

Elementary Number Theory

William Stein

September 2004

To my students and my wife, Clarita Lefthand.

Contents

Preface	3
1 Prime Numbers	5
1.1 Prime Factorization	5
1.2 The Sequence of Prime Numbers	13
1.3 Exercises	19
2 The Ring of Integers Modulo n	21
2.1 Congruences Modulo n	21
2.2 The Chinese Remainder Theorem	27
2.3 Quickly Computing Inverses and Huge Powers	29
2.4 Finding Primes	33
2.5 The Structure of $(\mathbf{Z}/p\mathbf{Z})^*$	34
2.6 Exercises	38
3 Public-Key Cryptography	43
3.1 The Diffie-Hellman Key Exchange	46
3.2 The RSA Cryptosystem	51
3.3 Attacking RSA	54
3.4 Exercises	58
4 Quadratic Reciprocity	59
4.1 Statement of the Quadratic Reciprocity Law	60
4.2 Euler's Criterion	62

4.3	First Proof of Quadratic Reciprocity	63
4.4	A Proof of Quadratic Reciprocity Using Gauss Sums	68
4.5	Finding Square Roots	72
4.6	Exercises	74
5	Continued Fractions	77
5.1	Finite Continued Fractions	78
5.2	Infinite Continued Fractions	83
5.3	The Continued Fraction of e	88
5.4	Quadratic Irrationals	91
5.5	Recognizing Rational Numbers	96
5.6	Sums of Two Squares	97
5.7	Exercises	100
6	Elliptic Curves	103
6.1	The Group Structure on an Elliptic Curve	104
6.2	Integer Factorization Using Elliptic Curves	107
6.3	Elliptic Curve Cryptography	112
6.4	Elliptic Curves Over the Rational Numbers	116
6.5	Exercises	121
7	Computational Number Theory	123
7.1	Prime Numbers	125
7.2	The Ring of Integers Modulo n	131
7.3	Public-Key Cryptography	139
7.4	Quadratic Reciprocity	145
7.5	Continued Fractions	148
7.6	Elliptic Curves	152
7.7	Exercises	165
	Answers and Hints	167
	References	175

Preface

This is a textbook about prime numbers, congruences, basic public-key cryptography, quadratic reciprocity, continued fractions, elliptic curves, and number theory algorithms. We assume the reader has some familiarity with groups, rings, and fields, and for Chapter 7 some programming experience. This book grew out of an undergraduate course that the author taught at Harvard University in 2001 and 2002.

Notation and Conventions. We let $\mathbf{N} = \{1, 2, 3, \dots\}$ denote the natural numbers, and use the standard notation \mathbf{Z} , \mathbf{Q} , \mathbf{R} , and \mathbf{C} for the rings of integer, rational, real, and complex numbers, respectively. In this book we will use the words proposition, theorem, lemma, and corollary as follows. Usually a proposition is a less important or less fundamental assertion, a theorem a deeper culmination of ideas, a lemma something that we will use later in this book to prove a proposition or theorem, and a corollary an easy consequence of a proposition, theorem, or lemma.

Acknowledgements. Brian Conrad and Ken Ribet made a large number of clarifying comments and suggestions throughout the book. Baurzhan Bektemirov, Lawrence Cabusora, and Keith Conrad read drafts of this book and made many comments. Frank Calegari used the course when teaching Math 124 at Harvard, and he and his students provided much feedback. Noam Elkies made comments and suggested Exercise 4.5. Seth Kleinerman wrote a version of Section 5.3 as a class project. Samit Dasgupta, George Stephanides, Kevin Stern, and Heidi Williams all suggested corrections. I

4 Contents

also benefited from conversations with Henry Cohn and David Savitt. I used Emacs, L^AT_EX, and Python in the preparation of this book.

1

Prime Numbers

In Section 1.1 we describe how the integers are built out of the prime numbers $2, 3, 5, 7, 11, \dots$. In Section 1.2 we discuss theorems about the set of primes numbers, starting with Euclid's proof that this set is infinite, then explore the distribution of primes via the prime number theorem and the Riemann Hypothesis (without proofs).

1.1 Prime Factorization

1.1.1 Primes

The set of *natural numbers* is

$$\mathbf{N} = \{1, 2, 3, 4, \dots\},$$

and the set of *integers* is

$$\mathbf{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}.$$

Definition 1.1.1 (Divides). If $a, b \in \mathbf{Z}$ we say that a *divides* b , written $a \mid b$, if $ac = b$ for some $c \in \mathbf{Z}$. In this case we say a is a *divisor* of b . We say that a *does not divide* b , written $a \nmid b$, if there is no $c \in \mathbf{Z}$ such that $ac = b$.

For example, we have $2 \mid 6$ and $-3 \mid 15$. Also, all integers divide 0, and 0 divides only 0. However, 3 does not divide 7 in \mathbf{Z} .

Remark 1.1.2. The notation $b \div a$ for “ b is divisible by a ” is common in Russian literature on number theory.

Definition 1.1.3 (Prime and Composite). An integer $n > 1$ is *prime* if it the only positive divisors of n are 1 and n . We call n *composite* if n is not prime.

The number 1 is neither prime nor composite. The first few primes of \mathbf{N} are

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, \dots ,

and the first few composites are

4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33, 34, \dots .

Remark 1.1.4. J.H. Conway argues in [Con97, viii] that -1 should be considered a prime, and in the 1914 table [Leh14], Lehmer considers 1 to be a prime. In this book we consider neither -1 nor 1 to be prime.

Every natural number is built, in a unique way, out of prime numbers:

Theorem 1.1.5 (Fundamental Theorem of Arithmetic). *Every natural number can be written as a product of primes uniquely up to order.*

Note that primes are the products with only one factor and 1 is the empty product.

Remark 1.1.6. Theorem 1.1.5, which we will prove in Section 1.1.4, is trickier to prove than you might first think. For example, unique factorization fails in the ring

$$\mathbf{Z}[\sqrt{-5}] = \{a + b\sqrt{-5} : a, b \in \mathbf{Z}\} \subset \mathbf{C},$$

where 6 factors into irreducible elements in two different ways:

$$2 \cdot 3 = 6 = (1 + \sqrt{-5}) \cdot (1 - \sqrt{-5}).$$

1.1.2 The Greatest Common Divisor

We will use the notion of greatest common divisor of two integers to prove that if p is a prime and $p \mid ab$, then $p \mid a$ or $p \mid b$. Proving this is the key step in our proof of Theorem 1.1.5.

Definition 1.1.7 (Greatest Common Divisor). Let

$$\gcd(a, b) = \max \{d \in \mathbf{Z} : d \mid a \text{ and } d \mid b\},$$

unless both a and b are 0 in which case $\gcd(0, 0) = 0$.

For example, $\gcd(1, 2) = 1$, $\gcd(6, 27) = 3$, and for any a , $\gcd(0, a) = \gcd(a, 0) = a$.

If $a \neq 0$, the greatest common divisor exists because if $d \mid a$ then $d \leq a$, and there are only a positive integers $\leq a$. Similarly, the gcd exists when $b \neq 0$.

Lemma 1.1.8. *For any integers a and b we have*

$$\gcd(a, b) = \gcd(b, a) = \gcd(\pm a, \pm b) = \gcd(a, b - a) = \gcd(a, b + a).$$

Proof. We only prove that $\gcd(a, b) = \gcd(a, b - a)$, since the other cases are proved in a similar way. Suppose $d \mid a$ and $d \mid b$, so there exist integers c_1 and c_2 such that $dc_1 = a$ and $dc_2 = b$. Then $b - a = dc_2 - dc_1 = d(c_2 - c_1)$, so $d \mid b - a$. Thus $\gcd(a, b) \leq \gcd(a, b - a)$, since the set over which we are taking the max for $\gcd(a, b)$ is a subset of the set for $\gcd(a, b - a)$. The same argument with a replaced by $-a$ and b replaced by $b - a$, shows that $\gcd(a, b - a) = \gcd(-a, b - a) \leq \gcd(-a, b) = \gcd(a, b)$, which proves that $\gcd(a, b) = \gcd(a, b - a)$. \square

Lemma 1.1.9. *Suppose $a, b, n \in \mathbf{Z}$. Then $\gcd(a, b) = \gcd(a, b - an)$.*

Proof. By repeated application of Lemma 1.1.8, we have

$$\gcd(a, b) = \gcd(a, b - a) = \gcd(a, b - 2a) = \cdots = \gcd(a, b - 2n).$$

\square

Assume for the moment that we have already proved Theorem 1.1.5. A natural (and naive!) way to compute $\gcd(a, b)$ is to factor a and b as a product of primes using Theorem 1.1.5; then the prime factorization of $\gcd(a, b)$ can read off from that of a and b . For example, if $a = 2261$ and $b = 1275$, then $a = 7 \cdot 17 \cdot 19$ and $b = 3 \cdot 5^2 \cdot 17$, so $\gcd(a, b) = 17$. It turns out that the greatest common divisor of two integers, even huge numbers (millions of digits), is surprisingly easy to compute using Algorithm 1.1.12 below, which computes $\gcd(a, b)$ without factoring a or b .

To motivate Algorithm 1.1.12, we compute $\gcd(2261, 1275)$ in a different way. First, we recall a helpful fact.

Proposition 1.1.10. *Suppose that a and b are integers with $b \neq 0$. Then there exists unique integers q and r such that $0 \leq r < |b|$ and $a = bq + r$.*

Proof. For simplicity, assume that both a and b are positive (we leave the general case to the reader). Let Q be the set of all nonnegative integers n such that $a - bn$ is nonnegative. Then Q is nonempty because $0 \in Q$ and Q is bounded because $a - bn < 0$ for all $n > a/b$. Let q be the largest element of Q . Then $r = a - bq < b$, otherwise $q + 1$ would also be in Q . Thus q and r satisfy the existence conclusion.

To prove uniqueness, suppose for the sake of contradiction that q' and $r' = a - bq'$ also satisfy the conclusion but that $q' \neq q$. Then $q' \in Q$ since $r' = a - bq' \geq 0$, so $q' < q$ and we can write $q' = q - m$ for some $m > 0$. But then $r' = a - bq' = a - b(q - m) = a - bq + bm = r + bm > b$ since $r \geq 0$, a contradiction. \square

For us an *algorithm* is a finite sequence of instructions that can be followed to perform a specific task, such as a sequence of instructions in a computer program, which must terminate on any valid input. The word “algorithm” is sometimes used more loosely (and sometimes more precisely) than defined here, but this definition will suffice for us.

Algorithm 1.1.11 (Division Algorithm). Suppose a and b are integers with $b \neq 0$. This algorithm computes integers q and r such that $0 \leq r < |b|$ and $a = bq + r$. We will not describe the actual steps of this algorithm, since it is just the familiar long division algorithm.

We use the division algorithm repeatedly to compute $\gcd(2261, 1275)$. Dividing 2261 by 1275 we find that

$$2261 = 1 \cdot 1275 + 986,$$

so $q = 1$ and $r = 986$. Notice that if a natural number d divides both 2261 and 1275, then d divides their difference 986 and d still divides 1275. On the other hand, if d divides both 1275 and 986, then it has to divide their sum 2261 as well! We have made progress:

$$\gcd(2261, 1275) = \gcd(1275, 986).$$

This equality also follows by repeated application of Lemma 1.1.8. Repeating, we have

$$1275 = 1 \cdot 986 + 289,$$

so $\gcd(1275, 986) = \gcd(986, 289)$. Keep going:

$$986 = 3 \cdot 289 + 119$$

$$289 = 2 \cdot 119 + 51$$

$$119 = 2 \cdot 51 + 17.$$

Thus $\gcd(2261, 1275) = \dots = \gcd(51, 17)$, which is 17 because $17 \mid 51$. Thus

$$\gcd(2261, 1275) = 17.$$

Aside from some tedious arithmetic, that computation was systematic, and it was not necessary to factor any integers (which is something we do not know how to do quickly if the numbers involved have hundreds of digits).

Algorithm 1.1.12 (Greatest Common Division). Given integers a, b , this algorithm computes $\gcd(a, b)$.

1. [Assume $a > b \geq 0$] We have $\gcd(a, b) = \gcd(|a|, |b|) = \gcd(|b|, |a|)$, so we may replace a and b by their absolute value and hence assume $a, b \geq 0$. If $a = b$ output a and terminate. Swapping if necessary we assume $a > b$.

2. [Quotient and Remainder] Using Algorithm 1.1.11, write $a = bq + r$, with $0 \leq r < b$ and $q \in \mathbf{Z}$.
3. [Finished?] If $r = 0$ then $b \mid a$, so we output b and terminate.
4. [Shift and Repeat] Set $a \leftarrow b$ and $b \leftarrow r$, then go to step 2.

Proof. Lemmas 1.1.8–1.1.9 imply that $\gcd(a, b) = \gcd(b, r)$ so the gcd does not change in step 4. Since the remainders form a decreasing sequence of nonnegative integers, the algorithm terminates. \square

See Section 7.1.1 for an implementation of Algorithm 1.1.12.

Example 1.1.13. Set $a = 15$ and $b = 6$.

$$\begin{aligned} 15 &= 6 \cdot 2 + 3 & \gcd(15, 6) &= \gcd(6, 3) \\ 6 &= 3 \cdot 2 + 0 & \gcd(6, 3) &= \gcd(3, 0) = 3 \end{aligned}$$

Note that we can just as easily do an example that is ten times as big, an observation that will be important in the proof of Theorem 1.1.17 below.

Example 1.1.14. Set $a = 150$ and $b = 60$.

$$\begin{aligned} 150 &= 60 \cdot 2 + 30 & \gcd(150, 60) &= \gcd(60, 30) \\ 60 &= 30 \cdot 2 + 0 & \gcd(60, 30) &= \gcd(30, 0) = 30 \end{aligned}$$

Lemma 1.1.15. *For any integers a, b, n , we have*

$$\gcd(an, bn) = \gcd(a, b) \cdot n.$$

Proof. The idea is to follow Example 1.1.14; we step through Euclid's algorithm for $\gcd(an, bn)$ and note that at every step the equation is the equation from Euclid's algorithm for $\gcd(a, b)$ but multiplied through by n . For simplicity, assume that both a and b are positive. We will prove the lemma by induction on $a + b$. The statement is true in the base case when $a + b = 2$, since then $a = b = 1$. Now assume a, b are arbitrary with $a \leq b$. Let q and r be such that $a = bq + r$ and $0 \leq r < b$. Then by Lemmas 1.1.8–1.1.9, we have $\gcd(a, b) = \gcd(b, r)$. Multiplying $a = bq + r$ by n we see that $an = bnq + rn$, so $\gcd(an, bn) = \gcd(bn, rn)$. Then

$$b + r = b + (a - bq) = a - b(q - 1) \leq a < a + b,$$

so by induction $\gcd(bn, rn) = \gcd(b, r) \cdot n$. Since $\gcd(a, b) = \gcd(b, r)$, this proves the lemma. \square

Lemma 1.1.16. *Suppose $a, b, n \in \mathbf{Z}$ are such that $n \mid a$ and $n \mid b$. Then $n \mid \gcd(a, b)$.*

Proof. Since $n \mid a$ and $n \mid b$, there are integers c_1 and c_2 , such that $a = nc_1$ and $b = nc_2$. By Lemma 1.1.15, $\gcd(a, b) = \gcd(nc_1, nc_2) = n \gcd(c_1, c_2)$, so n divides $\gcd(a, b)$. \square

At this point it would be natural to formally analyze the complexity of Algorithm 1.1.12. We will not do this, because the main reason we introduced Algorithm 1.1.12 is that it will allow us to prove Theorem 1.1.5, and we have not chosen to formally analyze the complexity of the other algorithms in this book. For an extensive analysis of the complexity of Algorithm 1.1.12, see [Knu98, §4.5.3].

With Algorithm 1.1.12, we can prove that if a prime divides the product of two numbers, then it has got to divide one of them. This result is the key to proving that prime factorization is unique.

Theorem 1.1.17 (Euclid). *Let p be a prime and $a, b \in \mathbf{N}$. If $p \mid ab$ then $p \mid a$ or $p \mid b$.*

You might think this theorem is “intuitively obvious”, but that might be because the fundamental theorem of arithmetic (Theorem 1.1.5) is deeply ingrained in your intuition. Yet Theorem 1.1.17 will be needed in our proof of the fundamental theorem of arithmetic.

Proof of Theorem 1.1.17. If $p \mid a$ we are done. If $p \nmid a$ then $\gcd(p, a) = 1$, since only 1 and p divide p . By Lemma 1.1.15, $\gcd(pb, ab) = b$. Since $p \mid pb$ and, by hypothesis, $p \mid ab$, it follows from Lemma 1.1.15 that

$$p \mid \gcd(pb, ab) = b.$$

□

1.1.3 Numbers Factor as Products of Primes

In this section, we prove that every natural number factors as a product of primes. Then we discuss the difficulty of finding such a decomposition in practice. We will wait until Section 1.1.4 to prove that factorization is unique.

As a first example, let $n = 1275$. The sum of the digits of n is divisible by 3, so n is divisible by 3 (see Proposition 2.1.3), and we have $n = 3 \cdot 425$. The number 425 is divisible by 5, since its last digit is 5, and we have $1275 = 3 \cdot 5 \cdot 85$. Again, dividing 85 by 5, we have $1275 = 3 \cdot 5^2 \cdot 17$, which is the prime factorization of 1275. Generalizing this process proves the following proposition:

Proposition 1.1.18. *Every natural number is a product of primes.*

Proof. Let n be a natural number. If $n = 1$, then n is the empty product of primes. If n is prime, we are done. If n is composite, then $n = ab$ with $a, b < n$. By induction, a and b are products of primes, so n is also a product of primes. □

Two questions immediately arise: (1) is this factorization unique, and (2) how quickly can we find such a factorization? Addressing (1), what if

we had done something differently when breaking apart 1275 as a product of primes? Could the primes that show up be different? Let's try: we have $1275 = 5 \cdot 255$. Now $255 = 5 \cdot 51$ and $51 = 17 \cdot 3$, and again the factorization is the same, as asserted by Theorem 1.1.5 above. We will prove uniqueness of the prime factorization of any integer in Section 1.1.4.

Regarding (2), there are algorithms for integer factorization; e.g., in Sections 6.2 and 7.1.3 we will study and implement some of them. It is a major open problem to decide how fast integer factorization algorithms can be.

Open Problem 1.1.19. *Is there an algorithm which can factor any integer n in polynomial time? (See below for the meaning of polynomial time.)*

By *polynomial time* we mean that there is a polynomial $f(x)$ such that for any n the number of steps needed by the algorithm to factor n is less than $f(\log_{10}(n))$. Note that $\log_{10}(n)$ is an approximation for the number of digits of the input n to the algorithm.

Peter Shor [Sho97] devised a polynomial time algorithm for factoring integers on quantum computers. We will not discuss his algorithm further, except to note that in 2001 IBM researchers built a quantum computer that used Shor's algorithm to factor 15 (see [LMG⁺01, IBM01]).

You can earn money by factoring certain large integers. Many cryptosystems would be easily broken if factoring certain large integers were easy. Since nobody has proven that factoring integers is difficult, one way to increase confidence that factoring is difficult is to offer cash prizes for factoring certain integers. For example, until recently there was a \$10000 bounty on factoring the following 174-digit integer (see [RSA]):

```
1881988129206079638386972394616504398071635633794173827007
6335642298885971523466548531906060650474304531738801130339
6716199692321205734031879550656996221305168759307650257059
```

This number is known as RSA-576 since it has 576 digits when written in binary (see Section 2.3.2 for more on binary numbers). It was factored at the German Federal Agency for Information Technology Security in December 2003 (see [Wei03]):

```
398075086424064937397125500550386491199064362342526708406
385189575946388957261768583317
×
472772146107435302536223071973048224632914695302097116459
852171130520711256363590397527
```

The previous RSA challenge was the 155-digit number

```
1094173864157052742180970732204035761200373294544920599091
3842131476349984288934784717997257891267332497625752899781
833797076537244027146743531593354333897.
```

It was factored on 22 August 1999 by a group of sixteen researchers in four months on a cluster of 292 computers (see [ACD⁺99]). They found that RSA-155 is the product of the following two 78-digit primes:

$$\begin{aligned} p &= 10263959282974110577205419657399167590071656780803806 \\ &\quad 6803341933521790711307779 \\ q &= 10660348838016845482092722036001287867920795857598929 \\ &\quad 1522270608237193062808643. \end{aligned}$$

The next RSA challenge is RSA-640:

$$\begin{aligned} &31074182404900437213507500358885679300373460228427275457201619 \\ &48823206440518081504556346829671723286782437916272838033415471 \\ &07310850191954852900733772482278352574238645401469173660247765 \\ &2346609, \end{aligned}$$

and its factorization is worth \$20000.

These RSA numbers were factored using an algorithm called the number field sieve (see [LL93]), which is the best-known general purpose factorization algorithm. A description of how the number field sieve works is beyond the scope of this book. However, the number field sieve makes extensive use of the elliptic curve factorization method, which we will describe in Section 6.2.

1.1.4 The Fundamental Theorem of Arithmetic

We are ready to prove Theorem 1.1.5 using the following idea. Suppose we have two factorizations of n . Using Theorem 1.1.17 we cancel common primes from each factorization, one prime at a time. At the end, we discover that the factorizations must consist of exactly the same primes. The technical details are given below.

Proof. If $n = 1$, then the only factorization is the empty product of primes, so suppose $n > 1$.

By Proposition 1.1.18, there exist primes p_1, \dots, p_d such that

$$n = p_1 p_2 \cdots p_d.$$

Suppose that

$$n = q_1 q_2 \cdots q_m$$

is another expression of n as a product of primes. Since

$$p_1 \mid n = q_1(q_2 \cdots q_m),$$

Euclid's theorem implies that $p_1 = q_1$ or $p_1 \mid q_2 \cdots q_m$. By induction, we see that $p_1 = q_i$ for some i .

Now cancel p_1 and q_i , and repeat the above argument. Eventually, we find that, up to order, the two factorizations are the same. \square

1.2 The Sequence of Prime Numbers

This section is concerned with three questions:

1. Are there infinitely many primes?
2. Given $a, b \in \mathbf{Z}$, are there infinitely many primes of the form $ax + b$?
3. How are the primes spaced along the number line?

We first show that there are infinitely many primes, then state Dirichlet's theorem that if $\gcd(a, b) = 1$, then $ax + b$ is a prime for infinitely many values of x . Finally, we discuss the Prime Number Theorem which asserts that there are asymptotically $x/\log(x)$ primes less than x , and we make a connection between this asymptotic formula and the Riemann Hypothesis.

1.2.1 There Are Infinitely Many Primes

Each number on the left in the following table is prime. We will see soon that this pattern does not continue indefinitely, but something similar works.

$$\begin{aligned} 3 &= 2 + 1 \\ 7 &= 2 \cdot 3 + 1 \\ 31 &= 2 \cdot 3 \cdot 5 + 1 \\ 211 &= 2 \cdot 3 \cdot 5 \cdot 7 + 1 \\ 2311 &= 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 + 1 \end{aligned}$$

Theorem 1.2.1 (Euclid). *There are infinitely many primes.*

Proof. Suppose that p_1, p_2, \dots, p_n are n distinct primes. We construct a prime p_{n+1} not equal to any of p_1, \dots, p_n as follows. If

$$N = p_1 p_2 p_3 \cdots p_n + 1, \tag{1.2.1}$$

then by Proposition 1.1.18 there is a factorization

$$N = q_1 q_2 \cdots q_m$$

with each q_i prime and $m \geq 1$. If $q_1 = p_i$ for some i , then $p_i \mid N$. Because of (1.2.1), we also have $p_i \mid N - 1$, so $p_i \mid 1 = N - (N - 1)$, which is a contradiction. Thus the prime $p_{n+1} = q_1$ is not in the list p_1, \dots, p_n , and we have constructed our new prime. \square

For example,

$$2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 + 1 = 30031 = 59 \cdot 509.$$

Multiplying together the first 6 primes and adding 1 doesn't produce a prime, but it produces an integer that is merely divisible by a new prime.

Joke 1.2.2 (Hendrik Lenstra). *There are infinitely many composite numbers. Proof.* To obtain a new composite number, multiply together the first n composite numbers and don't add 1.

1.2.2 Enumerating Primes

The Sieve of Eratosthenes is an efficient way to enumerate all primes up to n . The sieve works by first writing down all numbers up to n , noting that 2 is prime, and crossing off all multiples of 2. Next, note that the first number not crossed off is 3, which is prime, and cross off all multiples of 3, etc. Repeating this process, we obtain a list of the primes up to n . Formally, the algorithm is as follows:

Algorithm 1.2.3 (Sieve of Eratosthenes). Given a positive integer n , this algorithm computes a list of the primes up to n .

1. [Initialize] Let $X \leftarrow [3, 5, \dots]$ be the list of all odd integers between 3 and n . Let $P \leftarrow [2]$ be the list of primes found so far.
2. [Finished?] Let p to be the first element of X . If $p \geq \sqrt{n}$, append each element of X to P and terminate. Otherwise append p to P .
3. [Cross Off] Set X equal to the sublist of elements in X that are not divisible by p . Go to step 2.

For example, to list the primes ≤ 40 using the sieve, we proceed as follows. First $P = [2]$ and

$$X = [3, 5, 7, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39].$$

We append 3 to P and cross off all multiples of 3 to obtain the new list

$$X = [5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37].$$

Next we append 5 to P , obtaining $P = [2, 3, 5]$, and cross off the multiples of 5, to obtain $X = [7, 11, 13, 17, 19, 23, 29, 31, 37]$. Because $7^2 \geq 40$, we append X to P and find that the primes less than 40 are

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37.$$

Proof of Algorithm 1.2.3. The part of the algorithm that is not clear is that when the first element a of X satisfies $a \geq \sqrt{n}$, then each element of X is prime. To see this, suppose m is in X , so $\sqrt{n} \leq m \leq n$ and that m is divisible by no prime that is $\leq \sqrt{n}$. Write $m = \prod p_i^{e_i}$ with the p_i distinct primes and $p_1 < p_2 < \dots$. If $p_i > \sqrt{n}$ for each i and there is more than one p_i , then $m > n$, a contradiction. Thus some p_i is less than \sqrt{n} , which also contradicts our assumptions on m . \square

See Section 7.1.2 for an implementation of Algorithm 1.2.3.

1.2.3 The Largest Known Prime

Though Theorem 1.2.1 implies that there are infinitely many primes, it still makes sense to ask the question “What is the largest *known* prime?”

A *Mersenne prime* is a prime of the form $2^q - 1$. According to [Cal] the largest known prime as of July 2004 is the Mersenne prime

$$p = 2^{24036583} - 1,$$

which has 7235733 decimal digits, so writing it out would fill over 10 books the size of this book. Euclid’s theorem implies that there definitely is a prime bigger than this 7.2 million digit p . Deciding whether or not a number is prime is interesting, both as a motivating problem and for applications to cryptography, as we will see in Section 2.4 and Chapter 3.

1.2.4 Primes of the Form $ax + b$

Next we turn to primes of the form $ax + b$, where a and b are fixed integers with $a > 1$ and x varies over the natural numbers \mathbf{N} . We assume that $\gcd(a, b) = 1$, because otherwise there is no hope that $ax + b$ is prime infinitely often. For example, $2x + 2 = 2(x + 1)$ is only prime if $x = 0$, and is not prime for any other $x \in \mathbf{N}$.

Proposition 1.2.4. *There are infinitely many primes of the form $4x - 1$.*

Why might this be true? We list numbers of the form $4x - 1$ and underline those that are prime:

$$\underline{3}, \underline{7}, \underline{11}, 15, \underline{19}, \underline{23}, 27, \underline{31}, 35, 39, \underline{43}, \underline{47}, \dots$$

It is plausible that underlined numbers would continue to appear indefinitely.

Proof. Suppose p_1, p_2, \dots, p_n are distinct primes of the form $4x - 1$. Consider the number

$$N = 4p_1p_2 \cdots p_n - 1.$$

Then $p_i \nmid N$ for any i . Moreover, not every prime $p \mid N$ is of the form $4x + 1$; if they all were, then N would be of the form $4x + 1$. Thus there is a $p \mid N$ that is of the form $4x - 1$. Since $p \neq p_i$ for any i , we have found a new prime of the form $4x - 1$. We can repeat this process indefinitely, so the set of primes of the form $4x - 1$ cannot be finite. \square

Note that this proof does not work if $4x - 1$ is replaced by $4x + 1$, since a product of primes of the form $4x - 1$ can be of the form $4x + 1$.

Example 1.2.5. Set $p_1 = 3$, $p_2 = 7$. Then

$$N = 4 \cdot 3 \cdot 7 - 1 = \underline{83}$$

is a prime of the form $4x - 1$. Next

$$N = 4 \cdot 3 \cdot 7 \cdot 83 - 1 = \underline{6971},$$

which is again a prime of the form $4x - 1$. Again:

$$N = 4 \cdot 3 \cdot 7 \cdot 83 \cdot 6971 - 1 = 48601811 = 61 \cdot \underline{796751}.$$

This time 61 is a prime, but it is of the form $4x + 1 = 4 \cdot 15 + 1$. However, 796751 is prime and $796751 = 4 \cdot 199188 - 1$. We are unstoppable:

$$N = 4 \cdot 3 \cdot 7 \cdot 83 \cdot 6971 \cdot 796751 - 1 = \underline{5591} \cdot 6926049421.$$

This time the small prime, 5591, is of the form $4x - 1$ and the large one is of the form $4x + 1$.

Theorem 1.2.6 (Dirichlet). *Let a and b be integers with $\gcd(a, b) = 1$. Then there are infinitely many primes of the form $ax + b$.*

Proofs of this theorem typically use tools from advanced number theory, and are beyond the scope of this book (see e.g., [FT93, §VIII.4]).

1.2.5 How Many Primes are There?

We saw in Section 1.2.1 that there are infinitely many primes. In order to get a sense for just how many primes there are, we consider a few warm-up questions. Then we consider some numerical evidence and state the prime number theorem, which gives an asymptotic answer to our question, and connect this theorem with a form of the Riemann Hypothesis. Our discussion of counting primes in this section is very cursory; for more details, read Crandall and Pomerance's excellent book [CP01, §1.1.5].

The following vague discussion is meant to motivate a precise way to measure the number of primes. How many natural numbers are even? Answer: Half of them. How many natural numbers are of the form $4x - 1$? Answer: One fourth of them. How many natural numbers are perfect squares? Answer: Zero percent of all natural numbers, in the sense that the limit of the proportion of perfect squares to all natural numbers converges to 0. More precisely,

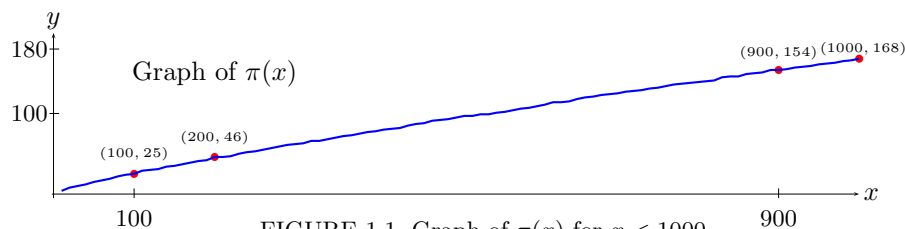
$$\lim_{x \rightarrow \infty} \frac{\#\{n \in \mathbf{N} : n \leq x \text{ and } n \text{ is a perfect square}\}}{x} = 0,$$

since the numerator is roughly \sqrt{x} and $\lim_{x \rightarrow \infty} \frac{\sqrt{x}}{x} = 0$. Likewise, it is an easy consequence of Theorem 1.2.8 below that zero percent of all natural numbers are prime (see Exercise 1.4).

We are thus led to ask another question: How many positive integers $\leq x$ are perfect squares? Answer: roughly \sqrt{x} . In the context of primes, we ask,

TABLE 1.1. Values of $\pi(x)$

x	100	200	300	400	500	600	700	800	900	1000
$\pi(x)$	25	46	62	78	95	109	125	139	154	168

FIGURE 1.1. Graph of $\pi(x)$ for $x < 1000$

Question 1.2.7. How many natural numbers $\leq x$ are prime?

Let

$$\pi(x) = \#\{p \in \mathbf{N} : p \leq x \text{ is a prime}\}.$$

For example,

$$\pi(6) = \#\{2, 3, 5\} = 3.$$

Some values of $\pi(x)$ are given in Table 1.1, and Figures 1.1 and 1.2 contain graphs of $\pi(x)$. These graphs look like straight lines, which maybe bend down slightly.

Gauss had a lifelong love of enumerating primes. Eventually he computed $\pi(3000000)$, though the author doesn't know whether or not Gauss got the right answer, which is 216816. Gauss conjectured the following asymptotic formula for $\pi(x)$, which was later proved independently by Hadamard and Vallée Poussin in 1896 (but will not be proved in this book):

Theorem 1.2.8 (Prime Number Theorem). *The function $\pi(x)$ is asymptotic to $x/\log(x)$, in the sense that*

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\log(x)} = 1.$$

We do nothing more here than motivate this deep theorem with a few further numerical observations.

The theorem implies that

$$\lim_{x \rightarrow \infty} \pi(x)/x = \lim_{x \rightarrow \infty} 1/\log(x) = 0,$$

so for any a ,

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/(\log(x) - a)} = \lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\log(x)} - \frac{a\pi(x)}{x} = 1.$$

Thus $x/(\log(x) - a)$ is also asymptotic to $\pi(x)$ for any a . See [CP01, §1.1.5] for a discussion of why $a = 1$ is the best choice. Table 1.2 compares $\pi(x)$ and $x/(\log(x) - 1)$ for several $x < 10000$.

TABLE 1.2. Comparison of $\pi(x)$ and $x/(\log(x) - 1)$

x	$\pi(x)$	$x/(\log(x) - 1)$ (approx)
1000	168	169.2690290604408165186256278
2000	303	302.9888734545463878029800994
3000	430	428.1819317975237043747385740
4000	550	548.3922097278253264133400985
5000	669	665.1418784486502172369455815
6000	783	779.2698885854778626863677374
7000	900	891.3035657223339974352567759
8000	1007	1001.602962794770080754784281
9000	1117	1110.428422963188172310675011
10000	1229	1217.976301461550279200775705

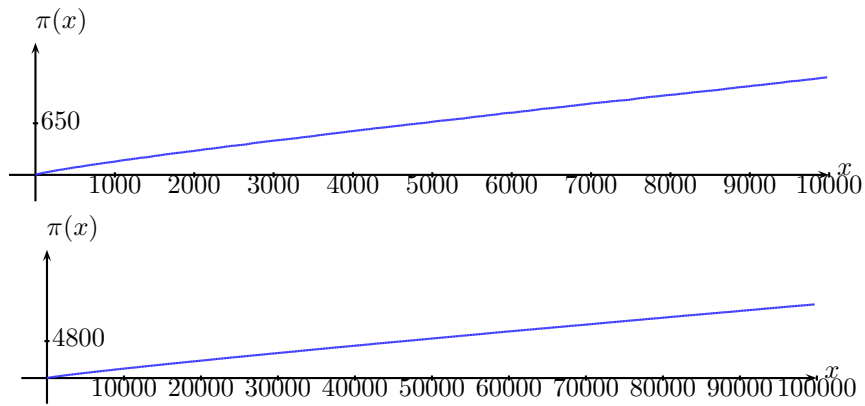


FIGURE 1.2. Graphs of $\pi(x)$ for $x < 10000$ and $x < 100000$

As of 2004, the record for counting primes appears to be

$$\pi(4 \cdot 10^{22}) = 783964159847056303858.$$

The computation of $\pi(4 \cdot 10^{22})$ reportedly took ten months on a 350 Mhz Pentium II (see [GS02] for more details).

For the reader familiar with complex analysis, we mention a connection between $\pi(x)$ and the Riemann Hypothesis. The Riemann zeta function $\zeta(s)$ is a complex analytic function on $\mathbf{C} \setminus \{1\}$ that extends the function defined on a right half plane by $\sum_{n=1}^{\infty} n^{-s}$. The Riemann Hypothesis is the conjecture that the zeros in \mathbf{C} of $\zeta(s)$ with positive real part lie on the line $\operatorname{Re}(s) = 1/2$. This conjecture is one of the Clay Math Institute million dollar millennium prize problems [Cla].

According to [CP01, §1.4.1], the Riemann Hypothesis is equivalent to the conjecture that

$$\operatorname{Li}(x) = \int_2^x \frac{1}{\log(t)} dt$$

is a “good” approximation to $\pi(x)$, in the following precise sense:

Conjecture 1.2.9 (Equivalent to the Riemann Hypothesis).

For all $x \geq 2.01$,

$$|\pi(x) - \operatorname{Li}(x)| \leq \sqrt{x} \log(x).$$

If $x = 2$, then $\pi(2) = 1$ and $\operatorname{Li}(2) = 0$, but $\sqrt{2} \log(2) = 0.9802 \dots$, so the inequality is not true for $x \geq 2$, but 2.01 is big enough. We will do nothing more to explain this conjecture, and settle for one numerical example.

Example 1.2.10. Let $x = 4 \cdot 10^{22}$. Then

$$\begin{aligned} \pi(x) &= 783964159847056303858, \\ \operatorname{Li}(x) &= 783964159852157952242.7155276025801473 \dots, \\ |\pi(x) - \operatorname{Li}(x)| &= 5101648384.71552760258014 \dots, \\ \sqrt{x} \log(x) &= 10408633281397.77913344605 \dots, \\ x/(\log(x) - 1) &= 783650443647303761503.5237113087392967 \dots \end{aligned}$$

1.3 Exercises

- 1.1 Compute the greatest common divisor $\operatorname{gcd}(455, 1235)$ by hand.
- 1.2 Use the Sieve of Eratosthenes to make a list of all primes up to 100.
- 1.3 Prove that there are infinitely many primes of the form $6x - 1$.
- 1.4 Use Theorem 1.2.8 to deduce that $\lim_{x \rightarrow \infty} \frac{\pi(x)}{x} = 0$.

2

The Ring of Integers Modulo n

This chapter is about the ring $\mathbf{Z}/n\mathbf{Z}$ of integers modulo n . First we discuss when linear equations modulo n have a solution, then introduce the Euler φ function and prove Fermat's Little Theorem and Wilson's theorem. Next we prove the Chinese Remainder Theorem, which addresses simultaneous solubility of several linear equations modulo coprime moduli. With these theoretical foundations in place, in Section 2.3 we introduce algorithms for doing interesting computations modulo n , including computing large powers quickly, and solving linear equations. We finish with a very brief discussion of finding prime numbers using arithmetic modulo n .

