n=1}^{\infty} \frac{\mu(n)}{n^s}$$

By the work we just did, this is $\sum_{n \notin A_k} \frac{\mu(n)}{n^s}$. This is the difference between the *infinite* sum and the partial product through the kth prime. Further, we know this error is finite for any given allowable s, because it's bounded by $\pm\zeta$, and ζ converges absolutely for $s > 1$ (recall the comparison test for infinite series).

Let's put absolute values on this error bound:

$$\left| \prod_{i=1}^{k} (1 - p_i^{-s}) - \sum_{n=1}^{\infty} \frac{\mu(n)}{n^s} \right| = \left| \sum_{n \notin A_k} \frac{\mu(n)}{n^s} \right|$$

To get a more explicit bound, we now deduce that any $n \notin A_k$ must be $n > p_k$, since n cannot have any of the first k primes as factors. Armed with this, the following Claim 24.5.6 will finish the proof:

$$\left| \sum_{n \notin A_k} \frac{\mu(n)}{n^s} \right| \leq \sum_{n > p_k} \frac{1}{n^s}$$

The latter error $\sum_{n>p_k} \frac{1}{n^s}$ must go to zero as $k \to \infty$, since this is the tail of a convergent infinite series. That means that the partial products converge to the series; we know that is finite, so everything converges and we have our Euler product for this Dirichlet series! □

Claim 24.5.6. *With all notation as in Fact 24.5.5, we have*

$$\left| \sum_{n \notin A_k} \frac{\mu(n)}{n^s} \right| \leq \sum_{n \notin A_k} \left| \frac{\mu(n)}{n^s} \right| \leq \sum_{n>p_k} \left| \frac{\mu(n)}{n^s} \right| \leq \sum_{n>p_k} \frac{1}{n^s}$$

Proof. The first inequality follows if we can put the absolute value inside the summation. This is an extended triangle inequality, which is only legitimate if the final thing converges; however, we already showed this at the end of the proof of the main fact.

The second inequality is due to the fact that any $n \notin A_k$ must be bigger than p_k, so the set of *all* integers above p_k would just yield a bigger sum (since all terms are now positive after the first step).

The final inequality uses that $\mu = 0, 1, -1$ always. □

24.6 Four Facts

We are now ready to work with four applied facts which we can prove, using these tools. Some have other types of proofs, but number theory combined with calculus really provides a unified framework for a huge number of problems.

- In Subsection 24.6.1, we will show that the probability that a random integer lattice point is 'visible' from the origin is $\frac{6}{\pi^2}$; this is Proposition 24.6.1.
- In Subsection 24.6.2, we see that the Dirichlet series for $f(n) = |\mu(n)|$ is $\zeta(s)/\zeta(2s)$; this is Proposition 24.6.2.
- In Subsection 24.6.4, we compute the average value of $\phi(n)$ to be $\frac{3n}{\pi^2}$; this is Proposition 24.6.4.
- In Subsection 24.6.3, we see that the **prime harmonic series** sum $\sum_{n=1}^{\infty} \frac{1}{p_n}$ diverges, with p_n the nth prime; this is Proposition 24.6.3.

24.6.1 Random integer lattice points

The following graphic will indicate what it means to have a point visible from the origin; is there a point *directly* between it and the origin or not? To rephrase, what is the probability that a point in the integer lattice has a line connecting the point to the origin that does not hit any other point? (We will explicitly avoid any discussion of why such infinitary probabilities are defined in this introductory text.)

Note that the probabilities estimated will vary wildly. Especially at a prime distance one should expect the computed probability to be higher than the theoretical one; why?

```
@interact
def _(viewsize=(5,[3..25])):
    var('x,y')
    P=Graphics()
```

```
grid_pts = [[i,j] for i in [-viewsize..viewsize] for j
    in [-viewsize..viewsize]]
P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
lattice_pts = [coords for coords in grid_pts if
    (gcd(coords[0],coords[1])==1)]
P += points(lattice_pts, rgbcolor = (0,0,1),
    pointsize=10)
linesegs=[line([[0,0],[spot[0],spot[1]]],
    rgbcolor=(1,0,0), linestyle="--",thickness=.5) for
    spot in lattice_pts]
for object in linesegs:
    P += object
show(P, figsize = [5,5], xmin = -viewsize, xmax =
    viewsize, ymin = -viewsize, ymax = viewsize,
    aspect_ratio=1)
pretty_print(html("Probability␣in␣view␣is␣$\\approx␣
    %s$"%( Integer(len(lattice_pts)) /
    Integer(len(grid_pts))).n()))
pretty_print(html("Theoretical␣probability␣is␣
    $1/\zeta(2)\\approx␣%s$"%(1/zeta(2)).n()))
```

It should be pretty clear from the picture that if x and y have a nontrivial common divisor d then, $\left(\frac{x}{d}, \frac{y}{d}\right)$ is right on the line of sight from the origin to (x, y) so that it is blocked off. This is most clearly so for $d = \gcd(x, y)$, so the following fact is the same thing as asking for the probability that two randomly chosen integers are relatively prime.

Proposition 24.6.1. *The chances that a random integer lattice point is visible from the origin is* $\frac{6}{\pi^2}$.

Proof. We will prove the statement about coprime random integers, or at least we will prove as much of it as we can without discussing infinite combinations of independent chances. We will also make an assumption about distribution of primes to simplify the proof; one can consider this a sketch, if necessary.

First, we know that $\gcd(x, y) = 1$ is true precisely if x and y are never simultaneously congruent to zero modulo p, for any prime p. (If there were such a p, of course it would be a divisor; by the Fundamental Theorem of Arithmetic we need only consider primes.)

For any given prime p, the chances that two integers will *both* be congruent to zero is

$$\left(\frac{1}{p}\right)\left(\frac{1}{p}\right).$$

This works because the probabilities are independent, since p is fixed, so we can just multiply probabilities.

Hence the probability that at least one of x or y will *not* be divisible by p is

$$1 - \left(\frac{1}{p}\right)\left(\frac{1}{p}\right) = 1 - \frac{1}{p^2} = 1 - p^{-2}.$$

(This may remind you of the so-called birthday problem in probability.)

Now comes our assumption. We will suppose that if $p \neq q$ are both prime, then the probability that any given integer is divisible by p has nothing to do with whether it is divisible by q. (Such properties are not true in general; if n is divisible by 4 it has a 100% likelihood of being divisible by 2, while if n is prime, it has almost no chance of being even.)

In such a case the probabilities are independent, so that (even in this infinitary case)

$$\prod_p (1-p^{-2}) = 1/\prod_p \frac{1}{1-p^{-2}} = 1/\zeta(2) = \frac{6}{\pi^2}\,.$$

□

This implies that a random pair of integers, selected from a large enough bound, will be relatively prime about 61% of the time.

```
(6/pi^2).n()
```

24.6.2 Dirichlet for the absolute Moebius

Proposition 24.6.2. *The Dirichlet series for* $|\mu(n)|$ *is* $\zeta(s)/\zeta(2s)$.

Proof. With all the tools we've gained, the proof is nearly completely symbolic at this point!

First, we have the following from the definition of Moebius in Definition 23.1.1, or from Fact 24.5.5:

$$\sum_{n=1}^{\infty} \frac{|\mu(n)|}{n^s} = \prod_p \left(1+\frac{1}{p^s}\right)\,.$$

Next, let us write $x = \frac{1}{p^s}$; then we can use the basic identity $(1+x) = \frac{1-x^2}{1-x}$ to rewrite the right-hand side as

$$\prod_p \left(1+\frac{1}{p^s}\right) = \frac{\prod_p \left(1-\frac{1}{p^{2s}}\right)}{\prod_p \left(1-\frac{1}{p^s}\right)}\,.$$

Now we just invert both numerator and denominator to get familiar friends:

$$\frac{\prod_p \left(1-\frac{1}{p^{2s}}\right)}{\prod_p \left(1-\frac{1}{p^s}\right)} = \frac{\prod_p 1/(1-\frac{1}{p^s})}{\prod_p 1/(1-\frac{1}{p^{2s}})}$$

which means the sum will be $\zeta(s)/\zeta(2s)$. □

Let's try this out computationally.

```
@interact
def _(s=[2,3,4,5]):
    S = sum([abs(moebius(n))/n^s for n in [1..10000]]).n()
    S2 = zeta(RR(s))/zeta(2*RR(s))
    pretty_print(html("The approximation is $%s$ while the
        zeta computation is $%s$."%(S,S2)))
```

24.6.3 The prime harmonic series

The divergence of the series created from the reciprocals of prime numbers is not necessarily a particularly obvious fact. Certainly it diverges much, much slower than the harmonic series (recalled before Definition 20.3.2), which already diverges very slowly.

```
@interact
def _(n=[10,100,1000,10000,100000,1000000]):
    out = sum([RealField(100)(1/p) for p in
        primes_first_n(n)])
    pretty_print(html("The sum of the reciprocals of the
        first $%s$ primes is $\\approx %s$"%(n,out)))
```

This proof doesn't actually use Dirichlet series, but has in common with them themes of convergence and estimation, so it is appropriate to include here.

Proposition 24.6.3 (Prime harmonic series diverges)**.** *Let p_n be the nth prime. Then the following series, which we call the* ***prime harmonic series****, diverges:*

$$\sum_{n=1}^{\infty} \frac{1}{p_n}$$

Proof. This is a fairly expanded form of the proofs in [C.1.1, Theorem 9.2] and [C.3.6, Theorem 1.13].

As with many other occasions to prove series divergence, we will focus on the 'tail's beyond a point that keeps getting further out. In this case, we'll choose the 'tail' beyond the first k primes,

$$T = \sum_{n>k} \frac{1}{p_n} .$$

By examining certain terms in this, and assuming (falsely) that it can be made finite, we will obtain a contradiction.