2.1 Congruences Modulo n

In this section we define the ring $\mathbf{Z}/n\mathbf{Z}$ of integers modulo n , introduce the Euler φ -function, and relate it to the multiplicative order of certain elements of $\mathbf{Z}/n\mathbf{Z}$.

If $a, b \in \mathbf{Z}$ and $n \in \mathbf{N}$, we say that a is *congruent to b modulo n* if $n \mid a - b$, and write $a \equiv b \pmod{n}$. Let $n\mathbf{Z} = (n)$ be the ideal of \mathbf{Z} generated by n .

Definition 2.1.1 (Integers Modulo n). The *ring of integers modulo n* is the quotient ring $\mathbf{Z}/n\mathbf{Z}$ of equivalence classes of integers modulo n . It is equipped with its natural ring structure:

$$(a + n\mathbf{Z}) + (b + n\mathbf{Z}) = (a + b) + n\mathbf{Z}$$

$$(a + n\mathbf{Z}) \cdot (b + n\mathbf{Z}) = (a \cdot b) + n\mathbf{Z}.$$

Example 2.1.2. For example,

$$\mathbf{Z}/3\mathbf{Z} = \{\{\dots, -3, 0, 3, \dots\}, \{\dots, -2, 1, 4, \dots\}, \{\dots, -1, 2, 5, \dots\}\}$$

We use the notation $\mathbf{Z}/n\mathbf{Z}$ because $\mathbf{Z}/n\mathbf{Z}$ is the quotient of the ring \mathbf{Z} by the ideal $n\mathbf{Z}$ of multiples of n . Because $\mathbf{Z}/n\mathbf{Z}$ is the quotient of a ring by an ideal, the ring structure on \mathbf{Z} induces a ring structure on $\mathbf{Z}/n\mathbf{Z}$. We often let a or $a \pmod{n}$ denote the equivalence class $a + n\mathbf{Z}$ of a . If p is a prime, then $\mathbf{Z}/p\mathbf{Z}$ is a field (see Exercise 2.11).

We call the natural reduction map $\mathbf{Z} \rightarrow \mathbf{Z}/n\mathbf{Z}$, which sends a to $a + n\mathbf{Z}$, *reduction modulo n* . We also say that a is a *lift* of $a + n\mathbf{Z}$. Thus, e.g., 7 is a lift of $1 \pmod{3}$, since $7 + 3\mathbf{Z} = 1 + 3\mathbf{Z}$.

We can use that arithmetic in $\mathbf{Z}/n\mathbf{Z}$ is well defined to derive tests for divisibility by n (see Exercise 2.7).

Proposition 2.1.3. *A number $n \in \mathbf{Z}$ is divisible by 3 if and only if the sum of the digits of n is divisible by 3.*

Proof. Write

$$n = a + 10b + 100c + \dots,$$

where the digits of n are a, b, c , etc. Since $10 \equiv 1 \pmod{3}$,

$$n = a + 10b + 100c + \dots \equiv a + b + c + \dots \pmod{3},$$

from which the proposition follows. \square

2.1.1 Linear Equations Modulo n

In this section, we are concerned with how to decide whether or not a linear equation of the form $ax \equiv b \pmod{n}$ has a solution modulo n . Algorithms for *computing* solutions to $ax \equiv b \pmod{n}$ are the topic of Section 2.3.

First we prove a proposition that gives a criterion under which one can cancel a quantity from both sides of a congruence.

Proposition 2.1.4 (Cancellation). *If $\gcd(c, n) = 1$ and*

$$ac \equiv bc \pmod{n},$$

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

Proof. By definition

$$n \mid ac - bc = (a - b)c.$$

Since $\gcd(n, c) = 1$, it follows from Theorem 1.1.5 that $n \mid a - b$, so

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

as claimed. \square

When a has a multiplicative inverse a' in $\mathbf{Z}/n\mathbf{Z}$ (i.e., $aa' \equiv 1 \pmod{n}$) then the equation $ax \equiv b \pmod{n}$ has a unique solution $x \equiv a'b \pmod{n}$ modulo n . Thus, it is of interest to determine the units in $\mathbf{Z}/n\mathbf{Z}$, i.e., the elements which have a multiplicative inverse.

We will use complete sets of residues to prove that the units in $\mathbf{Z}/n\mathbf{Z}$ are exactly the $a \in \mathbf{Z}/n\mathbf{Z}$ such that $\gcd(\tilde{a}, n) = 1$ for any lift \tilde{a} of a to \mathbf{Z} (it doesn't matter which lift).

Definition 2.1.5 (Complete Set of Residues). We call a subset $R \subset \mathbf{Z}$ of size n whose reductions modulo n are pairwise distinct a *complete set of residues* modulo n . In other words, a complete set of residues is a choice of representative for each equivalence class in $\mathbf{Z}/n\mathbf{Z}$.

For example,

$$R = \{0, 1, 2, \dots, n-1\}$$

is a complete set of residues modulo n . When $n = 5$, $R = \{0, 1, -1, 2, -2\}$ is a complete set of residues.

Lemma 2.1.6. *If R is a complete set of residues modulo n and $a \in \mathbf{Z}$ with $\gcd(a, n) = 1$, then $aR = \{ax : x \in R\}$ is also a complete set of residues modulo n .*

Proof. If $ax \equiv ax' \pmod{n}$ with $x, x' \in R$, then Proposition 2.1.4 implies that $x \equiv x' \pmod{n}$. Because R is a complete set of residues, this implies that $x = x'$. Thus the elements of aR have distinct reductions modulo n . It follows, since $\#aR = n$, that aR is a complete set of residues modulo n . \square

Proposition 2.1.7 (Units). *If $\gcd(a, n) = 1$, then the equation $ax \equiv b \pmod{n}$ has a solution, and that solution is unique modulo n .*

Proof. Let R be a complete set of residues modulo n , so there is a unique element of R that is congruent to b modulo n . By Lemma 2.1.6, aR is also a complete set of residues modulo n , so there is a unique element $ax \in aR$ that is congruent to b modulo n , and we have $ax \equiv b \pmod{n}$. \square

Algebraically, this proposition asserts that if $\gcd(a, n) = 1$, then the map $\mathbf{Z}/n\mathbf{Z} \rightarrow \mathbf{Z}/n\mathbf{Z}$ given by left multiplication by a is a bijection.

Example 2.1.8. Consider the equation $2x \equiv 3 \pmod{7}$, and the complete set $R = \{0, 1, 2, 3, 4, 5, 6\}$ of coset representatives. We have

$$2R = \{0, 2, 4, 6, 8 \equiv 1, 10 \equiv 3, 12 \equiv 5\},$$

so $2 \cdot 5 \equiv 3 \pmod{7}$.

When $\gcd(a, n) \neq 1$, then the equation $ax \equiv b \pmod{n}$ may or may not have a solution. For example, $2x \equiv 1 \pmod{4}$ has no solution, but $2x \equiv 2 \pmod{4}$ does, and in fact it has more than one mod 4 ($x = 1$ and $x = 3$). Generalizing Proposition 2.1.7, we obtain the following more general criterion for solvability.

Proposition 2.1.9 (Solvability). *The equation $ax \equiv b \pmod{n}$ has a solution if and only if $\gcd(a, n)$ divides b .*

Proof. Let $g = \gcd(a, n)$. If there is a solution x to the equation $ax \equiv b \pmod{n}$, then $n \mid (ax - b)$. Since $g \mid n$ and $g \mid a$, it follows that $g \mid b$.

Conversely, suppose that $g \mid b$. Then $n \mid (ax - b)$ if and only if

$$\frac{n}{g} \mid \left(\frac{a}{g}x - \frac{b}{g} \right).$$

Thus $ax \equiv b \pmod{n}$ has a solution if and only if $\frac{a}{g}x \equiv \frac{b}{g} \pmod{\frac{n}{g}}$ has a solution. Since $\gcd(a/g, n/g) = 1$, Proposition 2.1.7 implies this latter equation does have a solution. \square

In Chapter 4 we will study quadratic reciprocity, which gives a nice criterion for whether or not a quadratic equation modulo n has a solution.

2.1.2 Fermat's Little Theorem

The group of units $(\mathbf{Z}/n\mathbf{Z})^*$ of the ring $\mathbf{Z}/n\mathbf{Z}$ will be of great interest to us. Each element of this group has an order, and Lagrange's theorem from group theory implies that each element of $(\mathbf{Z}/n\mathbf{Z})^*$ has order that divides the order of $(\mathbf{Z}/n\mathbf{Z})^*$. In elementary number theory this fact goes by the monicker "Fermat's Little Theorem", and we reprove it from basic principles in this section.

Definition 2.1.10 (Order of an Element). Let $n \in \mathbf{N}$ and $x \in \mathbf{Z}$ and suppose that $\gcd(x, n) = 1$. The *order* of x modulo n is the smallest $m \in \mathbf{N}$ such that

$$x^m \equiv 1 \pmod{n}.$$

To show that the definition makes sense, we verify that such an m exists. Consider x, x^2, x^3, \dots modulo n . There are only finitely many residue classes modulo n , so we must eventually find two integers i, j with $i < j$ such that

$$x^j \equiv x^i \pmod{n}.$$

Since $\gcd(x, n) = 1$, Proposition 2.1.4 implies that we can cancel x 's and conclude that

$$x^{j-i} \equiv 1 \pmod{n}.$$

Definition 2.1.11 (Euler's phi-function). For $n \in \mathbf{N}$, let

$$\varphi(n) = \#\{a \in \mathbf{N} : a \leq n \text{ and } \gcd(a, n) = 1\}.$$

For example,

$$\begin{aligned}\varphi(1) &= \#\{1\} = 1, \\ \varphi(2) &= \#\{1\} = 1, \\ \varphi(5) &= \#\{1, 2, 3, 4\} = 4, \\ \varphi(12) &= \#\{1, 5, 7, 11\} = 4.\end{aligned}$$

Also, if p is any prime number then

$$\varphi(p) = \#\{1, 2, \dots, p-1\} = p-1.$$

In Section 2.2.1, we will prove that φ is a multiplicative function. This will yield an easy way to compute $\varphi(n)$ in terms of the prime factorization of n .

Theorem 2.1.12 (Fermat's Little Theorem). *If $\gcd(x, n) = 1$, then*

$$x^{\varphi(n)} \equiv 1 \pmod{n}.$$

Proof. As mentioned above, Fermat's Little Theorem has the following group-theoretic interpretation. The set of units in $\mathbf{Z}/n\mathbf{Z}$ is a group

$$(\mathbf{Z}/n\mathbf{Z})^* = \{a \in \mathbf{Z}/n\mathbf{Z} : \gcd(a, n) = 1\}.$$

which has order $\varphi(n)$. The theorem then asserts that the order of an element of $(\mathbf{Z}/n\mathbf{Z})^*$ divides the order $\varphi(n)$ of $(\mathbf{Z}/n\mathbf{Z})^*$. This is a special case of the more general fact (Lagrange's theorem) that if G is a finite group and $g \in G$, then the order of g divides the cardinality of G .

We now give an elementary proof of the theorem. Let

$$P = \{a : 1 \leq a \leq n \text{ and } \gcd(a, n) = 1\}.$$

In the same way that we proved Lemma 2.1.6, we see that the reductions modulo n of the elements of xP are the same as the reductions of the elements of P . Thus

$$\prod_{a \in P} (xa) \equiv \prod_{a \in P} a \pmod{n},$$

since the products are over the same numbers modulo n . Now cancel the a 's on both sides to get

$$x^{\#P} \equiv 1 \pmod{n},$$

as claimed. □

2.1.3 Wilson's Theorem

The following characterization of prime numbers, from the 1770s, is called "Wilson's Theorem", though it was first proved by Lagrange.

Proposition 2.1.13 (Wilson's Theorem). *An integer $p > 1$ is prime if and only if $(p - 1)! \equiv -1 \pmod{p}$.*

For example, if $p = 3$, then $(p - 1)! = 2 \equiv -1 \pmod{3}$. If $p = 17$, then

$$(p - 1)! = 20922789888000 \equiv -1 \pmod{17}.$$

But if $p = 15$, then

$$(p - 1)! = 87178291200 \equiv 0 \pmod{15},$$

so 15 is composite. Thus Wilson's theorem could be viewed as a primality test, though, from a computational point of view, it is probably the *least efficient* primality test since computing $(n - 1)!$ takes so many steps.

Proof. The statement is clear when $p = 2$, so henceforth we assume that $p > 2$. We first assume that p is prime and prove that $(p - 1)! \equiv -1 \pmod{p}$. If $a \in \{1, 2, \dots, p - 1\}$ then the equation

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

has a unique solution $a' \in \{1, 2, \dots, p - 1\}$. If $a = a'$, then $a^2 \equiv 1 \pmod{p}$, so $p \mid a^2 - 1 = (a - 1)(a + 1)$, so $p \mid (a - 1)$ or $p \mid (a + 1)$, so $a \in \{1, p - 1\}$. We can thus pair off the elements of $\{2, 3, \dots, p - 2\}$, each with their inverse. Thus

$$2 \cdot 3 \cdot \dots \cdot (p - 2) \equiv 1 \pmod{p}.$$

Multiplying both sides by $p - 1$ proves that $(p - 1)! \equiv -1 \pmod{p}$.

Next we assume that $(p - 1)! \equiv -1 \pmod{p}$ and prove that p must be prime. Suppose not, so that $p \geq 4$ is a composite number. Let ℓ be a prime divisor of p . Then $\ell < p$, so $\ell \mid (p - 1)!$. Also, by assumption,

$$\ell \mid p \mid ((p - 1)! + 1).$$

This is a contradiction, because a prime can not divide a number a and also divide $a + 1$, since it would then have to divide $(a + 1) - a = 1$. \square

Example 2.1.14. We illustrate the key step in the above proof in the case $p = 17$. We have

$$2 \cdot 3 \cdot \dots \cdot 15 = (2 \cdot 9) \cdot (3 \cdot 6) \cdot (4 \cdot 13) \cdot (5 \cdot 7) \cdot (8 \cdot 15) \cdot (10 \cdot 12) \cdot (14 \cdot 11) \equiv 1 \pmod{17},$$

where we have paired up the numbers a, b for which $ab \equiv 1 \pmod{17}$.

2.2 The Chinese Remainder Theorem

In this section we prove the Chinese Remainder Theorem, which gives conditions under which a system of linear equations is guaranteed to have a solution. In the 4th century a Chinese mathematician asked the following:

Question 2.2.1. There is a quantity whose number is unknown. Repeatedly divided by 3, the remainder is 2; by 5 the remainder is 3; and by 7 the remainder is 2. What is the quantity?

In modern notation, Question 2.2.1 asks us to find a positive integer solution to the following system of three equations:

$$\begin{aligned}x &\equiv 2 \pmod{3} \\x &\equiv 3 \pmod{5} \\x &\equiv 2 \pmod{7}\end{aligned}$$

The Chinese Remainder Theorem asserts that a solution exists, and the proof gives a method to find one. (See Section 2.3 for the necessary algorithms.)

Theorem 2.2.2 (Chinese Remainder Theorem). *Let $a, b \in \mathbf{Z}$ and $n, m \in \mathbf{N}$ such that $\gcd(n, m) = 1$. Then there exists $x \in \mathbf{Z}$ such that*

$$\begin{aligned}x &\equiv a \pmod{m}, \\x &\equiv b \pmod{n}.\end{aligned}$$

Moreover x is unique modulo mn .

Proof. If we can solve for t in the equation

$$a + tm \equiv b \pmod{n},$$

then $x = a + tm$ will satisfy both congruences. To see that we can solve, subtract a from both sides and use Proposition 2.1.7 together with our assumption that $\gcd(n, m) = 1$ to see that there is a solution.

For uniqueness, suppose that x and y solve both congruences. Then $z = x - y$ satisfies $z \equiv 0 \pmod{m}$ and $z \equiv 0 \pmod{n}$, so $m \mid z$ and $n \mid z$. Since $\gcd(n, m) = 1$, it follows that $nm \mid z$, so $x \equiv y \pmod{nm}$. \square

Algorithm 2.2.3 (Chinese Remainder Theorem). Given coprime integers m and n and integers a and b , this algorithm find an integer x such that $x \equiv a \pmod{m}$ and $x \equiv b \pmod{n}$.

1. [Extended GCD] Use Algorithm 2.3.3 below to find integers c, d such that $cm + dn = 1$.
2. [Answer] Output $x = a + (b - a)cm$ and terminate.

Proof. Since $c \in \mathbf{Z}$, we have $x \equiv a \pmod{m}$, and using that $cm + dn = 1$, we have $a + (b - a)cm \equiv a + (b - a) \equiv b \pmod{n}$. \square

Now we can answer Question 2.2.1. First, we use Theorem 2.2.2 to find a solution to the pair of equations

$$\begin{aligned}x &\equiv 2 \pmod{3}, \\x &\equiv 3 \pmod{5}.\end{aligned}$$

Set $a = 2$, $b = 3$, $m = 3$, $n = 5$. Step 1 is to find a solution to $t \cdot 3 \equiv 3 - 2 \pmod{5}$. A solution is $t = 2$. Then $x = a + tm = 2 + 2 \cdot 3 = 8$. Since any x' with $x' \equiv x \pmod{15}$ is also a solution to those two equations, we can solve all three equations by finding a solution to the pair of equations

$$\begin{aligned}x &\equiv 8 \pmod{15} \\x &\equiv 2 \pmod{7}.\end{aligned}$$

Again, we find a solution to $t \cdot 15 \equiv 2 - 8 \pmod{7}$. A solution is $t = 1$, so

$$x = a + tm = 8 + 15 = 23.$$

Note that there are other solutions. Any $x' \equiv x \pmod{3 \cdot 5 \cdot 7}$ is also a solution; e.g., $23 + 3 \cdot 5 \cdot 7 = 128$.

2.2.1 Multiplicative Functions

Definition 2.2.4 (Multiplicative Function). A function $f : \mathbf{N} \rightarrow \mathbf{Z}$ is *multiplicative* if, whenever $m, n \in \mathbf{N}$ and $\gcd(m, n) = 1$, we have

$$f(mn) = f(m) \cdot f(n).$$

Recall from Definition 2.1.11 that the *Euler φ -function* is

$$\varphi(n) = \#\{a : 1 \leq a \leq n \text{ and } \gcd(a, n) = 1\}.$$

Lemma 2.2.5. *Suppose that $m, n \in \mathbf{N}$ and $\gcd(m, n) = 1$. Then the map*

$$\psi : (\mathbf{Z}/mn\mathbf{Z})^* \rightarrow (\mathbf{Z}/m\mathbf{Z})^* \times (\mathbf{Z}/n\mathbf{Z})^*. \quad (2.2.1)$$

defined by

$$\psi(c) = (c \bmod m, c \bmod n)$$

is a bijection.

Proof. We first show that ψ is injective. If $\psi(c) = \psi(c')$, then $m \mid c - c'$ and $n \mid c - c'$, so $nm \mid c - c'$ because $\gcd(n, m) = 1$. Thus $c = c'$ as elements of $(\mathbf{Z}/mn\mathbf{Z})^*$.

Next we show that ψ is surjective. Given a and b with $\gcd(a, m) = 1$ and $\gcd(b, n) = 1$, Theorem 2.2.2 implies that there exists c with $c \equiv a \pmod{m}$ and $c \equiv b \pmod{n}$. We may assume that $1 \leq c \leq nm$, and since $\gcd(a, m) = 1$ and $\gcd(b, n) = 1$, we must have $\gcd(c, nm) = 1$. Thus $\psi(c) = (a, b)$. \square

Proposition 2.2.6 (Multiplicativity of φ). *The function φ is multiplicative.*

Proof. The map ψ of Lemma 2.2.5 is a bijection, so the set on the left in (2.2.1) has the same size as the product set on the right in (2.2.1). Thus

$$\varphi(mn) = \varphi(m) \cdot \varphi(n).$$

□

The proposition is helpful in computing $\varphi(n)$, at least if we assume we can compute the factorization of n (see Section 3.3.1 for a connection between factoring n and computing $\varphi(n)$). For example,

$$\varphi(12) = \varphi(2^2) \cdot \varphi(3) = 2 \cdot 2 = 4.$$

Also, for $n \geq 1$, we have

$$\varphi(p^n) = p^n - \frac{p^n}{p} = p^n - p^{n-1} = p^{n-1}(p-1), \quad (2.2.2)$$

since $\varphi(p^n)$ is the number of numbers less than p^n minus the number of those that are divisible by p . Thus, e.g.,

$$\varphi(389 \cdot 11^2) = 388 \cdot (11^2 - 11) = 388 \cdot 110 = 42680.$$

2.3 Quickly Computing Inverses and Huge Powers

This section is about how to solve the equation $ax \equiv 1 \pmod{n}$ when we know it has a solution, and how to efficiently compute $a^m \pmod{n}$. We also discuss a simple probabilistic primality test that relies on our ability to compute $a^m \pmod{n}$ quickly. All three of these algorithms are of fundamental importance to the cryptography algorithms of Chapter 3.

2.3.1 How to Solve $ax \equiv 1 \pmod{n}$

Suppose $a, n \in \mathbf{N}$ with $\gcd(a, n) = 1$. Then by Proposition 2.1.7 the equation $ax \equiv 1 \pmod{n}$ has a unique solution. How can we find it?

Proposition 2.3.1 (Extended Euclidean representation). *Suppose $a, b \in \mathbf{Z}$ and let $g = \gcd(a, b)$. Then there exists $x, y \in \mathbf{Z}$ such that*

$$ax + by = g.$$

Remark 2.3.2. If $e = cg$ is a multiple of g , then $cax + cby = cg = e$, so $e = (cx)a + (cy)b$ can also be written in terms of a and b .

Proof of Proposition 2.3.1. Let $g = \gcd(a, b)$. Then $\gcd(a/d, b/d) = 1$, so by Proposition 2.1.9 the equation

$$\frac{a}{g} \cdot x \equiv 1 \pmod{\frac{b}{g}} \quad (2.3.1)$$

has a solution $x \in \mathbf{Z}$. Multiplying (2.3.1) through by g yields $ax \equiv g \pmod{b}$, so there exists y such that $b \cdot (-y) = ax - g$. Then $ax + by = g$, as required. \square

Given a, b and $g = \gcd(a, b)$, our proof of Proposition 2.3.1 gives a way to explicitly find x, y such that $ax + by = g$, assuming one knows an algorithm to solve linear equations modulo n . Since we do not know such an algorithm, we now discuss a way to explicitly find x and y . This algorithm will in fact enable us to solve linear equations modulo n —to solve $ax \equiv 1 \pmod{n}$ when $\gcd(a, n) = 1$, use the algorithm below to find x and y such that $ax + ny = 1$. Then $ax \equiv 1 \pmod{n}$.

Suppose $a = 5$ and $b = 7$. The steps of Algorithm 1.1.12 to compute $\gcd(5, 7)$ are, as follows. Here we underline, because it clarifies the subsequent back substitution we will use to find x and y .

$$\begin{aligned} \underline{7} &= 1 \cdot \underline{5} + \underline{2} & \text{so } \underline{2} &= \underline{7} - \underline{5} \\ \underline{5} &= 2 \cdot \underline{2} + \underline{1} & \text{so } \underline{1} &= \underline{5} - 2 \cdot \underline{2} = \underline{5} - 2(\underline{7} - \underline{5}) = 3 \cdot \underline{5} - 2 \cdot \underline{7} \end{aligned}$$

On the right, we have back-substituted in order to write each partial remainder as a linear combination of a and b . In the last step, we obtain $\gcd(a, b)$ as a linear combination of a and b , as desired.

That example was not too complicated, so we try another one. Let $a = 130$ and $b = 61$. We have

$$\begin{aligned} \underline{130} &= 2 \cdot \underline{61} + \underline{8} & \underline{8} &= \underline{130} - 2 \cdot \underline{61} \\ \underline{61} &= 7 \cdot \underline{8} + \underline{5} & \underline{5} &= -7 \cdot \underline{130} + 15 \cdot \underline{61} \\ \underline{8} &= 1 \cdot \underline{5} + \underline{3} & \underline{3} &= 8 \cdot \underline{130} - 17 \cdot \underline{61} \\ \underline{5} &= 1 \cdot \underline{3} + \underline{2} & \underline{2} &= -15 \cdot \underline{130} + 32 \cdot \underline{61} \\ \underline{3} &= 1 \cdot \underline{2} + \underline{1} & \underline{1} &= 23 \cdot \underline{130} - 49 \cdot \underline{61} \end{aligned}$$

Thus $x = 23$ and $y = -49$ is a solution to $130x + 61y = 1$.

Algorithm 2.3.3 (Extended Euclidean Algorithm). Suppose a and b are integers and let $g = \gcd(a, b)$. This algorithm finds d, x and y such that $ax + by = g$. We describe only the steps when $a > b \geq 0$, since one can easily reduce to this case.

1. [Initialize] Set $x \leftarrow 1, y \leftarrow 0, r \leftarrow 0, s \leftarrow 1$.
2. [Finished?] If $b = 0$, set $g \leftarrow a$ and terminate.

3. [Quotient and Remainder] Use Algorithm 1.1.11 to write $a = qb + c$ with $0 \leq c < b$.
4. [Shift] Set $(a, b, r, s, x, y) \leftarrow (b, c, x - qr, y - qs, r, s)$ and go to step 2.

Proof. This algorithm is the same as Algorithm 1.1.12, except that we keep track of extra variables x, y, r, s , so it terminates and when it terminates $d = \gcd(a, b)$. We omit the rest of the inductive proof that the algorithm is correct, and instead refer the reader to [Knu97, §1.2.1] which contains a detailed proof in the context of a discussion of how one writes mathematical proofs. \square

Algorithm 2.3.4 (Inverse Modulo n). Suppose a and n are integers and $\gcd(a, n) = 1$. This algorithm finds an x such that $ax \equiv 1 \pmod{n}$.

1. [Compute Extended GCD] Use Algorithm 2.3.3 to compute integers x, y such that $ax + ny = \gcd(a, n) = 1$.
2. [Finished] Output x .

Proof. Reduce $ax + ny = 1$ modulo n to see that x satisfies $ax \equiv 1 \pmod{n}$. \square

See Section 7.2.1 for implementations of Algorithms 2.3.3 and 2.3.4.

Example 2.3.5. Solve $17x \equiv 1 \pmod{61}$. First, we use Algorithm 2.3.3 to find x, y such that $17x + 61y = 1$:

$$\begin{array}{ll} \underline{61} = 3 \cdot \underline{17} + \underline{10} & \underline{10} = \underline{61} - 3 \cdot \underline{17} \\ \underline{17} = 1 \cdot \underline{10} + \underline{7} & \underline{7} = -\underline{61} + 4 \cdot \underline{17} \\ \underline{10} = 1 \cdot \underline{7} + \underline{3} & \underline{3} = 2 \cdot \underline{61} - 7 \cdot \underline{17} \\ \underline{3} = 2 \cdot \underline{3} + \underline{1} & \underline{1} = -5 \cdot \underline{61} + 18 \cdot \underline{17} \end{array}$$

Thus $17 \cdot 18 + 61 \cdot (-5) = 1$ so $x = 18$ is a solution to $17x \equiv 1 \pmod{61}$.

2.3.2 How to Compute $a^m \pmod{n}$

Let a and n be integers, and m a nonnegative integer. In this section we describe an efficient algorithm to compute $a^m \pmod{n}$. For the cryptography applications in Chapter 3, m will have hundreds of digits.

The naive approach to computing $a^m \pmod{n}$ is to simply compute $a^m = a \cdot a \cdots a \pmod{n}$ by repeatedly multiplying by a and reducing modulo n . Note that after each arithmetic operation is completed, we reduce the result modulo n so that the sizes of the numbers involved do not get too large. Nonetheless, this algorithm is horribly inefficient because it takes $m - 1$ multiplications, which is huge if m has hundreds of digits.

A much more efficient algorithm for computing $a^m \pmod{n}$ involves writing m in binary, then expressing a^m as a product of expressions a^{2^i} , for

various i . These latter expressions can be computed by repeatedly squaring a^{2^i} . This more clever algorithm is not “simpler”, but it is vastly more efficient since the number of operations needed grows with the number of binary digits of m , whereas with the naive algorithm above the number of operations is $m - 1$.

Algorithm 2.3.6 (Write a number in binary). Let m be a nonnegative integer. This algorithm writes m in binary, so it finds $\varepsilon_i \in \{0, 1\}$ such that $m = \sum_{i=0}^r \varepsilon_i 2^i$ with each $\varepsilon_i \in \{0, 1\}$.

1. [Initialize] Set $i \leftarrow 0$.
2. [Finished?] If $m = 0$, terminate.
3. [Digit] If m is odd, set $\varepsilon_i \leftarrow 1$, otherwise $\varepsilon_i \leftarrow 0$. Increment i .
4. [Divide by 2] Set $m \leftarrow \lfloor \frac{m}{2} \rfloor$, the greatest integer $\leq m/2$. Goto step 2.

Algorithm 2.3.7 (Compute Power). Let a and n be integers and m a nonnegative integer. This algorithm computes a^m modulo n .

1. [Write in Binary] Write m in binary using Algorithm 2.3.6, so $a^m = \prod_{\varepsilon_i=1} a^{2^i} \pmod{n}$.
2. [Compute Powers] Compute $a, a^2, a^{2^2} = (a^2)^2, a^{2^3} = (a^{2^2})^2$, etc., up to a^{2^r} , where $r + 1$ is the number of binary digits of m .
3. [Multiply Powers] Multiply together the a^{2^i} such that $\varepsilon_i = 1$, always working modulo n .

See Section 7.2.2 for an implementation of Algorithms 2.3.6 and 2.3.7.

We can compute the last 2 digits of 6^{91} , by finding $6^{91} \pmod{100}$. Make a table whose first column, labeled i , contains 0, 1, 2, etc. The second column, labeled m , is got by dividing the entry above it by 2 and taking the integer part of the result. The third column, labeled ε_i , records whether or not the second column is odd. The fourth column is computed by squaring, modulo $n = 100$, the entry above it.

i	m	ε_i	$6^{2^i} \pmod{100}$
0	91	1	6
1	45	1	36
2	22	0	96
3	11	1	16
4	5	1	56
5	2	0	36
6	1	1	96

We have

$$6^{91} \equiv 6^{2^6} \cdot 6^{2^4} \cdot 6^{2^3} \cdot 6^2 \cdot 6 \equiv 96 \cdot 56 \cdot 16 \cdot 36 \cdot 6 \equiv 56 \pmod{100}.$$

That is easier than multiplying 6 by itself 91 times.

Remark 2.3.8. Alternatively, we could simplify the computation using Theorem 2.1.12. By that theorem, $6^{\varphi(100)} \equiv 1 \pmod{100}$, so since $\varphi(100) = \varphi(2^2 \cdot 5^2) = (2^2 - 2) \cdot (5^2 - 5) = 40$, we have $6^{91} \equiv 6^{11} \pmod{100}$.

2.4 Finding Primes

Theorem 2.4.1 (Pseudoprimality). *An integer $p > 1$ is prime if and only if for every $a \not\equiv 0 \pmod{p}$,*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof. If p is prime, then the statement follows from Proposition 2.1.13. If p is composite, then there is a divisor a of p with $a \neq 1, p$. If $a^{p-1} \equiv 1 \pmod{p}$, then $p \mid a^{p-1} - 1$. Since $a \mid p$, we have $a \mid a^{p-1} - 1$ hence $a \mid 1$, a contradiction. \square

Suppose $n \in \mathbf{N}$. Using this theorem and Algorithm 2.3.7, we can either quickly prove that n is not prime, or convince ourselves that n is likely prime (but not quickly prove that n is prime). For example, if $2^{n-1} \not\equiv 1 \pmod{n}$, then we have proved that n is not prime. On the other hand, if $a^{n-1} \equiv 1 \pmod{n}$ for a few a , it “seems likely” that n is prime, and we loosely refer to such a number that seems prime for several bases as a *pseudoprime*.

There are composite numbers n (called *Carmichael numbers*) with the amazing property that $a^{n-1} \equiv 1 \pmod{n}$ for *all* a with $\gcd(a, n) = 1$. The first Carmichael number is 561, and it is a theorem that there are infinitely many such numbers ([AGP94]).

Example 2.4.2. Is $p = 323$ prime? We compute $2^{322} \pmod{323}$. Making a table as above, we have

i	m	ε_i	$2^{2^i} \pmod{323}$
0	322	0	2
1	161	1	4
2	80	0	16
3	40	0	256
4	20	0	290
5	10	0	120
6	5	1	188
7	2	0	137
8	1	1	35

Thus

$$2^{322} \equiv 4 \cdot 188 \cdot 35 \equiv 157 \pmod{323},$$

so 323 is not prime, though this computation gives no information about 323 factors as a product of primes. In fact, one finds that $323 = 17 \cdot 19$.

It's possible to easily prove that a large number is composite, but the proof does not easily yield a factorization. For example if

$$n = 95468093486093450983409583409850934850938459083,$$

then $2^{n-1} \not\equiv 1 \pmod{n}$, so n is composite.

Another practical primality test is the Miller-Rabin test, which has the property that each time it is run on a number n it either correctly asserts that the number is definitely not prime, or that it is probably prime, and the probability of correctness goes up with each successive call. For a precise statement and implementation of Miller-Rabin, along with proof of correctness, see Section 7.2.4. If Miller-Rabin is called m times on n and in each case claims that n is probably prime, then one can in a precise sense bound the probability that n is composite in terms of m . For an implementation of Miller-Rabin, see Listing 7.2.9 in Chapter 7.

Until recently it was an open problem to give an algorithm (with proof) that decides whether or not any integer is prime in time bounded by a polynomial in the number of digits of the integer. Agrawal, Kayal, and Saxena recently found the first polynomial-time primality test (see [AKS02]). We will not discuss their algorithm further, because for our applications to cryptography Miller-Rabin or pseudoprimal tests will be sufficient.

2.5 The Structure of $(\mathbf{Z}/p\mathbf{Z})^*$

This section is about the structure of the group $(\mathbf{Z}/p\mathbf{Z})^*$ of units modulo a prime number p . The main result is that this group is always cyclic. We will use this result later in Chapter 4 in our proof of quadratic reciprocity.

Definition 2.5.1 (Primitive root). A *primitive root* modulo an integer n is an element of $(\mathbf{Z}/n\mathbf{Z})^*$ of order $\varphi(n)$.

We will prove that there is a primitive root modulo every prime p . Since the unit group $(\mathbf{Z}/p\mathbf{Z})^*$ has order $p-1$, this implies that $(\mathbf{Z}/p\mathbf{Z})^*$ is a cyclic group, a fact this will be extremely useful, since it completely determines the structure of $(\mathbf{Z}/p\mathbf{Z})^*$ as an abelian group.

If n is an odd prime power, then there is a primitive root modulo n (see Exercise 2.25), but there is no primitive root modulo the prime power 2^3 , and hence none mod 2^n for $n \geq 3$ (see Exercise 2.24).

Section 2.5.1 is the key input to our proof that $(\mathbf{Z}/p\mathbf{Z})^*$ is cyclic; here we show that for every divisor d of $p-1$ there are exactly d elements of $(\mathbf{Z}/p\mathbf{Z})^*$ whose order divides d . We then use this result in Section 2.5.2 to produce an element of $(\mathbf{Z}/p\mathbf{Z})^*$ of order q^r when q^r is a prime power that exactly divides $p-1$ (i.e., q^r divides $p-1$, but q^{r+1} does not divide $p-1$), and multiply together these elements to obtain an element of $(\mathbf{Z}/p\mathbf{Z})^*$ of order $p-1$.

2.5.1 Polynomials over $\mathbf{Z}/p\mathbf{Z}$

The polynomials $x^2 - 1$ has four roots in $\mathbf{Z}/8\mathbf{Z}$, namely 1, 3, 5, and 7. In contrast, the following proposition shows that a polynomial of degree d over a field, such as $\mathbf{Z}/p\mathbf{Z}$, can have at most d roots.

Proposition 2.5.2 (Root Bound). *Let $f \in k[x]$ be a nonzero polynomial over a field k . Then there are at most $\deg(f)$ elements $\alpha \in k$ such that $f(\alpha) = 0$.*

Proof. We prove the proposition by induction on $\deg(f)$. The cases in which $\deg(f) \leq 1$ are clear. Write $f = a_n x^n + \cdots + a_1 x + a_0$. If $f(\alpha) = 0$ then

$$\begin{aligned} f(x) &= f(x) - f(\alpha) \\ &= a_n(x^n - \alpha^n) + \cdots + a_1(x - \alpha) + a_0(1 - 1) \\ &= (x - \alpha)(a_n(x^{n-1} + \cdots + \alpha^{n-1}) + \cdots + a_2(x + \alpha) + a_1) \\ &= (x - \alpha)g(x), \end{aligned}$$

for some polynomial $g(x) \in k[x]$. Next suppose that $f(\beta) = 0$ with $\beta \neq \alpha$. Then $(\beta - \alpha)g(\beta) = 0$, so, since $\beta - \alpha \neq 0$, we have $g(\beta) = 0$. By our inductive hypothesis, g has at most $n - 1$ roots, so there are at most $n - 1$ possibilities for β . It follows that f has at most n roots. \square

Proposition 2.5.3. *Let p be a prime number and let d be a divisor of $p - 1$. Then $f = x^d - 1 \in (\mathbf{Z}/p\mathbf{Z})[x]$ has exactly d roots in $\mathbf{Z}/p\mathbf{Z}$.*

Proof. Let $e = (p - 1)/d$. We have

$$\begin{aligned} x^{p-1} - 1 &= (x^d)^e - 1 \\ &= (x^d - 1)((x^d)^{e-1} + (x^d)^{e-2} + \cdots + 1) \\ &= (x^d - 1)g(x), \end{aligned}$$

where $g \in (\mathbf{Z}/p\mathbf{Z})[x]$ and $\deg(g) = de - d = p - 1 - d$. Theorem 2.1.12 implies that $x^{p-1} - 1$ has exactly $p - 1$ roots in $\mathbf{Z}/p\mathbf{Z}$, since every nonzero element of $\mathbf{Z}/p\mathbf{Z}$ is a root! By Proposition 2.5.2, g has at most $p - 1 - d$ roots and $x^d - 1$ has at most d roots. Since a root of $(x^d - 1)g(x)$ is a root of either $x^d - 1$ or $g(x)$ and $x^{p-1} - 1$ has $p - 1$ roots, g must have exactly $p - 1 - d$ roots and $x^d - 1$ must have exactly d roots, as claimed. \square

We pause to reemphasize that the analogue of Proposition 2.5.3 is false when p is replaced by a composite integer n , since a root mod n of a product of two polynomials need not be a root of either factor. For example, $f = x^2 - 1 \in \mathbf{Z}/15\mathbf{Z}[x]$ has the four roots 1, 4, 11, and 14.