First, let's consider numbers of the form

$$p_1 p_2 p_3 \cdots p_k r + 1 = p_k\# \cdot r + 1$$

(Recall the primorial notation from Definition 22.2.6.) Such a number cannot be divisible by any of those first k primes, so by the Fundamental Theorem of Arithmetic any number like $p_k\# \cdot r + 1$ may be factored as

$$p_{n_1} p_{n_2} \cdots p_{n_\ell} ,$$

where all $n_i > k$ (some may be repeated).

Return to the 'tail'. Since this $p_k\# \cdot r + 1$ factors with ℓ factors, then *somewhere* in the ℓth power of the 'tail' we have the following term:

$$T^\ell = \left(\sum_{n>k} \frac{1}{p_n} \right)^\ell = \frac{1}{p_1 p_2 p_3 \cdots p_k r + 1} + \cdots .$$

Now assume that in fact the prime harmonic series converges; we will derive a contradiction.

First, for some k, the 'tail' T is less than $\frac{1}{2}$, i.e. $T = \sum_{n>k} \frac{1}{p_n} < \frac{1}{2}$. Since each term is positive, $T > 0$ and a geometric series involving the ℓth powers of T is very precisely bounded:

$$0 \leq \sum_{\ell=1}^{\infty} T^{\ell} = \sum_{\ell=1}^{\infty} \left(\sum_{n>k} \frac{1}{p_n} \right)^{\ell} \leq \sum_{\ell=1}^{\infty} \frac{1}{2^{\ell}} = 2\,.$$

Now we bring in the first discussion in this proof; every single term of the form $\frac{1}{p_1 p_2 p_3 \cdots p_k r+1}$ will appear *somewhere* within this sum of the ℓth powers, though naturally ℓ in each case will depend heavily upon r.

So the series of reciprocals of *just these* special terms is bounded.

$$0 < \sum_{r=1}^{\infty} \frac{1}{p_1 p_2 p_3 \cdots p_k r + 1} \leq \sum_{\ell=1}^{\infty} \left(\sum_{n>k} \frac{1}{p_n} \right)^{\ell} \leq 2\,.$$

A bounded series of all positive number should converge (e.g. by comparison).

Here comes the contradiction. The same series is bounded below as follows, *for each* integer k.

$$\sum_{r=1}^{\infty} \frac{1}{p_1 p_2 p_3 \cdots p_k r + 1} > \sum_{r=1}^{\infty} \frac{1}{p_1 p_2 p_3 \cdots p_k r + p_1 p_2 p_3 \cdots p_k}$$

$$= \frac{1}{p_1 p_2 p_3 \cdots p_k} \sum_{r=1}^{\infty} \frac{1}{r+1}$$

This series certainly *diverges*, as a multiple of the tail of the harmonic series!

Since no matter how big k is (and hence how far out in the 'tail' we go) we report that a certain series both converges and diverges, we have a contradiction. Hence our original assumption that we could choose k to make T finite was false, and the prime harmonic series must diverge! □

24.6.4 The average value of Euler phi

Finally, here is a really nice result to end with. Thinking about the average value of ϕ will put together many themes from this text.

You may recall Exercise 20.6.15 where you were asked to conjecture regarding this question. As there, it's useful here to try to graph the average values first; here I have incuded the correct long-term average as well.

```
def L(n):
    ls = []
    out = 0
    for i in range(1,n+1):
        out += euler_phi(i)
        ls.append((i,out/i))
    return ls

@interact
def _(n=100):
    P = line(L(n))
    show(P+plot(3/pi^2*x,(x,0,n), color='black',
        linestyle="--"))
    pretty_print(html("Blue␣is␣the␣average␣value␣of␣
        $\phi$$"))
    pretty_print(html("Black␣is␣$\\frac{3}{\pi^2}␣x$"))
```

Before formally proving this, let's look at a significant picture for conceptualizing the proof. This is similar to what we used for the average of τ and σ in Chapter 24.

```
@interact
def _(n=(6,range(2,50))):
    viewsize=n+1
    g(x)=1/x
    P=Graphics()
    P += plot(n*g,(x,0,n+1))
    grid_pts = [[i,j] for i in [1..viewsize] for j in
        [1..viewsize]]
    P += points(grid_pts,rgbcolor=(0,0,0),pointsize=2)
    lattice_pts = [coords for coords in grid_pts if
        (coords[0]*coords[1]<=n)]
    for thing in lattice_pts:
        P += text(moebius(thing[1])*thing[0],
            thing,rgbcolor=(0,0,0))
    show(P,ymax=viewsize,aspect_ratio=1)
```

The text at each lattice point is the value of horizontal coordinate, multiplied by a factor of Moebius of the vertical coordinate.

We will crucially use these two facts in the proof, based loosely on [C.3.6]; see it or [C.1.8] for much more related material – the latter is unusual in *starting* its discussion of averages with this example!

- From above (e.g. Fact 24.4.2),

$$\sum_{n=1}^{\infty} \frac{\mu(n)}{n^2} = \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$$

- From the previous chapter (e.g. Fact 23.3.2),

$$\phi = \mu \star N \Rightarrow \phi(n) = \sum_{d|n} \mu(d) \frac{n}{d}$$

Proposition 24.6.4. *The long-term average value of ϕ is given by*

$$\lim_{n\to\infty} \frac{\frac{1}{n}\sum_{k=1}^{n} \phi(k)}{\frac{3}{\pi^2} n} = 1$$

Proof. Consider the summation function for ϕ, $\sum_{k=1}^{n} \phi(n)$. As in Chapter 20, we will think of it as summing things up in two different ways.

In particular, look at the summation once we have replaced with the Moebius sum inside:

$$\sum_{k=1}^{n} \phi(k) = \sum_{k=1}^{n} \sum_{d|k} \mu(d) \frac{k}{d}$$

Now instead of considering it as a sum over divisors for each k, we can think of it as summing for each divisor over the various hyperbolas $xy = k$. This yields

$$\sum_{k=1}^{n} \sum_{d|k} \mu(d) \frac{k}{d} = \sum_{d=1}^{n} \mu(d) \left(\sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} k \right)$$

Now let's examine the terms of this sum. We will several times use Landau notation as in Definition 20.1.1.

Knowing about the sum of the first few consecutive integers (also used at the end of Subsection 20.3.2), we see that

$$\left(\sum_{k=1}^{\lfloor \frac{n}{d} \rfloor} k\right) = \frac{1}{2}\left(\frac{n}{d}\right)^2 + O\left(\frac{n}{d}\right) .$$

If we plug that in the double sum, we get

$$\sum_{k=1}^{n} \phi(n) = \frac{1}{2}n^2 \sum_{d=1}^{n} \frac{\mu(d)}{d^2} + nO\left(\sum_{d=1}^{n} \frac{\mu(d)}{d}\right) .$$

Let's examine this.

- The first term goes to $\frac{6}{\pi^2}$ as $n \to \infty$. Further, its error is $O(1/n)$, because $\mu(n)/n^2 < 1/n^2$ and $\int x^{-2}\,dx = -x^{-1}$.
- The second term is certainly $O(n\log(n))$, since it is n times a sum which is less than something $O(\log(n))$ (namely, ζ).

Plugging everything in, we get that

$$\sum_{k=1}^{n} \phi(n) = \frac{1}{2}n^2\frac{6}{\pi^2} + O(\text{ stuff less than } n^2)$$

Dividing by n and taking the limit, we get the asymptotic result.

$$\lim_{n\to\infty} \frac{\frac{1}{n}\sum_{k=1}^{n}\phi(n)}{\frac{3}{\pi^2}n}$$

$$= \lim_{n\to\infty} \frac{\frac{1}{2}\frac{n^2}{n}\frac{6}{\pi^2} + \frac{1}{n}O(\text{stuff less than } n^2)}{\frac{3}{\pi^2}n}$$

$$= \lim_{n\to\infty} \frac{\frac{3}{\pi^2}n + O(\text{ stuff less than } n)}{\frac{3}{\pi^2}n} = 1 .$$

□

24.7 Exercises

1. Write down your answers to the three questions about the definition of Dirichlet series after Definition 24.3.1.

2. Prove Theorem 24.5.4 in full generality, following that of Fact 24.5.5. (This is a good technical exercise in convergence.)

3. Learn more about the notion of **zero density** (recall Subsection 22.2.2). Then find other sets like $P = \{$ primes $\}$ such that the sum of the reciprocals diverges, but the set has zero density in the integers.

4. Use Sage or other computational tools to conjecture the rate of *growth* of the function

$$f(x) = \sum_{p\leq x} \frac{1}{p}$$

where p is of course prime. Hint: Typically one needs lumber to print a book, such as [C.3.5] (but don't peek there until you're really stuck!).

5. Recall ω from Definition 23.3.3 and $f(x)$ from the previous question. Confirm numerically that the average value to x (in the sense of Chapter 20) of ω is about the same as the size of $f(x)$. Give a reason why $\sum_{p \leq x} \frac{1}{p}$ should be related to $\sum_{n \leq x} \omega(n)$.

Chapter 25

Further Up and Further In

If you survived this book, hooray! You made it. You did a great job making it through a whole arc of number theory accessible at the undergraduate level.

Although we really did see a lot of the problems out there, there are many we *did not* see all the way through. We *were* able to prove some things about them. Here are just a few problems we started touching on.

- Solving higher-degree polynomial congruences, like $x^3 \equiv a \pmod{n}$. (Chapter 7)
- Knowing how to find the *first* integer point on hard things like the Pell (hyperbola) equation $x^2 - ny^2 = 1$. (Chapter 15)
- Writing a number not just in terms of a sum of squares, but a sum of cubes, or a sum like $x^2 + 7y^2$. (Chapter 14)
- The Prime Number Theorem, and finding ever better approximations to $\pi(x)$. (Chapter 21)

It's this last one we will focus on in this extended postscript, for it takes us to the very frontiers of the deepest questions about numbers.

25.1 Taking the PNT Further

Recall Gauss' approximating function for $\pi(x)$, the logarithmic integral function (Definition 21.2.2). Let's remind ourselves how it did.

```
@interact
def _(n=(1000,(1000,10^6))):
    P = plot(prime_pi,n-1000,n, color='black',
        legend_label='$\pi(x)$')
    P += plot(Li,n-1000,n, color='green',
        legend_label='$Li(x)$')
    show(P)
```

It wasn't too bad of an estimate. But, as mathematicians, we hope we could get a little closer. At that time, among several other things, we tried this fairly weird amended function:

$$Li(x) - \frac{1}{2}Li(\sqrt{x}) \ .$$

And this was indeed a better approximation.

```
@interact
def _(n=(1000,(1000,10^6))):
   P = plot(prime_pi,n-1000,n, color='black',
       legend_label='$\pi(x)$')
   P += plot(Li,n-1000,n, color='green',
       legend_label='$Li(x)$')
   P += plot(lambda x: Li(x) - .5*Li(sqrt(x)), n-1000,n,
       color='red',
       legend_label='$Li(x)-\\frac{1}{2}Li(\sqrt{x})$')
   show(P)
```

So one might think one could keep adding and subtracting

$$\frac{1}{n}Li(x^{1/n})$$

to get even closer, with this start to the pattern.

As it turns out, that is *not* quite the right pattern. In fact, the minus sign comes from $\mu(2)$, not from $(-1)^{2+1}$, as usually is the case in series which begin by alternating!

```
@interact
def _(n=(1000,(1000,10^6)),k=(3,[1..10])):
   P = plot(prime_pi,n-1000,n, color='black',
       legend_label='$\pi(x)$')
   P += plot(Li,n-1000,n, color='green',
       legend_label='$Li(x)$')
   F = lambda x: sum([Li(x^(1/j))*moebius(j)/j for j in
       [1..k]])
   P += plot(lambda x: Li(x) - .5*Li(sqrt(x)),n-1000,n,
       color='red',
       legend_label='$Li(x)-\\frac{1}{2}Li(\sqrt{x})$')
   P += plot(F,n-1000,n, color='blue',
       legend_label='$\sum_{j=1}^{%s} \\frac{\mu(j)}{j}
       Li(x^{1/j})$'%k)
   show(P)
```

This set of approximations doesn't really add a lot of accuracy beyond $k = 3$. In fact, at $x = 1000000$, taking the approximation with the sum $\sum_{j=1}^{3} \frac{\mu(j)}{j} Li(x^{1/j})$ is essentially the same as going all the way to infinity in $\sum_{j=1}^{\infty} \frac{\mu(j)}{j} Li(x^{1/j})$, and both of these are clearly not integers. This alone will not yield a *computable, exact* formula for $\pi(x)$. So here are some questions we might raise.

- So where does the Moebius μ in that approximation come from anyway?
- What else is there to the error

  $$|\pi(x) - Li(x)|$$

  since this one wasn't enough?
- Are there connections with things *other* than just $\pi(x)$?
- What does this have to do with winning a million dollars?

25.2 Improving the PNT

Let's look at a table of some of these results.

```
@interact
def _(k=(3,[2..11])):
    F = lambda x: sum([Li(x^(1/j))*moebius(j)/j for j in
        [1..k]])
    T = [['$i$', '$\pi(i)$', '$Li(i)$', '$\pi(i)-Li(i)$',
        '$\pi(i)-\sum_{j=1}^{%s} \\frac{\mu(j)}{j}
        Li(x^{1/j})$'%k]]
    for i in [100000,200000..1000000]:
        T.append([i, prime_pi(i), Li(i).n(digits=7),
            (prime_pi(i)-Li(i)).n(digits=4),
            (prime_pi(i)-F(i)).n(digits=4)])
    pretty_print(html(table(T,header_row = True, frame =
        True)))
```

This table shows the errors in Gauss' and our new estimates for every hundred thousand up to a million. Clearly Gauss is not exact, but the other error is not always perfect either.

After the Prime Number Theorem was proved, mathematicians wanted to get a better handle on the remaining *error* between the log integral and $\pi(x)$. In particular, the Swedish mathematician Helge Von Koch made a very interesting contribution in 1901.

Conjecture 25.2.1. *The error in the PNT is less than*

$$\frac{1}{8\pi}\sqrt{x}\log(x)\ .$$

This seems to work, broadly speaking.

```
@interact
def _(n=(1000,(1000,10^6))):
   P = plot(prime_pi,n-1000,n, color='black',
       legend_label='$\pi(x)$')
   P += plot(Li,n-1000,n, color='green',
       legend_label='$Li(x)$')
   P += plot(lambda x: Li(x) -
       1/(8*pi)*sqrt(x)*log(x),n-1000,n,
       color='blue',linestyle='--', legend_label="Von Koch
       error estimate")
   show(P)
```

Given the data in this graphic, the conjecture seems plausible, if not even improvable (though remember that Li and π switch places infinitely often, see Fact 21.2.3!). Of course, a conjecture is not a theorem, but luckily Von Koch had one of those as well.

Theorem 25.2.2. *The truth of the error estimate*

$$|\pi(x) - Li(x)| \leq \frac{1}{8\pi}\sqrt{x}\log(x)$$

for the prime number theorem is equivalent to saying that $\zeta(s)$ equals zero precisely where Riemann thought it would be zero in 1859 (see Conjecture 25.3.2).

This may seem like an odd statement. After all, ζ is just about reciprocals of all numbers, and can't directly measure primes. (And what do I mean by "thought it would be"?) But in fact, the original proofs of the PNT also used the ζ function in essential ways. So Von Koch was just formalizing the exact estimate it could give us on the error.

25.3 Toward the Riemann Hypothesis

Riemann, though, was after bigger fish. He didn't just want an error term. He wanted an *exact* formula for $\pi(x)$, one that could be *computed*. Computed by hand, or by machine, if such a machine came along, as close as one pleased. And this is where $\zeta(s)$ becomes important, because of the Euler product formula:

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_p \frac{1}{1-p^{-s}}$$

Somehow ζ *does* encode everything we want to know about prime numbers. And Riemann's paper, "On the Number of Primes Less Than a Given Magnitude", is the place where this magic really does happen. (The paper is also available in translation in the appendix of [C.3.4].) Seeing just how it happens is our goal to close the book.

We'll begin by plotting ζ, to see what's going on. As you can see, $\zeta(s)$ doesn't seem to hit zero very often. Maybe for negative s ...

```
plot(zeta,-10,10,ymax=10,ymin=-1)
```

25.3.1 Zeta beyond the series

Wait a minute! What was that plot? Shouldn't ζ diverge if you put negative numbers in for s? (Recall our definition in Definition 24.2.1.) After all, then for $s = -1$ we'd get things like

$$\sum_{i=1}^{\infty} n$$

and somehow I don't think that converges.

But it turns out that we can evaluate $\zeta(s)$ for nearly any *complex* number s we desire. The first graphic below color-codes where each complex number lands by matching it to the color in the second graphic.

```
graphics_array([complex_plot(zeta, (-20,20),
    (-20,20)),complex_plot(lambda z: z,
    (-3,3),(-3,3))]).show(figsize=[8,8])
```

The important point isn't the picture itself, but *that* there is a picture. To wit, ζ can be defined for (nearly) any complex number as input. Why would that be the case? One way to see that we could define this function for complex values comes by trying to define each term $\frac{1}{n^s}$ in $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$ more precisely.

Suppose we let s be a complex number, using the long-standing notational convention

$$s = \sigma + it$$

Then we can rewrite this term as

$$\frac{1}{n^s} = n^{-s} = e^{-s\log(n)} = e^{-(\sigma+it)\log(n)} = e^{-\sigma\log(n)}e^{-it\log(n)}$$

Now we use a fact you may remember from calculus, which is very easy to prove with Taylor series. (See Exercise 25.9.1):

$$e^{ix} = \cos(x) + i\sin(x)$$

Applying this, we get

$$\frac{1}{n^s} = e^{-\sigma\log(n)}e^{-it\log(n)} = n^{-\sigma}\left(\cos(t\log(n)) - i\sin(t\log(n))\right)$$

Using this analysis, if $\sigma > 1$, since cos and sin always have absolute value less than or equal to one, we still have the same convergence properties as with regular series. So if we take the imaginary and real parts separately, we can rewrite

$$\zeta(s) = \sum_{n=1}^{\infty}\frac{1}{n^s} = \sum_{n=1}^{\infty}\frac{\cos(t\log(n))}{n^s} + i\sum_{n=1}^{\infty}\frac{\sin(t\log(n))}{n^s}$$

That doesn't explain the part of the complex plane to the left of $\sigma = 1$ of the picture above. All I will say is that it is possible to extend ζ there, and Riemann did it. (In fact, Riemann is largely responsible for advanced complex analysis.) As an example, $\zeta(-1) = -\frac{1}{12}$, which is *very* close to saying that

$$\zeta(-1) = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + \cdots = -\frac{1}{12}$$

```
zeta(-1)
```

```
-1/12
```

Investigate further whether this has any meaning in Exercise 25.9.2.