2.5.2 Existence of Primitive Roots

Recall from Section 2.1.2 that the *order* of an element x in a finite group is the smallest $m \geq 1$ such that $x^m = 1$. In this section, we prove that $(\mathbf{Z}/p\mathbf{Z})^*$ is cyclic by using the results of Section 2.5.1 to produce an element of $(\mathbf{Z}/p\mathbf{Z})^*$ of order d for each prime power divisor d of $p - 1$, and then we multiply these together to obtain an element of order $p - 1$.

We will use the following lemma to assemble elements of each order dividing $p - 1$ to produce an element of order $p - 1$.

Lemma 2.5.4. *Suppose $a, b \in (\mathbf{Z}/n\mathbf{Z})^*$ have orders r and s , respectively, and that $\gcd(r, s) = 1$. Then ab has order rs .*

Proof. This is a general fact about commuting elements of any group; our proof only uses that $ab = ba$ and nothing special about $(\mathbf{Z}/n\mathbf{Z})^*$. Since

$$(ab)^{rs} = a^{rs}b^{rs} = 1,$$

the order of ab is a divisor of rs . Write this divisor as r_1s_1 where $r_1 \mid r$ and $s_1 \mid s$. Raise both sides of the equation

$$a^{r_1s_1}b^{r_1s_1} = (ab)^{r_1s_1} = 1.$$

to the power $r_2 = r/r_1$ to obtain

$$a^{r_1r_2s_1}b^{r_1r_2s_1} = 1.$$

Since $a^{r_1r_2s_1} = (a^{r_1r_2})^{s_1} = 1$, we have

$$b^{r_1r_2s_1} = 1,$$

so $s \mid r_1r_2s_1$. Since $\gcd(s, r_1r_2) = \gcd(s, r) = 1$, it follows that $s = s_1$. Similarly $r = r_1$, so the order of ab is rs . \square

Theorem 2.5.5 (Primitive Roots). *There is a primitive root modulo any prime p . In particular, the group $(\mathbf{Z}/p\mathbf{Z})^*$ is cyclic.*

Proof. The theorem is true if $p = 2$, since 1 is a primitive root, so we may assume $p > 2$. Write $p - 1$ as a product of distinct prime powers $q_i^{n_i}$:

$$p - 1 = q_1^{n_1}q_2^{n_2} \cdots q_r^{n_r}.$$

By Proposition 2.5.3, the polynomial $x^{q_i^{n_i}} - 1$ has exactly $q_i^{n_i}$ roots, and the polynomial $x^{q_i^{n_i-1}} - 1$ has exactly $q_i^{n_i-1}$ roots. There are $q_i^{n_i} - q_i^{n_i-1} = q_i^{n_i-1}(q_i - 1)$ elements $a \in \mathbf{Z}/p\mathbf{Z}$ such that $a^{q_i^{n_i}} = 1$ but $a^{q_i^{n_i-1}} \neq 1$; each of these elements has order $q_i^{n_i}$. Thus for each $i = 1, \dots, r$, we can choose an a_i of order $q_i^{n_i}$. Then, using Lemma 2.5.4 repeatedly, we see that

$$a = a_1a_2 \cdots a_r$$

has order $q_1^{n_1} \cdots q_r^{n_r} = p - 1$, so a is a primitive root modulo p . \square

Example 2.5.6. We illustrate the proof of Theorem 2.5.5 when $p = 13$. We have

$$p - 1 = 12 = 2^2 \cdot 3.$$

The polynomial $x^4 - 1$ has roots $\{1, 5, 8, 12\}$ and $x^2 - 1$ has roots $\{1, 12\}$, so we may take $a_1 = 5$. The polynomial $x^3 - 1$ has roots $\{1, 3, 9\}$, and we set $a_2 = 3$. Then $a = 5 \cdot 3 = 15 \equiv 2$ is a primitive root. To verify this, note that the successive powers of 2 (mod 13) are

$$2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7, 1.$$

Example 2.5.7. Theorem 2.5.5 is false if, e.g., p is replaced by a power of 2 bigger than 4. For example, the four elements of $(\mathbf{Z}/8\mathbf{Z})^*$ each have order dividing 2, but $\varphi(8) = 4$.

Theorem 2.5.8 (Primitive Roots mod p^n). *Let p^n be a power of an odd prime. Then there is a primitive root modulo p^n .*

The proof is left as Exercise 2.25.

Proposition 2.5.9 (Number of primitive roots). *If there is a primitive root modulo n , then there are exactly $\varphi(\varphi(n))$ primitive roots modulo n .*

Proof. The primitive roots modulo n are the generators of $(\mathbf{Z}/n\mathbf{Z})^*$, which by assumption is cyclic of order $\varphi(n)$. Thus they are in bijection with the generators of any cyclic group of order $\varphi(n)$. In particular, the number of primitive roots modulo n is the same as the number of elements of $\mathbf{Z}/\varphi(n)\mathbf{Z}$ with additive order $\varphi(n)$. An element of $\mathbf{Z}/\varphi(n)\mathbf{Z}$ has additive order $\varphi(n)$ if and only if it is coprime to $\varphi(n)$. There are $\varphi(\varphi(n))$ such elements, as claimed. \square

Example 2.5.10. For example, there are $\varphi(\varphi(17)) = \varphi(16) = 2^4 - 2^3 = 8$ primitive roots mod 17, namely 3, 5, 6, 7, 10, 11, 12, 14. The $\varphi(\varphi(9)) = \varphi(6) = 2$ primitive roots modulo 9 are 2 and 5. There are no primitive roots modulo 8, even though $\varphi(\varphi(8)) = \varphi(4) = 2 > 0$.

2.5.3 Artin's Conjecture

Conjecture 2.5.11 (Emil Artin). *Suppose $a \in \mathbf{Z}$ is not -1 or a perfect square. Then there are infinitely many primes p such that a is a primitive root modulo p .*

There is no single integer a such that Artin's conjecture is known to be true. For any given a , Pieter [Mor93] proved that there are infinitely many p such that the order of a is divisible by the largest prime factor of $p - 1$. Hooley [Hoo67] proved that something called the Generalized Riemann Hypothesis implies Conjecture 2.5.11.

Remark 2.5.12. Artin conjectured more precisely that if $N(x, a)$ is the number of primes $p \leq x$ such that a is a primitive root modulo p , then $N(x, a)$ is asymptotic to $C(a)\pi(x)$, where $C(a)$ is a positive constant that depends only on a and $\pi(x)$ is the number of primes up to x .

2.5.4 Computing Primitive Roots

Theorem 2.5.5 does not suggest an efficient algorithm for finding primitive roots. To actually find a primitive root mod p in practice, we try $a = 2$, then $a = 3$, etc., until we find an a that has order $p - 1$. Computing the order of an element of $(\mathbf{Z}/p\mathbf{Z})^*$ requires factoring $p - 1$, which we do not know how to do quickly in general, so finding a primitive root modulo p for large p seems to be a difficult problem.

See Section 7.2.3 for an implementation of this algorithm for finding a primitive root.

Algorithm 2.5.13 (Primitive Root). Given a prime p this algorithm computes the smallest positive integer a that generates $(\mathbf{Z}/p\mathbf{Z})^*$.

1. [$p = 2$?] If $p = 2$ output 1 and terminate. Otherwise set $a \leftarrow 2$.
2. [Prime Divisors] Compute the prime divisors p_1, \dots, p_r of $p - 1$ (see Section 7.1.3).
3. [Generator?] If for every p_i , we have $a^{(p-1)/p_i} \not\equiv 1 \pmod{p}$, then a is a generator of $(\mathbf{Z}/p\mathbf{Z})^*$, so output a and terminate.
4. [Try next] Set $a \leftarrow a + 1$ and go to step 3.

Proof. Let $a \in (\mathbf{Z}/p\mathbf{Z})^*$. The order of a is a divisor d of the order $p - 1$ of the group $(\mathbf{Z}/p\mathbf{Z})^*$. Write $d = (p - 1)/n$, for some divisor n of $p - 1$. If a is not a generator of $(\mathbf{Z}/p\mathbf{Z})^*$, then since $n \mid (p - 1)$, there is a prime divisor p_i of $p - 1$ such that $p_i \mid n$. Then

$$a^{(p-1)/p_i} = (a^{(p-1)/n})^{n/p_i} \equiv 1 \pmod{p}.$$

Conversely, if a is a generator, then $a^{(p-1)/p_i} \not\equiv 1 \pmod{p}$ for any p_i . Thus the algorithm terminates with step 3 if and only if the a under consideration is a primitive root. By Theorem 2.5.5 there is at least one primitive root, so the algorithm terminates. \square

We implement Algorithm 2.5.13 in Section 7.2.3.

2.6 Exercises

2.1 Compute the following gcd's using Algorithm 1.1.12:

$$\gcd(15, 35), \quad \gcd(247, 299), \quad \gcd(51, 897), \quad \gcd(136, 304)$$

2.2 Use Algorithm 2.3.3 to find $x, y \in \mathbf{Z}$ such that $2261x + 1275y = 17$.

2.3 Prove that if a and b are integers and p is a prime, then $(a + b)^p \equiv a^p + b^p \pmod{p}$. You may assume that the binomial coefficient

$$\frac{p!}{r!(p-r)!}$$

is an integer.

2.4 (a) Prove that if x, y is a solution to $ax + by = d$, then for all $c \in \mathbf{Z}$,

$$x' = x + c \cdot \frac{b}{d}, \quad y' = y - c \cdot \frac{a}{d} \quad (2.6.1)$$

is also a solution to $ax + by = d$.

(b) Find two distinct solutions to $2261x + 1275y = 17$.

(c) Prove that all solutions are of the form (2.6.1) for some c .

2.5 Let $f(x) = x^2 + ax + b \in \mathbf{Z}[x]$ be a quadratic polynomial with integer coefficients and positive leading coefficients, e.g., $f(x) = x^2 + x + 6$. Formulate a conjecture about when the set $\{f(n) : n \in \mathbf{Z} \text{ and } f(n) \text{ is prime}\}$ is infinite. Give numerical evidence that supports your conjecture.

2.6 Find four complete sets of residues modulo 7, where the i th set satisfies the i th condition: (1) nonnegative, (2) odd, (3) even, (4) prime.

2.7 Find rules in the spirit of Proposition 2.1.3 for divisibility of an integer by 5, 9, and 11, and prove each of these rules using arithmetic modulo a suitable n .

2.8 (*) The following problem is from the 1998 Putnam Competition. Define a sequence of decimal integers a_n as follows: $a_1 = 0$, $a_2 = 1$, and a_{n+2} is obtained by writing the digits of a_{n+1} immediately followed by those of a_n . For example, $a_3 = 10$, $a_4 = 101$, and $a_5 = 10110$. Determine the n such that a_n a multiple of 11, as follows:

(a) Find the smallest integer $n > 1$ such that a_n is divisible by 11.

(b) Prove that a_n is divisible by 11 if and only if $n \equiv 1 \pmod{6}$.

2.9 Find an integer x such that $37x \equiv 1 \pmod{101}$.

2.10 What is the order of 2 modulo 17?

2.11 Let p be a prime. Prove that $\mathbf{Z}/p\mathbf{Z}$ is a field.

2.12 Find an $x \in \mathbf{Z}$ such that $x \equiv -4 \pmod{17}$ and $x \equiv 3 \pmod{23}$.

2.13 Prove that if $n > 4$ is composite then

$$(n - 1)! \equiv 0 \pmod{n}.$$

2.14 For what values of n is $\varphi(n)$ odd?

2.15 (a) Prove that φ is multiplicative as follows. Suppose m, n are positive integers and $\gcd(m, n) = 1$. Show that the natural map $\psi : \mathbf{Z}/mn\mathbf{Z} \rightarrow \mathbf{Z}/m\mathbf{Z} \times \mathbf{Z}/n\mathbf{Z}$ is an injective homomorphism of rings, hence bijective by counting, then look at unit groups.

(b) Prove conversely that if $\gcd(m, n) > 1$ then the natural map $\psi : \mathbf{Z}/mn\mathbf{Z} \rightarrow \mathbf{Z}/m\mathbf{Z} \times \mathbf{Z}/n\mathbf{Z}$ is not an isomorphism.

2.16 Seven competitive math students try to share a huge hoard of stolen math books equally between themselves. Unfortunately, six books are left over, and in the fight over them, one math student is expelled. The remaining six math students, still unable to share the math books equally since two are left over, again fight, and another is expelled. When the remaining five share the books, one book is left over, and it is only after yet another math student is expelled that an equal sharing is possible. What is the minimum number of books which allow this to happen?

2.17 Show that if p is a positive integer such that both p and $p^2 + 2$ are prime, then $p = 3$.

2.18 Let $\varphi : \mathbf{N} \rightarrow \mathbf{N}$ be the Euler φ function.

(a) Find all natural numbers n such that $\varphi(n) = 1$.

(b) Do there exist natural numbers m and n such that $\varphi(mn) \neq \varphi(m) \cdot \varphi(n)$?

2.19 Find a formula for $\varphi(n)$ directly in terms of the prime factorization of n .

2.20 Find all *four* solutions to the equation

$$x^2 - 1 \equiv 0 \pmod{35}.$$

2.21 Prove that for any positive integer n the fraction $(12n + 1)/(30n + 2)$ is in reduced form.

2.22 Suppose a and b are positive integers.

(a) Prove that $\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a, b)} - 1$.

(b) Does it matter if 2 is replaced by an arbitrary prime p ?

(c) What if 2 is replaced by an arbitrary positive integer n ?

- 2.23 For every positive integer b , show that there exists a positive integer n such that the polynomial $x^2 - 1 \in (\mathbf{Z}/n\mathbf{Z})[x]$ has at least b roots.
- 2.24 (a) Prove that there is no primitive root modulo 2^n for any $n \geq 3$.
(b) (*) Prove that $(\mathbf{Z}/2^n\mathbf{Z})^*$ is generated by -1 and 5 .
- 2.25 Let p be an odd prime.
- (a) (*) Prove that there is a primitive root modulo p^2 . (Hint: Use that if a, b have orders n, m , with $\gcd(n, m) = 1$, then ab has order nm .)
- (b) Prove that for any n , there is a primitive root modulo p^n .
- (c) Explicitly find a primitive root modulo 125 .
- 2.26 (*) In terms of the prime factorization of n , characterize the integers n such that there is a primitive root modulo n .

3

Public-Key Cryptography



The author recently watched a TV show (not movie!) called La Femme Nikita about a woman named Nikita who is forced to be an agent for a shady anti-terrorist organization called Section One. Nikita has strong feelings for fellow agent Michael, and she most trusts Walter, Section One's ex-biker gadgets and explosives expert. Often Nikita's worst enemies are her superiors and coworkers at Section One.

A synopsis for a season three episode is as follows:

PLAYING WITH FIRE

On a mission to secure detonation chips from a terrorist organization's heavily armed base camp, Nikita is captured as a hostage by the enemy. Or so it is made to look. Michael and Nikita have actually created the scenario in order to secretly rendezvous with each other. The ruse works, but when Birkoff [Section One's master hacker] accidentally discovers encrypted messages between Michael and Nikita sent with Walter's help, Birkoff is forced to tell Madeline. Suspecting that Michael and Nikita may be planning a coup d'état, Operations and Madeline use a second team of operatives to track Michael and Nikita's next secret rendezvous... killing them if necessary.

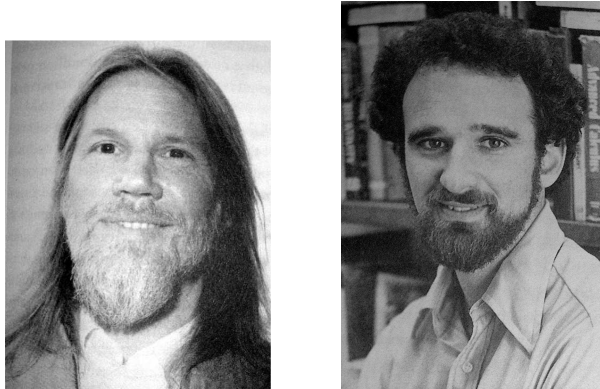


FIGURE 3.1. Diffie and Hellman (photos from [Sin99])

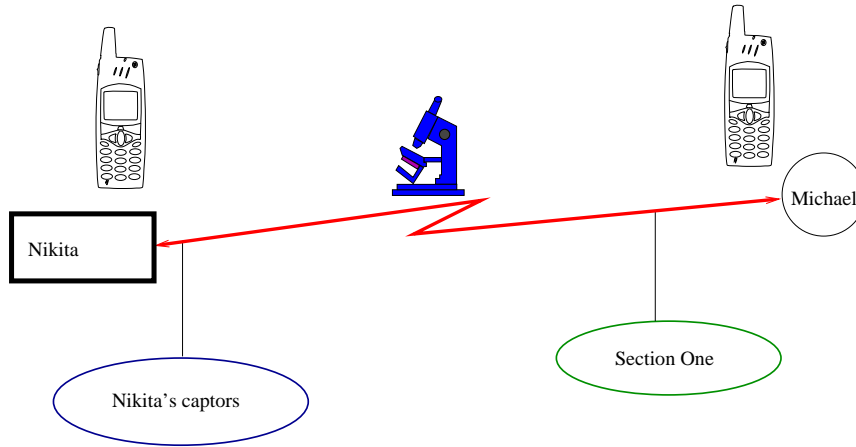
What sort of encryption might Walter have helped them to use? I let my imagination run free, and this is what I came up with. After being captured at the base camp, Nikita is given a phone by her captors, in hopes that she'll use it and they'll be able to figure out what she is really up to. Everyone is eagerly listening in on her calls.

Remark 3.0.1. In this book we will assume available a method for producing random integers. Methods for generating random integers are involved and interesting, but we will not discuss them in this book. For an in depth treatment of random numbers, see [Knu98, Ch. 3].

Nikita remembers a conversation with Walter about a public-key cryptosystem called the “Diffie-Hellman key exchange”. She remembers that it allows two people to agree on a secret key in the presence of eavesdroppers. Moreover, Walter mentioned that though Diffie-Hellman was the first ever public-key exchange system, it is still in common use today (e.g., in OpenSSH protocol version 2, see <http://www.openssh.com/>).

Nikita pulls out her handheld computer and phone, calls up Michael, and they do the following, which is *wrong* (try to figure out what is wrong as you read it).

1. Together they choose a big prime number p and a number g with $1 < g < p$.
2. Nikita *secretly* chooses an integer n .
3. Michael *secretly* chooses an integer m .
4. Nikita tells Michael $ng \pmod{p}$.
5. Michael tells $mg \pmod{p}$ to Nikita.
6. The “secret key” is $s = nmg \pmod{p}$, which both Nikita and Michael can easily compute.



Here's a very simple example with small numbers that illustrates what Michael and Nikita do. (They really used much larger numbers.)

1. $p = 97, g = 5$
2. $n = 31$
3. $m = 95$
4. $ng \equiv 58 \pmod{97}$
5. $mg \equiv 87 \pmod{97}$
6. $s = nmg = 78 \pmod{97}$

Nikita and Michael are foiled because everyone easily figures out s :

1. Everyone knows $p, g, ng \pmod{p}$, and $mg \pmod{p}$.
2. Using Algorithm 2.3.3, anyone can easily find $a, b \in \mathbf{Z}$ such that $ag + bp = 1$, which exist because $\gcd(g, p) = 1$.
3. Then $ang \equiv n \pmod{p}$, so everyone knows Nikita's secret key n , and hence can easily compute the shared secret s .

To taunt her, Nikita's captors give her a paragraph from a review of Diffie and Hellman's 1976 paper "New Directions in Cryptography" [DH76]:

"The authors discuss some recent results in communications theory [...] The first [method] has the feature that an unauthorized 'eavesdropper' will find it computationally infeasible to decipher the message [...] They propose a couple of techniques for implementing the system, but the reviewer was unconvinced."

3.1 The Diffie-Hellman Key Exchange

As night darkens Nikita's cell, she reflects on what has happened. Upon realizing that she mis-remembered how the system works, she phones Michael and they do the following:

1. Together Michael and Nikita choose a 200-digit integer p that is likely to be prime (see Section 2.4), and choose a number g with $1 < g < p$.
2. Nikita *secretly* chooses an integer n .
3. Michael *secretly* chooses an integer m .
4. Nikita computes $g^n \pmod{p}$ on her handheld computer and tells Michael the resulting number over the phone.
5. Michael tells Nikita $g^m \pmod{p}$.
6. The shared secret key is then

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{nm} \pmod{p},$$

which both Nikita and Michael can compute.

Here is a simplified example that illustrates what they did, that involves only relatively simple arithmetic.

1. $p = 97, g = 5$
2. $n = 31$
3. $m = 95$
4. $g^n \equiv 7 \pmod{p}$
5. $g^m \equiv 39 \pmod{p}$
6. $s \equiv (g^n)^m \equiv 14 \pmod{p}$

3.1.1 The Discrete Log Problem

Nikita communicates with Michael by encrypting everything using their agreed upon secret key. In order to understand the conversation, the eavesdropper needs s , but it takes a long time to compute s given only p, g, g^n , and g^m . One way would be to compute n from knowledge of g and g^n ; this is possible, but appears to be “computationally infeasible”, in the sense that it would take too long to be practical.

Let a , b , and n be real numbers with $a, b > 0$ and $n \geq 0$. Recall that the “log to the base b ” function characterized by

$$\log_b(a) = n \text{ if and only if } a = b^n.$$

We use the \log_b function in algebra to solve the following problem: Given a base b and a power a of b , find an exponent n such that

$$a = b^n.$$

That is, given $a = b^n$ and b , find n .

Example 3.1.1. The number $a = 19683$ is the n th power of $b = 3$ for some n . With a calculator we quickly find that

$$n = \log_3(19683) = \log(19683)/\log(3) = 9.$$

A calculator can quickly compute an approximation for $\log(x)$ by computing a partial sum of an appropriate rapidly-converging infinite series (at least for x in a certain range).

The discrete log problem is the analogue of this problem but in a finite group:

Problem 3.1.2 (Discrete Log Problem). Let G be a finite abelian group, e.g., $G = (\mathbf{Z}/p\mathbf{Z})^*$. Given $b \in G$ and a power a of b , find a positive integer n such that $b^n = a$.

As far as we know, finding discrete logarithms when p is large is difficult in practice. Over the years, many people have been very motivated to try. For example, if Nikita’s captors could efficiently solve Problem 3.1.2, then they could read the messages she exchanges with Michael. Unfortunately, we have no formal proof that computing discrete logarithms on a classical computer is difficult. Also, Peter Shor [Sho97] showed that if one could build a sufficiently complicated quantum computer, it could solve the discrete logarithm problem in time bounded by a polynomial function of the number of digits of $\#G$.

It is easy to give an inefficient algorithm that solves the discrete log problem. Simply try b^1, b^2, b^3 , etc., until we find an exponent n such that $b^n = a$. For example, suppose $a = 18$, $b = 5$, and $p = 23$. Working modulo 23 we have

$$b^1 = 5, b^2 = 2, b^3 = 10, \dots, b^{12} = 18,$$

so $n = 12$. When p is large, computing the discrete log this way soon becomes impractical, because increasing the number of digits of the modulus makes the computation take vastly longer.

Perhaps part of the reason that computing discrete logarithms is difficult, is that the logarithm in the real numbers is continuous, but the (minimum) logarithm of a number mod n bounces around at random. We illustrate this exotic behavior in Figure 3.2.

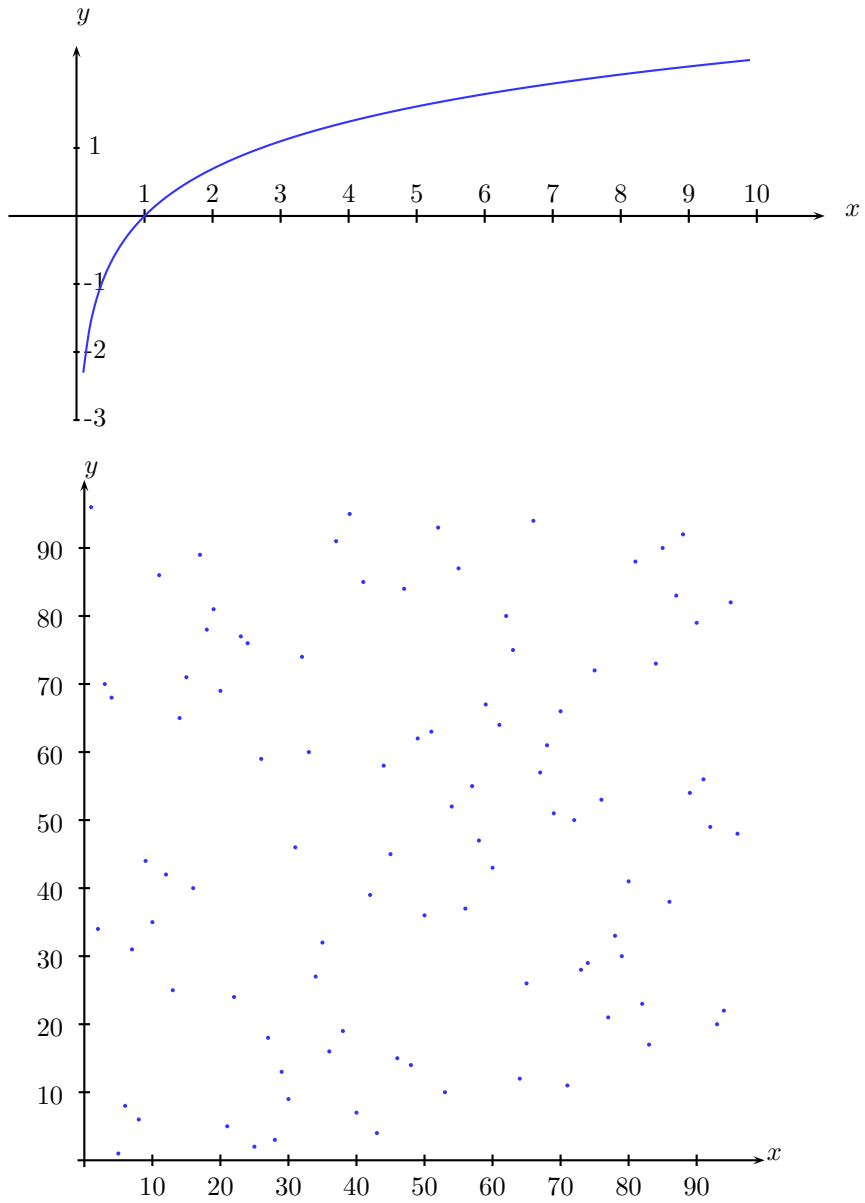


FIGURE 3.2. Graphs of the continuous log and of the discrete log modulo 97. Which looks easier to compute?

3.1.2 Realistic Diffie-Hellman Example

In this section we present an example that uses bigger numbers. First we prove a proposition that we can use to choose a prime p in such a way that it is easy to find a $g \in (\mathbf{Z}/p\mathbf{Z})^*$ with order $p - 1$. We have already seen in Section 2.5 that for every prime p there exists an element g of order $p - 1$, and we gave Algorithm 2.5.13 for finding a primitive root for any prime. The significance of the proposition below is that it suggests an algorithm for finding a primitive root that is easier to use in practice when p is large, because it does not require factoring $p - 1$. Of course, one could also just use a random g for Diffie-Hellman; it is not essential that g generates $(\mathbf{Z}/p\mathbf{Z})^*$.

Proposition 3.1.3. *Suppose p is a prime such that $(p - 1)/2$ is also prime. Then the elements of $(\mathbf{Z}/p\mathbf{Z})^*$ have order either 1, 2, $(p - 1)/2$, or $p - 1$.*

Proof. Since p is prime, the group $(\mathbf{Z}/p\mathbf{Z})^*$ has order $p - 1$. By assumption, the prime factorization of $p - 1$ is $2 \cdot ((p - 1)/2)$. Let $a \in (\mathbf{Z}/p\mathbf{Z})^*$. Then by Theorem 2.1.12, $a^{p-1} = 1$, so the order of a is a divisor of $p - 1$, which proves the proposition. \square

Given a prime p with $(p - 1)/2$ prime, find an element of order $p - 1$ as follows. If 2 has order $p - 1$ we are done. If not, 2 has order $(p - 1)/2$ since 2 doesn't have order either 1 or 2. Then -2 has order $p - 1$.

Let $p = 93450983094850938450983409611$. Then p is prime, but $(p - 1)/2$ is not. So we keep adding 2 to p and testing pseudoprimality using Section 2.4 until we find that the next pseudoprime after p is

$$q = 93450983094850938450983409623.$$

It turns out that q pseudoprime and $(q - 1)/2$ is also pseudoprime. We find that 2 has order $(q - 1)/2$, so $g = -2$ has order $q - 1$ and is hence a generator of $(\mathbf{Z}/q\mathbf{Z})^*$, at least assuming that q is really prime.

The secret random numbers generated by Nikita and Michael are

$$n = 18319922375531859171613379181$$

and

$$m = 82335836243866695680141440300.$$

Nikita sends

$$g^n = 45416776270485369791375944998 \in (\mathbf{Z}/p\mathbf{Z})^*$$

to Michael, and Michael sends

$$g^m = 15048074151770884271824225393 \in (\mathbf{Z}/p\mathbf{Z})^*$$

to Nikita. They agree on the secret key

$$g^{nm} = 85771409470770521212346739540 \in (\mathbf{Z}/p\mathbf{Z})^*.$$

Remark 3.1.4. See Section 7.3.1 for a computer implementation of the Diffie-Hellman key exchange.

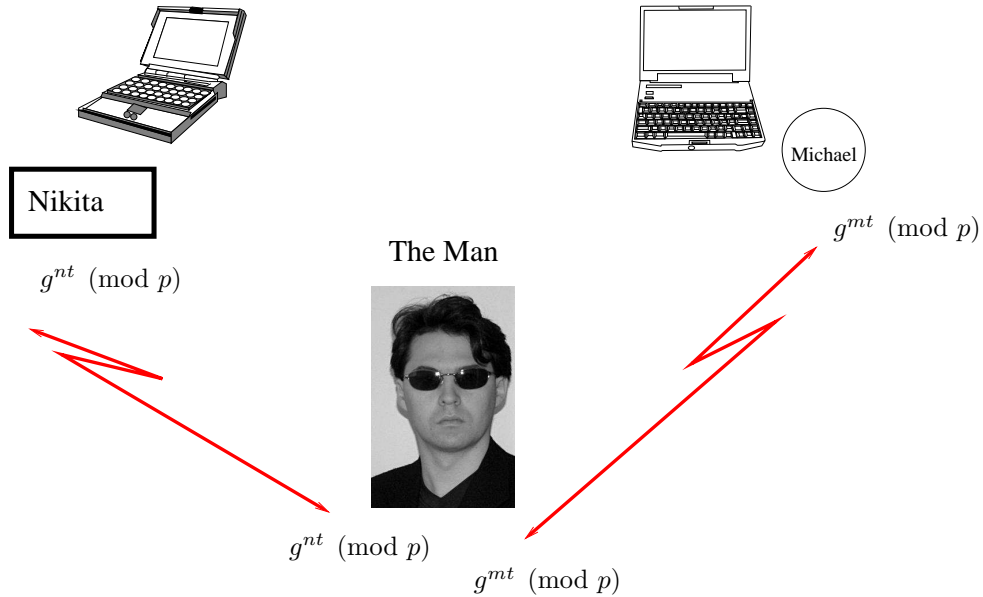


FIGURE 3.3. The Man in the Middle Attack

3.1.3 The Man in the Middle Attack

After their first system was broken, instead of talking on the phone, Michael and Nikita can now only communicate via text messages. One of her captors, The Man, is watching each of the transmissions; moreover, he can intercept messages and send false messages. When Nikita sends a message to Michael announcing $g^n \pmod{p}$, The Man intercepts this message, and sends his own number $g^t \pmod{p}$ to Michael. Eventually, Michael and The Man agree on the secret key $g^{tm} \pmod{p}$, and Nikita and The Man agree on the key $g^{tn} \pmod{p}$. When Nikita sends a message to Michael she unwittingly uses the secret key $g^{tn} \pmod{p}$; The Man then intercepts it, decrypts it, changes it, and re-encrypts it using the key $g^{tm} \pmod{p}$, and sends it on to Michael. This is bad because now The Man can read every message sent between Michael and Nikita, and moreover, he can change them in transmission in subtle ways.

One way to get around this attack is to use a digital signature scheme based on the RSA cryptosystem. We will not discuss digital signatures further in this book, but will discuss RSA in the next section.

3.2 The RSA Cryptosystem

The Diffie-Hellman key exchange has drawbacks. As discussed in Section 3.1.3, it is susceptible to the man in the middle attack. This section is about the RSA public-key cryptosystem of Rivest, Shamir, and Adleman [RSA78], which is an alternative to Diffie-Hellman that is more flexible in some ways.

We first describe the RSA cryptosystem, then discuss several ways to attack it. It is important to be aware of such weaknesses, in order to avoid foolish mistakes when implementing RSA. We barely scratched the surface here of the many possible attacks on specific implementations of RSA or other cryptosystems.

3.2.1 How RSA works

The fundamental idea behind RSA is to try to construct a trap-door or one-way function on a set X , that is, an invertible function

$$E : X \rightarrow X$$

such that it is easy for Nikita to compute E^{-1} , but extremely difficult for anybody else to do so.

Here is how Nikita makes a one-way function E on the set of integers modulo n .

1. Using a method hinted at in Section 2.4, Nikita picks two large primes p and q , and lets $n = pq$.
2. It is then easy for Nikita to compute

$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1).$$

3. Nikita next chooses a random integer e with

$$1 < e < \varphi(n) \text{ and } \gcd(e, \varphi(n)) = 1.$$

4. Nikita uses the algorithm from Section 2.3.2 to find a solution $x = d$ to the equation

$$ex \equiv 1 \pmod{\varphi(n)}.$$

5. Finally, Nikita defines a function $E : \mathbf{Z}/n\mathbf{Z} \rightarrow \mathbf{Z}/n\mathbf{Z}$ by

$$E(x) = x^e \in \mathbf{Z}/n\mathbf{Z}.$$

Anybody can compute E fairly quickly using the repeated-squaring algorithm from Section 2.3.2.

Nikita's *public key* is the pair of integers (n, e) , which is just enough information for people to easily compute E . Nikita knows a number d such that $ed \equiv 1 \pmod{\varphi(n)}$, so, as we will see, she can quickly compute E^{-1} .

To send Nikita a message, proceed as follows. Encode your message, in some way, as a sequence of numbers modulo n (see Section 3.2.2)

$$m_1, \dots, m_r \in \mathbf{Z}/n\mathbf{Z},$$

then send

$$E(m_1), \dots, E(m_r)$$

to Nikita. (Recall that $E(m) = m^e$ for $m \in \mathbf{Z}/n\mathbf{Z}$.)

When Nikita receives $E(m_i)$, she finds each m_i by using that $E^{-1}(m) = m^d$, a fact that follows from the following proposition.

Proposition 3.2.1 (Decryption key). *Let n be an integer that is a product of distinct primes and let $d, e \in \mathbf{N}$ be such that $p - 1 \mid de - 1$ for each prime $p \mid n$. Then $a^{de} \equiv a \pmod{n}$ for all $a \in \mathbf{Z}$.*

Proof. Since $n \mid a^{de} - a$ if and only if $p \mid a^{de} - a$ for each prime divisor p of n , it suffices to prove that $a^{de} \equiv a \pmod{p}$ for each prime divisor p of n . If $\gcd(a, p) \neq 0$, then $a \equiv 0 \pmod{p}$, so $a^{de} \equiv a \pmod{p}$. If $\gcd(a, p) = 1$, then Theorem 2.1.12 asserts that $a^{p-1} \equiv 1 \pmod{p}$. Since $p - 1 \mid de - 1$, we have $a^{de-1} \equiv 1 \pmod{p}$ as well. Multiplying both sides by a shows that $a^{de} \equiv a \pmod{p}$. \square

Thus to decrypt $E(m_i)$ Nikita computes

$$E(m_i)^d = (m_i^e)^d = m_i.$$

For an implementation of RSA see Section 7.3.3.

3.2.2 Encoding a Phrase in a Number

In order to use the RSA cryptosystem to encrypt messages, it is necessary to encode them as a sequence of numbers of size less than $n = pq$. We now describe a simple way to do this. For an implementation of a slightly more general encoding that includes extra randomness so that plain text encodes differently each time, see Section 7.3.2.

Suppose s is a sequence of capital letters and spaces, and that s does not begin with a space. We encode s as a number in base 27 as follows: a single space corresponds to 0, the letter A to 1, B to 2, ..., Z to 26. Thus "RUN NIKITA" is a number written in base 27:

$$\begin{aligned} \text{RUN NIKITA} &\leftrightarrow 27^9 \cdot 18 + 27^8 \cdot 21 + 27^7 \cdot 14 + 27^6 \cdot 0 + 27^5 \cdot 14 \\ &\quad + 27^4 \cdot 9 + 27^3 \cdot 11 + 27^2 \cdot 9 + 27 \cdot 20 + 1 \\ &= 143338425831991 \text{ (in decimal)}. \end{aligned}$$

To recover the letters from the decimal number, repeatedly divide by 27 and read off the letter corresponding to each remainder:

$$\begin{array}{rcll}
 143338425831991 & = & 5308830586370 \cdot 27 & + & 1 & \text{“A”} \\
 5308830586370 & = & 196623355050 \cdot 27 & + & 20 & \text{“T”} \\
 196623355050 & = & 7282346483 \cdot 27 & + & 9 & \text{“I”} \\
 7282346483 & = & 269716536 \cdot 27 & + & 11 & \text{“K”} \\
 269716536 & = & 9989501 \cdot 27 & + & 9 & \text{“I”} \\
 9989501 & = & 369981 \cdot 27 & + & 14 & \text{“N”} \\
 369981 & = & 13703 \cdot 27 & + & 0 & \text{“ ”} \\
 13703 & = & 507 \cdot 27 & + & 14 & \text{“N”} \\
 507 & = & 18 \cdot 27 & + & 21 & \text{“U”} \\
 18 & = & 0 \cdot 27 & + & 18 & \text{“R”}
 \end{array}$$

If $27^k \leq n$, then any sequence of k letters can be encoded as above using a positive integer $\leq n$. Thus if we use can encrypt integers of size at most n , then we must break our message up into blocks of size at most $\log_{27}(n)$.