25.3.2 Zeta on some lines

Let's get a sense for what the ζ function looks like. First, observe a three-dimensional plot of its absolute value for σ between 0 and 1 (which will turn out to be all that is important for our purposes).

```
plot3d(lambda x,y: abs(zeta(x+i*y)),(0,1),(-20,20),
    plot_points=100)+plot3d(0,(0,1),(-20,20),
    color='green',alpha=.5)
```

To get a better idea of what happens, we compare two plots. One is a one-dimensional plot of $|\zeta|$ for different inputs with the same σ. On the other side is the two-dimensional colored complex plot of $\zeta(\sigma + it)$, where σ is the real part, chosen by you, and then we plot t out as far as requested. The line which we are viewing on the complex plane in the first graphic is dashed in the second one.

Remark 25.3.1. It is not really possible to fully visualize a complex function of complex input. So we often pick some line in the complex plane, such as where the real part equals 1 (sort of like $x = 1$) or where the imaginary part equals 1 (sort of like $y = 1$); then we either treat this as input to a parametric curve, or similarly look at the output and in one way or another reduce it to one real number, and plot this as normal.

```
@interact
def _(sig=slider(.01, .99, .01, 0.5, label='\(\sigma\)'),
    end=slider(2,100,1,40, label='end␣of␣\(t\)')):
    p = plot(lambda t: abs(zeta(sig+t*i)), -end,end,
        rgbcolor=hue(0.7))
    q = complex_plot(zeta,(0,.99),(-end,end),
        aspect_ratio=1/end) + line([(sig,-end),(sig,end)],
        linestyle='--')
    show(graphics_array([p,q]),figsize=[5,3])
```

You'll notice that the only places the function has absolute value zero (which means the only places it hits zero) are when $\sigma = 1/2$.

Another way to visualize ζ in a useful way is with the parametric graph of each vertical line in the complex plane as mapped to the complex plane. You can think of this as where an infinitely thin slice of the complex plane is "wrapped" to.

```
@interact
def _(sig=slider(.01, .99, .01, 0.5, label='\(\sigma\)')):
    end=30
    p = parametric_plot((lambda t: zeta(sig+t*i).real(),
        lambda t: zeta(sig+t*i).imag()), (0,end),
        rgbcolor=hue(0.7),plot_points=300)
    q = complex_plot(zeta,(0,.99),(0,end),
        aspect_ratio=1/end) + line([(sig,0),(sig,end)],
        linestyle='--')
    show(graphics_array([p,q]), figsize=[5,3])
```

This image is reasonably famous, because the *only* time the curve seems to hit the origin at all is precisely at $\sigma = 1/2$, and at $\sigma = 1/2$ the curve seems to hit the origin *lots* of times. For any other σ the curve just misses the origin, somehow.

Now it's true that ζ is also zero at negative even integer input, but these are well understood. The pictures demonstrate the mysterious part. And so we have the following crucial question – where is $\zeta(s) = 0$?

Conjecture 25.3.2 (Riemann Hypothesis)**.** *All the zeros of* $\zeta(s) = \zeta(\sigma + it)$ *where* $t \neq 0$ *are on* $\sigma = 1/2$*.*

The importance of this problem is evidenced by it having been selected as one of the seven Millennium Prize problems by the Clay Math Institute (each holding a million-dollar award), as well as having many recent popular books devoted to it. In what follows we will loosely follow the very interesting exposition of *Prime Obsession* by John Derbyshire, [C.3.1].

25.4 Connecting to the Primes

The last few sections of this final chapter are devoted to seeing why the Riemann Hypothesis might be related to the distribution of prime numbers. For motivation, think of Von Koch's result Theorem 25.2.2 connecting the RH to a bound on the error between $\pi(x)$ and the log integral. Our goal is more detailed, however.

We'll pursue this connection in three steps.

1. Our first step is to see the connection between $\pi(x)$ and $\mu(n)$ (25.4.1).

2. Then we'll see the connection between these and ζ (25.5).
3. Finally, we'll see how the zeros of ζ come into play (25.6).

25.4.1 Connecting to Moebius

Let's begin by defining a new function.

```
def J(x):
    end = floor(log(x)/log(2))
    out = 0
    for j in [1..end]:
        out += 1/j*prime_pi(x^(1/j))
    return out
```

```
L1 = [(n,J(n)) for n in [1..20]]
L2 = [(n,J(n)) for n in [1..150]]
graphics_array([plot_step_function(L1),
    plot_step_function(L2)]).show(figsize=[10,5])
```

Riemann called the function above f. Following [C.3.4] and [C.3.1], we will call it $J(x)$. It is very similar to $\pi(x)$ in its definition, so it's not surprising that it looks similar.

Definition 25.4.1. We define

$$J(x) = \pi(x) + \frac{1}{2}\pi(\sqrt{x}) + \frac{1}{3}\pi(\sqrt[3]{x}) + \frac{1}{4}\pi(\sqrt[4]{x}) + \cdots = \sum_{n=1}^{\infty} \frac{1}{n}\pi\left(x^{1/n}\right)$$

This looks like it's an infinite sum, but for any given x, it is finite. For instance, let's calculate $J(20)$:

$$J(20) = \pi(20) + \frac{1}{2}\pi(\sqrt{20}) + \frac{1}{3}\pi(\sqrt[3]{20}) + \frac{1}{4}\pi(\sqrt[4]{20}) = 8 + \frac{2}{2} + \frac{1}{3} + \frac{1}{4} = 9\frac{7}{12}$$

because $\sqrt[5]{20} \approx 1.8$ and $\pi(\sqrt[5]{20}) \approx \pi(1.8) = 0$, so the sum ends there, and we can see that on the graph.

Okay, so we have this new function. Yet another arithmetic function. So what?

Ah, but what have we been doing to all our arithmetic functions to see what they can do, to get formulas for them? We've been *Moebius inverting* them, naturally! (Recall Section 23.2.) In this case, Moebius inversion could be really great, since it would give us information about the thing being added, which is the all-important $\pi(x)$.

The only thing standing in our way is that

$$J(x) = \sum_{n=1}^{\infty} \frac{1}{n}\pi\left(x^{1/n}\right)$$

is not a sum over divisors. But it turns out that, just like when we took the limits of the sum over divisors $\sum_{d|n} \frac{1}{d}$, we got $\sum_{n=1}^{\infty} \frac{1}{n}$, we can do the same thing with Moebius inversion.

Fact 25.4.2. *If* $\sum_{n=1}^{\infty} f(x/n)$ *and* $\sum_{n=1}^{\infty} g(x/n)$ *both converge absolutely, then*

$$g(x) = \sum_{n=1}^{\infty} f(x/n) \iff f(x) = \sum_{n=1}^{\infty} \mu(n) g(x/n)\ .$$

We can use this by setting $g = J$ with $f(x/n) = \frac{1}{n}\pi\left(x^{1/n}\right)$. Applying this, we achieve a very important result writing $\pi(x)$ in terms of J.

$$\pi(x) = \sum_{n=1}^{\infty} \mu(n)\frac{J(x^{1/n})}{n} = J(x) - \frac{1}{2}J(\sqrt{x}) - \frac{1}{3}J(\sqrt[3]{x}) - \frac{1}{5}J(\sqrt[5]{x}) + \frac{1}{6}J(\sqrt[6]{x}) + \cdots \tag{25.4.1}$$

Remark 25.4.3. If that last use of Moebius inversion looked a little sketchy, it does to me too, but I cannot find a single source where it's complained about that $f(x/n) = \frac{1}{n}\pi\left(x^{1/n}\right)$ is really a function of x and n, not just x/n. In any case, the result is correct, via a somewhat different explanation of this version of inversion in a footnote in Edwards' discussion of this matter in [C.3.4].

25.5 Connecting to Zeta

25.5.1 Turning the golden key

Now, this looks just as hopeless as before. How is J going to help us calculate π, if we can only calculate J in terms of π anyway?

Here is where Riemann "turns the Golden Key", as Derbyshire puts it. Because ζ has an Euler product over the *set of primes*, we can just possibly connect it to each prime. It turns out this will in fact connect ζ to J. This is the goal of the rest of the current section.

In the next section, we will see how the zeros of ζ give us an *exact* formula for J; then we will finally plug J back into the Moebius-inverted formula for π to get an exact formula for π in Section 25.7. Here is a plot of that formula, as a foretaste.

```
var('y')
import mpmath
L = lcalc.zeros_in_interval(10,150,0.1)
n=100
P = plot(prime_pi,n-50,n, color='black',
    legend_label='$\pi(x)$')
P += plot(Li,n-50,n, color='green', legend_label='$Li(x)$')
G = lambda x: sum([mpmath.li(x^(1/j))*moebius(j)/j for j
    in [1..3]])
P += plot(G,n-50,n,color='red', legend_label =
    '$\sum_{j=1}^{%s} \\frac{\mu(j)}{j} Li(x^{1/j})$'%3)
F = lambda x: sum([(mpmath.li(x^(1/j))-log(2) +
    numerical_integral( 1/(y*(y^2-1)*log(y)),
    x^(1/j),oo)[0] )*moebius(j)/j for j in [1..3]]) -
    sum([(mpmath.ei(log(x)*((0.5+l[0]*i)/j)) +
    mpmath.ei(log(x)*((0.5-l[0]*i)/j))).real for l in L for
    j in [1..3]])
P += plot(F,n-50,n,color='blue', legend_label='Really good
    estimate',plot_points=50)
show(P)
```

We can see above that this has the potential to be a *very* good approximation, even given that I did limited calculations here. The most interesting thing is the gentle waves you should see; this is quite different from the other types of approximations we had, and seems to have the potential to mimic the more abrupt nature of the actual $\pi(x)$ function much better in the long run. (See [C.3.3] for more details along these lines, connecting to Fourier series, which we will not pursue.)