3.2.3 Examples

So the arithmetic is easy to follow, we use small primes p and q and encrypt the single letter “X” using the RSA cryptosystem.

1. Choose p and q : Let $p = 17$, $q = 19$, so $n = pq = 323$.
2. Compute $\varphi(n)$:

$$\begin{aligned}
 \varphi(n) &= \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1)(q-1) \\
 &= pq - p - q + 1 = 323 - 17 - 19 + 1 = 288.
 \end{aligned}$$

3. Randomly choose an $e < 288$: We choose $e = 95$.
4. Solve

$$95x \equiv 1 \pmod{288}.$$

Using the GCD algorithm, we find that $d = 191$ solves the equation.

The public key is $(323, 95)$, so the encryption function is

$$E(x) = x^{95},$$

and the decryption function is $D(x) = x^{191}$.

Next, we encrypt the letter “X”. It is encoded as the number 24, since X is the 24th letter of the alphabet. We have

$$E(24) = 24^{95} = 294 \in \mathbf{Z}/323\mathbf{Z}.$$

To decrypt, we compute E^{-1} :

$$E^{-1}(294) = 294^{191} = 24 \in \mathbf{Z}/323\mathbf{Z}.$$

This next example illustrates RSA but with bigger numbers. Let

$$p = 738873402423833494183027176953, \quad q = 3787776806865662882378273.$$

Then

$$n = p \cdot q = 2798687536910915970127263606347911460948554197853542169$$

and

$$\begin{aligned} \varphi(n) &= (p-1)(q-1) \\ &= 2798687536910915970127262867470721260308194351943986944. \end{aligned}$$

Using a pseudo-random number generator on a computer, the author randomly chose the integer

$$e = 1483959194866204179348536010284716655442139024915720699.$$

Then

$$d = 2113367928496305469541348387088632973457802358781610803$$

Since $\log_{27}(n) \approx 38.04$, we can encode then encrypt single blocks of up to 38 letters. Let's encrypt "RUN NIKITA", which encodes as $m = 143338425831991$. We have

$$\begin{aligned} E(m) &= m^e \\ &= 1504554432996568133393088878600948101773726800878873990. \end{aligned}$$

Remark 3.2.2. In practice one usually chooses e to be small, since that does not seem to reduce the security of RSA, and makes the key size smaller. For example, in the OpenSSL documentation (see <http://www.openssl.org/>) about their implementation of RSA it states that "The exponent is an odd number, typically 3, 17 or 65537."

3.3 Attacking RSA

Suppose Nikita's public key is (n, e) and her decryption key is d , so $ed \equiv 1 \pmod{\varphi(n)}$. If somehow we compute the factorization $n = pq$, then we can compute $\varphi(n) = (p-1)(q-1)$ and hence compute d . Thus if we can factor n then we can break the corresponding RSA public-key cryptosystem.

3.3.1 Factoring n Given $\varphi(n)$

Suppose $n = pq$. Given $\varphi(n)$, it is very easy to compute p and q . We have

$$\varphi(n) = (p-1)(q-1) = pq - (p+q) + 1,$$

so we know both $pq = n$ and $p+q = n+1 - \varphi(n)$. Thus we know the polynomial

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

whose roots are p and q . These roots can be found using the quadratic formula.

Example 3.3.1. The number $n = pq = 31615577110997599711$ is a product of two primes, and $\varphi(n) = 31615577098574867424$. We have

$$\begin{aligned} f &= x^2 - (n+1 - \varphi(n))x + n \\ &= x^2 - 12422732288x + 31615577110997599711 \\ &= (x - 3572144239)(x - 8850588049), \end{aligned}$$

where the factorization step is easily accomplished using the quadratic formula:

$$\begin{aligned} &\frac{-b + \sqrt{b^2 - 4ac}}{2a} \\ &= \frac{12422732288 + \sqrt{12422732288^2 - 4 \cdot 31615577110997599711}}{2} \\ &= 8850588049. \end{aligned}$$

We conclude that $n = 3572144239 \cdot 8850588049$.

3.3.2 When p and q are Close

Suppose that p and q are “close” to each other. Then it is easy to factor n using a factorization method of Fermat.

Suppose $n = pq$ with $p > q$, say. Then

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2.$$

Since p and q are “close”,

$$s = \frac{p-q}{2}$$

is small,

$$t = \frac{p+q}{2}$$

is only slightly larger than \sqrt{n} , and $t^2 - n = s^2$ is a perfect square. So we just try

$$t = \lceil \sqrt{n} \rceil, \quad t = \lceil \sqrt{n} \rceil + 1, \quad t = \lceil \sqrt{n} \rceil + 2, \dots$$

until $t^2 - n$ is a perfect square s^2 . (Here $\lceil x \rceil$ denotes the least integer $n \geq x$.)
Then

$$p = t + s, \quad q = t - s.$$

Example 3.3.2. Suppose $n = 23360947609$. Then

$$\sqrt{n} = 152842.88\dots$$

If $t = 152843$, then $\sqrt{t^2 - n} = 187.18\dots$

If $t = 152844$, then $\sqrt{t^2 - n} = 583.71\dots$

If $t = 152845$, then $\sqrt{t^2 - n} = 804 \in \mathbf{Z}$.

Thus $s = 804$. We find that $p = t + s = 153649$ and $q = t - s = 152041$.

3.3.3 Factoring n Given d

In this section, we show that finding the decryption key d for an RSA cryptosystem is, in practice, at least as difficult as factoring n . We give a probabilistic algorithm that given a decryption key determines the factorization of n .

Consider an RSA cryptosystem with modulus n and encryption key e . Suppose we somehow find an integer d such that

$$a^{ed} \equiv a \pmod{n}$$

for all a . Then $m = ed - 1$ satisfies $a^m \equiv 1 \pmod{n}$ for all a that are coprime to n . As we saw in Section 3.3.1, knowing $\varphi(n)$ leads directly to a factorization of n . Unfortunately, knowing d does not seem to lead easily to a factorization of n . However, there is a probabilistic procedure that, given an m such that $a^m \equiv 1 \pmod{n}$, will find a factorization of n with “high probability” (we will not analyze the probability here).

Algorithm 3.3.3 (Probabilistic Algorithm to Factor n). Let $n = pq$ be the product of two distinct odd primes, and suppose m is an integer such that $a^m \equiv 1 \pmod{n}$ for all a coprime to n . This probabilistic algorithm factors n with “high probability”. In the steps below, a always denotes an integer coprime to $n = pq$.

1. [Divide out powers of 2] If $a^{m/2} \equiv 1 \pmod{n}$ for several randomly chosen a , set $m \leftarrow m/2$, and go to step 1, otherwise let a be such that $a^{m/2} \not\equiv 1 \pmod{n}$.
2. [Compute GCD's] Compute $g \leftarrow \gcd(a^{m/2} - 1, n)$.
3. [Terminate?] If g is a proper divisor of n , output g and terminate. Otherwise go to step 1 and choose a different a .

In step 1, note that m is even since $(-1)^m \equiv 1 \pmod{n}$, so it makes sense to consider $m/2$. It is not practical to determine whether or not $a^{m/2} \equiv 1 \pmod{n}$ for all a , because it would require doing a computation for too

many a . Instead, we try a few random a ; if $a^{m/2} \equiv 1 \pmod{n}$ for the a we check, we divide m by 2. Also note that if there exists even a single a such that $a^{m/2} \not\equiv 1 \pmod{n}$, then half the a have this property, since then $a \mapsto a^{m/2}$ is a surjective homomorphism $(\mathbf{Z}/n\mathbf{Z})^* \rightarrow \{\pm 1\}$ and the kernel has index 2.

Proposition 2.5.2 implies that if $x^2 \equiv 1 \pmod{p}$ then $x = \pm 1 \pmod{p}$. In step 2, since $(a^{m/2})^2 \equiv 1 \pmod{n}$, we also have $(a^{m/2})^2 \equiv 1 \pmod{p}$ and $(a^{m/2})^2 \equiv 1 \pmod{q}$, so $a^{m/2} \equiv \pm 1 \pmod{p}$ and $a^{m/2} \equiv \pm 1 \pmod{q}$. Since $a^{m/2} \not\equiv 1 \pmod{n}$, there are three possibilities for these signs, so with probability $2/3$, one of the following two possibilities occurs:

1. $a^{m/2} \equiv +1 \pmod{p}$ and $a^{m/2} \equiv -1 \pmod{q}$
2. $a^{m/2} \equiv -1 \pmod{p}$ and $a^{m/2} \equiv +1 \pmod{q}$.

The only other possibility is that both signs are -1 . In the first case,

$$p \mid a^{m/2} - 1 \quad \text{but} \quad q \nmid a^{m/2} - 1,$$

so $\gcd(a^{m/2} - 1, pq) = p$, and we have factored n . Similarly, in the second case, $\gcd(a^{m/2} - 1, pq) = q$, and we again factor n .

Example 3.3.4. Somehow we discover that the RSA cryptosystem with

$$n = 32295194023343 \quad \text{and} \quad e = 29468811804857$$

has decryption key $d = 11127763319273$. We use this information and Algorithm 3.3.3 to factor n . If

$$m = ed - 1 = 327921963064646896263108960,$$

then $\varphi(pq) \mid m$, so $a^m \equiv 1 \pmod{n}$ for all a coprime to n . For each $a \leq 20$ we find that $a^{m/2} \equiv 1 \pmod{n}$, so we replace m by

$$\frac{m}{2} = 163960981532323448131554480.$$

Again, we find with this new m that for each $a \leq 20$, $a^{m/2} \equiv 1 \pmod{n}$, so we replace m by $81980490766161724065777240$. Yet again, for each $a \leq 20$, $a^{m/2} \equiv 1 \pmod{n}$, so we replace m by $40990245383080862032888620$. This is enough, since $2^{m/2} \equiv 4015382800099 \pmod{n}$. Then

$$\gcd(2^{m/2} - 1, n) = \gcd(4015382800098, 32295194023343) = 737531,$$

and we have found a factor of n . Dividing, we find that

$$n = 737531 \cdot 43788253.$$

3.3.4 Further Remarks

If one were to implement an actual RSA cryptosystem, there are many additional tricks and ideas to keep in mind. For example, one can add some extra random letters to each block of text, so that a given string will encrypt differently each time it is encrypted. This makes it more difficult for an attacker who knows the encrypted and plaintext versions of one message to gain information about subsequent encrypted messages. For an example implementation that incorporates this randomness, see Listing 7.3.4. In any particular implementation, there might be attacks that would be devastating in practice, but which wouldn't require factoring the RSA modulus.

RSA is in common use, e.g., it is used in OpenSSH protocol version 1 (see <http://www.openssh.com/>).

We will consider the ElGamal cryptosystem in Sections 6.3.2. It has a similar flavor to RSA, but is more flexible in some ways.

3.4 Exercises

- 3.1 This problem concerns encoding phrases using numbers using the encoding of Section 3.2.2. What is the longest that an arbitrary sequence of letters (no spaces) can be if it must fit in a number that is less than 10^{20} ?
- 3.2 Suppose Michael creates an RSA cryptosystem with a very large modulus n for which the factorization of n cannot be found in a reasonable amount of time. Suppose that Nikita sends messages to Michael by representing each alphabetic character as an integer between 0 and 26 (A corresponds to 1, B to 2, etc., and a space \square to 0), then encrypts each number *separately* using Michael's RSA cryptosystem. Is this method secure? Explain your answer.
- 3.3 For any $n \in \mathbf{N}$, let $\sigma(n)$ be the sum of the divisors of n ; for example, $\sigma(6) = 1 + 2 + 3 + 6 = 12$ and $\sigma(10) = 1 + 2 + 5 + 10 = 18$. Suppose that $n = pqr$ with p , q , and r distinct primes. Devise an "efficient" algorithm that given n , $\varphi(n)$ and $\sigma(n)$, computes the factorization of n . For example, if $n = 105$, then $p = 3$, $q = 5$, and $r = 7$, so the input to the algorithm would be

$$n = 105, \quad \varphi(n) = 48, \quad \text{and} \quad \sigma(n) = 192,$$

and the output would be 3, 5, and 7.

For computational exercises about cryptosystems, see the exercises for Chapter 7.

4

Quadratic Reciprocity

The linear equation

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

has a solution if and only if $\gcd(a, n)$ divides b (see Proposition 2.1.9). This chapter is about some amazing mathematics motivated by the search for a criterion for whether or not a quadratic equation

$$ax^2 + bx + c \equiv 0 \pmod{n}$$

has a solution. In many cases, the Chinese Remainder Theorem and the quadratic formula reduce this question to the key question of whether a given integer a is a perfect square modulo a prime p .

The quadratic reciprocity law of Gauss provides a precise answer to the following question: For which primes p is the image of a in $(\mathbf{Z}/p\mathbf{Z})^*$ a perfect square? Amazingly, the answer depends only on the reduction of p modulo $4a$.

There are over a hundred proofs of the quadratic reciprocity law (see [Lem] for a long list). We give two proofs. The first, which we give in Section 4.3, is completely elementary and involves keeping track of integer points in intervals. It is satisfying because one can understand every detail without much abstraction, but it is unsatisfying because it is difficult to conceptualize what is going on. In sharp contrast, our second proof, which we give in Section 4.4, is more abstract and uses a conceptual development of properties of Gauss sums. You should read Sections 4.1 and 4.2, then at least one of Section 4.3 or Section 4.4, depending on your taste and how much abstract algebra you know.

In Section 4.5, we return to the computational question of actually finding square roots and solving quadratic equations in practice.

4.1 Statement of the Quadratic Reciprocity Law

In this section we state the quadratic reciprocity law.

Definition 4.1.1 (Quadratic Residue). Fix a prime p . An integer a not divisible by p is *quadratic residue* modulo p if a is a square modulo p ; otherwise, a is a *quadratic nonresidue*.

The quadratic reciprocity theorem connects the question of whether or not a is a quadratic residue modulo p to the question of whether p is a quadratic residue modulo each of the prime divisors of a . To express it precisely, we introduce some new notation.

Definition 4.1.2 (Legendre Symbol). Let p be an odd prime and let a be an integer coprime to p . Set

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a \text{ is a quadratic residue, and} \\ -1 & \text{otherwise.} \end{cases}$$

We call this symbol the *Legendre Symbol*.

This notation is well entrenched in the literature, even though it is also the notation for “ a divided by p ”; be careful not to confuse the two.

Since $\left(\frac{a}{p}\right)$ only depends on $a \pmod{p}$, it makes sense to define $\left(\frac{a}{p}\right)$ for $a \in \mathbf{Z}/p\mathbf{Z}$ to be $\left(\frac{\tilde{a}}{p}\right)$ for any lift \tilde{a} of a to \mathbf{Z} .

Lemma 4.1.3. *The map $\psi : (\mathbf{Z}/p\mathbf{Z})^* \rightarrow \{\pm 1\}$ given by $\psi(a) = \left(\frac{a}{p}\right)$ is a surjective group homomorphism.*

Proof. By Theorem 2.5.5, $G = (\mathbf{Z}/p\mathbf{Z})^*$ is a cyclic group of order $p - 1$. Because p is odd, G has even order, so the subgroup H of squares of elements of G has index 2 in G . Since $\left(\frac{a}{p}\right) = 1$ if and only if $a \in H$, we see that ψ is the composition $G \rightarrow G/H \cong \{\pm 1\}$, where we identify the nontrivial element of G/H with -1 . \square

Remark 4.1.4. We could also prove that ψ is surjective without using that $(\mathbf{Z}/p\mathbf{Z})^*$ is cyclic, as follows. If $a \in (\mathbf{Z}/p\mathbf{Z})^*$ is a square, say $a \equiv b^2 \pmod{p}$, then $a^{(p-1)/2} = b^{p-1} \equiv 1 \pmod{p}$, so a is a root of $f = x^{(p-1)/2} - 1$. By Proposition 2.5.2, the polynomial f has at most $(p-1)/2$ roots. Thus there must be an $a \in (\mathbf{Z}/p\mathbf{Z})^*$ that is not a root of f , and for that a , we have $\psi(a) = \left(\frac{a}{p}\right) = -1$, and trivially $\psi(1) = 1$, so the map ψ is surjective. Note

TABLE 4.1. When is 5 a square modulo p ?

p	$\left(\frac{5}{p}\right)$	$p \bmod 5$	p	$\left(\frac{5}{p}\right)$	$p \bmod 5$
7	-1	2	29	1	4
11	1	1	31	1	1
13	-1	3	37	-1	2
17	-1	2	41	1	1
19	1	4	43	-1	3
23	-1	3	47	-1	2

that this argument does not prove that ψ is a homomorphism, though it can be extended to one that does.

The symbol $\left(\frac{a}{p}\right)$ only depends on the residue class of a modulo p , so making a table of values $\left(\frac{a}{p}\right)$ for many values of a would be easy. Would it be easy to make a table of $\left(\frac{5}{p}\right)$ for many p ? Probably, since there is a simple pattern in Table 4.1. It appears that $\left(\frac{5}{p}\right)$ depends only on the congruence class of p modulo 5. More precisely, $\left(\frac{5}{p}\right) = 1$ if and only if $p \equiv 1, 4 \pmod{5}$, i.e., $\left(\frac{5}{p}\right) = 1$ if and only if p is a square modulo 5.

Based on similar observations, in the 18th century various mathematicians found a conjectural explanation for the mystery suggested by Table 4.1. Finally, on April 8, 1796, at the age of 19, Gauss proved the following theorem.

Theorem 4.1.5 (Gauss's Quadratic Reciprocity Law). *Suppose p and q are distinct odd primes. Then*

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}} \left(\frac{q}{p}\right).$$

Also

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} \quad \text{and} \quad \left(\frac{2}{p}\right) = \begin{cases} 1 & \text{if } p \equiv \pm 1 \pmod{8} \\ -1 & \text{if } p \equiv \pm 3 \pmod{8}. \end{cases}$$

We will give two proofs of Gauss's formula relating $\left(\frac{p}{q}\right)$ to $\left(\frac{q}{p}\right)$. The first elementary proof is in Section 4.3, and the second more algebraic proof is in Section 4.4.

In our example Gauss's theorem implies that

$$\left(\frac{5}{p}\right) = (-1)^{2 \cdot \frac{p-1}{2}} \left(\frac{p}{5}\right) = \left(\frac{p}{5}\right) = \begin{cases} +1 & \text{if } p \equiv 1, 4 \pmod{5} \\ -1 & \text{if } p \equiv 2, 3 \pmod{5}. \end{cases}$$

As an application, the following example illustrates how to answer questions like “is a a square modulo b ” using Theorem 4.1.5.

Example 4.1.6. Is 69 a square modulo the prime 389? We have

$$\left(\frac{69}{389}\right) = \left(\frac{3 \cdot 23}{389}\right) = \left(\frac{3}{389}\right) \cdot \left(\frac{23}{389}\right) = (-1) \cdot (-1) = 1.$$

Here

$$\left(\frac{3}{389}\right) = \left(\frac{389}{3}\right) = \left(\frac{2}{3}\right) = -1,$$

and

$$\begin{aligned} \left(\frac{23}{389}\right) &= \left(\frac{389}{23}\right) = \left(\frac{21}{23}\right) = \left(\frac{-2}{23}\right) \\ &= \left(\frac{-1}{23}\right) \left(\frac{2}{23}\right) = (-1)^{\frac{23-1}{2}} \cdot 1 = -1. \end{aligned}$$

Thus 69 is a square modulo 389.

Though we know that 69 is a square modulo 389, we don’t know an explicit x such that $x^2 \equiv 69 \pmod{389}$! This is reminiscent of how we could prove using Theorem 2.1.12 that certain numbers are composite without knowing a factorization.

Remark 4.1.7. The Jacobi symbol is an extension of the Legendre symbol to composite moduli. For more details, see Exercise 4.8.

4.2 Euler’s Criterion

Let p be an odd prime and a an integer not divisible by p . Euler used the existence of primitive roots to show that $\left(\frac{a}{p}\right)$ is congruent to $a^{(p-1)/2}$ modulo p . We will use this fact repeatedly below in both proofs of Theorem 4.1.5.

Proposition 4.2.1 (Euler’s Criterion). *We have $\left(\frac{a}{p}\right) = 1$ if and only if*

$$a^{(p-1)/2} \equiv 1 \pmod{p}.$$

Proof. The map $\varphi : (\mathbf{Z}/p\mathbf{Z})^* \rightarrow (\mathbf{Z}/p\mathbf{Z})^*$ given by $\varphi(a) = a^{(p-1)/2}$ is a group homomorphism, since powering is a group homomorphism of any abelian group. Let $\psi : (\mathbf{Z}/p\mathbf{Z})^* \rightarrow \{\pm 1\}$ be the homomorphism $\psi(a) = \left(\frac{a}{p}\right)$ of Lemma 4.1.3. If $a \in \ker(\psi)$, then $a = b^2$ for some $b \in \mathbf{Z}/p\mathbf{Z}$, so

$$\varphi(a) = a^{(p-1)/2} = (b^2)^{(p-1)/2} = b^{p-1} = 1.$$

Thus $\ker(\psi) \subset \ker(\varphi)$. By Lemma 4.1.3, $\ker(\psi)$ has index 2 in $(\mathbf{Z}/p\mathbf{Z})^*$, so either $\ker(\varphi) = \ker(\psi)$ or $\varphi = 1$. If $\varphi = 1$, the polynomial $x^{(p-1)/2} - 1$

has $p - 1$ roots in the field $\mathbf{Z}/p\mathbf{Z}$, which contradicts Proposition 2.5.2, so $\ker(\varphi) = \ker(\psi)$, which proves the proposition. \square

From a computational point of view, Corollary 4.2.2 provides a convenient way to compute $\left(\frac{a}{p}\right)$. See Section 7.4.1 for an implementation.

Corollary 4.2.2. *The equation $x^2 \equiv a \pmod{p}$ has no solution if and only if $a^{(p-1)/2} \equiv -1 \pmod{p}$. Thus $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}$.*

Proof. This follows from Proposition 4.2.1 and the fact that the polynomial $x^2 - 1$ has no roots besides $+1$ and -1 (which follows from Proposition 2.5.3). \square

As additional computational motivation for the value of Corollary 4.2.2, note that to evaluate $\left(\frac{a}{p}\right)$ using Theorem 4.1.5 would not be practical if a and p both very large, because it would require factoring a . However, Corollary 4.2.2 provides a method for evaluating $\left(\frac{a}{p}\right)$ without factoring a .

Example 4.2.3. Suppose $p = 11$. By squaring each element of $(\mathbf{Z}/11\mathbf{Z})^*$, we see that the squares modulo 11 are $\{1, 3, 4, 5, 9\}$. We compute $a^{(p-1)/2} = a^5$ for each $a \in (\mathbf{Z}/11\mathbf{Z})^*$ and get

$$\begin{aligned} 1^5 &= 1, & 2^5 &= -1, & 3^5 &= 1, & 4^5 &= 1, & 5^5 &= 1, \\ 6^5 &= -1, & 7^5 &= -1, & 8^5 &= -1, & 9^5 &= 1, & 10^5 &= -1. \end{aligned}$$

Thus the a with $a^5 = 1$ are $\{1, 3, 4, 5, 9\}$, just as Proposition 4.2.1 predicts.

Example 4.2.4. We determine whether or not 3 is a square modulo the prime $p = 726377359$. Using a computer we find that

$$3^{(p-1)/2} \equiv -1 \pmod{726377359}.$$

Thus 3 is not a square modulo p . This computation wasn't difficult, but it would have been tedious by hand. The law of quadratic reciprocity provides a way to answer this question, which could easily be carried out by hand:

$$\begin{aligned} \left(\frac{3}{726377359}\right) &= (-1)^{(3-1)/2 \cdot (726377359-1)/2} \left(\frac{726377359}{3}\right) \\ &= (-1) \cdot \left(\frac{1}{3}\right) = -1. \end{aligned}$$

4.3 First Proof of Quadratic Reciprocity

Our first proof of quadratic reciprocity is elementary. The proof involves keeping track of integer points in intervals. Proving Gauss's lemma is the

first step; this lemma computes $\left(\frac{a}{p}\right)$ in terms of the number of integers of a certain type that lie in a certain interval. Next we prove Lemma 4.3.2, which controls how the parity of the number of integer points in an interval changes when an endpoint of the interval is changed. Then we prove that $\left(\frac{a}{p}\right)$ depends only on p modulo $4a$ by applying Gauss's lemma and keeping careful track of intervals as they are rescaled and their endpoints are changed. Finally, in Section 4.3.2 we use some basic algebra to deduce the quadratic reciprocity law using the tools we've just developed. Our proof follows the one given in [Dav99] closely.

Lemma 4.3.1 (Gauss's Lemma). *Let p be an odd prime and let a be an integer $\not\equiv 0 \pmod{p}$. Form the numbers*

$$a, 2a, 3a, \dots, \frac{p-1}{2}a$$

and reduce them modulo p to lie in the interval $(-\frac{p}{2}, \frac{p}{2})$. Let ν be the number of negative numbers in the resulting set. Then

$$\left(\frac{a}{p}\right) = (-1)^\nu.$$

Proof. In defining ν , we expressed each number in

$$S = \left\{ a, 2a, \dots, \frac{p-1}{2}a \right\}$$

as congruent to a number in the set

$$\left\{ 1, -1, 2, -2, \dots, \frac{p-1}{2}, -\frac{p-1}{2} \right\}.$$

No number $1, 2, \dots, \frac{p-1}{2}$ appears more than once, with either choice of sign, because if it did then either two elements of S are congruent modulo p or 0 is the sum of two elements of S , and both events are impossible. Thus the resulting set must be of the form

$$T = \left\{ \varepsilon_1 \cdot 1, \varepsilon_2 \cdot 2, \dots, \varepsilon_{(p-1)/2} \cdot \frac{p-1}{2} \right\},$$

where each ε_i is either $+1$ or -1 . Multiplying together the elements of S and of T , we see that

$$\begin{aligned} (1a) \cdot (2a) \cdot (3a) \cdots \left(\frac{p-1}{2}a\right) &\equiv \\ (\varepsilon_1 \cdot 1) \cdot (\varepsilon_2 \cdot 2) \cdots \left(\varepsilon_{(p-1)/2} \cdot \frac{p-1}{2}\right) &\pmod{p}, \end{aligned}$$

so

$$a^{(p-1)/2} \equiv \varepsilon_1 \cdot \varepsilon_2 \cdots \varepsilon_{(p-1)/2} \pmod{p}.$$

The lemma then follows from Proposition 4.2.1, since $\left(\frac{a}{p}\right) = a^{(p-1)/2}$. \square

4.3.1 Euler's Proposition

For rational numbers $a, b \in \mathbf{Q}$, let

$$(a, b) \cap \mathbf{Z} = \{x \in \mathbf{Z} : a \leq x \leq b\}$$

be the set of integers between a and b . The following lemma will help us to keep track of how many integers lie in certain intervals.

Lemma 4.3.2. *Let $a, b \in \mathbf{Q}$. Then for any integer n ,*

$$\#((a, b) \cap \mathbf{Z}) \equiv \#((a, b + 2n) \cap \mathbf{Z}) \pmod{2}$$

and

$$\#((a, b) \cap \mathbf{Z}) \equiv \#((a - 2n, b) \cap \mathbf{Z}) \pmod{2},$$

provided that each interval involved in the congruence is nonempty.

Note that if one of the intervals is empty, then the statement may be false; e.g., if $(a, b) = (-1/2, 1/2)$ and $n = -1$ then $\#((a, b) \cap \mathbf{Z}) = 1$ but $\#(a, b - 2) \cap \mathbf{Z} = 0$.

Proof. Let $\lceil x \rceil$ denotes the least integer $\geq x$. Since $n > 0$,

$$(a, b + 2n) = (a, b) \cup [b, b + 2n),$$

where the union is disjoint. There are $2n$ integers,

$$\lceil b \rceil, \lceil b \rceil + 1, \dots, \lceil b \rceil + 2n - 1,$$

in the interval $[b, b + 2n)$, so the first congruence of the lemma is true in this case. We also have

$$(a, b - 2n) = (a, b) \text{ minus } [b - 2n, b)$$

and $[b - 2n, b)$ contains exactly $2n$ integers, so the lemma is also true when n is negative. The statement about $\#((a - 2n, b) \cap \mathbf{Z})$ is proved in a similar manner. \square

Once we have proved the following proposition, it will be easy to deduce the quadratic reciprocity law.

Proposition 4.3.3 (Euler). *Let p be an odd prime and let a be a positive integer with $p \nmid a$. If q is a prime with $q \equiv \pm p \pmod{4a}$, then $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right)$.*

Proof. We will apply Lemma 4.3.1 to compute $\left(\frac{a}{p}\right)$. Let

$$S = \left\{ a, 2a, 3a, \dots, \frac{p-1}{2}a \right\}$$

and

$$I = \left(\frac{1}{2}p, p\right) \cup \left(\frac{3}{2}p, 2p\right) \cup \cdots \cup \left(\left(b - \frac{1}{2}\right)p, bp\right),$$

where $b = \frac{1}{2}a$ or $\frac{1}{2}(a-1)$, whichever is an integer. We check that every element of S that reduces to something in the interval $(-\frac{p}{2}, 0)$ lies in I . This is clear if $b = \frac{1}{2}a < \frac{p-1}{2}a$. If $b = \frac{1}{2}(a-1)$, then $bp + \frac{p}{2} > \frac{p-1}{2}a$, so $((b - \frac{1}{2})p, bp)$ is the last interval that could contain an element of S that reduces to $(-\frac{p}{2}, 0)$. Note that the integer endpoints of I are not in S , since those endpoints are divisible by p , but no element of S is divisible by p . Thus, by Lemma 4.3.1,

$$\left(\frac{a}{p}\right) = (-1)^{\#(S \cap I)}.$$

To compute $\#(S \cap I)$, first rescale by a to see that

$$\#(S \cap I) = \#\left(\mathbf{Z} \cap \frac{1}{a}I\right),$$

where

$$\frac{1}{a}I = \left(\left(\frac{p}{2a}, \frac{p}{a}\right) \cup \left(\frac{3p}{2a}, \frac{2p}{a}\right) \cup \cdots \cup \left(\frac{(2b-1)p}{2a}, \frac{bp}{a}\right)\right).$$

Write $p = 4ac + r$, and let

$$J = \left(\left(\frac{r}{2a}, \frac{r}{a}\right) \cup \left(\frac{3r}{2a}, \frac{2r}{a}\right) \cup \cdots \cup \left(\frac{(2b-1)r}{2a}, \frac{br}{a}\right)\right).$$

The only difference between I and J is that the endpoints of intervals are changed by addition of an even integer. By Lemma 4.3.2,

$$\nu = \#\left(\mathbf{Z} \cap \frac{1}{a}I\right) \equiv \#(\mathbf{Z} \cap J) \pmod{2}.$$

Thus $\left(\frac{a}{p}\right) = (-1)^\nu$ depends only on r , i.e., only on p modulo $4a$. Thus if $q \equiv p \pmod{4a}$, then $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right)$.

If $q \equiv -p \pmod{4a}$, then the only change in the above computation is that r is replaced by $4a - r$. This changes $\frac{1}{a}I$ into

$$K = \left(2 - \frac{r}{2a}, 4 - \frac{r}{a}\right) \cup \left(6 - \frac{3r}{2a}, 8 - \frac{2r}{a}\right) \cup \cdots \\ \cup \left(4b - 2 - \frac{(2b-1)r}{2a}, 4b - \frac{br}{a}\right).$$

Thus K is the same as $-\frac{1}{a}I$, except even integers have been added to the endpoints. By Lemma 4.3.2,

$$\#(K \cap \mathbf{Z}) \equiv \# \left(\left(\frac{1}{a}I \right) \cap \mathbf{Z} \right) \pmod{2},$$

so $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right)$, which completes the proof. \square

The following more careful analysis in the special case when $a = 2$ helps illustrate the proof of the above lemma, and the result is frequently useful in computations. For an alternative proof of the proposition, see Exercise 4.5.

Proposition 4.3.4 (Legendre symbol of 2). *Let p be an odd prime. Then*

$$\left(\frac{2}{p}\right) = \begin{cases} 1 & \text{if } p \equiv \pm 1 \pmod{8} \\ -1 & \text{if } p \equiv \pm 3 \pmod{8}. \end{cases}$$

Proof. When $a = 2$, the set $S = \{a, 2a, \dots, 2 \cdot \frac{p-1}{2}\}$ is

$$\{2, 4, 6, \dots, p-1\}.$$

We must count the parity of the number of elements of S that lie in the interval $I = (\frac{p}{2}, p)$. Writing $p = 8c + r$, we have

$$\begin{aligned} \#(I \cap S) &= \# \left(\frac{1}{2}I \cap \mathbf{Z} \right) = \# \left(\left(\frac{p}{4}, \frac{p}{2} \right) \cap \mathbf{Z} \right) \\ &= \# \left(\left(2c + \frac{r}{4}, 4c + \frac{r}{2} \right) \cap \mathbf{Z} \right) \equiv \# \left(\left(\frac{r}{4}, \frac{r}{2} \right) \cap \mathbf{Z} \right) \pmod{2}, \end{aligned}$$

where the last equality comes from Lemma 4.3.2. The possibilities for r are 1, 3, 5, 7. When $r = 1$, the cardinality is 0, when $r = 3, 5$ it is 1, and when $r = 7$ it is 2. \square

4.3.2 Proof of Quadratic Reciprocity

It is now straightforward to deduce the quadratic reciprocity law.

First Proof of Theorem 4.1.5. First suppose that $p \equiv q \pmod{4}$. By swapping p and q if necessary, we may assume that $p > q$, and write $p - q = 4a$. Since $p = 4a + q$,

$$\left(\frac{p}{q}\right) = \left(\frac{4a + q}{q}\right) = \left(\frac{4a}{q}\right) = \left(\frac{4}{q}\right) \left(\frac{a}{q}\right) = \left(\frac{a}{q}\right),$$

and

$$\left(\frac{q}{p}\right) = \left(\frac{p - 4a}{p}\right) = \left(\frac{-4a}{p}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{a}{p}\right).$$

Proposition 4.3.3 implies that $\left(\frac{a}{q}\right) = \left(\frac{a}{p}\right)$, since $p \equiv q \pmod{4a}$. Thus

$$\left(\frac{p}{q}\right) \cdot \left(\frac{q}{p}\right) = \left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}},$$

where the last equality is because $\frac{p-1}{2}$ is even if and only if $\frac{q-1}{2}$ is even.

Next suppose that $p \not\equiv q \pmod{4}$, so $p \equiv -q \pmod{4}$. Write $p+q = 4a$. We have

$$\left(\frac{p}{q}\right) = \left(\frac{4a-q}{q}\right) = \left(\frac{a}{q}\right), \quad \text{and} \quad \left(\frac{q}{p}\right) = \left(\frac{4a-p}{p}\right) = \left(\frac{a}{p}\right).$$

Since $p \equiv -q \pmod{4a}$, Proposition 4.3.3 implies that $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$. Since $(-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}} = 1$, the proof is complete. \square

4.4 A Proof of Quadratic Reciprocity Using Gauss Sums

In this section we present a beautiful proof of Theorem 4.1.5 using algebraic identities satisfied by sums of “roots of unity”. The objects we introduce in the proof are of independent interest, and provide a powerful tool to prove higher-degree analogues of quadratic reciprocity. (For more on higher reciprocity see [IR90]. See also Section 6 of [IR90] on which the proof below is modeled.)

Definition 4.4.1 (Root of Unity). An n th root of unity is a complex number ζ such that $\zeta^n = 1$. A root of unity ζ is a *primitive* n th root of unity if n is the smallest positive integer such that $\zeta^n = 1$.

For example, -1 is a primitive second root of unity, and $\zeta = \frac{\sqrt{-3}-1}{2}$ is a primitive cube root of unity. More generally, for any $n \in \mathbf{N}$ the complex number

$$\zeta_n = \cos(2\pi/n) + i \sin(2\pi/n)$$

is a primitive n th root of unity (this follows from the identity $e^{i\theta} = \cos(\theta) + i \sin(\theta)$). For the rest of this section, we fix an odd prime p and the primitive p th root $\zeta = \zeta_p$ of unity.

Definition 4.4.2 (Gauss Sum). Fix an odd prime p . The *Gauss sum* associated to an integer a is

$$g_a = \sum_{n=0}^{p-1} \left(\frac{n}{p}\right) \zeta^{an},$$

where $\zeta = \zeta_p = \cos(2\pi/p) + i \sin(2\pi/p)$.

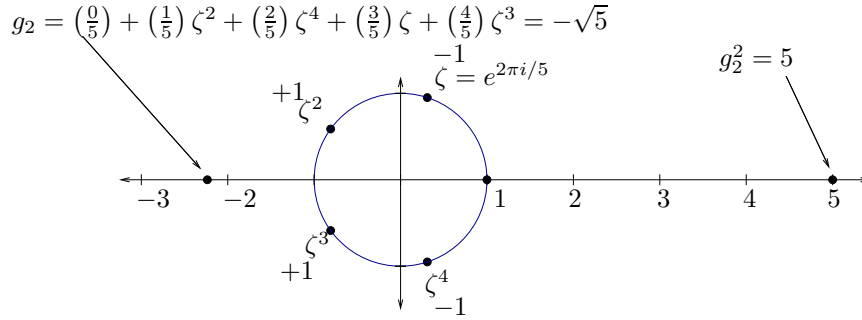


FIGURE 4.1. Gauss sum g_2 for $p = 5$

Note that p is implicit in the definition of g_a . If we were to change p , then the Gauss sum g_a associated to a would be different. The definition of g_a also depends on our choice of ζ ; we've chosen $\zeta = \zeta_p$, but could have chosen a different ζ and then g_a could be different.

Figure 4.1 illustrates the Gauss sum g_2 for $p = 5$. The Gauss sum is obtained by adding the points on the unit circle, with signs as indicated, to obtain the real number $-\sqrt{5}$. This suggests the following proposition, whose proof will require some work.