25.5.2 Detailing the connections

Now let's connect J and ζ. Recall the Euler product for ζ again:

$$\zeta(s) = \prod_p \frac{1}{1-p^{-s}}$$

The trick to getting information about primes out of this, as well as connecting to J, is to take the *logarithm* of the whole thing. This will turn the product into a sum, something we can work with much more easily[1].

$$\log(\zeta(s)) = \sum_p \log\left(\frac{1}{1-p^{-s}}\right) = \sum_p -\log\left(1-p^{-s}\right) \tag{25.5.1}$$

Adding just fractions would have perhaps allowed using a geometric series to make this a sum, but what could we do with a sum of logarithms?

Question 25.5.1. What can we do with $-\log()$ of some sum, not a product?

Solution. We can use its Taylor series!

$$-\log(1-x) = \sum_{k=1}^{\infty} \frac{x^k}{k}$$

So we plug it in:

$$\log(\zeta(s)) = \sum_p \sum_{k=1}^{\infty} \frac{(p^{-s})^k}{k} \tag{25.5.2}$$

Now we will manipulate this in two big steps. First we'll rewrite the fraction as an integral, and then we will try to somehow add up the integrals.

Standard improper integral work from second-semester calculus (Exercise 25.9.3) shows that

$$\frac{(p^{-s})^k}{k} = \frac{s}{k}\int_{p^k}^{\infty} x^{-s-1}dx$$

That means we can rewrite the logarithm of ζ as

$$\log(\zeta(s)) = \sum_p \sum_{k=1}^{\infty} \frac{(p^{-s})^k}{k}$$

$$= \sum_p \sum_{k=1}^{\infty} \frac{s}{k}\int_{p^k}^{\infty} x^{-s-1}dx = s\sum_p \sum_{k=1}^{\infty} \int_{p^k}^{\infty} \frac{1}{k}x^{-s-1}dx$$

This is a very large sum of integrals. We can rewrite this as a single integral, but we will need to pay close attention.

First, we can unify all these integrals from p^k to ∞ by making them all have the same endpoints. This is done somewhat artificially, by writing

$$\int_{p^k}^{\infty} \frac{1}{k}x^{-s-1}dx = \int_1^{p^k} \frac{1}{k}\cdot 0 \cdot x^{-s-1}\,dx + \int_{p^k}^{\infty} \frac{1}{k}x^{-s-1}dx$$

[1] This reminds me of the old joke about Noah's ark and logarithms. So, after the ark lands, all the animals are … having baby animals, let's say. Except the snakes. No baby snakes. Noah asks what the problem is – they seem to be missing the point. Snakes say, no worries, just give us a wooden bench or sawhorse or something. Noah wonders what's up, but gives it to them. Next morning, tons of baby snakes! Naturally Noah has to ask where the magic was. "Simple; adders need a log table to multiply."

This yields the integral of a piecewise-defined function, but it for every k and p it is defined from 1 to ∞.

Now comes the most surprising part. What function would I get if I added up all those integrals in the double sum (originally at (25.5.2))? To see this, let us add up *all* of the piecewise integrands, organizing by the powers k for any given prime p.

- Whenever x reaches $p^1 = p$, the sum of all those functions would add $\frac{1}{1}x^{-s-1}$. Adding up all of these for all p means the total function would include
$$\pi(x)x^{-s-1}\dots$$
- Whenever x reaches p^2, the sum of all those functions would add $\frac{1}{2}x^{-s-1}$. This, however, is the same thing as when $\sqrt{x}$ hits a prime, so we can add it to the previous point. The total function would include would include
$$\frac{1}{2}\pi(\sqrt{x})x^{-s-1}\dots$$
- When x reaches a cube of a prime, the sum adds $\frac{1}{3}x^{-s-1}$. This is the same thing as adding a new part when $\sqrt[3]{x}$ hits a prime, that is adding
$$\frac{1}{3}\pi(\sqrt[3]{x})x^{-s-1}$$
- And so forth for each k.

In short, adding up all these piecewise integrands seems to give a big integrand
$$\left(\pi(x) + \frac{1}{2}\pi(\sqrt{x}) + \frac{1}{3}\pi(\sqrt[3]{x}) + \cdots\right)x^{-s-1}$$

But this sum of all the piecewise integrands is $J(x)$, multiplied by x^{-s-1}. Hence
$$\log(\zeta(s)) = s\sum_p\sum_{k=1}^{\infty}\int_{p^k}^{\infty}\frac{1}{k}x^{-s-1}dx = s\int_1^{\infty}J(x)x^{-s-1}dx \tag{25.5.3}$$

This completes our connection of ζ and J.

25.6 Connecting to Zeros

25.6.1 Where are the zeros?

Our next goal is to see how this connection
$$\log(\zeta(s)) = s\int_1^{\infty}J(x)x^{-s-1}dx$$
relates to the zeros of the ζ function (and hence the Riemann Hypothesis).

```
L = lcalc.zeros_in_interval(10,100,0.1)
[l[0] for l in L]
```

We see all the zeros for $\sigma = 1/2$ between 0 and 100; there are 29 of them.

We will connect to ζ by means of a very powerful analogy, the one which Euler used to prove $\zeta(2) = \frac{\pi^2}{6}$ (see the end of Subsection 20.4.2) and which, correctly done, does yield the right answer.

Begin the analogy by recalling basic algebra. The Fundamental Theorem of Algebra states that *every* polynomial factors over the complex numbers. For instance,

$$f(x) = 5x^3 - 5x = 5(x-0)(x-1)(x+1) \ .$$

If we take the logarithm of such a factorization, we can say things like

$$\log(f(x)) = \log(5) + \log(x-0) + \log(x-1) + \log(x+1)$$

Then if it turned out that $\log(f(x))$ was useful to us for some other reason R, it would be reasonable to say that we can get information about the otherwise-mysterious R from adding up information about the zeros of f (and the constant 5), because of the addition of $\log(x-r)$ for all the roots r.

You can't really do this with arbitrary functions, of course. Disappointingly, ζ is definitely a function where this doesn't work, mostly because $\zeta(1)$ diverges so badly, no matter how you define the complex version of ζ.

But it so happens that ζ is very close to a function you *can* analyze this way, $(s-1)\zeta(s)$. Applying the logarithm factoring idea to $(s-1)\zeta(s)$ (and doing lots of relatively hard complex integrals, or some other formal business with difficult convergence considerations) allows us to essentially invert the equation

$$\log(\zeta(s)) = s\int_1^\infty J(x)x^{-s-1}dx$$

to the even more surprising formula

$$J(x) = Li(x) - \sum_\rho Li(x^\rho) - \log(2) + \int_x^\infty \frac{dt}{t(t^2-1)\log(t)} \qquad (25.6.1)$$

25.6.2 Analyzing the connection

It is hard to overestimate the importance of the formula (25.6.1). Each piece comes from something inside ζ itself, inverted in this special way.

- First, $Li(x)$ comes from the fact that we needed $(s-1)\zeta(s)$ to apply this inversion, not just $\zeta(s)$. In fact, this particular inversion can be seen by integrating, as it's true that

  $$s\int_1^\infty Li(x)x^{-s-1}dx = -\log(s-1)$$

  so one can see that $s-1$ and Li seem to correspond.

- Second, each $Li(x^\rho)$ comes from each of the zeros of ζ on the line $\sigma = 1/2$ in the complex plane. This is the part which most closely corresponds to the factoring.

- The constant term $\log(2)$ comes from the constant when you do the factoring, similarly to the 5 in the example above using $f(x) = 5x^3 - 5x$.

- Finally, the integral in (25.6.1) comes from the zeros of ζ at $-2n$ we mentioned just before the statement of 25.3.2.

To give you a sense of how complicated (25.6.1) really is, here is a plot of just one small piece of it.

```
import mpmath
parametric_plot((lambda t:
    mpmath.ei(log(20)*(0.5+i*RR(t))).real, lambda t:
    mpmath.ei(log(20)*(0.5+i*RR(t))).imag), (0,14.1),
    rgbcolor=hue(0.7), plot_points=300) +
    point((mpmath.ei(log(20)*(0.5+i*14.1)).real,
    mpmath.ei(log(20)*(0.5+i*14.1)).imag),
    color='red',size=20)
```

This is the plot of

$$Li(20^{1/2+it})$$

up through the *first* zero of ζ above the real axis. It's beautiful, but also forbidding. After all, if takes that much twisting and turning to get to Li of the first zero, what is in store if we have to add up over all *infinitely many* of them to calculate $J(20)$?

So at the very least, it would be helpful to know *where* all of those mysterious zeros live! This is why the Riemann Hypothesis is so important; it pins them down quite dramatically.

25.7 The Riemann Explicit Formula

Now we are finally ready to see Riemann's result, by plugging in this formula for J into the Moebius inverted formula for π given by

$$\pi(x) = J(x) - \frac{1}{2}J(\sqrt{x}) - \frac{1}{3}J(\sqrt[3]{x}) - \frac{1}{5}J(\sqrt[5]{x}) + \frac{1}{6}J(\sqrt[6]{x}) + \cdots$$

It is true that Riemann did not prove the following formula fully rigorously, and indeed one of the provers of the Prime Number Theorem mentioned taking *decades* as part of that effort just to prove all the statements Riemann made in this one paper. Nonetheless, it is certainly Riemann's formula for $\pi(x)$, and an amazing one:

Fact 25.7.1 (Riemann explicit formula)**.**

$$\pi(x) = \sum_{n=1}^{\infty} \frac{\mu(n)}{n} \left[Li(x^{1/n}) - \sum_{\rho} \left(Li(x^{\rho/n}) + Li(x^{\bar{\rho}/n}) \right) + \int_{x^{1/n}}^{\infty} \frac{dt}{t(t^2-1)\log(t)} \right]$$

It is worth making two points about the transition to this formula. First, if you're wondering where the log(2) at the end of the previous section went, it went to 0 because $\sum_{n=1}^{\infty} \frac{\mu(n)}{n} = 0$, though this is very hard to prove. (In fact, it is a consequence of the Prime Number Theorem; see Exercise 25.9.5.)

Secondly, each ρ is a zero above the real axis, and then $\bar{\rho}$ is the corresponding one below the real axis. The summation is over every single zero not on the real axis. In particular, these ρ are conjectured by the Riemann Hypothesis to all have real part equal to 1/2, which would make things particularly tidy.

Now let's see this formula in action.

```
import mpmath
var('y')
L = lcalc.zeros_in_interval(10,50,0.1)
@interact
```

```
def _(n=(100,(60,10^3))):
    P = plot(prime_pi,n-50,n, color='black',
        legend_label='$\pi(x)$')
    P += plot(Li,n-50,n, color='green',
        legend_label='$Li(x)$')
    G = lambda x: sum([mpmath.li(x^(1/j)) * moebius(j)/j
        for j in [1..3]])
    P += plot(G,n-50,n, color='red', legend_label =
        '$\sum_{j=1}^{%s} \\frac{\mu(j)}{j} Li(x^{1/j})$'%3)
    F = lambda x: sum([(mpmath.li(x^(1/j))-log(2) +
        numerical_integral( 1/(y*(y^2-1)*log(y)),
        x^(1/j),oo)[0] )*moebius(j)/j for j in [1..3]]) -
        sum([(mpmath.ei(log(x)*((0.5+l[0]*i)/j)) +
        mpmath.ei(log(x)*((0.5-l[0]*i)/j))).real for l in L
        for j in [1..3]])
    P += plot(F,n-50,n,color='blue', legend_label='Really
        good estimate',plot_points=50)
    show(P)
```

This graphic shows just how good it can get. Again, notice the waviness, which allows it to approximate $\pi(x)$ not just once per "step" of the function, but *along* the steps.

We can also just check out some numerical values.

```
var('y')
L = lcalc.zeros_in_interval(10,300,0.1)
F = lambda x: sum([(mpmath.li(x^(1/j))-log(2) +
    numerical_integral(1/(y*(y^2-1)*log(y)),x^(1/j),oo)[0]
    )*moebius(j)/j for j in [1..3]]) -
    sum([(mpmath.ei(log(x)*((0.5+l[0]*i)/j)) +
    mpmath.ei(log(x)*((0.5-l[0]*i)/j))).real for l in L for
    j in [1..3]])
var('y')
L = lcalc.zeros_in_interval(10,300,0.1)
F = lambda x: sum([(mpmath.li(x^(1/j))-log(2) +
    numerical_integral(1/(y*(y^2-1)*log(y)),x^(1/j),oo)[0]
    )*moebius(j)/j for j in [1..3]]) -
    sum([(mpmath.ei(log(x)*((0.5+l[0]*i)/j)) +
    mpmath.ei(log(x)*((0.5-l[0]*i)/j))).real for l in L for
    j in [1..3]])
@interact
def _(n=300):
    print F(n)
    print prime_pi(n)
    print Li(n.n())
    print Li(n.n()) - 1/2*Li(sqrt(n.n())) -
        1/3*Li((n.n())^(1/3))
```

Many wonderful facts would follow from the truth of the Riemann Hypothesis, or from a natural generalization.

Fact 25.7.2 (Consequences of the (generalized) Riemann Hypothesis)**.** *The following follow from the Riemann Hypothesis or a generalization for things like general Dirichlet series.*

- *The Dirichlet series of the Möbius function would be the multiplicative inverse of the zeta function for lots more complex values than just the real ones we proved it for in .*

- *The* value *(not just average) of $\sigma(n)$ would have the following bound once n is big enough:*
$$\sigma(n) < e^{\gamma}\log(\log(n))$$
- *The biggest gap between consecutive prime numbers could not be too big (to be precise, $O(\sqrt{p}\log(p))$.*
- *We would know exactly what it means for a type of prime to win the 'prime races' (see Section 22.1).*
- *Artin's conjecture (Conjecture 17.5.2) on primitive roots follows from a generalization as well.*

So can *you* prove that there are no other zeros other than those on the critical line to contribute to these approximations to $\pi(x)$? If so, welcome to the future of number theory!

25.8 Epilogue

Let's see just a little more of the future of number theory. The Riemann zeta function and counting primes is truly only the beginning of research in modern number theory.

For instance, research in finding and counting points on curves (as in Chapter 15) leads to more complicated series like ζ, called L-functions. There is a version of the Riemann Hypothesis for them, too (see the end of the previous subsection). Even without that, they gives truly interesting, strange, and beautiful results. Here is a recent result of interest.

Recall from Example 14.2.3 that the notation $r_{12}(n)$ should denote the number of ways to write n as a sum of *twelve* squares. Here, order and sign both matter, so $(1,2)$ and $(2,1)$ and $(-2,1)$ are all different.

Theorem 25.8.1. *As we let p run through the set of all prime numbers, the distribution of the fraction*
$$\frac{r_{12}(p) - 8(p^5+1)}{32p^{5/2}}$$
is precisely as this circular function in the long run:
$$\frac{2}{\pi}\sqrt{1-t^2}$$

Proof. Needless to say, this is ***far*** beyond the level of this course – but maybe you will make the next contribution? Initially this result is a corollary of the proof of the Sato-Tate conjecture by Barnet-Lamb, Geraghty, Harris, and Taylor; that proof crucially used the so-called "Fundamental Lemma" of Gérard Laumon and Ngô Bảo Châu, the latter of whom won the Fields Medal based on proving it in very full generality. □

Sage note 25.8.2 (Into the future)**.** The following graphic is based on one due to William Stein, the original founder and developer of Sage, in personal communication. The higher the number, the closer the values should group to the distribution; change the number of bins in the histogram to see it more clearly.

```
def sqrt2():
    PI = float(pi)
    return plot(lambda x: (2/PI)*math.sqrt(1-x^2), -1,1,
        plot_points=200,
              rgbcolor=(0.3,0.1,0.1), thickness=2)

delta = delta_qexp(10^5)

@interact
def delta_dist(bins=(20,[10..150]), number =
    [500,1000,..,delta.prec()]):
    D = delta[:number]
    w = [float(D[p])/(2*float(p)^(5.5)) for p in
        prime_range(number + 1)]
    show(histogram(w, bins=bins, normed=True) + sqrt2(),
        frame=True, gridlines=True)
```

What an amazing result. These ideas are at the forefront of all types of number theory research today, and my hope is that you will enjoy exploring more of it, both with paper and pencil and using tools like Sage!

25.9 Exercises

1. Prove that $e^{ix} = \cos(x) + i\sin(x)$ using Taylor series. Try to include proofs of the convergence of everything involved.

2. Many books have a chain of reasoning interpreting the value $\zeta(-1) = \frac{1}{12}$. Find a *physical* one and summarize the argument. (The Specialized References and Other References may have some suggestions.) Do you buy that adding all positive integers could possibly have a meaning?

3. Show all details for the improper integrals in Section 25.5.

4. Differentiate the function $h(x) = x^x$. Why is this question appropriate for this chapter?

5. Verify numerically that $\sum_{n=1}^{\infty} \frac{\mu(n)}{n} \to 0$ – by calculator, then by computer. How close can you get to zero before your computer gives up?

6. Read one of the several excellent introductions to the Riemann Hypothesis intended for the "general reader". (Some are listed in the Specialized References.)

7. What is the Birch-Swinnerton-Dyer Conjecture? Find out as much about it as you can.

8. Answer one of these questions, or all of them.

- What are partitions of a number?
- What are continued fractions?
- What is an elliptic curve, and how is it used in cryptography?
- What is a number field?

9. What else do you want to know about numbers? What are you inspired to discover?

Appendix A

List of Sage notes

There are many great Sage references. But for the convenience of users of this text, we collect all the many Sage notes from the text here in one place.

(Continued on next page)

Appendix B

Notation

This is a quick guide to possibly unfamiliar notation. Page numbers or references usually refer to the first appearance of a notation with that meaning, occasionally to a definition.