Proposition 4.4.3 (Gauss sum). *For any a not divisible by p ,*

$$g_a^2 = (-1)^{(p-1)/2} p.$$

In order to prove the proposition, we introduce a few lemmas.

Lemma 4.4.4. *For any integer a ,*

$$\sum_{n=0}^{p-1} \zeta^{an} = \begin{cases} p & \text{if } a \equiv 0 \pmod{p}, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. If $a \equiv 0 \pmod{p}$, then $\zeta^a = 1$, so the sum equals the number of summands, which is p . If $a \not\equiv 0 \pmod{p}$, then we use the identity

$$x^p - 1 = (x - 1)(x^{p-1} + \dots + x + 1)$$

with $x = \zeta^a$. We have $\zeta^a \neq 1$, so $\zeta^a - 1 \neq 0$ and

$$\sum_{n=0}^{p-1} \zeta^{an} = \frac{\zeta^{ap} - 1}{\zeta^a - 1} = \frac{1 - 1}{\zeta^a - 1} = 0.$$

□

Lemma 4.4.5. *If x and y are arbitrary integers, then*

$$\sum_{n=0}^{p-1} \zeta^{(x-y)n} = \begin{cases} p & \text{if } x \equiv y \pmod{p}, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. This follows from Lemma 4.4.4 by setting $a = x - y$. \square

Lemma 4.4.6. *We have $g_0 = 0$.*

Proof. By definition

$$g_0 = \sum_{n=0}^{p-1} \left(\frac{n}{p}\right). \quad (4.4.1)$$

By Lemma 4.1.3, the map

$$\left(\frac{\cdot}{p}\right) : (\mathbf{Z}/p\mathbf{Z})^* \rightarrow \{\pm 1\}$$

is a surjective homomorphism of groups. Thus half the elements of $(\mathbf{Z}/p\mathbf{Z})^*$ map to $+1$ and half map to -1 (the subgroup that maps to $+1$ has index 2). Since $\left(\frac{0}{p}\right) = 0$, the sum (4.4.1) is 0. \square

Lemma 4.4.7. *For any integer a ,*

$$g_a = \left(\frac{a}{p}\right) g_1.$$

Proof. When $a \equiv 0 \pmod{p}$ the lemma follows from Lemma 4.4.6, so suppose that $a \not\equiv 0 \pmod{p}$. Then

$$\left(\frac{a}{p}\right) g_a = \left(\frac{a}{p}\right) \sum_{n=0}^{p-1} \left(\frac{n}{p}\right) \zeta^{an} = \sum_{n=0}^{p-1} \left(\frac{an}{p}\right) \zeta^{an} = \sum_{m=0}^{p-1} \left(\frac{m}{p}\right) \zeta^m = g_1.$$

Here we use that multiplication by a is an automorphism of $\mathbf{Z}/p\mathbf{Z}$. Finally, multiply both sides by $\left(\frac{a}{p}\right)$ and use that $\left(\frac{a}{p}\right)^2 = 1$. \square

We have enough lemmas to prove Proposition 4.4.3.

Proof of Proposition 4.4.3. We evaluate the sum $\sum_{a=0}^{p-1} g_a g_{-a}$ in two different ways. By Lemma 4.4.7, since $a \not\equiv 0 \pmod{p}$ we have

$$g_a g_{-a} = \left(\frac{a}{p}\right) g_1 \left(\frac{-a}{p}\right) g_1 = \left(\frac{-1}{p}\right) \left(\frac{a}{p}\right)^2 g_1^2 = (-1)^{(p-1)/2} g_1^2,$$

where the last step follows from Proposition 4.2.1 and that $\left(\frac{a}{p}\right) \in \{\pm 1\}$. Thus

$$\sum_{a=0}^{p-1} g_a g_{-a} = (p-1)(-1)^{(p-1)/2} g_1^2. \quad (4.4.2)$$

On the other hand, by definition

$$\begin{aligned} g_a g_{-a} &= \sum_{n=0}^{p-1} \left(\frac{n}{p}\right) \zeta^{an} \cdot \sum_{m=0}^{p-1} \left(\frac{m}{p}\right) \zeta^{-am} \\ &= \sum_{n=0}^{p-1} \sum_{m=0}^{p-1} \left(\frac{n}{p}\right) \left(\frac{m}{p}\right) \zeta^{an} \zeta^{-am} \\ &= \sum_{n=0}^{p-1} \sum_{m=0}^{p-1} \left(\frac{n}{p}\right) \left(\frac{m}{p}\right) \zeta^{an-am}. \end{aligned}$$

Let $\delta(n, m) = 1$ if $n \equiv m \pmod{p}$ and 0 otherwise. By Lemma 4.4.5,

$$\begin{aligned} \sum_{a=0}^{p-1} g_a g_{-a} &= \sum_{a=0}^{p-1} \sum_{n=0}^{p-1} \sum_{m=0}^{p-1} \left(\frac{n}{p}\right) \left(\frac{m}{p}\right) \zeta^{an-am} \\ &= \sum_{n=0}^{p-1} \sum_{m=0}^{p-1} \left(\frac{n}{p}\right) \left(\frac{m}{p}\right) \sum_{a=0}^{p-1} \zeta^{an-am} \\ &= \sum_{n=0}^{p-1} \sum_{m=0}^{p-1} \left(\frac{n}{p}\right) \left(\frac{m}{p}\right) p\delta(n, m) \\ &= \sum_{n=0}^{p-1} \left(\frac{n}{p}\right)^2 p \\ &= p(p-1). \end{aligned}$$

Equate (4.4.2) and the above equality, then cancel $(p-1)$ to see that

$$g_1^2 = (-1)^{(p-1)/2} p.$$

Since $a \not\equiv 0 \pmod{p}$, we have $\left(\frac{a}{p}\right)^2 = 1$, so by Lemma 4.4.7,

$$g_a^2 = \left(\frac{a}{p}\right)^2 g_1^2 = g_1^2,$$

and the proposition is proved. \square

4.4.1 Proof of Quadratic Reciprocity

We are now ready to prove Theorem 4.1.5 using Gauss sums.

Proof. Let q be an odd prime with $q \neq p$. Set $p^* = (-1)^{(p-1)/2} p$ and recall that Proposition 4.4.3 asserts that $p^* = g^2$, where $g = g_1 = \sum_{n=0}^{p-1} \left(\frac{n}{p}\right) \zeta^n$.

Proposition 4.2.1 implies that

$$(p^*)^{(q-1)/2} \equiv \left(\frac{p^*}{q}\right) \pmod{q}.$$

We have $g^{q-1} = (g^2)^{(q-1)/2} = (p^*)^{(q-1)/2}$, so multiplying both sides of the displayed equation by g yields a congruence

$$g^q \equiv g \left(\frac{p^*}{q}\right) \pmod{q}. \quad (4.4.3)$$

But wait, what does this congruence mean, given that g^q is not an integer? It means that the difference $g^q - g \left(\frac{p^*}{q}\right)$ lies in the ideal (q) in the ring $\mathbf{Z}[\zeta]$ of all polynomials in ζ with coefficients in \mathbf{Z} .

The ring $\mathbf{Z}[\zeta]/(q)$ has characteristic q , so if $x, y \in \mathbf{Z}[\zeta]$, then $(x + y)^q \equiv x^q + y^q \pmod{q}$. Applying this to (4.4.3), we see that

$$g^q = \left(\sum_{n=0}^{p-1} \binom{n}{p} \zeta^n\right)^q \equiv \sum_{n=0}^{p-1} \binom{n}{p}^q \zeta^{nq} \equiv \sum_{n=0}^{p-1} \binom{n}{p} \zeta^{nq} \equiv g_q \pmod{q}.$$

By Lemma 4.4.7,

$$g^q \equiv g_q \equiv \left(\frac{q}{p}\right) g \pmod{q}.$$

Combining this with (4.4.3) yields

$$\left(\frac{q}{p}\right) g \equiv \left(\frac{p^*}{q}\right) g \pmod{q}.$$

Since $g^2 = p^*$ and $p \neq q$, we can cancel g from both sides to find that $\left(\frac{q}{p}\right) \equiv \left(\frac{p^*}{q}\right) \pmod{q}$. Since both residue symbols are ± 1 and q is odd, it follows that $\left(\frac{q}{p}\right) = \left(\frac{p^*}{q}\right)$. Finally, we note using Proposition 4.2.1 that

$$\left(\frac{p^*}{q}\right) = \left(\frac{(-1)^{(p-1)/2} p}{q}\right) = \left(\frac{-1}{q}\right)^{(p-1)/2} \left(\frac{p}{q}\right) = (-1)^{\frac{q-1}{2} \cdot \frac{p-1}{2}} \cdot \left(\frac{p}{q}\right).$$

□

4.5 Finding Square Roots

We return in this section to the question of computing square roots. If K is a field in which $2 \neq 0$, and $a, b, c \in K$, with $a \neq 0$, then the solutions to the quadratic equation $ax^2 + bx + c = 0$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Now assume $K = \mathbf{Z}/p\mathbf{Z}$, with p an odd prime. Using Theorem 4.1.5, we can decide whether or not $b^2 - 4ac$ is a perfect square in $\mathbf{Z}/p\mathbf{Z}$, and hence whether or not $ax^2 + bx + c = 0$ has a solution in $\mathbf{Z}/p\mathbf{Z}$. However Theorem 4.1.5 says nothing about how to actually find a solution when there is one. Also, note that for this problem we do *not* need the full quadratic reciprocity law; in practice to decide whether an element of $\mathbf{Z}/p\mathbf{Z}$ is a perfect square Proposition 4.2.1 is quite fast, in view of Section 2.3.

Suppose $a \in \mathbf{Z}/p\mathbf{Z}$ is a nonzero quadratic residue. If $p \equiv 3 \pmod{4}$ then $b = a^{\frac{p+1}{4}}$ is a square root of a because

$$b^2 = a^{\frac{p+1}{2}} = a^{\frac{p-1}{2}+1} = a^{\frac{p-1}{2}} \cdot a = \left(\frac{a}{p}\right) \cdot a = a.$$

We can compute b in time polynomial in the number of digits of p using the powering algorithm of Section 2.3.

We do not know a deterministic polynomial-time algorithm to compute a square root of a when $p \equiv 1 \pmod{4}$. The following is a standard probabilistic algorithm to compute a square root of a , which works well in practice. Consider the quotient ring

$$R = (\mathbf{Z}/p\mathbf{Z})[x]/(x^2 - a),$$

by which we mean the following. We have

$$R = \{u + v\alpha : u, v \in \mathbf{Z}/p\mathbf{Z}\}$$

with multiplication defined by

$$(u + v\alpha)(z + w\alpha) = (uz + awv) + (uw + vz)\alpha.$$

Here α corresponds to the class of x in the quotient ring. Let b and c be the square roots of a in $\mathbf{Z}/p\mathbf{Z}$ (though we cannot easily compute b and c yet, we can consider them in order to deduce an algorithm to find them). We have ring homomorphisms $f : R \rightarrow \mathbf{Z}/p\mathbf{Z}$ and $g : R \rightarrow \mathbf{Z}/p\mathbf{Z}$ given by $f(u + v\alpha) = u + vb$ and $g(u + v\alpha) = u + vc$. Together these define a ring isomorphism

$$\varphi : R \longrightarrow \mathbf{Z}/p\mathbf{Z} \times \mathbf{Z}/p\mathbf{Z}$$

given by $\varphi(u + v\alpha) = (u + vb, u + vc)$. Choose in some way a random element z of $(\mathbf{Z}/p\mathbf{Z})^*$, and define $u, v \in \mathbf{Z}/p\mathbf{Z}$ by

$$u + v\alpha = (1 + z\alpha)^{\frac{p-1}{2}},$$

where we compute $(1 + z\alpha)^{\frac{p-1}{2}}$ quickly using an analogue of the binary powering algorithm of Section 2.3.2. If $v = 0$ we try again with another random z . If $v \neq 0$ we can quickly find the desired square roots b and c as follows. The quantity $u + vb$ is a $(p-1)/2$ power in $\mathbf{Z}/p\mathbf{Z}$, so it equals

either 0, 1, or -1 , so $b = -u/v$, $(1-u)/v$, or $(-1-u)/v$, respectively. Since we know u and v we can try each of $-u/v$, $(1-u)/v$, and $(-1-u)/v$ and see which is a square root of a .

We implement this algorithm in Section 7.4.2.

Example 4.5.1. Continuing Example 4.1.6, we find a square root of 69 modulo 389. We apply the algorithm described above in the case $p \equiv 1 \pmod{4}$. We first choose the random $z = 24$ and find that $(1 + 24\alpha)^{194} = -1$. The coefficient of α in the power is 0, and we try again with $z = 51$. This time we have $(1 + 51\alpha)^{194} = 239\alpha = u + v\alpha$. The inverse of 239 in $\mathbf{Z}/389\mathbf{Z}$ is 153, so we consider the following three possibilities for a square root of 69:

$$-\frac{u}{v} = 0 \quad \frac{1-u}{v} = 153 \quad -\frac{1-u}{v} = -153.$$

Thus 153 and -153 are the square roots of 69 in $\mathbf{Z}/389\mathbf{Z}$.

4.6 Exercises

4.1 Calculate the following by hand: $\left(\frac{3}{97}\right)$, $\left(\frac{3}{389}\right)$, $\left(\frac{22}{11}\right)$, and $\left(\frac{51}{7}\right)$.

4.2 Use Theorem 4.1.5 to show that for $p \geq 5$ prime,

$$\left(\frac{3}{p}\right) = \begin{cases} 1 & \text{if } p \equiv 1, 11 \pmod{12}, \\ -1 & \text{if } p \equiv 5, 7 \pmod{12}. \end{cases}$$

4.3 (*) Use that $(\mathbf{Z}/p\mathbf{Z})^*$ is cyclic to give a direct proof that $\left(\frac{-3}{p}\right) = 1$ when $p \equiv 1 \pmod{3}$. (Hint: There is an $c \in (\mathbf{Z}/p\mathbf{Z})^*$ of order 3. Show that $(2c+1)^2 = -3$.)

4.4 (*) If $p \equiv 1 \pmod{5}$, show directly that $\left(\frac{5}{p}\right) = 1$ by the method of Exercise 4.3. (Hint: Let $c \in (\mathbf{Z}/p\mathbf{Z})^*$ be an element of order 5. Show that $(c+c^4)^2 + (c+c^4) - 1 = 0$, etc.)

4.5 (*) Let p be an odd prime. In this exercise you will prove that $\left(\frac{2}{p}\right) = 1$ if and only if $p \equiv \pm 1 \pmod{8}$.

(a) Prove that

$$x = \frac{1-t^2}{1+t^2}, \quad y = \frac{2t}{1+t^2}$$

is a parameterization of the set of solutions to $x^2 + y^2 \equiv 1 \pmod{p}$, in the sense that the solutions $(x, y) \in \mathbf{Z}/p\mathbf{Z}$ are in bijection with the $t \in \mathbf{Z}/p\mathbf{Z} \cup \{\infty\}$ such that $1+t^2 \not\equiv 0 \pmod{p}$.

Here $t = \infty$ corresponds to the point $(-1, 0)$. (Hint: if (x_1, y_1) is a solution, consider the line $y = t(x + 1)$ through (x_1, y_1) and $(-1, 0)$, and solve for x_1, y_1 in terms of t .)

- (b) Prove that the number of solutions to $x^2 + y^2 \equiv 1 \pmod{p}$ is $p + 1$ if $p \equiv 3 \pmod{4}$ and $p - 1$ if $p \equiv 1 \pmod{4}$.
- (c) Consider the set S of pairs $(a, b) \in (\mathbf{Z}/p\mathbf{Z})^* \times (\mathbf{Z}/p\mathbf{Z})^*$ such that $a + b = 1$ and $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right) = 1$. Prove that $\#S = (p + 1 - 4)/4$ if $p \equiv 3 \pmod{4}$ and $\#S = (p - 1 - 4)/4$ if $p \equiv 1 \pmod{4}$. Conclude that $\#S$ is odd if and only if $p \equiv \pm 1 \pmod{8}$.
- (d) The map $\sigma(a, b) = (b, a)$ that swaps coordinates is a bijection of the set S . It has exactly one fixed point if and only if there is an $a \in \mathbf{Z}/p\mathbf{Z}$ such that $2a = 1$ and $\left(\frac{a}{p}\right) = 1$. Also, prove that $2a = 1$ has a solution $a \in \mathbf{Z}/p\mathbf{Z}$ with $\left(\frac{a}{p}\right) = 1$ if and only if $\left(\frac{2}{p}\right) = 1$.
- (e) Finish by showing that σ has exactly one fixed point if and only if $\#S$ is odd, i.e., if and only if $p \equiv \pm 1 \pmod{8}$.

Remark: The method of proof of this exercise can be generalized to give a proof of the full quadratic reciprocity law.

4.6 How many natural numbers $x < 2^{13}$ satisfy the equation

$$x^2 \equiv 5 \pmod{2^{13} - 1}?$$

You may assume that $2^{13} - 1$ is prime.

4.7 Find the natural number $x < 97$ such that $x \equiv 4^{48} \pmod{97}$. Note that 97 is prime.

4.8 In this problem we will formulate an analogue of quadratic reciprocity for a symbol like $\left(\frac{a}{q}\right)$, but without the restriction that q be a prime.

Suppose n is a positive integer, which we factor as $\prod_{i=1}^k p_i^{e_i}$. We define the Jacobi symbol $\left(\frac{a}{n}\right)$ as follows:

$$\left(\frac{a}{n}\right) = \prod_{i=1}^k \left(\frac{a}{p_i}\right)^{e_i}.$$

- (a) Give an example to show that $\left(\frac{a}{n}\right) = 1$ need not imply that a is a perfect square modulo n .
- (b) (*) Let n be odd and a and b be integers. Prove that the following holds:

- i. $\left(\frac{a}{n}\right)\left(\frac{b}{n}\right) = \left(\frac{ab}{n}\right)$. (Thus $a \mapsto \left(\frac{a}{n}\right)$ induces a homomorphism from $(\mathbf{Z}/n\mathbf{Z})^*$ to $\{\pm 1\}$.)
- ii. $\left(\frac{-1}{n}\right) \equiv n \pmod{4}$.
- iii. $\left(\frac{2}{n}\right) = 1$ if $n \equiv \pm 1 \pmod{8}$ and -1 otherwise.
- iv. $\left(\frac{a}{n}\right) = (-1)^{\frac{a-1}{2} \cdot \frac{n-1}{2}} \left(\frac{n}{a}\right)$

4.9 (*) Prove that for any $n \in \mathbf{Z}$ the integer $n^2 + n + 1$ does not have any divisors of the form $6k - 1$.

5

Continued Fractions

A *continued fraction* is an expression of the form

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots}}}$$

In this book we will assume that the a_i are real numbers and $a_i > 0$ for $i \geq 1$, and the expression may or may not go on indefinitely. More general notions of continued fractions have been extensively studied, but they are beyond the scope of this book. We will be most interested in the case when the a_i are all integers.

We denote the continued fraction displayed above by

$$[a_0, a_1, a_2, \dots].$$

For example,

$$\begin{aligned} [1, 2] &= 1 + \frac{1}{2} = \frac{3}{2}, \\ [3, 7, 15, 1, 292] &= 3 + \frac{1}{7 + \frac{1}{15 + \frac{1}{1 + \frac{1}{292}}}} \\ &= \frac{103993}{33102} = 3.14159265301190260407\dots, \end{aligned}$$

and

$$\begin{aligned}
 [2, 1, 2, 1, 1, 4, 1, 1, 6] &= 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{4 + \frac{1}{1 + \frac{1}{1 + \frac{1}{6}}}}}}}}} \\
 &= \frac{1264}{465} \\
 &= 2.7182795698924731182795698\dots
 \end{aligned}$$

The second two examples were chosen to foreshadow that continued fractions can be used to obtain good rational approximations to irrational numbers. Note that the first approximates π and the second e .

Continued fractions have many applications. For example, they provide an algorithmic way to recognize a decimal approximation to a rational number. Continued fractions also suggest a sense in which e might be “less complicated” than π (see Example 5.2.3 and Section 5.3).

In Section 5.1 we study continued fractions $[a_0, a_1, \dots, a_n]$ of finite length and lay the foundations for our later investigations. In Section 5.2 we give the continued fraction procedure, which associates to a real number x a sequence a_0, a_1, \dots of integers such that $x = \lim_{n \rightarrow \infty} [a_0, a_1, \dots, a_n]$. We also prove that if a_0, a_1, \dots is any infinite sequence of positive integers, then the sequence $c_n = [a_0, a_1, \dots, a_n]$ converges; more generally, we prove that if the a_n are arbitrary positive real numbers and $\sum_{n=0}^{\infty} a_n$ diverges then (c_n) converges. In Section 5.4, we prove that a continued fraction with $a_i \in \mathbf{N}$ is (eventually) periodic if and only if its value is a non-rational root of a quadratic polynomial, then discuss open questions concerning continued fractions of roots of irreducible polynomials of degree greater than 2. We conclude the chapter with applications of continued fractions to recognizing approximations to rational numbers (Section 5.5) and writing integers as sums of two squares (Section 5.6).

The reader is encouraged to read more about continued fractions in [HW79, Ch. X], [Khi63], [Bur89, §13.3], and [NZM91, Ch. 7].

5.1 Finite Continued Fractions

This section is about continued fractions of the form $[a_0, a_1, \dots, a_m]$ for some $m \geq 0$. We give an inductive definition of numbers p_n and q_n such

that for all $n \leq m$

$$[a_0, a_1, \dots, a_n] = \frac{p_n}{q_n}. \quad (5.1.1)$$

We then give related formulas for the determinants of the 2×2 matrices $\begin{pmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{pmatrix}$ and $\begin{pmatrix} p_n & p_{n-2} \\ q_n & q_{n-2} \end{pmatrix}$, which we will repeatedly use to deduce properties of the sequence of partial convergents $[a_0, \dots, a_k]$. We will use Algorithm 1.1.12 to prove that every rational number is represented by a continued fraction, as in (5.1.1).

Definition 5.1.1 (Finite Continued Fraction). A *finite continued fraction* is an expression

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots + \frac{1}{a_n}}}},$$

where each a_m is a real number and $a_m > 0$ for all $m \geq 1$.

Definition 5.1.2 (Simple Continued Fraction). A *simple continued fraction* is a finite or infinite continued fraction in which the a_i are all integers.

To get a feeling for continued fractions, observe that

$$\begin{aligned} [a_0] &= a_0, \\ [a_0, a_1] &= a_0 + \frac{1}{a_1} = \frac{a_0 a_1 + 1}{a_1}, \\ [a_0, a_1, a_2] &= a_0 + \frac{1}{a_1 + \frac{1}{a_2}} = \frac{a_0 a_1 a_2 + a_0 + a_2}{a_1 a_2 + 1}. \end{aligned}$$

Also,

$$\begin{aligned} [a_0, a_1, \dots, a_{n-1}, a_n] &= \left[a_0, a_1, \dots, a_{n-2}, a_{n-1} + \frac{1}{a_n} \right] \\ &= a_0 + \frac{1}{[a_1, \dots, a_n]} \\ &= [a_0, [a_1, \dots, a_n]]. \end{aligned}$$

5.1.1 Partial Convergents

Fix a finite continued fraction $[a_0, \dots, a_m]$. We do not assume at this point that the a_i are integers.

Definition 5.1.3 (Partial convergents). For $0 \leq n \leq m$, the n th *convergent* of the continued fraction $[a_0, \dots, a_m]$ is $[a_0, \dots, a_n]$. These convergents for $n < m$ are also called *partial convergents*.

For each n with $-2 \leq n \leq m$, define real numbers p_n and q_n as follows:

$$\begin{aligned} p_{-2} = 0, & \quad p_{-1} = 1, & \quad p_0 = a_0, & \quad \cdots & \quad p_n = a_n p_{n-1} + p_{n-2} & \quad \cdots, \\ q_{-2} = 1, & \quad q_{-1} = 0, & \quad q_0 = 1, & \quad \cdots & \quad q_n = a_n q_{n-1} + q_{n-2} & \quad \cdots. \end{aligned}$$

Proposition 5.1.4 (Partial Convergents). *For $n \geq 0$ we have*

$$[a_0, \dots, a_n] = \frac{p_n}{q_n}.$$

Proof. We use induction. The assertion is obvious when $n = 0, 1$. Suppose the proposition is true for all continued fractions of length $n - 1$. Then

$$\begin{aligned} [a_0, \dots, a_n] &= [a_0, \dots, a_{n-2}, a_{n-1} + \frac{1}{a_n}] \\ &= \frac{\left(a_{n-1} + \frac{1}{a_n}\right) p_{n-2} + p_{n-3}}{\left(a_{n-1} + \frac{1}{a_n}\right) q_{n-2} + q_{n-3}} \\ &= \frac{(a_{n-1} a_n + 1) p_{n-2} + a_n p_{n-3}}{(a_{n-1} a_n + 1) q_{n-2} + a_n q_{n-3}} \\ &= \frac{a_n (a_{n-1} p_{n-2} + p_{n-3}) + p_{n-2}}{a_n (a_{n-1} q_{n-2} + q_{n-3}) + q_{n-2}} \\ &= \frac{a_n p_{n-1} + p_{n-2}}{a_n q_{n-1} + q_{n-2}} \\ &= \frac{p_n}{q_n}. \end{aligned}$$

□

Proposition 5.1.5. *For $n \geq 0$ we have*

$$p_n q_{n-1} - q_n p_{n-1} = (-1)^{n-1} \tag{5.1.2}$$

and

$$p_n q_{n-2} - q_n p_{n-2} = (-1)^n a_n. \tag{5.1.3}$$

Equivalently,

$$\frac{p_n}{q_n} - \frac{p_{n-1}}{q_{n-1}} = (-1)^{n-1} \cdot \frac{1}{q_n q_{n-1}}$$

and

$$\frac{p_n}{q_n} - \frac{p_{n-2}}{q_{n-2}} = (-1)^n \cdot \frac{a_n}{q_n q_{n-2}}.$$

Proof. The case for $n = 0$ is obvious from the definitions. Now suppose $n > 0$ and the statement is true for $n - 1$. Then

$$\begin{aligned} p_n q_{n-1} - q_n p_{n-1} &= (a_n p_{n-1} + p_{n-2}) q_{n-1} - (a_n q_{n-1} + q_{n-2}) p_{n-1} \\ &= p_{n-2} q_{n-1} - q_{n-2} p_{n-1} \\ &= -(p_{n-1} q_{n-2} - p_{n-2} q_{n-1}) \\ &= -(-1)^{n-2} = (-1)^{n-1}. \end{aligned}$$

This completes the proof of (5.1.2). For (5.1.3), we have

$$\begin{aligned} p_n q_{n-2} - p_{n-2} q_n &= (a_n p_{n-1} + p_{n-2}) q_{n-2} - p_{n-2} (a_n q_{n-1} + q_{n-2}) \\ &= a_n (p_{n-1} q_{n-2} - p_{n-2} q_{n-1}) \\ &= (-1)^n a_n. \end{aligned}$$

□

Remark 5.1.6. Expressed in terms of matrices, the proposition asserts that the determinant of $\begin{pmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{pmatrix}$ is $(-1)^{n-1}$, and of $\begin{pmatrix} p_n & p_{n-2} \\ q_n & q_{n-2} \end{pmatrix}$ is $(-1)^n a_n$.

Corollary 5.1.7 (Convergents in lowest terms). *If $[a_0, a_1, \dots, a_m]$ is a simple continued fraction, so each a_i is an integer, then the p_n and q_n are integers and the fraction p_n/q_n is in lowest terms.*

Proof. It is clear that the p_n and q_n are integers, from the formula that defines them. If d is a positive divisor of both p_n and q_n , then $d \mid (-1)^{n-1}$, so $d = 1$. □

5.1.2 The Sequence of Partial Convergents

Let $[a_0, \dots, a_m]$ be a continued fraction and for $n \leq m$ let

$$c_n = [a_0, \dots, a_n] = \frac{p_n}{q_n}$$

denote the n th convergent. Recall that by definition of continued fraction, $a_n > 0$ for $n > 0$, which gives the partial convergents of a continued fraction additional structure. For example, the partial convergents of $[2, 1, 2, 1, 1, 4, 1, 1, 6]$ are

$$2, 3, 8/3, 11/4, 19/7, 87/32, 106/39, 193/71, 1264/465.$$

To make the size of these numbers clearer, we approximate them using decimals. We also underline every other number, to illustrate some extra structure.

$$\underline{2}, 3, \underline{2.66667}, 2.75000, \underline{2.71429}, 2.71875, \underline{2.71795}, 2.71831, \underline{2.71828}$$

The underlined numbers are smaller than all of the non-underlined numbers, and the sequence of underlined numbers is strictly increasing, whereas the non-underlined numbers strictly decrease. We next prove that this extra structure is a general phenomenon.

Proposition 5.1.8 (How convergents converge). *The even indexed convergents c_{2n} increase strictly with n , and the odd indexed convergents c_{2n+1} decrease strictly with n . Also, the odd indexed convergents c_{2n+1} are greater than all of the even indexed convergents c_{2m} .*

Proof. The a_n are positive for $n \geq 1$, so the q_n are positive. By Proposition 5.1.5, for $n \geq 2$,

$$c_n - c_{n-2} = (-1)^n \cdot \frac{a_n}{q_n q_{n-2}},$$

which proves the first claim.

Suppose for the sake of contradiction that there exist integers r, m such that $c_{2m+1} < c_{2r}$. Proposition 5.1.5 implies that for $n \geq 1$,

$$c_n - c_{n-1} = (-1)^{n-1} \cdot \frac{1}{q_n q_{n-1}}$$

has sign $(-1)^{n-1}$, so for all $s \geq 0$ we have $c_{2s+1} > c_{2s}$. Thus it is impossible that $r = m$. If $r < m$, then by what we proved in the first paragraph, $c_{2m+1} < c_{2r} < c_{2m}$, a contradiction (with $s = m$). If $r > m$, then $c_{2r+1} < c_{2m+1} < c_{2r}$, which is also a contradiction (with $s = r$). \square

5.1.3 Every Rational Number is Represented

Proposition 5.1.9 (Rational continued fractions). *Every nonzero rational number can be represented by a simple continued fraction.*

Proof. Without loss of generality we may assume that the rational number is a/b , with $b \geq 1$ and $\gcd(a, b) = 1$. Algorithm 1.1.12 gives:

$$\begin{aligned} a &= b \cdot a_0 + r_1, & 0 < r_1 < b \\ b &= r_1 \cdot a_1 + r_2, & 0 < r_2 < r_1 \\ &\dots & \\ r_{n-2} &= r_{n-1} \cdot a_{n-1} + r_n, & 0 < r_n < r_{n-1} \\ r_{n-1} &= r_n \cdot a_n + 0. \end{aligned}$$

Note that $a_i > 0$ for $i > 0$ (also $r_n = 1$ since $\gcd(a, b) = 1$). Rewrite the equations as follows:

$$\begin{aligned} a/b &= a_0 + r_1/b = a_0 + 1/(b/r_1), \\ b/r_1 &= a_1 + r_2/r_1 = a_1 + 1/(r_1/r_2), \\ r_1/r_2 &= a_2 + r_3/r_2 = a_2 + 1/(r_2/r_3), \\ &\dots \\ r_{n-1}/r_n &= a_n. \end{aligned}$$

It follows that

$$\frac{a}{b} = [a_0, a_1, \dots, a_n].$$

\square

The proof of Proposition 5.1.9 leads to an algorithm for computing the continued fraction of a rational number. See Section 7.5 for an implementation.

A nonzero rational number can be represented in exactly two ways; for example, $2 = [1, 1] = [2]$ (see Exercise 5.2).

5.2 Infinite Continued Fractions

This section begins with the continued fraction procedure, which associates to a real number x a sequence a_0, a_1, \dots of integers. After giving several examples, we prove that $x = \lim_{n \rightarrow \infty} [a_0, a_1, \dots, a_n]$ by proving that the odd and even partial convergents become arbitrarily close to each other. We also show that if a_0, a_1, \dots is any infinite sequence of positive integers, then the sequence of $c_n = [a_0, a_1, \dots, a_n]$ converges, and, more generally, if a_n is an arbitrary sequence of positive reals such that $\sum_{n=0}^{\infty} a_n$ diverges then (c_n) converges.

5.2.1 The Continued Fraction Procedure

Let $x \in \mathbf{R}$ and write

$$x = a_0 + t_0$$

with $a_0 \in \mathbf{Z}$ and $0 \leq t_0 < 1$. We call the number a_0 the *floor* of x , and we also sometimes write $a_0 = \lfloor x \rfloor$. If $t_0 \neq 0$, write

$$\frac{1}{t_0} = a_1 + t_1$$

with $a_1 \in \mathbf{N}$ and $0 \leq t_1 < 1$. Thus $t_0 = \frac{1}{a_1 + t_1} = [0, a_1 + t_1]$, which is a (non-simple) continued fraction expansion of t_0 . Continue in this manner so long as $t_n \neq 0$ writing

$$\frac{1}{t_n} = a_{n+1} + t_{n+1}$$

with $a_{n+1} \in \mathbf{N}$ and $0 \leq t_{n+1} < 1$. We call this procedure, which associates to a real number x the sequence of integers a_0, a_1, a_2, \dots , the *continued fraction process*. We implement it in on a computer in Section 7.5.

Example 5.2.1. Let $x = \frac{8}{3}$. Then $x = 2 + \frac{2}{3}$, so $a_0 = 2$ and $t_0 = \frac{2}{3}$. Then $\frac{1}{t_0} = \frac{3}{2} = 1 + \frac{1}{2}$, so $a_1 = 1$ and $t_1 = \frac{1}{2}$. Then $\frac{1}{t_1} = 2$, so $a_2 = 2$, $t_2 = 0$, and the sequence terminates. Notice that

$$\frac{8}{3} = [2, 1, 2],$$

so the continued fraction procedure produces the continued fraction of $\frac{8}{3}$.

Example 5.2.2. Let $x = \frac{1+\sqrt{5}}{2}$. Then

$$x = 1 + \frac{-1 + \sqrt{5}}{2},$$

so $a_0 = 1$ and $t_0 = \frac{-1+\sqrt{5}}{2}$. We have

$$\frac{1}{t_0} = \frac{2}{-1 + \sqrt{5}} = \frac{-2 - 2\sqrt{5}}{-4} = \frac{1 + \sqrt{5}}{2}$$

so again $a_1 = 1$ and $t_1 = \frac{-1+\sqrt{5}}{2}$. Likewise, $a_n = 1$ for all n . As we will see below, the following exciting equality makes sense.

$$\frac{1 + \sqrt{5}}{2} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \dots}}}}}$$

Example 5.2.3. Suppose $x = e = 2.71828182\dots$. Using the continued fraction procedure, we find that

$$a_0, a_1, a_2, \dots = 2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, \dots$$

For example, $a_0 = 2$ is the floor of e . Subtracting 2 and inverting, we obtain $1/0.718\dots = 1.3922\dots$, so $a_1 = 1$. Subtracting 1 and inverting yields $1/0.3922\dots = 2.5496\dots$, so $a_2 = 2$. We will prove in Section 5.3 that the continued fraction of e obeys a simple pattern.

The 5th partial convergent of the continued fraction of e is

$$[a_0, a_1, a_2, a_3, a_4, a_5] = \frac{87}{32} = 2.71875,$$

which is a good rational approximation to e , in the sense that

$$\left| \frac{87}{32} - e \right| = 0.000468\dots$$

Note that $0.000468\dots < 1/32^2 = 0.000976\dots$, which illustrates the bound in Corollary 5.2.10 below.

Let's do the same thing with $\pi = 3.14159265358979\dots$: Applying the continued fraction procedure, we find that the continued fraction of π is

$$a_0, a_1, a_2, \dots = 3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, \dots$$

The first few partial convergents are

$$3, \frac{22}{7}, \frac{333}{106}, \frac{355}{113}, \frac{103993}{33102}, \dots$$

These are good rational approximations to π ; for example,

$$\frac{103993}{33102} = 3.14159265301\dots$$

Notice that the continued fraction of e exhibits a nice pattern (see Section 5.3 for a proof), whereas the continued fraction of π exhibits no pattern that is obvious to the author. The continued fraction of π has been extensively studied, and over 20 million terms have been computed. The data suggests that every integer appears infinitely often as a partial convergent. For much more about the continued fraction of π or of any other sequence in this book, type the first few terms of the sequence into [Slo].

5.2.2 Convergence of Infinite Continued Fractions

Lemma 5.2.4. *For every n such that a_n is defined, we have*

$$x = [a_0, a_1, \dots, a_n + t_n],$$

and if $t_n \neq 0$ then $x = [a_0, a_1, \dots, a_n, \frac{1}{t_n}]$.

Proof. We use induction. The statements are both true when $n = 0$. If the second statement is true for $n - 1$, then

$$\begin{aligned} x &= \left[a_0, a_1, \dots, a_{n-1}, \frac{1}{t_{n-1}} \right] \\ &= [a_0, a_1, \dots, a_{n-1}, a_n + t_n] \\ &= \left[a_0, a_1, \dots, a_{n-1}, a_n, \frac{1}{t_n} \right]. \end{aligned}$$

Similarly, the first statement is true for n if it is true for $n - 1$. □

Theorem 5.2.5 (Continued Fraction Limit). *Let a_0, a_1, \dots be a sequence of integers such that $a_n > 0$ for all $n \geq 1$, and for each $n \geq 0$, set $c_n = [a_0, a_1, \dots, a_n]$. Then $\lim_{n \rightarrow \infty} c_n$ exists.*

Proof. For any $m \geq n$, the number c_n is a partial convergent of $[a_0, \dots, a_m]$. By Proposition 5.1.8 the even convergents c_{2n} form a strictly *increasing* sequence and the odd convergents c_{2n+1} form a strictly *decreasing* sequence. Moreover, the even convergents are all $\leq c_1$ and the odd convergents are all $\geq c_0$. Hence $\alpha_0 = \lim_{n \rightarrow \infty} c_{2n}$ and $\alpha_1 = \lim_{n \rightarrow \infty} c_{2n+1}$ both exist and $\alpha_0 \leq \alpha_1$. Finally, by Proposition 5.1.5

$$|c_{2n} - c_{2n-1}| = \frac{1}{q_{2n} \cdot q_{2n-1}} \leq \frac{1}{2n(2n-1)} \rightarrow 0,$$

so $\alpha_0 = \alpha_1$. □

We define

$$[a_0, a_1, \dots] = \lim_{n \rightarrow \infty} c_n.$$

Example 5.2.6. We illustrate the theorem with $x = \pi$. As in the proof of Theorem 5.2.5, let c_n be the n th partial convergent to π . The c_n with n odd converge down to π

$$c_1 = 3.1428571\dots, c_3 = 3.1415929\dots, c_5 = 3.1415926\dots$$

whereas the c_n with n even converge up to π

$$c_2 = 3.1415094\dots, c_4 = 3.1415926\dots, c_6 = 3.1415926\dots$$

Theorem 5.2.7. *Let a_0, a_1, a_2, \dots be a sequence of real numbers such that $a_n > 0$ for all $n \geq 1$, and for each $n \geq 0$, set $c_n = [a_0, a_1, \dots, a_n]$. Then $\lim_{n \rightarrow \infty} c_n$ exists if and only if the sum $\sum_{n=0}^{\infty} a_n$ diverges.*

Proof. We only prove that if $\sum a_n$ diverges then $\lim_{n \rightarrow \infty} c_n$ exists. A proof of the converse can be found in [Wal48, Ch. 2, Thm. 6.1].

Let q_n be the sequence of “denominators” of the partial convergents, as defined in Section 5.1.1, so $q_{-2} = 1$, $q_{-1} = 0$, and for $n \geq 0$,

$$q_n = a_n q_{n-1} + q_{n-2}.$$

As we saw in the proof of Theorem 5.2.5, the limit $\lim_{n \rightarrow \infty} c_n$ exists provided that the sequence $\{q_n q_{n-1}\}$ diverges to positive infinity.

For n even,

$$\begin{aligned} q_n &= a_n q_{n-1} + q_{n-2} \\ &= a_n q_{n-1} + a_{n-2} q_{n-3} + q_{n-4} \\ &= a_n q_{n-1} + a_{n-2} q_{n-3} + a_{n-4} q_{n-5} + q_{n-6} \\ &= a_n q_{n-1} + a_{n-2} q_{n-3} + \dots + a_2 q_1 + q_0 \end{aligned}$$

and for n odd,

$$q_n = a_n q_{n-1} + a_{n-2} q_{n-3} + \dots + a_1 q_0 + q_{-1}.$$

Since $a_n > 0$ for $n > 0$, the sequence $\{q_n\}$ is increasing, so $q_i \geq 1$ for all $i \geq 0$. Applying this fact to the above expressions for q_n , we see that for n even

$$q_n \geq a_n + a_{n-2} + \dots + a_2,$$

and for n odd

$$q_n \geq a_n + a_{n-2} + \dots + a_1.$$

If $\sum a_n$ diverges, then at least one of $\sum a_{2n}$ or $\sum a_{2n+1}$ must diverge. The above inequalities then imply that at least one of the sequences $\{q_{2n}\}$ or $\{q_{2n+1}\}$ diverge to infinity. Since $\{q_n\}$ is an increasing sequence, it follows that $\{q_n q_{n-1}\}$ diverges to infinity. \square

Example 5.2.8. Let $a_n = \frac{1}{n \log(n)}$ for $n \geq 2$ and $a_0 = a_1 = 0$. By the integral test, $\sum a_n$ diverges, so by Theorem 5.2.7 the continued fraction $[a_0, a_1, a_2, \dots]$ converges. This convergence is very slow, since, e.g.

$$[a_0, a_1, \dots, a_{9999}] = 0.5750039671012225425930 \dots$$

yet

$$[a_0, a_1, \dots, a_{10000}] = 0.7169153932917378550424 \dots$$

Theorem 5.2.9. *Let $x \in \mathbf{R}$ be a real number. Then x is the value of the (possibly infinite) simple continued fraction $[a_0, a_1, a_2, \dots]$ produced by the continued fraction procedure.*

Proof. If the sequence is finite then some $t_n = 0$ and the result follows by Lemma 5.2.4. Suppose the sequence is infinite. By Lemma 5.2.4,

$$x = [a_0, a_1, \dots, a_n, \frac{1}{t_n}].$$

By Proposition 5.1.4 (which we apply in a case when the partial quotients of the continued fraction are not integers!), we have

$$x = \frac{\frac{1}{t_n} \cdot p_n + p_{n-1}}{\frac{1}{t_n} \cdot q_n + q_{n-1}}.$$

Thus if $c_n = [a_0, a_1, \dots, a_n]$, then

$$\begin{aligned} x - c_n &= x - \frac{p_n}{q_n} \\ &= \frac{\frac{1}{t_n} p_n q_n + p_{n-1} q_n - \frac{1}{t_n} p_n q_n - p_n q_{n-1}}{q_n \left(\frac{1}{t_n} q_n + q_{n-1} \right)} \\ &= \frac{p_{n-1} q_n - p_n q_{n-1}}{q_n \left(\frac{1}{t_n} q_n + q_{n-1} \right)} \\ &= \frac{(-1)^n}{q_n \left(\frac{1}{t_n} q_n + q_{n-1} \right)}. \end{aligned}$$

Thus

$$\begin{aligned} |x - c_n| &= \frac{1}{q_n \left(\frac{1}{t_n} q_n + q_{n-1} \right)} \\ &< \frac{1}{q_n (a_{n+1} q_n + q_{n-1})} \\ &= \frac{1}{q_n \cdot q_{n+1}} \leq \frac{1}{n(n+1)} \rightarrow 0. \end{aligned}$$

In the inequality we use that a_{n+1} is the integer part of $\frac{1}{t_n}$, and is hence $\leq \frac{1}{t_n} < 1$, since $t_n < 1$. \square

This corollary follows from the proof of the above theorem.

Corollary 5.2.10 (Convergence of continued fraction). *Let a_0, a_1, \dots define a simple continued fraction, and let $x = [a_0, a_1, \dots] \in \mathbf{R}$ be its value. Then for all m ,*

$$\left| x - \frac{p_m}{q_m} \right| < \frac{1}{q_m \cdot q_{m+1}}.$$

Proposition 5.2.11. *If x is a rational number then the sequence a_0, a_1, \dots produced by the continued fraction procedure terminates.*

Proof. Let $[b_0, b_1, \dots, b_m]$ be the continued fraction representation of x that we obtain using Algorithm 1.1.12, so the b_i are the partial quotients at each step. If $m = 0$, then x is an integer, so we may assume $m > 0$. Then

$$x = b_0 + 1/[b_1, \dots, b_m].$$

If $[b_1, \dots, b_m] = 1$ then $m = 1$ and $b_1 = 1$, which will not happen using Algorithm 1.1.12, since it would give $[b_0 + 1]$ for the continued fraction of the integer $b_0 + 1$. Thus $[b_1, \dots, b_m] > 1$, so in the continued fraction algorithm we choose $a_0 = b_0$ and $t_0 = 1/[b_1, \dots, b_m]$. Repeating this argument enough times proves the claim. \square

5.3 The Continued Fraction of e

The continued fraction expansion of e begins $[2, 1, 2, 1, 1, 4, 1, 1, 6, \dots]$. The obvious pattern in fact does continue, as Euler proved in 1737 (see [Eul85]), and we will prove in this section. As an application, Euler gave a proof that e is irrational by noting that its continued fraction is infinite.

The proof we give below draws heavily on the proof in [Coh], which describes a slight variant of a proof of Hermite (see [Old70]). The continued fraction representation of e is also treated in the German book [Per57], but the proof requires substantial background from elsewhere in that text.

5.3.1 Preliminaries

First, we write the continued fraction of e in a slightly different form. Instead of $[2, 1, 2, 1, 1, 4, \dots]$, we can start the sequence of coefficients

$$[1, 0, 1, 1, 2, 1, 1, 4, \dots]$$

to make the pattern the same throughout. (Everywhere else in this chapter we assume that the partial quotients a_n for $n \geq 1$ are positive, but

temporarily relax that condition here and allow $a_1 = 0$.) The numerators and denominators of the convergents given by this new sequence satisfy a simple recurrence. Using r_i as a stand-in for p_i or q_i , we have

$$\begin{aligned} r_{3n} &= r_{3n-1} + r_{3n-2} \\ r_{3n-1} &= r_{3n-2} + r_{3n-3} \\ r_{3n-2} &= 2(n-1)r_{3n-3} + r_{3n-4}. \end{aligned}$$

Our first goal is to collapse these three recurrences into one recurrence that only makes mention of r_{3n} , r_{3n-3} , and r_{3n-6} . We have

$$\begin{aligned} r_{3n} &= r_{3n-1} + r_{3n-2} \\ &= (r_{3n-2} + r_{3n-3}) + (2(n-1)r_{3n-3} + r_{3n-4}) \\ &= (4n-3)r_{3n-3} + 2r_{3n-4}. \end{aligned}$$

This same method of simplification also shows us that

$$r_{3n-3} = 2r_{3n-7} + (4n-7)r_{3n-6}.$$

To get rid of $2r_{3n-4}$ in the first equation, we make the substitutions

$$\begin{aligned} 2r_{3n-4} &= 2(r_{3n-5} + r_{3n-6}) \\ &= 2((2(n-2)r_{3n-6} + r_{3n-7}) + r_{3n-6}) \\ &= (4n-6)r_{3n-6} + 2r_{3n-7}. \end{aligned}$$

Substituting for $2r_{3n-4}$ and then $2r_{3n-7}$, we finally have the needed collapsed recurrence,

$$r_{3n} = 2(2n-1)r_{3n-3} + r_{3n-6}.$$

5.3.2 Two Integral Sequences

We define the sequences $x_n = p_{3n}$, $y_n = q_{3n}$. Since the $3n$ -convergents will converge to the same real number that the n -convergents do, x_n/y_n also converges to the limit of the continued fraction. Each sequence $\{x_n\}$, $\{y_n\}$ will obey the recurrence relation derived in the previous section (where z_n is a stand-in for x_n or y_n):

$$z_n = 2(2n-1)z_{n-1} + z_{n-2}, \text{ for all } n \geq 2. \quad (5.3.1)$$

The two sequences can be found in Table 5.1. (The initial conditions $x_0 = 1$, $x_1 = 3$, $y_0 = y_1 = 1$ are taken straight from the first few convergents of the original continued fraction.) Notice that since we are skipping several convergents at each step, the ratio x_n/y_n converges to e very quickly.

TABLE 5.1. Convergents

n	0	1	2	3	4	...
x_n	1	3	19	193	2721	...
y_n	1	1	7	71	1001	...
x_n/y_n	1	3	2.714...	2.71830...	2.7182817...	...

5.3.3 A Related Sequence of Integrals

Now, we define a sequence of real numbers T_0, T_1, T_2, \dots by the following integrals:

$$T_n = \int_0^1 \frac{t^n(t-1)^n}{n!} e^t dt.$$

Below, we compute the first two terms of this sequence explicitly. (When we compute T_1 , we are doing the integration by parts $u = t(t-1)$, $dv = e^t dt$. Since the integral runs from 0 to 1, the boundary condition is 0 when evaluated at each of the endpoints. This vanishing will be helpful when we do the integral in the general case.)

$$\begin{aligned} T_0 &= \int_0^1 e^t dt = e - 1, \\ T_1 &= \int_0^1 t(t-1)e^t dt \\ &= - \int_0^1 ((t-1) + t)e^t dt \\ &= -(t-1)e^t \Big|_0^1 - te^t \Big|_0^1 + 2 \int_0^1 e^t dt \\ &= 1 - e + 2(e - 1) = e - 3. \end{aligned}$$

The reason that we defined this series now becomes apparent: $T_0 = y_0 e - x_0$ and that $T_1 = y_1 e - x_1$. In general, it will be true that $T_n = y_n e - x_n$. We will now prove this fact.

It is clear that if the T_n were to satisfy the same recurrence that the x_i and y_i do, in equation (5.3.1), then the above statement holds by induction. (The initial conditions are correct, as needed.) So we simplify T_n by

integrating by parts twice in succession:

$$\begin{aligned}
T_n &= \int_0^1 \frac{t^n(t-1)^n}{n!} e^t dt \\
&= - \int_0^1 \frac{t^{n-1}(t-1)^n + t^n(t-1)^{n-1}}{(n-1)!} e^t dt \\
&= \int_0^1 \left(\frac{t^{n-2}(t-1)^n}{(n-2)!} + n \frac{t^{n-1}(t-1)^{n-1}}{(n-1)!} \right. \\
&\quad \left. + n \frac{t^{n-1}(t-1)^{n-1}}{(n-1)!} + \frac{t^n(t-1)^{n-2}}{(n-2)!} \right) e^t dt \\
&= 2nT_{n-1} + \int_0^1 \frac{t^{n-2}(t-1)^{n-2}}{n-2!} (2t^2 - 2t + 1) e^t dt \\
&= 2nT_{n-1} + 2 \int_0^1 \frac{t^{n-1}(t-1)^{n-1}}{n-2!} e^t dt + \int_0^1 \frac{t^{n-2}(t-1)^{n-2}}{n-2!} e^t dt \\
&= 2nT_{n-1} + 2(n-1)T_{n-1} + T_{n-2} \\
&= 2(2n-1)T_{n-1} + T_{n-2},
\end{aligned}$$

which is the desired recurrence.

Therefore $T_n = y_n e - x_n$. To conclude the proof, we consider the limit as n approaches infinity:

$$\lim_{n \rightarrow \infty} \int_0^1 \frac{t^n(t-1)^n}{n!} e^t dt = 0,$$

by inspection, and therefore

$$\lim_{n \rightarrow \infty} \frac{x_n}{y_n} = \lim_{n \rightarrow \infty} \left(e - \frac{T_n}{y_n} \right) = e.$$

Therefore, the ratio x_n/y_n approaches e , and the continued fraction expansion $[2, 1, 2, 1, 1, 4, 1, 1, \dots]$ does in fact converge to e .

5.3.4 Extensions of the Argument

The method of proof of this section generalizes to show that the continued fraction expansion of $e^{1/n}$ is

$$[1, (n-1), 1, 1, (3n-1), 1, 1, (5n-1), 1, 1, (7n-1), \dots]$$

for all $n \in \mathbf{N}$ (see Exercise 5.6).

5.4 Quadratic Irrationals

The main result of this section is that the continued fraction expansion of a number is eventually repeating if and only if the number is a quadratic

irrational. This can be viewed as an analogue for continued fractions of the familiar fact that the decimal expansion of x is eventually repeating if and only if x is rational. The proof that continued fractions of quadratic irrationals eventually repeats is surprisingly difficult and involves an interesting finiteness argument. Section 5.4.2 emphasizes our striking ignorance about continued fractions of real roots of irreducible polynomials over \mathbf{Q} of degree bigger than 2.

Definition 5.4.1 (Quadratic Irrational). A real number $\alpha \in \mathbf{R}$ is a *quadratic irrational* if it is irrational and satisfies a quadratic polynomial with coefficients in \mathbf{Q} .

Thus, e.g., $(1 + \sqrt{5})/2$ is a quadratic irrational. Recall that

$$\frac{1 + \sqrt{5}}{2} = [1, 1, 1, \dots].$$

The continued fraction of $\sqrt{2}$ is $[1, 2, 2, 2, 2, \dots]$, and the continued fraction of $\sqrt{389}$ is

$$[19, 1, 2, 1, 1, 1, 1, 2, 1, 38, 1, 2, 1, 1, 1, 1, 2, 1, 38, \dots].$$

Does the $[1, 2, 1, 1, 1, 1, 2, 1, 38]$ pattern repeat over and over again?

5.4.1 Periodic Continued Fractions

Definition 5.4.2 (Periodic Continued Fraction). A *periodic continued fraction* is a continued fraction $[a_0, a_1, \dots, a_n, \dots]$ such that

$$a_n = a_{n+h}$$

for some fixed positive integer h and all sufficiently large n . We call the minimal such h the *period of the continued fraction*.

Example 5.4.3. Consider the periodic continued fraction $[1, 2, 1, 2, \dots] = \overline{[1, 2]}$. What does it converge to? We have

$$\overline{[1, 2]} = 1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \dots}}}}$$

so if $\alpha = \overline{[1, 2]}$ then

$$\alpha = 1 + \frac{1}{2 + \frac{1}{\alpha}} = 1 + \frac{1}{\frac{2\alpha + 1}{\alpha}} = 1 + \frac{\alpha}{2\alpha + 1} = \frac{3\alpha + 1}{2\alpha + 1}.$$

Thus $2\alpha^2 - 2\alpha - 1 = 0$, so

$$\alpha = \frac{1 + \sqrt{3}}{2}.$$

Theorem 5.4.4 (Periodic Characterization). *An infinite simple continued fraction is periodic if and only if it represents a quadratic irrational.*

Proof. (\implies) First suppose that

$$[a_0, a_1, \dots, a_n, \overline{a_{n+1}, \dots, a_{n+h}}]$$

is a periodic continued fraction. Set $\alpha = [a_{n+1}, a_{n+2}, \dots]$. Then

$$\alpha = [a_{n+1}, \dots, a_{n+h}, \alpha],$$

so by Proposition 5.1.4

$$\alpha = \frac{\alpha p_{n+h} + p_{n+h-1}}{\alpha q_{n+h} + q_{n+h-1}}.$$

Here we use that α is the last partial quotient. Thus, α satisfies a quadratic equation with coefficients in \mathbf{Q} . Computing as in Example 5.4.3 and rationalizing the denominators, and using that the a_i are all integers, shows that

$$\begin{aligned} [a_0, a_1, \dots] &= [a_0, a_1, \dots, a_n, \alpha] \\ &= a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \dots + \frac{1}{\alpha}}} \end{aligned}$$

is of the form $c + d\alpha$, with $c, d \in \mathbf{Q}$, so $[a_0, a_1, \dots]$ also satisfies a quadratic polynomial over \mathbf{Q} .

The continued fraction procedure applied to the value of an infinite simple continued fraction yields that continued fraction back, so by Proposition 5.2.11, $\alpha \notin \mathbf{Q}$ because it is the value of an infinite continued fraction.

(\impliedby) Suppose $\alpha \in \mathbf{R}$ is an irrational number that satisfies a quadratic equation

$$a\alpha^2 + b\alpha + c = 0 \tag{5.4.1}$$

with $a, b, c \in \mathbf{Z}$ and $a \neq 0$. Let $[a_0, a_1, \dots]$ be the continued fraction expansion of α . For each n , let

$$r_n = [a_n, a_{n+1}, \dots],$$

so

$$\alpha = [a_0, a_1, \dots, a_{n-1}, r_n].$$

We will prove periodicity by showing that the set of r_n 's is finite. If we have shown finiteness, then there exists $n, h > 0$ such that $r_n = r_{n+h}$, so

$$\begin{aligned} [a_0, \dots, a_{n-1}, r_n] &= [a_0, \dots, a_{n-1}, a_n, \dots, a_{n+h-1}, r_{n+h}] \\ &= [a_0, \dots, a_{n-1}, a_n, \dots, a_{n+h-1}, r_n] \\ &= [a_0, \dots, a_{n-1}, a_n, \dots, a_{n+h-1}, a_n, \dots, a_{n+h-1}, r_{n+h}] \\ &= [a_0, \dots, a_{n-1}, \overline{a_n, \dots, a_{n+h-1}}]. \end{aligned}$$

It remains to show there are only finitely many distinct r_n . We have

$$\alpha = \frac{p_n}{q_n} = \frac{r_n p_{n-1} + p_{n-2}}{r_n q_{n-1} + q_{n-2}}.$$

Substituting this expression for α into the quadratic equation (5.4.1), we see that

$$A_n r_n^2 + B_n r_n + C_n = 0,$$

where

$$\begin{aligned} A_n &= ap_{n-1}^2 + bp_{n-1}q_{n-1} + cq_{n-1}^2, \\ B_n &= 2ap_{n-1}p_{n-2} + b(p_{n-1}q_{n-2} + p_{n-2}q_{n-1}) + 2cq_{n-1}q_{n-2}, \text{ and} \\ C_n &= ap_{n-2}^2 + bp_{n-2}q_{n-2} + cp_{n-2}^2. \end{aligned}$$

Note that $A_n, B_n, C_n \in \mathbf{Z}$, that $C_n = A_{n-1}$, and that

$$B^2 - 4A_n C_n = (b^2 - 4ac)(p_{n-1}q_{n-2} - q_{n-1}p_{n-2})^2 = b^2 - 4ac.$$

Recall from the proof of Theorem 5.2.9 that

$$\left| \alpha - \frac{p_{n-1}}{q_{n-1}} \right| < \frac{1}{q_n q_{n-1}}.$$

Thus

$$|\alpha q_{n-1} - p_{n-1}| < \frac{1}{q_n} < \frac{1}{q_{n-1}},$$

so

$$p_{n-1} = \alpha q_{n-1} + \frac{\delta}{q_{n-1}} \quad \text{with } |\delta| < 1.$$

Hence

$$\begin{aligned} A_n &= a \left(\alpha q_{n-1} + \frac{\delta}{q_{n-1}} \right)^2 + b \left(\alpha q_{n-1} + \frac{\delta}{q_{n-1}} \right) q_{n-1} + cq_{n-1}^2 \\ &= (a\alpha^2 + b\alpha + c)q_{n-1}^2 + 2a\alpha\delta + a\frac{\delta^2}{q_{n-1}} + b\delta \\ &= 2a\alpha\delta + a\frac{\delta^2}{q_{n-1}} + b\delta. \end{aligned}$$

Thus

$$|A_n| = \left| 2a\alpha\delta + a\frac{\delta^2}{q_{n-1}^2} + b\delta \right| < 2|a\alpha| + |a| + |b|.$$

Thus there are only finitely many possibilities for the integer A_n . Also,

$$|C_n| = |A_{n-1}| \quad \text{and} \quad |B_n| = \sqrt{b^2 - 4(ac - A_n C_n)},$$

so there are only finitely many triples (A_n, B_n, C_n) , and hence only finitely many possibilities for r_n as n varies, which completes the proof. (The proof above closely follows [HW79, Thm. 177, pg.144–145].) \square

5.4.2 Continued Fractions of Algebraic Numbers of Higher Degree

Definition 5.4.5 (Algebraic Number). An *algebraic number* is a root of a polynomial $f \in \mathbf{Q}[x]$.

Open Problem 5.4.6. Give a simple description of the complete continued fractions expansion of the algebraic number $\sqrt[3]{2}$. It begins

$$[1, 3, 1, 5, 1, 1, 4, 1, 1, 8, 1, 14, 1, 10, 2, 1, 4, 12, 2, 3, 2, 1, 3, 4, 1, 1, 2, 14, 3, 12, 1, 15, 3, 1, 4, 534, 1, 1, 5, 1, 1, \dots]$$

The author does not see a pattern, and the 534 reduces his confidence that he will. Lang and Trotter (see [LT72]) analyzed many terms of the continued fraction of $\sqrt[3]{2}$ statistically, and their work suggests that $\sqrt[3]{2}$ has an “unusual” continued fraction; later work in [LT74] suggests that maybe it does not.

Khinchine (see [Khi63, pg. 59])

No properties of the representing continued fractions, analogous to those which have just been proved, are known for algebraic numbers of higher degree [as of 1963]. [...] It is of interest to point out that up till the present time *no continued fraction development of an algebraic number of higher degree than the second is known* [emphasis added]. It is not even known if such a development has bounded elements. Generally speaking the problems associated with the continued fraction expansion of algebraic numbers of degree higher than the second are extremely difficult and virtually unstudied.

Richard Guy (see [Guy94, pg. 260])

Is there an algebraic number of degree greater than two whose simple continued fraction has unbounded partial quotients? Does every such number have unbounded partial quotients?

Baum and Sweet [BS76] answered the analogue of Richard Guy's question but with algebraic numbers replaced by elements of a field K other than \mathbf{Q} . (The field K is $\mathbf{F}_2((1/x))$, the field of Laurent series in the variable $1/x$ over the finite field with two elements. An element of K is a polynomial in x plus a formal power series in $1/x$.) They found an α of degree three over K whose continued fraction has all terms of bounded degree, and other elements of various degrees greater than 2 over K whose continued fractions have terms of unbounded degree.

5.5 Recognizing Rational Numbers

Suppose that somehow you can compute approximations to some rational number, and want to figure what the rational number probably is. Computing the approximation to high enough precision to find a period in the decimal expansion is not a good approach, because the period can be huge (see below). A much better approach is to compute the simple continued fraction of the approximation, and truncate it before a large partial quotient a_n , then compute the value of the truncated continued fraction. This results in a rational number that has relatively small numerator and denominator, and is close to the approximation of the rational number, since the tail end of the continued fraction is at most $1/a_n$.

We begin with a contrived example, which illustrates how to recognize a rational number. Let

$$x = 9495/3847 = 2.46815700545879906420587470756433584611385\dots$$

The continued fraction of the truncation 2.468157005458799064 is

$$[2, 2, 7, 2, 1, 5, 1, 1, 1, 1, 1, 1, 328210621945, 2, 1, 1, 1, \dots]$$

We have

$$[2, 2, 7, 2, 1, 5, 1, 1, 1, 1, 1] = \frac{9495}{3847}.$$

Notice that no repetition is evident in the digits of x given above, though we know that the decimal expansion of x must be eventually periodic, since all decimal expansions of rational numbers are eventually periodic. In fact, the length of the period of the decimal expansion of $1/3847$ is 3846, which is the order of 10 modulo 3847 (see Exercise 5.7).

For a slightly less contrived application of this idea, suppose $f(x) \in \mathbf{Z}[x]$ is a polynomial with integer coefficients, and we know for some reason that one root of f is a rational number. Then we can find that rational number by using Newton's method to approximate each root, and continued fractions to decide whether each root is a rational number (we can substitute the value of the continued fraction approximation into f to see if it

is actually a root). One could also use the well-known rational root theorem, which asserts that any rational root n/d of f , with $n, d \in \mathbf{Z}$ coprime, has the property that n divides the constant term of f and d the leading coefficient of f . However, using that theorem to find n/d would require factoring the constant and leading terms of f , which could be completely impractical if they have a few hundred digits (see Section 1.1.3). In contrast, Newton's method and continued fractions should quickly find n/d , assuming the degree of f isn't too large.

For example, suppose $f = 3847x^2 - 14808904x + 36527265$. To apply Newton's method, let x_0 be a guess for a root of f . Then iterate using the recurrence

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Choosing $x_0 = 0$, approximations of first two iterates are

$$x_1 = 2.466574501394566404103909378,$$

and

$$x_2 = 2.468157004807401923043166846.$$

The continued fraction of the approximations x_1 and x_2 are

$$[2, 2, 6, 1, 47, 2, 1, 4, 3, 1, 5, 8, 2, 3]$$

and

$$[2, 2, 7, 2, 1, 5, 1, 1, 1, 1, 1, 1, 103, 8, 1, 2, 3, \dots].$$

Truncating the continued fraction of x_2 before 103 gives

$$[2, 2, 7, 2, 1, 5, 1, 1, 1, 1, 1],$$

which evaluates to $9495/3847$, which is a rational root of f .

Another computational application of continued fractions, which we can only hint at, is that there are functions in certain parts of advanced number theory (that are beyond the scope of this book) that take rational values at certain points, and which can only be computed efficiently via approximations; using continued fractions as illustrated above to evaluate such functions is crucial.

5.6 Sums of Two Squares

In this section we apply continued fractions to prove the following theorem.

Theorem 5.6.1. *A positive integer n is a sum of two squares if and only if all prime factors of $p \mid n$ such that $p \equiv 3 \pmod{4}$ have even exponent in the prime factorization of n .*

We first consider some examples. Notice that $5 = 1^2 + 2^2$ is a sum of two squares, but 7 is not a sum of two squares. Since 2001 is divisible by 3 (because $2 + 1$), but not by 9 (since $2 + 1$ is not), Theorem 5.6.1 implies that 2001 is not a sum of two squares. The theorem also implies that $2 \cdot 3^4 \cdot 5 \cdot 7^2 \cdot 13$ is a sum of two squares.

Definition 5.6.2 (Primitive). A representation $n = x^2 + y^2$ is *primitive* if x and y are coprime.

Lemma 5.6.3. *If n is divisible by a prime $p \equiv 3 \pmod{4}$, then n has no primitive representations.*

Proof. Suppose n has a primitive representation, $n = x^2 + y^2$, and let p be any prime factor of n . Then

$$p \mid x^2 + y^2 \quad \text{and} \quad \gcd(x, y) = 1,$$

so $p \nmid x$ and $p \nmid y$. Since $\mathbf{Z}/p\mathbf{Z}$ is a field we may divide by y^2 in the equation $x^2 + y^2 \equiv 0 \pmod{p}$ to see that $(x/y)^2 \equiv -1 \pmod{p}$. Thus the quadratic residue symbol $\left(\frac{-1}{p}\right)$ equals $+1$. However, by Proposition 4.2.1,

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$$

so $\left(\frac{-1}{p}\right) = 1$ if and only if $(p-1)/2$ is even, which is to say $p \equiv 1 \pmod{4}$. □

Proof of Theorem 5.6.1 (\implies). Suppose that $p \equiv 3 \pmod{4}$ is a prime, that $p^r \mid n$ but $p^{r+1} \nmid n$ with r odd, and that $n = x^2 + y^2$. Letting $d = \gcd(x, y)$, we have

$$x = dx', \quad y = dy', \quad \text{and} \quad n = d^2 n'$$

with $\gcd(x', y') = 1$ and

$$(x')^2 + (y')^2 = n'.$$

Because r is odd, $p \mid n'$, so Lemma 5.6.3 implies that $\gcd(x', y') > 1$, a contradiction. □

To prepare for our proof of (\impliedby), we reduce the problem to the case when n is prime. Write $n = n_1^2 n_2$ where n_2 has no prime factors $p \equiv 3 \pmod{4}$. It suffices to show that n_2 is a sum of two squares, since

$$(x_1^2 + y_1^2)(x_2^2 + y_2^2) = (x_1 x_2 - y_1 y_2)^2 + (x_1 y_2 + x_2 y_1)^2, \quad (5.6.1)$$

so a product of two numbers that are sums of two squares is also a sum of two squares. Since $2 = 1^2 + 1^2$ is a sum of two squares, it suffices to show that any prime $p \equiv 1 \pmod{4}$ is a sum of two squares.

Lemma 5.6.4. *If $x \in \mathbf{R}$ and $n \in \mathbf{N}$, then there is a fraction $\frac{a}{b}$ in lowest terms such that $0 < b \leq n$ and*

$$\left| x - \frac{a}{b} \right| \leq \frac{1}{b(n+1)}.$$

Proof. Consider the continued fraction $[a_0, a_1, \dots]$ of x . By Corollary 5.2.10, for each m

$$\left| x - \frac{p_m}{q_m} \right| < \frac{1}{q_m \cdot q_{m+1}}.$$

Since $q_{m+1} \geq q_m + 1$ and $q_0 = 1$, either there exists an m such that $q_m \leq n < q_{m+1}$, or the continued fraction expansion of x is finite and n is larger than the denominator of the rational number x , in which case we take $\frac{a}{b} = x$ and are done. In the first case,

$$\left| x - \frac{p_m}{q_m} \right| < \frac{1}{q_m \cdot q_{m+1}} \leq \frac{1}{q_m \cdot (n+1)},$$

so $\frac{a}{b} = \frac{p_m}{q_m}$ satisfies the conclusion of the lemma. \square

Proof of Theorem 5.6.1 (\Leftarrow). As discussed above, it suffices to prove that any prime $p \equiv 1 \pmod{4}$ is a sum of two squares. Since $p \equiv 1 \pmod{4}$,

$$(-1)^{(p-1)/2} = 1,$$

so Proposition 4.2.1 implies that -1 is a square modulo p ; i.e., there exists $r \in \mathbf{Z}$ such that $r^2 \equiv -1 \pmod{p}$. Lemma 5.6.4, with $n = \lfloor \sqrt{p} \rfloor$ and $x = -\frac{r}{p}$, implies that there are integers a, b such that $0 < b < \sqrt{p}$ and

$$\left| -\frac{r}{p} - \frac{a}{b} \right| \leq \frac{1}{b(n+1)} < \frac{1}{b\sqrt{p}}.$$

Letting $c = rb + pa$, we have that

$$|c| < \frac{pb}{b\sqrt{p}} = \frac{p}{\sqrt{p}} = \sqrt{p}$$

so

$$0 < b^2 + c^2 < 2p.$$

But $c \equiv rb \pmod{p}$, so

$$b^2 + c^2 \equiv b^2 + r^2b^2 \equiv b^2(1 + r^2) \equiv 0 \pmod{p}.$$

Thus $b^2 + c^2 = p$. \square

Remark 5.6.5. Our proof of Theorem 5.6.1 leads to an efficient algorithm to compute a representation of any $p \equiv 1 \pmod{4}$ as a sum of two squares. See Listing 7.5.5 for an implementation.

5.7 Exercises

- 5.1 If $c_n = p_n/q_n$ is the n th convergent of $[a_0, a_1, \dots, a_n]$ and $a_0 > 0$, show that

$$[a_n, a_{n-1}, \dots, a_1, a_0] = \frac{p_n}{p_{n-1}}$$

and

$$[a_n, a_{n-1}, \dots, a_2, a_1] = \frac{q_n}{q_{n-1}}.$$

(Hint: In the first case, notice that $\frac{p_n}{p_{n-1}} = a_n + \frac{p_{n-2}}{p_{n-1}} = a_n + \frac{1}{\frac{p_{n-1}}{p_{n-2}}}$.)

- 5.2 Show that every nonzero rational number can be represented in exactly two ways by a finite simple continued fraction. (For example, 2 can be represented by $[1, 1]$ and $[2]$, and $1/3$ by $[0, 3]$ and $[0, 2, 1]$.)
- 5.3 Evaluate the infinite continued fraction $[2, \overline{1, 2, 1}]$.
- 5.4 Determine the infinite continued fraction of $\frac{1+\sqrt{13}}{2}$.
- 5.5 Let $a_0 \in \mathbf{R}$ and a_1, \dots, a_n and b be positive real numbers. Prove that

$$[a_0, a_1, \dots, a_n + b] < [a_0, a_1, \dots, a_n]$$

if and only if n is odd.

- 5.6 (*) Extend the method presented in the text to show that the continued fraction expansion of $e^{1/k}$ is

$$[1, (k-1), 1, 1, (3k-1), 1, 1, (5k-1), 1, 1, (7k-1), \dots]$$

for all $k \in \mathbf{N}$.

- (a) Compute $p_0, p_3, q_0,$ and q_3 for the above continued fraction. Your answers should be in terms of k .
- (b) Condense three steps of the recurrence for the numerators and denominators of the above continued fraction. That is, produce a simple recurrence for r_{3n} in terms of r_{3n-3} and r_{3n-6} whose coefficients are polynomials in n and k .
- (c) Define a sequence of real numbers by

$$T_n(k) = \frac{1}{k^n} \int_0^{1/k} \frac{(kt)^n (kt-1)^n}{n!} e^t dt.$$

- i. Compute $T_0(k)$, and verify that it equals $q_0 e^{1/k} - p_0$.
- ii. Compute $T_1(k)$, and verify that it equals $q_3 e^{1/k} - p_3$.

iii. Integrate $T_n(k)$ by parts twice in succession, as in Section 5.3, and verify that $T_n(k)$, $T_{n-1}(k)$, and $T_{n-2}(k)$ satisfy the recurrence produced in part 6b, for $n \geq 2$.

(d) Conclude that the continued fraction

$$[1, (k-1), 1, 1, (3k-1), 1, 1, (5k-1), 1, 1, (7k-1), \dots]$$

represents $e^{1/k}$.

- 5.7 Let d be an integer that is coprime to 10. Prove that the decimal expansion of $\frac{1}{d}$ has period equal to the order of 10 modulo d . (Hint: For every positive integer r , we have $\frac{1}{1-10^r} = \sum_{n \geq 1} 10^{-rn}$.)
- 5.8 Find a positive integer that has at least three different representations as the sum of two squares, disregarding signs and the order of the summands.
- 5.9 Show that if a natural number n is the sum of two rational squares it is also the sum of two integer squares.
- 5.10 (*) Let p be an odd prime. Show that $p \equiv 1, 3 \pmod{8}$ if and only if p can be written as $p = x^2 + 2y^2$ for some choice of integers x and y .
- 5.11 Prove that of any four consecutive integers, at least one is not representable as a sum of two squares.

6

Elliptic Curves

Definition 6.0.1 (Elliptic Curve). An *elliptic curve* over a field K is a curve of the form

$$y^2 = x^3 + ax + b,$$

where $a, b \in K$ and $-16(4a^3 + 27b^2) \neq 0$.

In Section 6.1 we equip the set

$$E(K) = \{(x, y) \in K \times K : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

of K -rational points on an elliptic curve E over K with an abelian group structure. Here \mathcal{O} may be thought of as a point on E “at infinity”. In Figure 6.1 we graph $y^2 = x^3 + x$ over the finite field $\mathbf{Z}/7\mathbf{Z}$, and in Figure 6.2 we graph $y^2 = x^3 + x$ over the field $K = \mathbf{R}$ of real numbers.

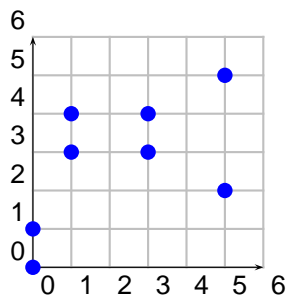
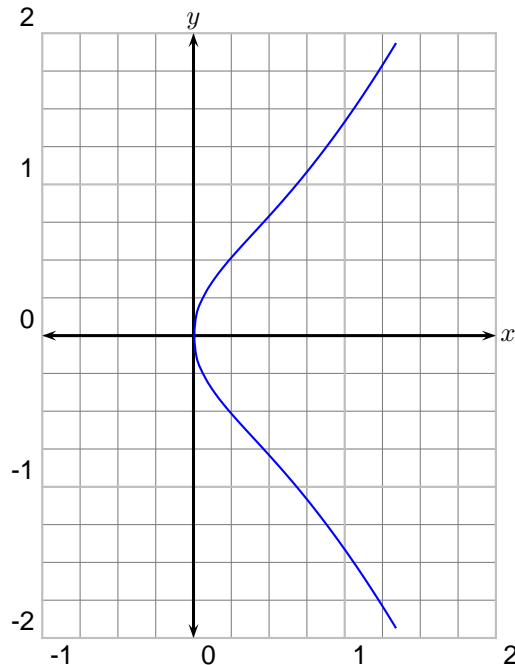


FIGURE 6.1. The Elliptic Curve $y^2 = x^3 + x$ over $\mathbf{Z}/7\mathbf{Z}$

FIGURE 6.2. The Elliptic Curve $y^2 = x^3 + x$ over \mathbf{R}

In Sections 6.2–6.3 we see how elliptic curves play a role in integer factorization algorithms, and how elliptic curves over finite fields provide cryptosystems that may in some ways be better than the cryptosystems from Chapter 3. In Section 6.4 we survey some results and conjectures about the groups $E(\mathbf{Q})$ for elliptic curves E over \mathbf{Q} .