Symbol	Description	Page
$\mathbb{Z}$	(ring of) integers	1
$\mathbb{N}$	counting numbers (starting at zero)	1
$a \mid b$	a is a divisor of b	3
$\gcd(a, b)$	greatest common divisor of a and b	10
$\lfloor x \rfloor$	greatest integer (floor) function	23
$a \equiv b \pmod{n}$	a is congruent to b modulo n	36
$[a]$	the equivalence class of a modulo some fixed n	39
a^{-1}	multiplicative inverse of a number modulo some fixed n	53
$\prod_{i=1}^{n} p_i$	product of unspecified, possible identical, primes	65
$\prod p$	short form for product of primes	65
$\prod q$	alternate short form for product of primes	65
$\prod_{i=1}^{n} p_i^{e_i}$	product of unspecified distinct prime power	65
$\prod p^e$	short form for product of prime powers	65
$p^k \parallel n$	for p prime, $p^k \mid n$ *but* p^{k+1} does not divide n	68
$n!$	n factorial	69
$\mathbb{Z}_n$	(ring of) integers modulo n	87
$A \setminus \{a\}$	the set of all elements in A except $a \in A$	93
$\lvert G \rvert$	order of a group G	94
$\lvert x \rvert$	order of a group element $x \in G$	95
U_n	group of units modulo n	100
$\phi(n)$	order of the group of units of n (Euler function)	102
F_n	Fermat number $2^{2^n} + 1$	149
M_n	Mersenne number $2^n - 1$	151
$r_2(n)$	number of different ways to write n as a sum of two squares	186
$r_k(n)$	number of different ways to write n as a sum of k perfect squares	192
QR	abbreviation for 'quadratic residue'	220
$\left(\frac{a}{p}\right)$	Legendre symbol, for p prime	226

(Continued on next page)

Symbol	Description	Page
aE	multiples of even numbers by a (in a given residue system)	233
$\left(\frac{a}{n}\right)$	Jacobi symbol, n odd	240
$r(n)$	alternate notation for $r_2(n)$	254
$\sigma_k(n)$	sum of kth powers of divisors of n	259
$\tau(n)$	number of (positive) divisors of n	259
$\sigma(n)$	sum of (positive) divisors of n	259
$u(n)$	unit function	262
$N(n)$	identity function	262
$\sigma^{-1}(n)$	abundancy index of n	268
$O(g(x))$	'Big Oh' notation that a function is less in absolute value than $Cg(x)$, for some constant C	276
$\log(n)$	natural (base e) logarithm	283
γ	Euler-Mascheroni gamma constant, limit of difference between the harmonic series and natural logarithm	285
Γ	Gamma function factorial extension	285
$\phi(n, a)$	number of integers coprime to first a primes	293
$Li(x)$	logarithmic integral $\int_2^x \frac{dt}{\log(t)}$	294
$\Theta(x)$	Chebyshev theta function	300
$a(n)$	prime number indicator function	301
$p\#$	primorial (product of primes up to p)	312
C_2	twin prime constant	315
$\mu(n)$	Moebius function of n	319
$f \star g$	Dirichlet product of arithmetic functions f and g	322
$I(n)$	Dirichlet product identity function	323
$\omega(n)$	number of unique prime divisors of n	325
$\nu(n)$	alternate notation for $\omega(n)$	325
$\lambda(n)$	Liouville's function	325
$\zeta(s)$	Riemann zeta function	333
$J(x)$	auxiliary function in Riemann explicit formula	355

Appendix C

References and Further Resources

There are so many resources I used in preparation of this book it would be very hard to list all of them. Still, I have a lot of recommendations for further reading, places for instructors to look for alternate examples, proofs, exercises, etc., and most of these are books I have actively used at some point. I attempted to include a canonical website for each book, though be aware that especially publisher pages may change at short notice. I've also included some valuable articles I have benefited from.

C.1 General References

There are many good introductory number theory texts.

[1] Gareth A. and J. Mary Jones, *Elementary Number Theory*, Springer, London, (2005). (Website)
A good introduction with an emphasis on groups, containing interleaved exercises with full answers.

[2] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, fifth edition, Oxford, (1979) (Website for expanded sixth edition)
A highly regarded text with copious notes, but sometimes more than a little hard to parse with its consecutively numbered theorems and very dense prose.

[3] William Stein, *Elementary Number Theory: Primes, Congruences, and Secrets*, Springer, (2008) (Website)
Freely available and the first Sage-enabled number theory text, by the founder of Sage (a number theorist).

[4] Ken Rosen, *Elementary Number Theory and its Applications*, Pearson, (2011). (Website)
A venerable text with programming exercises that still wear well.

[5] David C. Marshall, Edward Odell, Michael Starbird, *Number Theory through Inquiry*, Mathematical Association of America, Washington, (2007). (Website)
The topics are very standard, but the approach is quite different; no proofs, only statements. This turns out to be a highly effective pedagogy; see the Academy of Inquiry Based Learning for more information.

[6] R. P. Burn, *A pathway into number theory*, Cambridge, (1996) (Website)
A very fun inquiry-driven text before there were such things, with a lot of extremely good examples, especially in things like quadratic forms.

[7] John Stillwell, *Elements of Number Theory*, Springer, (2003) (Website)
More algebraically oriented, with good material on the Pell equation and Gaussian integers – noteworthy for a good treatment of Conway's river concepts.

[8] Harold Shapiro, *Introduction to the Theory of Numbers*, Dover, (2008) (No website)
Incredibly comprehensive, at a fairly high level. Good material on averages and odd perfection, immense bibliography and notes in style of [C.1.2], and also inquiry-driven "do-it-yourself" sections. Appears to be out of print.

[9] Anthony Gioia, *The Theory of Numbers*, Dover, (2001) (No website)
Surprisingly detailed and high-level but has good coverage of several unusual topics such as geometry of numbers.

[10] Marty Erickson, Anthony Vazzana, David Garth, *Introduction to Number Theory*, second edition, CRC, (2016). (Website)
Enough material for two courses, some fairly advanced, and newly endowed with downloadable Sage worksheets.

[11] George Andrews, *Number Theory*, Dover, (1994) (Website)
Yet another nice reprint from Dover, this one with (as one would expect of the author) great combinatorial content.

[12] H. M. Edwards, *Higher Arithmetic: An Algorithmic Introduction to Number Theory*, American Mathematical Society, (2006) (Website)
Not so algorithmic, but very, *very* concrete and constructive. Squares are □s, which grows on the reader.

[13] Neville Robbins, *Beginning Number Theory*, Jones and Bartlett, (2006) (No website)
An out-of-print standard text with many similar topics and interesting historical comments.

[14] Oystein Ore, *Invitation to Number Theory*, Mathematical Association of America, (1967) (Website)
An older text that is still worth the conversational tone.

C.2 Proof and Programming References

The first few books here are good resources for an introduction to proof, which should cover anything needed as a prerequisite for this text.

In addition to the many good programming exercises in several books in the General References, the latter books will give you an introduction to the programming side of things.

[1] Richard Hammack, *The Book of Proof*, (2013). (Website)
A quality middle-of-the-road introduction to proof, used reasonably widely and covering all standard topics for a proof transition course.

[2] Joseph Fields, *A Gentle Introduction to the Art of Mathematics*, (2013). (Website)
The title is pretty accurate; this is a quite gentle open text usable for self-study.

[3] Edward Burger, *Extending the Frontiers of Mathematics*, Key College, (2007) (Website)
This book is not necessarily just an introduction to proof, but has a wonderful attitude to conjecture. Essentially, one should view every proof as an opportunity to extend, and every disproof as a chance to rescue.

[4] Gregory Bard, *Sage for Undergraduates*, American Mathematical Society, (2015) (Website)
This is a very good guide to Sage for anyone starting out with basic college math knowledge; the author has taught using Sage for some time. Did I mention it is freely downloadable as well as available in print?

[5] Craig Finch, *Sage: Beginner's Guide*, Packt, (2011) (Website)
This guide is not free, but is comprehensive (for the time it was written) and has the unique perspective of someone not involved in the Sage community.

[6] Allen Downey, *Think Python*, O'Reilly, (2012) (Website)
A very good introduction to programming from scratch in Python, usable from the website or as a hard-copy text.

[7] Zed Shaw, *Learn Python the Hard Way*, Addison-Wesley, (2013) (Website)
A preternaturally idiosyncratic take on how to program, but well worth the effort to learn things the hard way if you have the time to push through it.

C.3 Specialized References

Number Theory is a huge field, and even at an introductory level there are many wonderful resources to be aware of. I have used most of the following in one way or another in preparation of this text, and if you are intrigued by a specific facet of number theory, I encourage you to get these from your library! Most of these are more specialized, but a few are not really texts but intended for the "casual" reader.

[1] John Derbyshire, *Prime Obsession*, Joseph Henry Press, (2003) (Website)
A marvelous achievement of bringing the Riemann Hypothesis to the (determined) lay reader while simultaneously making you care about post-Napoleonic Europe. If I do say so myself.

[2] Roland van der Veen and Jan van de Craats, *The Riemann Hypothesis*, Mathematical Association of America, (2016). (Website)
Interesting lecture notes leading to a basic understanding of the Riemann Hypothesis, based on a high-school enrichment program in the Netherlands.

[3] Barry Mazur and William Stein, *Prime Numbers and the Riemann Hypothesis*, Cambridge University Press, (2016). (Website)
This book goes straight for the jugular of the Riemann Hypothesis, starting from scratch. That requires a lot of investment, but you won't find it from the perspective of working number theorists in other books, either.

[4] H. M. Edwards, *Riemann's Zeta Function*, Dover, (2001) (Website)
Still useful comprehensive first text on this important topic.

[5] Jeffrey Stopple, *A Primer of Analytic Number Theory*, Cambridge, (2003). (Website)
Very innovative book on exactly what it says; second half not neces-

sarily for every US undergraduate, but easiest introduction to Birch-Swinnerton-Dyer I could find! Covers most traditional material, too, and has copious entertaining historical notes.

[6] Tom Apostol, *Introduction to Analytic Number Theory*, Springer, (1976). (Website)
The canonical "undergraduate" analytic number theory book. Monumental but very difficult; zillions of interesting results in exercises.

[7] Stan Wagon and David Bressoud, *A Course in Computational Number Theory*, Wiley, (2008). (Website)
Contains Mathematica code to visualize and explore a lot of interesting number theory, and is very consistent with the computational viewpoint throughout.

[8] Paul Pollack, *Not Always Buried Deep*, American Mathematical Society, (2009). (Website)
Definitely a second course in number theory, as the subtitle says, with good material on arithmetic progressions and the Hilbert-Waring problem (the latter is difficult to find in a textbook).

[9] Harold Davenport, *The Higher Arithmetic*, Cambridge University Press, (2008). (Website)
Another well-known general resource, with a very good description of how to find if a rational conic has a rational point (which directly connects to integer points on conics as well).