Remark 6.0.2. If K has characteristic 2 (e.g., $K = \mathbf{Z}/2\mathbf{Z}$), then for any choice of a, b , the quantity $-16(4a^3 + 27b^2) \in K$ is 0, so according to Definition 6.0.1 there are no elliptic curves over K . There is a slightly more general definition of elliptic curves, which allows for elliptic curves in characteristic 2; these elliptic curves are popular in cryptography because arithmetic on them lends itself to computer implementation.

6.1 The Group Structure on an Elliptic Curve

Let E be an elliptic curve over a field K , given by an equation $y^2 = x^3 + ax + b$. We begin by defining a binary operation $+$ on $E(K)$.

Algorithm 6.1.1 (Elliptic Curve Group Law). Given $P_1, P_2 \in E(K)$, this algorithm computes a third point $R = P_1 + P_2 \in E(K)$.

1. [One Point \mathcal{O}] If $P_1 = \mathcal{O}$ set $R = P_2$ or if $P_2 = \mathcal{O}$ set $R = P_1$ and terminate. Otherwise write $P_i = (x_i, y_i)$.

2. [Negatives] If $x_1 = x_2$ and $y_1 = -y_2$, set $R = \mathcal{O}$ and terminate.
3. [Compute λ] Set $\lambda = \begin{cases} (3x_1^2 + a)/(2y_1) & \text{if } P_1 = P_2, \\ (y_1 - y_2)/(x_1 - x_2) & \text{otherwise.} \end{cases}$
Note: If $y_1 = 0$ and $P_1 = P_2$, output \mathcal{O} and terminate.
4. [Compute Sum] Then $R = (\lambda^2 - x_1 - x_2, -\lambda x_3 - \nu)$, where $\nu = y_1 - \lambda x_1$ and x_3 is the x coordinate of R .

We implement this algorithm in Section 7.6.1.

Theorem 6.1.2. *The binary operation $+$ defined above endows the set $E(K)$ with an abelian group structure, in which \mathcal{O} is the identity element.*

Before discussing why the theorem is true, we reinterpret $+$ geometrically, so that it will be easier for us to visualize. The sum $P_1 + P_2$ is obtained by finding the third point P_3 of intersection between E and the line L determined by P_1 and P_2 , then reflecting P_3 about the x -axis. (This description requires suitable interpretation in cases 1, 2, and when $P_1 = P_2$.) This is illustrated in Figure 6.3, in which $(0, 2) + (1, 0) = (3, 4)$ on $y^2 = x^3 - 5x + 4$. To further clarify this geometric interpretation, we prove the following proposition.

Proposition 6.1.3 (Geometric group law). *Suppose $P_i = (x_i, y_i)$, $i = 1, 2$ are distinct point on an elliptic curve $y^2 = x^3 + ax + b$, and that $x_1 \neq x_2$. Let L be the unique line through P_1 and P_2 . Then L intersects the graph of E at exactly one other point*

$$Q = (\lambda^2 - x_1 - x_2, \lambda x_3 + \nu),$$

where $\lambda = (y_1 - y_2)/(x_1 - x_2)$ and $\nu = y_1 - \lambda x_1$.

Proof. The line L through P_1, P_2 is $y = y_1 + (x - x_1)\lambda$. Substituting this into $y^2 = x^3 + ax + b$ we get

$$(y_1 + (x - x_1)\lambda)^2 = x^3 + ax + b.$$

Simplifying we get $f(x) = x^3 - \lambda^2 x^2 + \dots = 0$, where we omit the coefficients of x and the constant term. Since P_1 and P_2 are in $L \cap E$, the polynomial f has x_1 and x_2 as roots. By Proposition 2.5.2 f can have at most three roots. Writing $f = \prod (x - x_i)$ and equating terms, we see that $x_1 + x_2 + x_3 = \lambda^2$. Thus $x_3 = \lambda^2 - x_1 - x_2$, as claimed. Also, from the equation for L we see that $y_3 = y_1 + (x_3 - x_1)\lambda = \lambda x_3 + \nu$, which completes the proof. \square

To prove Theorem 6.1.2 means to show that $+$ satisfies the three axioms of an abelian group: existence of inverses, commutativity, and associativity. The existence of inverses follows immediately from the definition, since $(x, y) + (x, -y) = \mathcal{O}$. Commutativity is also clear from the definition of group law, since in parts 1–3, the recipe is unchanged if we swap P_1 and

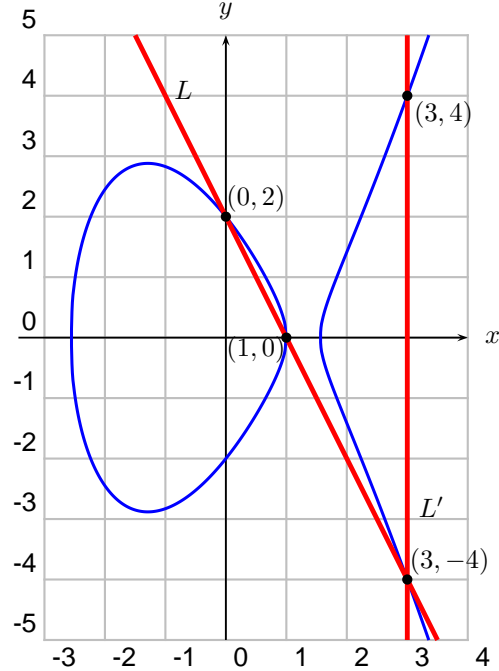


FIGURE 6.3. The Group Law: $(1, 0) + (0, 2) = (3, 4)$ on $y^2 = x^3 - 5x + 4$

P_2 ; in part 4 swapping P_1 and P_2 does not change the line determined by P_1 and P_2 , so by Proposition 6.1.3 it doesn't change the sum $P_1 + P_2$.

It is more difficult to prove that $+$ satisfies the associative axiom, i.e., that $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$. This can be proved in at least three ways. The first is to reinterpret the group law geometrically (extending Proposition 6.1.3 to all cases), and thus transfer the problem to a question in plane geometry. The reader who wishes to find such a proof as an exercise is encouraged to read [ST92, §I.2], which contains a helpful discussion and diagrams illustrating how such a geometric argument might proceed, but does not culminate with a complete proof. The second approach is to use the formulas that define $+$ to reduce associativity to checking specific algebraic identities; this is something that would be tedious to do by hand, but can be done using a computer. We give part of such a computer proof in Section 7.6.4 below. The third approach (see e.g. [Sil86]) is to develop a general theory of “divisors on algebraic curves”, from which associativity of the group law falls out as a corollary. The third approach is the best, because it opens up many new vistas; however we will not pursue it further because it is beyond the scope of this book.

6.2 Integer Factorization Using Elliptic Curves

In 1987, Hendrik Lenstra published the landmark paper [Len87] that introduces and analyzes the Elliptic Curve Method (ECM), which is a powerful algorithm for factoring integers using elliptic curves. Lenstra's method is also described in [ST92, §IV.4], [Dav99, §VIII.5], and [Coh93, §10.3].

Lenstra's algorithm is well suited for finding "medium sized" factors of an integer N , which today means 10 to 20 decimal digits. The ECM method is not directly useful for factoring RSA challenge numbers (see Section 1.1.3), but surprisingly it is used in intermediate steps of some of the algorithms that are used for hunting for such factorizations. Implementation of ECM typically requires little memory.



Lenstra

6.2.1 Pollard's $(p-1)$ -Method

Lenstra's discovery of ECM was inspired by Pollard's $(p-1)$ -method, which we describe in this section.

Definition 6.2.1 (Power smooth). Let B be a positive integer. If n is a positive integer with prime factorization $n = \prod p_i^{e_i}$, then n is B -power smooth if $p_i^{e_i} \leq B$ for all i .

Thus $30 = 2 \cdot 3 \cdot 5$ is B power smooth for $B = 5, 7$, but $150 = 2 \cdot 3 \cdot 5^2$ is not 5-power smooth (it is $B = 25$ -power smooth).

We will use the following algorithm in both the Pollard $p-1$ and elliptic curve factorization methods.

Algorithm 6.2.2 (Least Common Multiple of First B Integers). Given a positive integer B , this algorithm computes the least common multiple of the positive integers up to B .

1. [Sieve] Using the Sieve of Eratosthenes (Algorithm 1.2.3), compute a list P of all primes $p \leq B$.
2. [Multiply] Compute and output the product $\prod_{p \in P} \lfloor \log_p(B) \rfloor$.

Proof. Let $m = \text{lcm}(1, 2, \dots, B)$. Then

$$\text{ord}_p(m) = \max(\{\text{ord}_p(n) : 1 \leq n \leq B\}) = \text{ord}_p(p^r),$$

where p^r is the largest power of p that satisfies $p^r \leq B$. Since $p^r \leq B < p^{r+1}$, we have $r = \lfloor \log_p(B) \rfloor$. \square

We implement Algorithm 6.2.2 in Section 7.6.2.

Let N be a positive integer that we wish to factor. We use the Pollard $(p-1)$ -method to look for a nontrivial factor of N as follows. First we choose a positive integer B , usually with at most six digits. Suppose that there is a prime divisor p of N such that $p-1$ is B -power smooth. We try to find p computationally using the following strategy. If $a > 1$ is an integer not divisible by p then by Theorem 2.1.12,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Let $m = \text{lcm}(1, 2, 3, \dots, B)$, and observe that our assumption that $p-1$ is B -power smooth implies that $p-1 \mid m$, so

$$a^m \equiv 1 \pmod{p}.$$

Thus

$$p \mid \gcd(a^m - 1, N) > 1.$$

If $\gcd(a^m - 1, N) < N$ also then $\gcd(a^m - 1, N)$ is a nontrivial factor of N . If $\gcd(a^m - 1, N) = N$, then $a^m \equiv 1 \pmod{q^r}$ for every prime power divisor q^r of N . In this case, repeat the above steps but with a smaller choice of B or possibly a different choice of a . Also, it is a good idea to check from the start whether or not N is not a perfect power M^r , and if so replace N by M . We formalize the algorithm as follows:

Algorithm 6.2.3 (Pollard $p-1$ Method). Given a positive integer N and a bound B , this algorithm attempts to find a nontrivial factor m of N such that each prime $p \mid m$ has the property that $p-1$ is B -power smooth.

1. [Compute lcm] Use Algorithm 6.2.2 to compute $m = \text{lcm}(1, 2, \dots, B)$.
2. [Initialize] Set $a \leftarrow 2$.
3. [Power and gcd] Compute $x \leftarrow a^m - 1 \pmod{N}$ and $g = \gcd(x, N)$.
4. [Finished?] If $g \neq 1$ or N , output g and terminate.
5. [Try Again?] If $a < 10$ (say), set $a \leftarrow a + 1$ and go to step 3. Otherwise terminate.

We implement Algorithm 6.2.3 in Section 7.6.2.

For fixed B , Algorithm 6.2.3 often splits N when N is divisible by a prime p such that $p-1$ is B -power smooth. Approximately 15% of primes p in the interval from 10^{15} and $10^{15} + 10000$ are such that $p-1$ is 10^6 power-smooth, so the Pollard method with $B = 10^6$ already fails nearly 85% of the time at finding 15-digit primes in this range (see also Exercise 7.14). We will not analyze Pollard's method further, since it was mentioned here only to set the stage for the elliptic curve factorization method.

The following examples illustrate the Pollard $(p-1)$ -method.

Example 6.2.4. In this example, Pollard works perfectly. Let $N = 5917$. We try to use the Pollard $p - 1$ method with $B = 5$ to split N . We have $m = \text{lcm}(1, 2, 3, 4, 5) = 60$; taking $a = 2$ we have

$$2^{60} - 1 \equiv 3416 \pmod{5917}$$

and

$$\gcd(2^{60} - 1, 5917) = \gcd(3416, 5917) = 61,$$

so 61 is a factor of 5917.

Example 6.2.5. In this example, we replace B by larger integer. Let $N = 779167$. With $B = 5$ and $a = 2$ we have

$$2^{60} - 1 \equiv 710980 \pmod{779167},$$

and $\gcd(2^{60} - 1, 779167) = 1$. With $B = 15$, we have

$$m = \text{lcm}(1, 2, \dots, 15) = 360360,$$

$$2^{360360} - 1 \equiv 584876 \pmod{779167},$$

and

$$\gcd(2^{360360} - 1, N) = 2003,$$

so 2003 is a nontrivial factor of 779167.

Example 6.2.6. In this example, we replace B by a smaller integer. Let $N = 4331$. Suppose $B = 7$, so $m = \text{lcm}(1, 2, \dots, 7) = 420$,

$$2^{420} - 1 \equiv 0 \pmod{4331},$$

and $\gcd(2^{420} - 1, 4331) = 4331$, so we do not obtain a factor of 4331. If we replace B by 5, Pollard's method works:

$$2^{60} - 1 \equiv 1464 \pmod{4331},$$

and $\gcd(2^{60} - 1, 4331) = 61$, so we split 4331.

Example 6.2.7. In this example, $a = 2$ does not work, but $a = 3$ does. Let $N = 187$. Suppose $B = 15$, so $m = \text{lcm}(1, 2, \dots, 15) = 360360$,

$$2^{360360} - 1 \equiv 0 \pmod{187},$$

and $\gcd(2^{360360} - 1, 187) = 187$, so we do not obtain a factor of 187. If we replace $a = 2$ by $a = 3$, then Pollard's method works:

$$3^{360360} - 1 \equiv 66 \pmod{187},$$

and $\gcd(3^{360360} - 1, 187) = 11$. Thus $187 = 11 \cdot 17$.



FIGURE 6.4. Hendrik Lenstra

6.2.2 Motivation for the Elliptic Curve Method

Fix a positive integer B . If $N = pq$ with p and q prime and $p - 1$ and $q - 1$ are not B -power smooth, then the Pollard $(p - 1)$ -method is unlikely to work. For example, let $B = 20$ and suppose that $N = 59 \cdot 101 = 5959$. Note that neither $59 - 1 = 2 \cdot 29$ nor $101 - 1 = 4 \cdot 25$ is B -power smooth. With $m = \text{lcm}(1, 2, 3, \dots, 20) = 232792560$, we have

$$2^m - 1 \equiv 5944 \pmod{N},$$

and $\gcd(2^m - 1, N) = 1$, so we do not find a factor of N .

As remarked above, the problem is that $p - 1$ is not 20-power smooth for either $p = 59$ or $p = 101$. However, notice that $p - 2 = 3 \cdot 19$ is 20-power smooth. Lenstra's ECM replaces $(\mathbf{Z}/p\mathbf{Z})^*$, which has order $p - 1$, by the group of points on an elliptic curve E over $\mathbf{Z}/p\mathbf{Z}$. It is a theorem that

$$\#E(\mathbf{Z}/p\mathbf{Z}) = p + 1 \pm s$$

for some nonnegative integer $s < 2\sqrt{p}$ (see e.g., [Sil86, §V.1] for a proof). For example, if E is the elliptic curve

$$y^2 = x^3 + x + 54$$

over $\mathbf{Z}/59\mathbf{Z}$ then by enumerating points one sees that $E(\mathbf{Z}/59\mathbf{Z})$ is cyclic of order 57. The set of numbers $59 + 1 \pm s$ for $s \leq 15$ contains 14 numbers that are B -power smooth for $B = 20$ (see Exercise 7.14). Thus working with an elliptic curve gives us more flexibility. For example, $60 = 59 + 1 + 0$ is 5-power smooth and $70 = 59 + 1 + 10$ is 7-power smooth.

6.2.3 Lenstra's Elliptic Curve Factorization Method

Algorithm 6.2.8 (Elliptic Curve Factorization Method). Given a positive integer N and a bound B , this algorithm attempts to find a nontrivial factor m of N . Assume that N is prime, carry out the following steps:

1. [Compute lcm] Use Algorithm 6.2.2 to compute $m = \text{lcm}(1, 2, \dots, B)$.

2. [Choose Random Elliptic Curve] Choose a random $a \in \mathbf{Z}/N\mathbf{Z}$ such that $4a^3 + 27 \in (\mathbf{Z}/N\mathbf{Z})^*$. Then $P = (0, 1)$ is a point on the elliptic curve $y^2 = x^3 + ax + 1$ over $\mathbf{Z}/N\mathbf{Z}$.
3. [Compute Multiple] Attempt to compute mP using an elliptic curve analogue of Algorithm 2.3.7. If at some point we cannot compute a sum of points because some denominator in step 3 of Algorithm 6.1.1 is not coprime to N , we compute the gcd of this denominator with N . If this gcd is a nontrivial divisor, output it. If every denominator is coprime to N , output “Fail”.

We implement Algorithm 6.2.8 in Section 7.6.2.

If Algorithm 6.2.8 fails for one random elliptic curve, there is an option that is unavailable with Pollard’s $(p-1)$ -method—we may repeat the above algorithm with a different elliptic curve. With Pollard’s method we always work with the group $(\mathbf{Z}/N\mathbf{Z})^*$, but here we can try many groups $E(\mathbf{Z}/N\mathbf{Z})$ for many curves E . One can prove that number of points on E over $\mathbf{Z}/p\mathbf{Z}$ is of the form $p + 1 - t$ for some t with $|t| < 2\sqrt{p}$, and that Algorithm 6.2.8 is “likely” to succeed if $p + 1 - t$ is B -power-smooth.

6.2.4 Examples

For simplicity, we use an elliptic curve of the form

$$y^2 = x^3 + ax + 1,$$

which has the point $P = (0, 1)$ already on it.

We factor $N = 5959$ using the elliptic curve method. Let

$$m = \text{lcm}(1, 2, \dots, 20) = 232792560 = 1101111000000010000111110000_2,$$

where x_2 means x is written in binary. First we choose $a = 1201$ at random and consider $y^2 = x^3 + 1201x + 1$ over $\mathbf{Z}/5959\mathbf{Z}$. Using the formula for $P+P$ from Algorithm 6.1.1 implemented on a computer (see Section 7.6) we compute $2^i \cdot P = 2^i \cdot (0, 1)$ for $i \in B = \{4, 5, 6, 7, 8, 13, 21, 22, 23, 24, 26, 27\}$. Then $\sum_{i \in B} 2^i P = mP$. It turns out that during no step of this computation does a number not coprime to 5959 appear in any denominator, so we do not split N using $a = 1201$. Next we try $a = 389$ and at some stage in the computation we add $P = (2051, 5273)$ and $Q = (637, 1292)$. When computing the group law explicitly we try to compute $\lambda = (y_1 - y_2)/(x_1 - x_2)$ in $(\mathbf{Z}/5959\mathbf{Z})^*$, but fail since $x_1 - x_2 = 1414$ and $\text{gcd}(1414, 5959) = 101$. We thus find a nontrivial factor 101 of 5959.

For bigger examples and an implementation of the algorithm, see Section 7.6.2.

6.2.5 A Heuristic Explanation

Let N be a positive integer and for simplicity of exposition assume that $N = p_1 \cdots p_r$ with the p_i distinct primes. It follows from Lemma 2.2.5 that there is a natural isomorphism

$$f : (\mathbf{Z}/N\mathbf{Z})^* \longrightarrow (\mathbf{Z}/p_1\mathbf{Z})^* \times \cdots \times (\mathbf{Z}/p_r\mathbf{Z})^*.$$

When using Pollard's method, we choose an $a \in (\mathbf{Z}/N\mathbf{Z})^*$, compute a^m , then compute $\gcd(a^m - 1, N)$. This gcd is divisible exactly by the primes p_i such that $a^m \equiv 1 \pmod{p_i}$. To reinterpret Pollard's method using the above isomorphism, let $(a_1, \dots, a_r) = f(a)$. Then $(a_1^m, \dots, a_r^m) = f(a^m)$, and the p_i that divide $\gcd(a^m - 1, N)$ are exactly the p_i such that $a_i^m = 1$. By Theorem 2.1.12, these p_i include the primes p_j such that $p_j - 1$ is B -power smooth, where $m = \text{lcm}(1, \dots, m)$.

We will not define $E(\mathbf{Z}/N\mathbf{Z})$ when N is composite, since this is not needed for the algorithm (where we assume that N is prime and hope for a contradiction). However, for the remainder of this paragraph, we pretend that $E(\mathbf{Z}/N\mathbf{Z})$ is meaningful and describe a heuristic connection between Lenstra and Pollard's methods. The significant difference between Pollard's method and the elliptic curve method is that the isomorphism f is replaced by an isomorphism (in quotes)

$$g : E(\mathbf{Z}/N\mathbf{Z}) \rightarrow E(\mathbf{Z}/p_1\mathbf{Z}) \times \cdots \times E(\mathbf{Z}/p_r\mathbf{Z})$$

where E is $y^2 = x^3 + ax + 1$, and the a of Pollard's method is replaced by $P = (0, 1)$. We put the isomorphism in quotes to emphasize that we have not defined $E(\mathbf{Z}/N\mathbf{Z})$. When carrying out the elliptic curve factorization algorithm, we attempt to compute mP and if some components of $f(Q)$ are 0, for some point Q that appears during the computation, but others are nonzero, we find a nontrivial factor of N .

6.3 Elliptic Curve Cryptography

In this section we discuss an analogue of Diffie-Hellman that uses an elliptic curve instead of $(\mathbf{Z}/p\mathbf{Z})^*$. The idea to use elliptic curves in cryptography was independently proposed by Neil Koblitz and Victor Miller in the mid 1980s. We then discuss the ElGamal elliptic curve cryptosystem.

6.3.1 Elliptic Curve Analogues of Diffie-Hellman

The Diffie-Hellman key exchange from Section 3.1 works well on an elliptic curve with no serious modification. Michael and Nikita agree on a secret key as follows:

1. Michael and Nikita agree on a prime p , an elliptic curve E over $\mathbf{Z}/p\mathbf{Z}$, and a point $P \in E(\mathbf{Z}/p\mathbf{Z})$.
2. Michael secretly chooses a random m and sends mP .
3. Nikita secretly chooses a random n and sends nP .
4. The secret key is nmP , which both Michael and Nikita can compute.

Presumably, an adversary can not compute nmP without solving the discrete logarithm problem (see Problem 3.1.2 and Section 6.3.3 below) in $E(\mathbf{Z}/p\mathbf{Z})$. For well-chosen E , P , and p experience suggests that the discrete logarithm problem in $E(\mathbf{Z}/p\mathbf{Z})$ is much more difficult than the discrete logarithm problem in $(\mathbf{Z}/p\mathbf{Z})^*$ (see Section 6.3.3 for more on the elliptic curve discrete log problem).

6.3.2 The ElGamal Cryptosystem and Digital Rights Management

This section is about the ElGamal cryptosystem, which works well on an elliptic curves. This section draws on a paper by an actual computer hacker named Beale Screamer who cracked a “Digital Rights Management” (DRM) system.

The elliptic curve used in the DRM is an elliptic curve over the finite field $k = \mathbf{Z}/p\mathbf{Z}$, where

$$p = 785963102379428822376694789446897396207498568951.$$

In base 16 the number p is

$$89ABCDEF012345672718281831415926141424F7,$$

which includes counting in hexadecimal, and digits of e , π , and $\sqrt{2}$. The elliptic curve E is

$$y^2 = x^3 + 317689081251325503476317476413827693272746955927x \\ + 79052896607878758718120572025718535432100651934.$$

We have

$$\#E(k) = 785963102379428822376693024881714957612686157429,$$

and the group $E(k)$ is cyclic with generator

$$B = (771507216262649826170648268565579889907769254176, \\ 390157510246556628525279459266514995562533196655).$$

Our heroes Nikita and Michael share digital music when they are not out fighting terrorists. When Nikita installed the DRM software on her computer, it generated a private key

$$n = 670805031139910513517527207693060456300217054473,$$

which it hides in bits and pieces of files. In order for Nikita to play Juno Reactor's latest hit *juno*, her web browser contacts a web site that sells music. After Nikita sends her credit card number, that web site allows Nikita to download a license file that allows her audio player to unlock and play *juno*.

As we will see below, the license file was created using the ElGamal public-key cryptosystem in the group $E(k)$. Nikita can now use her license file to unlock *juno*. However, when she shares both *juno* and the license file with Michael, he is frustrated because even with the license his computer still does not play *juno*. This is because Michael's computer does not know Nikita's computer's private key (the integer n above), so Michael's computer can not decrypt the license file.



We now describe the ElGamal cryptosystem, which lends itself well to implementation in the group $E(\mathbf{Z}/p\mathbf{Z})$. To illustrate ElGamal, we describe how Nikita would set up an ElGamal cryptosystem that anyone could use to encrypt messages for her. Nikita chooses a prime p , an elliptic curve E over $\mathbf{Z}/p\mathbf{Z}$, and a point $B \in E(\mathbf{Z}/p\mathbf{Z})$, and publishes p , E , and B . She also chooses a random integer n , which she keeps secret, and publishes nB . Her public key is the four-tuple (p, E, B, nB) .

Suppose Michael wishes to encrypt a message for Nikita. If the message is encoded as an element $P \in E(\mathbf{Z}/p\mathbf{Z})$, Michael computes a random integer r and the points rB and $P + r(nB)$ on $E(\mathbf{Z}/p\mathbf{Z})$. Then P is encrypted as the pair $(rB, P + r(nB))$. To decrypt the encrypted message, Nikita multiplies rB by her secret key n to find $n(rB) = r(nB)$, then subtracts this from $P + r(nB)$ to obtain

$$P = P + r(nB) - r(nB).$$

We implement this cryptosystem in Section 7.6.3.

Remark 6.3.1. It also make sense to construct an ElGamal cryptosystem in the group $(\mathbf{Z}/p\mathbf{Z})^*$.

Returning out our story, Nikita's license file is an encrypted message to her. It contains the pair of points $(rB, P + r(nB))$, where

$$rB = (179671003218315746385026655733086044982194424660, \\ 697834385359686368249301282675141830935176314718)$$

and

$$P + r(nB) = (137851038548264467372645158093004000343639118915, \\ 110848589228676224057229230223580815024224875699).$$

When Nikita's computer plays `juno`, it loads the secret key

$$n = 670805031139910513517527207693060456300217054473$$

into memory and computes

$$n(rB) = (328901393518732637577115650601768681044040715701, \\ 586947838087815993601350565488788846203887988162).$$

It then subtracts this from $P + r(nB)$ to obtain

$$P = (14489646124220757767, \\ 669337780373284096274895136618194604469696830074).$$

The x -coordinate 14489646124220757767 is the key that unlocks `juno`.

If Nikita knew the private key n that her computer generated, she could compute P herself and unlock `juno` and share her music with Michael. Beale Screamer found a weakness in the implementation of the DRM that allows Nikita to find n , which is not surprising since n is stored on her computer.

6.3.3 The Elliptic Curve Discrete Logarithm Problem

Problem 6.3.2 (Elliptic Curve Discrete Log Problem). Suppose E is an elliptic curve over $\mathbf{Z}/p\mathbf{Z}$ and $P \in E(\mathbf{Z}/p\mathbf{Z})$. Given a multiple Q of P , the *elliptic curve discrete log problem* is to find $n \in \mathbf{Z}$ such that $nP = Q$.

For example, let E be the elliptic curve given by $y^2 = x^3 + x + 1$ over the field $\mathbf{Z}/7\mathbf{Z}$. We have

$$E(\mathbf{Z}/7\mathbf{Z}) = \{\mathcal{O}, (2, 2), (0, 1), (0, 6), (2, 5)\}.$$

If $P = (2, 2)$ and $Q = (0, 6)$, then $3P = Q$, so $n = 3$ is a solution to the discrete logarithm problem.

If $E(\mathbf{Z}/p\mathbf{Z})$ has order p or $p \pm 1$ or is a product of reasonably small primes, then there are some methods for attacking the discrete log problem on E , which are beyond the scope of this book. It is thus important to be able to compute $\#E(\mathbf{Z}/p\mathbf{Z})$ efficiently, in order to verify that the elliptic curve one wishes to use for a cryptosystem doesn't have any obvious vulnerabilities. The naive algorithm to compute $\#E(\mathbf{Z}/p\mathbf{Z})$ is to try each value of $x \in \mathbf{Z}/p\mathbf{Z}$ and count how often $x^3 + ax + b$ is a perfect square mod p , but this is of no use when p is large enough to be useful for cryptography. Fortunately, there is an algorithm due to Schoof, Elkies, and Atkin for computing $\#E(\mathbf{Z}/p\mathbf{Z})$ efficiently, but we will not describe this algorithm because it uses many ideas beyond the scope of this book.

In Section 3.1.1 we discussed the discrete log problem in $(\mathbf{Z}/p\mathbf{Z})^*$. There are general attacks called “index calculus attacks” on the discrete log problem in $(\mathbf{Z}/p\mathbf{Z})^*$ that are slow, but still faster than the known algorithms for solving the discrete log in a “general” group (one with no extra structure). For most elliptic curves, there is no known analogue of index calculus attacks on the discrete log problem. At present it appears that given p the discrete log problem in $E(\mathbf{Z}/p\mathbf{Z})$ is much harder than the discrete log problem in the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$. This suggests that by using an elliptic curve-based cryptosystem instead of one based on $(\mathbf{Z}/p\mathbf{Z})^*$ one gets equivalent security with much smaller numbers, which is one reason why building cryptosystems using elliptic curves is attractive to some cryptographers. For example, Certicom, a company that strongly supports elliptic curve cryptography, claims:

“[Elliptic curve crypto] devices require less storage, less power, less memory, and less bandwidth than other systems. This allows you to implement cryptography in platforms that are constrained, such as wireless devices, handheld computers, smart cards, and thin-clients. It also provides a big win in situations where efficiency is important.”

For an up-to-date list of elliptic curve discrete log challenge problems that Certicom sponsors, see [Cer]. For example, in April 2004 a specific cryptosystem was cracked that was based on an elliptic curve over $\mathbf{Z}/p\mathbf{Z}$, where p has 109 bits. The first unsolved challenge problem involves an elliptic curve over $\mathbf{Z}/p\mathbf{Z}$, where p has 131 bits, and the next challenge after that is one in which p has 163 bits. Certicom claims at [Cer] that the 163-bit challenge problem is computationally infeasible.

6.4 Elliptic Curves Over the Rational Numbers

Let E be an elliptic curve defined over \mathbf{Q} . The following is a deep theorem about the group $E(\mathbf{Q})$.



FIGURE 6.5. Louis J. Mordell

Theorem 6.4.1 (Mordell). *The group $E(\mathbf{Q})$ is finitely generated. That is, there are points $P_1, \dots, P_s \in E(\mathbf{Q})$ such that every element of $E(\mathbf{Q})$ is of the form $n_1P_1 + \dots + n_sP_s$ for integers $n_1, \dots, n_s \in \mathbf{Z}$.*

Mordell's theorem implies that it makes sense to ask whether or not we can compute $E(\mathbf{Q})$, where by “compute” we mean find a finite set P_1, \dots, P_s of points on E that generate $E(\mathbf{Q})$ as an abelian group. There is a systematic approach to computing $E(\mathbf{Q})$ (see e.g., [Cre97, Cre, Sil86]), and it is widely believed this method always succeeds, but nobody has yet proved that it always does. Proving that it does is one of the central open problems in number theory.

The details of the above approach to computing $E(\mathbf{Q})$ are beyond the scope of this book. In several places below we will simply assert that $E(\mathbf{Q})$ has a certain structure or is generated by certain elements. In each case, we computed $E(\mathbf{Q})$ using a computer implementation of this method.

6.4.1 The Torsion Subgroup of $E(\mathbf{Q})$ and the Rank

For any abelian group G , let G_{tor} be the subgroup of elements of finite order. If E is an elliptic curve over \mathbf{Q} , then $E(\mathbf{Q})_{\text{tor}}$ is a subgroup of $E(\mathbf{Q})$, which must be finite because of Theorem 6.4.1 (see Exercise 6.5). For example, if E is $y^2 = x^3 - 5x + 4$, then $E(\mathbf{Q})_{\text{tor}} = \{\mathcal{O}, (1, 0)\} \cong \mathbf{Z}/2\mathbf{Z}$.

The possibilities for $E(\mathbf{Q})_{\text{tor}}$ are known.

Theorem 6.4.2 (Mazur, 1976). *Let E be an elliptic curve over \mathbf{Q} . Then $E(\mathbf{Q})_{\text{tor}}$ is isomorphic to one of the following 15 groups:*

$$\begin{array}{ll} \mathbf{Z}/n\mathbf{Z} & \text{for } n \leq 10 \text{ or } n = 12, \\ \mathbf{Z}/2 \times \mathbf{Z}/2n & \text{for } n \leq 4. \end{array}$$

The quotient $E(\mathbf{Q})/E(\mathbf{Q})_{\text{tor}}$ is a finitely generated free abelian group, so it is isomorphic to \mathbf{Z}^r for some integer r , called the *rank* of $E(\mathbf{Q})$.

Conjecture 6.4.3. *There are elliptic curves over \mathbf{Q} of arbitrarily large rank.*

The “world record” is the following curve, whose rank is at least 24:

$$y^2 + xy + y = x^3 - 120039822036992245303534619191166796374x \\ + 504224992484910670010801799168082726759443756222911415116$$

It was discovered in January 2000 by Roland Martin and William McMillen of the National Security Agency. For several months they were not allowed to release the actual curve to the public.

6.4.2 The Congruent Number Problem

Definition 6.4.4 (Congruent Number). We call a nonzero rational number n a *congruent number* if $\pm n$ is the area of a right triangle with rational side lengths. Equivalently, n is a *congruent number* if the system of two equations

$$a^2 + b^2 = c^2 \\ \frac{1}{2}ab = n$$

has a solution with $a, b, c \in \mathbf{Q}$.

For example, 6 is the area of the right triangle with side lengths 3, 4, and 5, so 6 is a congruent number. Less obvious is that 5 is also a congruent number; it is the area of the right triangle with side lengths $3/2$, $20/3$, and $41/6$. It is nontrivial to prove that 1, 2, 3, and 4 are not congruent numbers. Here is a list of the integer congruent numbers up to 50:

5, 6, 7, 13, 14, 15, 20, 21, 22, 23, 24, 28, 29, 30, 31, 34, 37, 38, 39, 41, 45, 46, 47.

Every congruence class modulo 8 except 3 is represented in this list, which incorrectly suggests that if $n \equiv 3 \pmod{8}$ then n is not a congruent number. Though no $n \leq 218$ with $n \equiv 3 \pmod{8}$ is a congruent number, $n = 219$ is a congruent number congruent and $219 \equiv 3 \pmod{8}$.

Deciding whether an integer n is a congruent number can be subtle since the simplest triangle with area n can be very complicated. For example, as Zagier pointed out, the number 157 is a congruent number, and the “simplest” rational right triangle with area 157 has side lengths

$$a = \frac{6803298487826435051217540}{411340519227716149383203} \text{ and } b = \frac{411340519227716149383203}{21666555693714761309610}.$$

This solution would be difficult to find by a brute force search.

We call congruent numbers “congruent” because of the following proposition, which asserts that any congruent number is the common “congruence” between three perfect squares.

Proposition 6.4.5. *Suppose n is the area of a right triangle with rational side lengths a, b, c , with $a \leq b < c$. Let $A = (c/2)^2$. Then*

$$A - n, \quad A, \quad \text{and} \quad A + n$$

are all perfect squares of rational numbers.

Proof. We have

$$\begin{aligned} a^2 + b^2 &= c^2 \\ \frac{1}{2}ab &= n \end{aligned}$$

Add or subtract 4 times the second equation to the first to get

$$\begin{aligned} a^2 \pm 2ab + b^2 &= c^2 \pm 4n \\ (a \pm b)^2 &= c^2 \pm 4n \\ \left(\frac{a \pm b}{2}\right)^2 &= \left(\frac{c}{2}\right)^2 \pm n \\ &= A \pm n \end{aligned}$$

□

The following open problem has motivated much work on congruent numbers.

Open Problem 6.4.6. *Give an algorithm which, given n , outputs whether or not n is a congruent number.*

The following proposition establishes a link between elliptic curves and the congruent number problem.

Proposition 6.4.7 (Congruent numbers and elliptic curves). *Let n be a rational number. There is a bijection between*

$$A = \left\{ (a, b, c) \in \mathbf{Q}^3 : \frac{ab}{2} = n, a^2 + b^2 = c^2 \right\}$$

and

$$B = \{(x, y) \in \mathbf{Q}^2 : y^2 = x^3 - n^2x, \text{ with } y \neq 0\}$$

given explicitly by the maps

$$f(a, b, c) = \left(-\frac{nb}{a+c}, 2n^2a+c \right)$$

and

$$g(x, y) = \left(\frac{n^2 - x^2}{y}, -\frac{2xn}{y}, \frac{n^2 + x^2}{y} \right).$$

The proof of this proposition is not deep, but involves substantial algebra and we will not prove it in this book.

For $n \neq 0$, let E_n be the elliptic curve $y^2 = x^3 - n^2x$.

Proposition 6.4.8 (Congruent number criterion). *The rational number n is a congruent number if and only if there is a point $P = (x, y) \in E_n(\mathbf{Q})$ with $y \neq 0$.*

Proof. The number n is a congruent number if and only if the set A from Proposition 6.4.7 is nonempty. By the proposition A is nonempty if and only if B is nonempty. \square

Example 6.4.9. Let $n = 5$. Then E_n is $y^2 = x^3 - 25x$, and we find by a brute force search the point $(-4, -6) \in E_n(\mathbf{Q})$. Then

$$g(-4, -6) = \left(\frac{25 - 16}{-6}, -\frac{-40}{-6}, \frac{25 + 16}{-6} \right) = \left(-\frac{3}{2}, -\frac{20}{3}, -\frac{41}{6} \right).$$

Multiplying through by -1 yields the side lengths of a rational right triangle with area 5.