[10] Stephen Richards, *A Number for Your Thoughts*, S. P. Richards, (1982) (No website)
Many very interesting topics for the general reader, from repunits to all sorts of other topics. Intriguing story must lie behind the essentially identical book by a different author several years later.

[11] Samuel S. Wagstaff, Jr., *The Joy of Factoring*, American Mathematical Society, (2013). (Website)
The title says it all, and more accessible to college students than one would think. By one of the leaders in the field.

[12] George Andrews and Kimmo Eriksson, *Integer Partitions*, Cambridge University Press, (2004). (Website)
A brilliant, accessible, inventive book which makes me very sad there is only enough time for so many topics in a one-semester course. Indispensable for bringing partitions to undergraduates.

[13] Richard Friedberg, *An Adventurer's Guide to Number Theory*, Dover, (1995) (Website)
Very conversational and enjoyable; not really a textbook. Key feature is a detailed discussion of how Euler missed what is essentially unique factorization in a certain number field for two of his more interesting results – and he does it without actually proving unique factorization!

[14] Julian Havil, *Gamma: Exploring Euler's Constant*, Princeton, (2009). (Website)
This book turns out to be about both Γ the function and γ the constant (recall Definition 20.3.2), and includes a description of Apéry's tomb (see Subsection 24.4.1 and $\zeta(3)$).

[15] C. D. Olds, Anneli Lax, Giuliana Davidoff, *The Geometry of Numbers*, Mathematical Association of America, (2000) (Website)
Delightful introduction to and inspiration for many of the lattice topics

pursued in this text. The second half goes fairly deep, and is more than worth pursuing as a directed study with undergraduates.

[16] Paulo Ribenboim, *The Little Book of Bigger Primes*, Springer, (2004) (Website)
This book has incredible amounts of interesting detail regarding many of the prime topics considered here. An example: a discourse on whether the pseudoprime criterion base 2 was really discovered by ancient Chinese mathematicians.

[17] Paulo Ribenboim, *My Numbers, My Friends*, Springer, (2000) (Website)
Based on a series of lectures, this book is rather higher level, but has correspondingly more truly interesting material, including an entire chapter inspired by 1093 and a very early prime-generating algorithm by a certain Pocklington.

C.4 Historical References

Number Theory is also a very old field, as should be clear from using this book. Here I have collated references intended both for mathematicians and the fabled 'educated laity'. (Note that many of the other books referenced here have significant historical content, notably [C.3.5].)

[1] Jim Tattersall, *Elementary Number Theory in Nine Chapters*, Cambridge University Press, (2005) (Website)
Oodles of class-tested historical material and many, many exercises, including a welter of them on topics surrounding amicable numbers.

[2] John J. Watkins, *Number Theory: A Historical Approach*, Princeton, (2013). (Website)
A very nice historically-oriented approach to elementary number theory. Includes Sage material in an appendix.

[3] Oystein Ore, *Number Theory and Its History*, Dover, (1948). (Website)
Another conversational classic by Ore, with plenty of historical goodies.

[4] Jay Goldman, *The Queen of Mathematics*, AK Peters, (1997) (Website)
A truly historical sojourn through much of number theory up through the early twentieth century, with extensive primary source material and investigation of Gauss' monumental work. Sadly, largely beyond the level of this text.

[5] William Dunham, *Journey Through Genius*, Wiley, (1990). (Website)
This is intended for those without calculus, but has many great number-theoretic bits all the same.

[6] William Dunham, *Euler: The Master of Us All*, Mathematical Association of America, (1999). (Website)
This book has some nice discussion of Euler's number theory alongside many other historical vignettes with real math power.

[7] A. Knoebel et al., *Mathematical Masterpieces: Further Chronicles by the Explorers*, Springer, (2007). (Website)
Collection of additional classroom resources focused on primary source material, including the Basel problem and quadratic reciprocity.

C.5 Other References

Some books are just interesting, even if they are not primarily about number theory. I enjoyed all of these a great deal and recommend them.

[1] Richard Evans Schwartz, *You Can Count on Monsters*, A K Peters, (2010) (Website)
This delightful picture book has a different monster for each *prime* number, with bizarre combinations for composites. Personal experience says it satisfies for ages three and up.

[2] Nathan Carter, *Visual Group Theory*, Mathematical Association of America, (2009). (Website)
Visualize group theory; gorgeous pictures.

[3] John H. Conway and Richard Guy, *The Book of Numbers*, Springer, (1996). (Website)
A joyous and pictorially engaging romp.

[4] Arthur T. Benjamin and Ezra Brown (eds.), *Biscuits of Number Theory*, Mathematical Association of America, (2009). (Website)
A very good compendium of many articles (published throughout the years) most appropriate for teachers of undergraduate number theory.

[5] Kerins et al., *Famous Functions in Number Theory*, American Mathematical Society, (2015). (Website)
Aimed at bringing number theory to in-practice or pre-practice educators, this has a very nice treatment of arithmetic functions. Once you've heard of summation and Moebius inversion as 'parent' and 'child' relationships, you'll never think of them the same again.

[6] Kerins et al., *Applications of Algebra and Geometry to the Work of Teaching*, American Mathematical Society, (2015). (Website)
Aimed at bringing algebra and geometry to in-practice or pre-practice educators; manages to bring Gaussian and Eisenstein integers and some quadratic forms in at the ground level.

[7] T. S. Michael, *How to Guard an Art Gallery*, Johns Hopkins, (2009) (Website)
The subtitle is "and other discrete mathematical adventures", and that about says it. Covers a surprising amount of number theory in very visual ways.

[8] Robert Young, *Excursions in Calculus: An Interplay of the Continuous and Discrete*, Mathematical Association of America, (1992) (Website)
Unfortunately no longer in print, but a very good source of ideas for connecting what we usually think of as the continuous world of calculus and various discrete topics (not just number theory, though this shows up in several chapters).

C.6 Useful Articles

Throughout the text, I've attempted to reference articles in so-called 'generalist' mathematics publications which have been useful or intriguing. See also the collection [C.5.4], where some of these appear. For any comments, see the locations they are referenced.

[1] Ivan Niven and Barry Powell, *Primes in Certain Arithmetic Progressions*, The American Mathematical Monthly, June-July 1976, **83** no. 6, 467–469.

[**2**] D. Zagier, *A One-Sentence Proof That Every Prime $p \equiv 1 (mod\ 4)$ Is a Sum of Two Squares*, The American Mathematical Monthly, February 1990, **97** no. 2, 144–144.

[**3**] Andrew Granville and Greg Martin, *Prime Number Races*, The American Mathematical Monthly, January 2006, **113** no. 1, 1–33.

[**4**] David A. Cox, *Why Eisenstein Proved the Eisenstein Criterion and Why Schönemann Discovered It First*, The American Mathematical Monthly, January 2011, **118** no. 1, 3–21.

[**5**] Steven H. Weintraub, *On Legendre's Work on the Law of Quadratic Reciprocity*, The American Mathematical Monthly, March 2011, **118** no. 3, 210–216.

[**6**] Jonathan Bayless and Dominic Klyve, *Reciprocal Sums as a Knowledge Metric: Theory, Computation, and Perfect Numbers*, The American Mathematical Monthly, November 2013, **120** no. 9, 822–831.

[**7**] Xianzu Lin, *Infinitely Many Primes in the Arithmetic Progression $kn-1$*, The American Mathematical Monthly, January 2015, **122** no. 1, 48–51.

[**8**] Reinhard Laubenbacher and David Pengelley, *Eisenstein's Misunderstood Geometric Proof of the Quadratic Reciprocity Theorem*, The College Mathematics Journal, January 1994, **25** no. 1, 29–34.

[**9**] Roger B. Nelsen, *Proof Without Words: Square Triangular Numbers and Almost Isosceles Pythagorean Triples*, College Mathematics Journal, May 2016, **47** no. 3, 179–179.

[**10**] David Lowry-Duda, *Unexpected Conjectures about -5 Modulo Primes*, College Mathematics Journal, January 2015, **46** no. 1, 56–57.

[**11**] William G. Stanton and Judy A. Holdener, *Abundancy "Outlaws" of the Form $\frac{\sigma(N)+t}{N}$*, Journal of Integer Sequences, **10**

[**12**] D. R. Slavitt, *Give Way To God, or The Dying Christ – Pierre de Fermat*, The Mathematical Intelligencer, Summer 2012, **34** no. 2, 3–5.

[**13**] Paul Nahin, *The Mysterious Mr. Graham*, The Mathematical Intelligencer, Spring 2016, **38** no. 1, 48–51.

[**14**] P. A. Weiner, *The abundancy index, a measure of perfection*, Mathematics Magazine, October 2000, **73** no. 4, 307–310.

[**15**] Andrew Bremner, *Positively prodigious powers or how Dudeney done it?*, Mathematics Magazine, April 2011, **84** no. 2, 120–125.

[**16**] Rafael Jakimczuk, *The Quadratic Character of 2*, Mathematics Magazine, April 2011, **84** no. 2, 126–127.

[**17**] Russell A. Gordon, *Properties of Eisenstein Triples*, Mathematics Magazine, February 2012, **85** no. 1, 12–25.

[**18**] Roger B. Nelsen, *Proof Without Words: Infinitely Many Almost-Isosceles Pythagorean Triples Exist*, Mathematics Magazine, April 2016, **89** no. 2, 103–104.

[**19**] C. Edward Sandifer, *How Euler Did It: Odd Perfect Numbers*, MAA Online, November 2006

[**20**] Matthias Beck, *How to change coins, M&M's, or chicken nuggets: The linear Diophantine problem of Frobenius*, in Resources for Teaching Discrete Mathematics: Classroom Projects, History Modules, and Articles (B. Hopkins, ed.), Mathematical Association of America, 2009, 65–74.

Index