We can apply the map g to any point in $E_n(\mathbf{Q})$ with $y \neq 0$. Using the group law we find that $2(-4, -6) = (1681/144, 62279/1728)$, and

$$g(2(-4, -6)) = \left(-\frac{1519}{492}, -\frac{4920}{1519}, \frac{3344161}{747348} \right).$$

Example 6.4.10. Let $n = 1$, so E_1 is defined by $y^2 = x^3 - x$. Since 1 is not a congruent number, the elliptic curve E_1 has no point with $y \neq 0$. See Exercise 6.9.

Example 6.4.9 foreshadows the following theorem.

Theorem 6.4.11 (Infinitely Many Triangles). *If n is a congruent number, then there are infinitely many distinct right triangles with rational side lengths and area n .*

We will not prove this theorem, except to note that one proves it by showing that $E_n(\mathbf{Q})_{\text{tor}} = \{\mathcal{O}, (0, 0), (n, 0), (-n, 0)\}$, so the elements of the set B in Proposition 6.4.7 all have infinite order, hence B is infinite so A is infinite.

There is a theorem of Tunnell and a conjecture of Birch and Swinnerton-Dyer, which if true, would imply the existence of an elementary way to decide whether or not an integer n is a congruent number. We state this elementary way in the form of a conjecture.

Conjecture 6.4.12. *Let a, b, c denote integers. If n is an even square-free integer then n is a congruent number if and only if*

$$\# \left\{ (a, b, c) \in \mathbf{Z}^3 : 4a^2 + b^2 + 8c^2 = \frac{n}{2} : c \text{ is even} \right\}$$

$$= \# \left\{ (a, b, c) : 4a^2 + b^2 + 8c^2 = \frac{n}{2} : c \text{ is odd} \right\}.$$

If n is odd and square free then n is a congruent number if and only if

$$\begin{aligned} & \# \{ (a, b, c) : 2a^2 + b^2 + 8c^2 = n : c \text{ is even} \} \\ &= \# \{ (a, b, c) : 2a^2 + b^2 + 8c^2 = n : c \text{ is odd} \}. \end{aligned}$$

The book [Kob84] is about congruent numbers and Conjecture 6.4.12. The Birch and Swinnerton-Dyer conjecture is a Clay Math Institute million dollar millennium prize problem (see [Cla, Wil00]).

6.5 Exercises

6.1 One rational solution to the equation $y^2 = x^3 - 2$ is $(3, 5)$. Find a rational solution with $x \neq 3$ by drawing the tangent line to $(3, 5)$ and computing the second point of intersection.

6.2 Let E be the elliptic curve over the finite field $K = \mathbf{Z}/5\mathbf{Z}$ defined by the equation

$$y^2 = x^3 + x + 1.$$

- (a) List all 9 elements of $E(K)$.
- (b) What is the structure of $E(K)$, as a product of cyclic groups?

6.3 Let E be the elliptic curve defined by the equation $y^2 = x^3 + 1$. For each prime $p \geq 5$, let N_p be the cardinality of the group $E(\mathbf{Z}/p\mathbf{Z})$ of points on this curve having coordinates in $\mathbf{Z}/p\mathbf{Z}$. For example, we have that $N_5 = 6, N_7 = 12, N_{11} = 12, N_{13} = 12, N_{17} = 18, N_{19} = 12, N_{23} = 24,$ and $N_{29} = 30$ (you do not have to prove this).

- (a) For the set of primes satisfying $p \equiv 2 \pmod{3}$, can you see a pattern for the values of N_p ? Make a general conjecture for the value of N_p when $p \equiv 2 \pmod{3}$.
- (b) (*) Prove your conjecture.

6.4 Let E be an elliptic curve over the real numbers \mathbf{R} . Prove that $E(\mathbf{R})$ is not a finitely generated abelian group.

6.5 (*) Suppose G is a finitely generated abelian group. Prove that the subgroup G_{tor} of elements of finite order in G is finite.

6.6 Suppose $y^2 = x^3 + ax + b$ with $a, b \in \mathbf{Q}$ defines an elliptic curve. Show that there is another equation $Y^2 = X^3 + AX + B$ with $A, B \in \mathbf{Z}$ whose solutions are in bijection with the solutions to $y^2 = x^3 + ax + b$.

- 6.7 Suppose a, b, c are relatively prime integers with $a^2 + b^2 = c^2$. Then there exist integers x and y with $x > y$ such that $c = x^2 + y^2$ and either $a = x^2 - y^2, b = 2xy$ or $a = 2xy, b = x^2 - y^2$.
- 6.8 (*) Fermat's Last Theorem for exponent 4 asserts that any solution to the equation $x^4 + y^4 = z^4$ with $x, y, z \in \mathbf{Z}$ satisfies $xyz = 0$. Prove of Fermat's Last Theorem for exponent 4, as follows.
- Show that if the equation $x^2 + y^4 = z^4$ has no integer solutions with $xyz \neq 0$, then Fermat's Last Theorem for exponent 4 is true.
 - Prove that $x^2 + y^4 = z^4$ has no integer solutions with $xyz \neq 0$ as follows. Suppose $n^2 + k^4 = m^4$ is a solution with $m > 0$ minimal amongst all solutions. Show that there exists a solution with m smaller using Exercise 6.7 (consider two cases).
- 6.9 (*) Prove that 1 is not a congruent number by showing that the elliptic curve $y^2 = x^3 - x$ has no rational solutions except $(0, 1)$ and $(0, 0)$, as follows:
- Write $y = \frac{p}{q}$ and $x = \frac{r}{s}$, where p, q, r, s are all positive integers and $\gcd(p, q) = \gcd(r, s) = 1$. Prove that $s \mid q$, so $q = sk$ for some $k \in \mathbf{Z}$.
 - Prove that $s = k^2$, and substitute to see that $p^2 = r^3 - rk^4$.
 - Prove that r is a perfect square by supposing there is a prime ℓ such that $\text{ord}_\ell(r)$ is odd and analyzing ord_ℓ of both sides of $p^2 = r^3 - rk^4$.
 - Write $r = m^2$, and substitute to see that $p^2 = m^6 - m^2k^4$. Prove that $m \mid p$.
 - Divide through by m^2 and deduce a contradiction to Exercise 6.8.

7

Computational Number Theory

In this chapter, we discuss how to use the computer language Python to do computations with many of the mathematical objects discussed in this book. One reason we separate this chapter from the other chapters is that the best order for presenting theory is in many cases not the best order for presenting algorithms that rely on that theory. For example, in Section 2.1.1 we gave theoretical criterion for whether or not a linear equation $ax \equiv b \pmod{n}$ has a solution, and it wasn't until Section 2.3 that we described an algorithm for solving them. Moreover, extensive asides on issues related to implementing algorithms would obstruct the flow of the earlier chapters.

We use Python [Ros] because it is free and includes arbitrary precision integer arithmetic, but does *not* include substantial number theoretic functionality. If we were to use one of the major packages such as Mathematica, Maple, MATLAB, or MAGMA, then this chapter would be a manual describing how to use various builtin functions, instead of a chapter about how those functions actually work. Also, Python code is concise and easy to read. A drawback to using Python is that some of the algorithms we implemented for this book run more slowly than they would if implemented in certain other languages. We believe the clarity of having complete implementations of the relevant algorithms for this book easily available in a readable form is worth the tradeoff.

If you do not wish to use Python, you can still learn from this chapter. View the Python listings as pseudocode, and try to understand the details of how the algorithms work. In contrast, if you would like to understand Python well, great places to start are <http://docs.python.org/tut> and

<http://diveintopython.org>. Also, in this chapter we will describe new language features as we first encounter them.

Python is freely available from <http://www.python.org>. The examples in this chapter assume you are using Python version at least 2.3. You can download a file that contains all of the code printed on the following pages from

<http://modular.fas.harvard.edu/ent/>.

Put the file `ent.py` in a directory, start up Python, and load the functions from `ent.py` by typing the following:

```
>>> from ent import *
```

You might also install IPython (<http://ipython.scipy.org>), which provides a friendly interface to Python with better support for mathematics and documentation.

The examples in this chapter have been automatically tested using the default Python 2.3 shell. Some examples contain numbers that are obtained using randomized algorithms, so output may be different for you. Lines containing such output are indicated by a comment `#rand`.

Some of the functions defined in this chapter use the Python functions `log` and `sqrt` from the Python `math` library, and the `randrange` function from the `random` library. The code below assumes these three functions have been imported as follows:

```
from random import randrange
from math import log, sqrt
```

In Python the notation `==` means “equals”, `!=` means “not equals”, `>=` means \geq and `<=` means \leq . Another important convention in Python is that if n and m are integers, then the expression `n/m` evaluates to the biggest integer $\leq n/m$, as the following examples illustrate:

```
>>> 7/5
1
>>> -2/3
-1
```

To obtain a floating point approximation to a rational number use a decimal point or coerce at least one of the integers to a `float`

```
>>> 1.0/3
0.33333333333333331
>>> float(2)/3
0.66666666666666663
```

7.1 Prime Numbers

The main algorithms relevant to Chapter 1 are Algorithm 1.1.12 for computing greatest common divisors, an algorithm for integer factorization, and Algorithm 1.2.3 which computes all primes up to a certain bound.

7.1.1 Greatest Common Divisors

The following is an implementation of Algorithm 1.1.12.

Listing 7.1.1 (Greatest Common Divisor).

```
def gcd(a, b):                                     # (1)
    """
    Returns the greatest common divisor of a and b.
    Input:
        a -- an integer
        b -- an integer
    Output:
        an integer, the gcd of a and b
    Examples:
    >>> gcd(97,100)
    1
    >>> gcd(97 * 10**15, 19**20 * 97**2)          # (2)
    97L
    """
    if a < 0: a = -a
    if b < 0: b = -b
    if a == 0: return b
    if b == 0: return a
    while b != 0:                                  # (3)
        (a, b) = (b, a%b)                          # (4)
    return a
```

In line (1) we declare the name of the function and the two input arguments a and b . Notice how the rest of the function is indented. In Python indentation has meaning, e.g., it determines the scope of the definition of the `gcd` function and the `while` loop in lines (3) and (4). The part of Listing 7.1.1 between triple quotes is a documentation string; it is where we describe the `gcd` function, its input and output, and gives examples of usage. All functions defined in this chapter include such a documentation string, which is usually longer than the actual code that implements the function. From within IPython the documentation string can be accessed by typing `gcd?`.

In line (2) notice that exponentiation x^y in Python is denoted `x**y`. The output of the second example is `97L` instead of `97` because Python

implements two types of integers, `int` and `long`. The `int` type represents integers that fit within the “word size” of the computer. The `long` type represents integers of arbitrary size, but computations with them are slower than with `int`. When a computation involving an `int` results in an integer that is larger than can fit in an `int`, the result is of type `long`. The reason `97L` is printed instead of `97` is that `longs` are printed with a trailing `L`, as the following example illustrates.

```
>>> 100**2
10000
>>> 10**20
100000000000000000000L
```

The rest of the code implements Algorithm 1.1.12. The expression `a%b`, read “ a mod b ”, in the while loop is Python’s notation for the the unique integer r such that $0 \leq r < |b|$ and $a = bq+r$ for some $q \in \mathbf{Z}$. The command `(a,b)=(b,a%b)` simultaneously sets `a` to `b` and `b` to the remainder `a%b`.

7.1.2 Enumerating Primes

Listing 7.1.2 contains an implementation of Algorithm 1.2.3.

Listing 7.1.2 (Sieve of Eratosthenes).

```
def primes(n):
    """
    Returns a list of the primes up to n, computed
    using the Sieve of Eratosthenes.
    Input:
        n -- a positive integer
    Output:
        list -- a list of the primes up to n
    Examples:
    >>> primes(10)
    [2, 3, 5, 7]
    >>> primes(45)
    [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]
    """
    if n <= 1: return []
    X = range(3,n+1,2) # (1)
    P = [2] # (2)
    sqrt_n = sqrt(n) # (3)
    while len(X) > 0 and X[0] <= sqrt_n: # (4)
        p = X[0] # (5)
        P.append(p) # (6)
        X = [a for a in X if a%p != 0] # (7)
    return P + X # (8)
```

In the line labeled (1) we create the list X of odd numbers i with $3 \leq i < n+1$ using Python's `range` function. In line (2) we create the list P with the single element 2. In line (3) we compute \sqrt{n} using the `sqrt` library function imported earlier. Line (4) sets up a while loop that iterates until either X is empty or the first element of X is greater than \sqrt{n} . Line (5) sets p equal to the first element of X , then line (6) appends p to the end of P . Line (7) deletes the elements of X that are divisible by p . Finally line (8) is executed after the while loop terminates, and returns the concatenation of P and X .

Our implementation of `primes` makes extensive use the Python `list` data type. The following examples further illustrate use of lists:

```
>>> range(10)           # range(n) is from 0 to n-1
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(3,10)        # range(a,b) is from a to b-1
[3, 4, 5, 6, 7, 8, 9]
>>> [x**2 for x in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> [x**2 for x in range(10) if x%4 == 1]
[1, 25, 81]
>>> [1,2,3] + [5,6,7]   # concatenation
[1, 2, 3, 5, 6, 7]
>>> len([1,2,3,4,5])   # length of a list
5
>>> x = [4,7,10,'gcd']  # mixing types is fine
>>> x[0]                # 0-based indexing
4
>>> x[3]
'gcd'
>>> x[3] = 'lagrange'   # assignment
>>> x.append("fermat")  # append to end of list
>>> x
[4, 7, 10, 'lagrange', 'fermat']
>>> del x[3]           # delete entry 3 from list
>>> x
[4, 7, 10, 'fermat']
```

The following examples illustrate an application of the `primes` function to computation of the number $\pi(x)$ of primes up to x .

```
>>> v = primes(10000)
>>> len(v)             # this is pi(10000)
1229
>>> len([x for x in v if x < 1000]) # pi(1000)
168
>>> len([x for x in v if x < 5000]) # pi(5000)
```

7.1.3 Integer Factorization

We implement integer factorization using two functions. The first function splits off a factor using an algorithm such as trial division, the Pollard $p - 1$ method, or the elliptic curve method. The second splits off factors until n is completely factored. Listing 7.1.4 contains an implementation of a factorization algorithm, which by default uses the trial division splitting algorithm implemented in Listing 7.1.3. In Section 7.6 we will see how to use the Pollard $p - 1$ and elliptic curve algorithms for splitting off factors.

Trial division is a simple method for splitting off the smallest prime factor of an integer. If it splits off a factor, then that factor is guaranteed to be prime. The implementation below quickly factors numbers with up to about 12 digits, and can also be used to factor off small primes from a large number.

Listing 7.1.3 (Trial Division).

```
def trial_division(n, bound=None):
    """
    Return the smallest prime divisor <= bound of the
    positive integer n, or n if there is no such prime.
    If the optional argument bound is omitted, then bound=n.
    Input:
        n -- a positive integer
        bound - (optional) a positive integer
    Output:
        int -- a prime p<=bound that divides n, or n if
              there is no such prime.
    Examples:
    >>> trial_division(15)
    3
    >>> trial_division(91)
    7
    >>> trial_division(11)
    11
    >>> trial_division(387833, 300)
    387833
    >>> # 300 is not big enough to split off a
    >>> # factor, but 400 is.
    >>> trial_division(387833, 400)
    389
    """
    if n == 1: return 1
```

```

for p in [2, 3, 5]:
    if n%p == 0: return p
if bound == None: bound = n
dif = [6, 4, 2, 4, 2, 4, 6, 2]
m = 7; i = 1
while m <= bound and m*m <= n:
    if n%m == 0:
        return m
    m += dif[i%8]
    i += 1
return n

```

When declaring `trial_division` the second argument is `bound=None`. This means the second argument is optional, and if the user omits it when calling `trial_division`, then `bound` is set equal to `None`. In the while loop we use `+=`, e.g., in the line `i += 1`. This has exactly the same effect as `i=i+1`, but may be implemented more efficiently.

The following two observations are needed to see that the implementation in Listing 7.1.3 is correct. First, in order to find a divisor of n it is only necessary to consider integers $m \leq \sqrt{n}$. This is because if $m > \sqrt{n}$ and $m \mid n$, then n/m also divides n and $n/m < \sqrt{n}$. Second, for efficiency the implementation above does not simply march through all $m \leq \sqrt{n}$, but after checking that none of 2, 3, 5 divides n , starts with $m = 7$ and increments m by each of 4, 2, 4, 2, 4, 6, 2, 6 in turn, cycling around. This has the effect of skipping those m that are divisible by 2, 3, or 5. The reason is that the numbers modulo 30 that are coprime to 2, 3, 5 are exactly 7, 7+4, 7+4+2, 7+4+2+4, etc. One could, of course, replace 30 by $210 = 2 \cdot 3 \cdot 5 \cdot 7$ at the expense of replacing `dif` by a longer list (see Exercise 7.2).

Listing 7.1.4 contains an implementation of a factorization algorithm that uses `trial_division`.

Listing 7.1.4 (Integer Factorization).

```

def factor(n):
    """
    Returns the factorization of the integer n as
    a sorted list of tuples (p,e), where the integers p
    are output by the split algorithm.
    Input:
        n -- an integer
    Output:
        list -- factorization of n
    Examples:
    >>> factor(500)
    [(2, 2), (5, 3)]
    >>> factor(-20)

```

```

[(2, 2), (5, 1)]
>>> factor(1)
[]
>>> factor(2004)
[(2, 2), (3, 1), (167, 1)]
"""
if n in [-1, 0, 1]: return []
if n < 0: n = -n
F = []
while n != 1:
    p = trial_division(n)
    e = 1
    n /= p
    while n%p == 0:
        e += 1; n /= p
    F.append((p,e))
F.sort()
return F

```

The pairs (p, e) in the factorization are represented as tuples. The tuple type is similar to the list type, with some exceptions. The following examples illustrate usage of the tuple type:

```

>>> x=(1, 2, 3)      # creation
>>> x[1]
2
>>> (1, 2, 3) + (4, 5, 6) # concatenation
(1, 2, 3, 4, 5, 6)
>>> (a, b) = (1, 2)    # assignment assigns to each member
>>> print a, b
1 2
>>> for (c, d) in [(1,2), (5,6)]:
...     print c, d
1 2
5 6
>>> x = 1, 2          # parentheses optional in creation
>>> x
(1, 2)
>>> c, d = x          # parentheses also optional
>>> print c, d
1 2

```

7.2 The Ring of Integers Modulo n

The main algorithmic issues of Chapter 2 are solving linear equations and systems of linear equations in one variable modulo n , computing powers quickly, finding a generator of $(\mathbf{Z}/p\mathbf{Z})^*$, and determining whether or not a number is prime.

7.2.1 Linear Equations Modulo n

Listing 7.2.1 is an implementation of Algorithm 2.3.4 for computing g and integers x, y such that $ax + by = g$.

Listing 7.2.1 (Extended GCD).

```
def xgcd(a, b):
    """
    Returns g, x, y such that g = x*a + y*b = gcd(a,b).
    Input:
        a -- an integer
        b -- an integer
    Output:
        g -- an integer, the gcd of a and b
        x -- an integer
        y -- an integer
    Examples:
    >>> xgcd(2,3)
    (1, -1, 1)
    >>> xgcd(10, 12)
    (2, -1, 1)
    >>> g, x, y = xgcd(100, 2004)
    >>> print g, x, y
    4 -20 1
    >>> print x*100 + y*2004
    4
    """
    if a == 0 and b == 0: return (0, 0, 1)
    if a == 0: return (abs(b), 0, b/abs(b))
    if b == 0: return (abs(a), a/abs(a), 0)
    x_sign = 1; y_sign = 1
    if a < 0: a = -a; x_sign = -1
    if b < 0: b = -b; y_sign = -1
    x = 1; y = 0; r = 0; s = 1
    while b != 0:
        (c, q) = (a%b, a/b)
        (a, b, r, s, x, y) = (b, c, x-q*r, y-q*s, r, s)
    return (a, x*x_sign, y*y_sign)
```

Using Proposition 2.1.9 and `xgcd` we obtain the following algorithm for computing the inverse of $a \pmod n$.

Listing 7.2.2 (Inverse Modulo).

```
def inversemod(a, n):
    """
    Returns the inverse of a modulo n, normalized to
    lie between 0 and n-1. If a is not coprime to n,
    raise an exception (this will be useful later for
    the elliptic curve factorization method).
    Input:
        a -- an integer coprime to n
        n -- a positive integer
    Output:
        an integer between 0 and n-1.
    Examples:
    >>> inversemod(1,1)
    0
    >>> inversemod(2,5)
    3
    >>> inversemod(5,8)
    5
    >>> inversemod(37,100)
    73
    """
    g, x, y = xgcd(a, n)
    if g != 1:
        raise ZeroDivisionError, (a,n)
    assert g == 1, "a must be coprime to n."
    return x%n
```

Proposition 2.1.9 leads to the algorithm implemented in Listing 7.2.3 for solving a linear equation $ax \equiv b \pmod n$. In line (1) we compute c such that $ac \equiv g \pmod n$; also in line (1) the underscore means that the third value returned by `xgcd` should be ignored (not saved to a variable). Since $g \mid a$ and $g \mid n$, we have $(a/g)c \equiv 1 \pmod{n/g}$, and multiplying by b , rearranging, and using that $g \mid b$, yields $a(b/g)c \equiv b \pmod{bn/g}$. Thus $(b/g)c$ solves the equation $ax \equiv b \pmod n$.

Listing 7.2.3 (Solve Linear Modulo).

```
def solve_linear(a,b,n):
    """
    If the equation ax = b (mod n) has a solution, return a
```

solution normalized to lie between 0 and $n-1$, otherwise returns None.

Input:

```
a -- an integer
b -- an integer
n -- an integer
```

Output:

```
an integer or None
```

Examples:

```
>>> solve_linear(4, 2, 10)
```

```
8
```

```
>>> solve_linear(2, 1, 4) == None
```

```
True
```

```
"""
```

```
g, c, _ = xgcd(a,n) # (1)
```

```
if b%g != 0: return None
```

```
return ((b/g)*c) % n
```

In Listing 7.2.4 we implement Algorithm 2.2.3 for solving Chinese Remainder Theorem problems.

Listing 7.2.4 (Chinese Remainder Theorem).

```
def crt(a, b, m, n):
```

```
    """
```

```
    Return the unique integer between 0 and  $m*n - 1$ 
    that reduces to  $a$  modulo  $n$  and  $b$  modulo  $m$ , where
    the integers  $m$  and  $n$  are coprime.
```

```
    Input:
```

```
        a, b, m, n -- integers, with  $m$  and  $n$  coprime
```

```
    Output:
```

```
        int -- an integer between 0 and  $m*n - 1$ .
```

```
    Examples:
```

```
>>> crt(1, 2, 3, 4)
```

```
10
```

```
>>> crt(4, 5, 10, 3)
```

```
14
```

```
>>> crt(-1, -1, 100, 101)
```

```
10099
```

```
"""
```

```
g, c, _ = xgcd(m, n)
```

```
assert g == 1, "m and n must be coprime."
```

```
return (a + (b-a)*c*m) % (m*n)
```

7.2.2 Computation of Powers

In Listing 7.2.5 we implement Algorithm 2.3.7 for quickly computing large powers of an integer modulo n .

Listing 7.2.5 (Power Modulo).

```
def powermod(a, m, n):
    """
    The m-th power of a modulo n.
    Input:
        a -- an integer
        m -- a nonnegative integer
        n -- a positive integer
    Output:
        int -- an integer between 0 and n-1
    Examples:
    >>> powermod(2,25,30)
    2
    >>> powermod(19,12345,100)
    99
    """
    assert m >= 0, "m must be nonnegative." # (1)
    assert n >= 1, "n must be positive."    # (2)
    ans = 1
    apow = a
    while m != 0:
        if m%2 != 0:
            ans = (ans * apow) % n          # (3)
            apow = (apow * apow) % n       # (4)
            m /= 2
    return ans % n
```

The two `assert` statements in lines (1) and (2) express conditions that must be satisfied by the input to the function. If either condition is not satisfied, the function terminates and the corresponding error message is printed. In the while loop, in lines (3) and (4), we reduce each intermediate integer modulo n , since otherwise the integers involved could be huge.

7.2.3 Finding a Primitive Root

Listing 7.2.6 contains an implementation of Algorithm 2.5.13 for computing a primitive root modulo p .

Listing 7.2.6 (Primitive Root).

```
def primitive_root(p):
```

```

"""
Returns first primitive root modulo the prime p.
(If p is not prime, this return value of this function
is not meaningful.)
Input:
    p -- an integer that is assumed prime
Output:
    int -- a primitive root modulo p
Examples:
>>> primitive_root(7)
3
>>> primitive_root(389)
2
>>> primitive_root(5881)
31
"""
if p == 2: return 1
F = factor(p-1)
a = 2
while a < p:
    generates = True
    for q, _ in F:
        if powermod(a, (p-1)/q, p) == 1:
            generates = False
            break
    if generates: return a
    a += 1
assert False, "p must be prime."

```

7.2.4 *Determining Whether a Number is Prime*

In Listing 7.2.7 we define a function that decides whether or not an integer is a pseudoprime to several bases. See Section 2.4 for the connection between primes and pseudo-primes.

Listing 7.2.7 (Is Pseudoprime).

```

def is_pseudoprime(n, bases = [2,3,5,7]):
    """
    Returns True if n is a pseudoprime to the given bases,
    in the sense that  $n > 1$  and  $b^{n-1} = 1 \pmod{n}$  for each
    elements b of bases, with b not a multiple of n, and
    False otherwise.
    Input:
        n -- an integer
    """

```

```

    bases -- a list of integers
Output:
    bool
Examples:
>>> is_pseudoprime(91)
False
>>> is_pseudoprime(97)
True
>>> is_pseudoprime(1)
False
>>> is_pseudoprime(-2)
True
>>> s = [x for x in range(10000) if is_pseudoprime(x)]
>>> t = primes(10000)
>>> s == t
True
>>> is_pseudoprime(29341) # first non-prime pseudoprime
True
>>> factor(29341)
[(13, 1), (37, 1), (61, 1)]
"""
if n < 0: n = -n
if n <= 1: return False
for b in bases:
    if b%n != 0 and powermod(b, n-1, n) != 1:
        return False
return True

```

We iterate over the elements b of `bases`, and for each b that is not a multiple of n , we decide whether $b^{n-1} \equiv 1 \pmod{n}$. If not, then n is definitely not prime so we return `False`; if the congruence is satisfied for all b , return `True`.

The following session illustrates that for the default bases 2, 3, 5, 7, the first non-prime pseudoprime is 29341, and for the bases 2, 3, 5, 7, 11, 13, then the first non-prime pseudoprime is 162401:

```

>>> P = [p for p in range(200000) if is_pseudoprime(p)]
>>> Q = primes(200000)
>>> R = [x for x in P if not (x in Q)]; print R
[29341, 46657, 75361, 115921, 162401]
>>> [n for n in R if is_pseudoprime(n, [2,3,5,7,11,13])]
[162401]
>>> factor(162401)
[(17, 1), (41, 1), (233, 1)]

```

We next turn to the Miller-Rabin primality test. First we state the algorithm precisely with proof, and give an implementation in Listing 7.2.9

Algorithm 7.2.8 (Miller-Rabin Primality Test). Given an integer $n \geq 5$ this algorithm outputs either true or false. If it outputs true, then n is “probably prime”, and if it outputs false, then n is definitely composite.

1. [Split Off Power of 2] Compute the unique integers m and k such that m is odd and $n - 1 = 2^k \cdot m$.
2. [Random Base] Choose a random integer a with $1 < a < n$.
3. [Odd Power] Set $b \leftarrow a^m \pmod{n}$. If $b \equiv \pm 1 \pmod{n}$ output true and terminate.
4. [Even Powers] If $b^{2^r} \equiv -1 \pmod{n}$ for any r with $1 \leq r \leq k - 1$, output true and terminate. Otherwise output false.

If Miller-Rabin outputs true for n , we can call it again with n and if it again outputs true then the probability that n is prime increases.

Proof. We will prove that the algorithm is correct, but will prove nothing about how likely the algorithm is to assert that a composite is prime. We must prove that if the algorithm pronounces an integer n composite, then n really is composite. Thus suppose n is prime, yet the algorithm pronounces n composite. Then $a^m \not\equiv \pm 1 \pmod{n}$, and for all r with $1 \leq r \leq k - 1$ we have $a^{2^r m} \not\equiv -1 \pmod{n}$. Since n is prime and $2^{k-1}m = (n-1)/2$, Proposition 4.2.1 implies that $a^{2^{k-1}m} \equiv \pm 1 \pmod{n}$, so by our hypothesis $a^{2^{k-1}m} \equiv 1 \pmod{n}$. But then $(a^{2^{k-2}m})^2 \equiv 1 \pmod{n}$, so by Proposition 2.5.2, we have $a^{2^{k-2}m} \equiv \pm 1 \pmod{n}$. Again, by our hypothesis, this implies $a^{2^{k-2}m} \equiv 1 \pmod{n}$. Repeating this argument inductively we see that $a^m \equiv \pm 1 \pmod{n}$, which contradicts our hypothesis on a . \square

The implementation of Algorithm 7.2.8 in Listing 7.2.9 runs the Miller-Rabin primality test on n several times (a default of 4) and returns true only if n is declared probably prime every time. One of the examples illustrate how Miller-Rabin sometimes gives incorrect results.

Listing 7.2.9 (Miller-Rabin Primality Test).

```
def miller_rabin(n, num_trials=4):
    """
    True if n is likely prime, and False if n
    is definitely not prime. Increasing num_trials
    increases the probability of correctness.
    (One can prove that the probability that this
    function returns True when it should return
    False is at most (1/4)**num_trials.)
```

Input:

```
n -- an integer
num_trials -- the number of trials with the
              primality test.
```

Output:

```
bool -- whether or not n is probably prime.
```

Examples:

```
>>> miller_rabin(91)
False                                     #rand
>>> miller_rabin(97)
True                                      #rand
>>> s = [x for x in range(1000) if miller_rabin(x, 1)]
>>> t = primes(1000)
>>> print len(s), len(t) # so 1 in 25 wrong
175 168                                  #rand
>>> s = [x for x in range(1000) if miller_rabin(x)]
>>> s == t
True                                      #rand
"""
if n < 0: n = -n
if n in [2,3]: return True
if n <= 4: return False
m = n - 1
k = 0
while m%2 == 0:
    k += 1; m /= 2
# Now n - 1 = (2**k) * m with m odd
for i in range(num_trials):
    a = randrange(2,n-1)                    # (1)
    apow = powermod(a, m, n)
    if not (apow in [1, n-1]):
        some_minus_one = False
        for r in range(k-1):              # (2)
            apow = (apow**2)%n
            if apow == n-1:
                some_minus_one = True
                break                      # (3)
    if (apow in [1, n-1]) or some_minus_one:
        prob_prime = True
    else:
        return False
return True
```

In line (1) we use `randrange`; the command `randrange(a,b)` returns a random integer in the interval $[a, b - 1]$. Line (3) uses the `break` statement, which exists the immediately enclosing `for` or `while` loop; in this case the `for` loop starting at line (2).

7.3 Public-Key Cryptography

The main algorithms in Chapter 3 deal with implementing the Diffie-Hellman and RSA cryptosystems, and with some attacks on RSA in special cases. In this section we give a function for encoding an arbitrary string as a sequence of numbers of some bounded size, and vice-versa, then implement each of Diffie-Hellman and RSA.

7.3.1 The Diffie-Hellman Key Exchange

In order for two parties to agree on a secret key using Diffie-Hellman, we need a function to generate a large random prime.

Listing 7.3.1 (Random Prime).

```
def random_prime(num_digits, is_prime = miller_rabin):
    """
    Returns a random prime with num_digits digits.
    Input:
        num_digits -- a positive integer
        is_prime -- (optional argument)
                   a function of one argument n that
                   returns either True if n is (probably)
                   prime and False otherwise.
    Output:
        int -- an integer
    Examples:
    >>> random_prime(10)
    8599796717L          #rand
    >>> random_prime(40)
    1311696770583281776596904119734399028761L #rand
    """
    n = randrange(10**(num_digits-1), 10**num_digits)
    if n%2 == 0: n += 1
    while not is_prime(n): n += 2
    return n
```

Suppose p is a large random prime. Then it is extremely unlikely that 2 will have small order modulo p , so we will use $g = 2$ as the base for the

key exchange. The function `dh_init` below computes and returns a random integer n and $2^n \pmod{p}$. Thus Nikita and Michael should each call `dh_init` with input p , and send the resulting $2^n \pmod{p}$ to each other. Then each calls `dh_secret` with the powers of 2 they received to compute the the shared secret key. After defining `dh_init` and `dh_secret` below, we give a complete nontrivial example.

Listing 7.3.2 (Initialize Diffie-Hellman).

```
def dh_init(p):
    """
    Generates and returns a random positive
    integer n < p and the power 2^n (mod p).
    Input:
        p -- an integer that is prime
    Output:
        int -- a positive integer < p, a secret
        int -- 2^n (mod p), send to other user
    Examples:
    >>> p = random_prime(20)
    >>> dh_init(p)
    (15299007531923218813L, 4715333264598442112L) #rand
    """
    n = randrange(2,p)
    return n, powermod(2,n,p)
```

Listing 7.3.3 (Diffie-Hellman Secret).

```
def dh_secret(p, n, mpow):
    """
    Computes the shared Diffie-Hellman secret key.
    Input:
        p -- an integer that is prime
        n -- an integer: output by dh_init for this user
        mpow-- an integer: output by dh_init for other user
    Output:
        int -- the shared secret key.
    Examples:
    >>> p = random_prime(20)
    >>> n, npow = dh_init(p)
    >>> m, mpow = dh_init(p)
    >>> dh_secret(p, n, mpow)
    15695503407570180188L #rand
    >>> dh_secret(p, m, npow)
    15695503407570180188L #rand
    """
```

```
return powermod(mpow,n,p)
```

First Nikita and Michael generate a prime.

```
>>> p = random_prime(50)
>>> p
13537669335668960267902317758600526039222634416221L #rand
```

Nikita generates her secret n and computes $2^n \pmod{p}$.

```
>>> n, npow = dh_init(p)
>>> n
8520467863827253595224582066095474547602956490963L #rand
>>> npow
3206478875002439975737792666147199399141965887602L #rand
```

Michael generates his secret m and computes $2^m \pmod{p}$.

```
>>> m, mpow = dh_init(p)
>>> m
3533715181946048754332697897996834077726943413544L #rand
>>> mpow
3465862701820513569217254081716392362462604355024L #rand
```

At this point Nikita publicly announces `npow` and Michael publicly announces `mpow`. Nikita and Michael can now compute the shared secret key.

```
>>> dh_secret(p, n, mpow)
12931853037327712933053975672241775629043437267478L #rand
>>> dh_secret(p, m, npow)
12931853037327712933053975672241775629043437267478L #rand
```

7.3.2 Encoding Strings as Lists of Integers

In order to encrypt actual messages, instead of single integers, we define a function that converts an arbitrary string to a list of integers, and another that converts a list of integers back to a string.

A chosen plain text attack is an attack on a cryptosystem in which the attacker knows the unencrypted and encrypted versions of some messages, and can use that information to deduce something about future encrypted messages. For example, if a remote weather station encrypts the temperature and sends it encrypted, then an attacker who knows the temperature at the weather station might know how that temperature is encrypted. To reduce the chance that such attacks could weaken the cryptosystems implemented in this chapter, the function `str_to_numlist` randomizes its output, so the same string will usually be encoded differently, depending on when the function is called.

Listing 7.3.4 (String to Number List).

```

def str_to_numlist(s, bound):
    """
    Returns a sequence of integers between 0 and bound-1
    that encodes the string s. Randomization is included,
    so the same string is very likely to encode differently
    each time this function is called.
    Input:
        s -- a string
        bound -- an integer >= 256
    Output:
        list -- encoding of s as a list of integers
    Examples:
    >>> str_to_numlist("Run!", 1000)
    [82, 117, 110, 33]          #rand
    >>> str_to_numlist("TOP SECRET", 10**20)
    [4995371940984439512L, 92656709616492L] #rand
    """
    assert bound >= 256, "bound must be at least 256."
    n = int(log(bound) / log(256))      # (1)
    salt = min(int(n/8) + 1, n-1)      # (2)
    i = 0; v = []
    while i < len(s):                   # (3)
        c = 0; pow = 1
        for j in range(n):              # (4)
            if j < salt:
                c += randrange(1,256)*pow # (5)
            else:
                if i >= len(s): break
                c += ord(s[i])*pow       # (6)
                i += 1
        pow *= 256
    v.append(c)
    return v

```

In Listing 7.3.4, we view a string as a sequence of integers between 0 and 255. In line (1) we compute the number of characters that can be encoded in an integer up to `bound`; this is the block size. In line (2) we determine the number of random characters in each block. The while loop (3) iterates until we have encoded every character of the string in the list `v` of numbers. The for loop (4) iterates over the number of characters in a block, forming a number in base 256. The lower order digits are random (line 5), and the rest encode actual text of the message (line 6). The function `ord` used in line (6) converts a character to a number between 0 and 255. Listing 7.3.5

takes a sequence of integers output by `str_to_numlist` and returns the corresponding string.

Listing 7.3.5 (Number List to String).

```
def numlist_to_str(v, bound):
    """
    Returns the string that the sequence v of
    integers encodes.
    Input:
        v -- list of integers between 0 and bound-1
        bound -- an integer >= 256
    Output:
        str -- decoding of v as a string
    Examples:
    >>> print numlist_to_str([82, 117, 110, 33], 1000)
    Run!
    >>> x = str_to_numlist("TOP SECRET MESSAGE", 10**20)
    >>> print numlist_to_str(x, 10**20)
    TOP SECRET MESSAGE
    """
    assert bound >= 256, "bound must be at least 256."
    n = int(log(bound) / log(256))
    s = ""
    salt = min(int(n/8) + 1, n-1)
    for x in v:
        for j in range(n):
            y = x%256
            if y > 0 and j >= salt:
                s += chr(y)
            x /= 256
    return s
```

7.3.3 The RSA Cryptosystem

Listings 7.3.6–7.3.8 contain an implementation of the RSA cryptosystem.

Listing 7.3.6 (Initialize RSA).

```
def rsa_init(p, q):
    """
    Returns defining parameters (e, d, n) for the RSA
    cryptosystem defined by primes p and q. The
    primes p and q may be computed using the
    random_prime functions.
    Input:
```

```

    p -- a prime integer
    q -- a prime integer
Output:
    Let m be (p-1)*(q-1).
    e -- an encryption key, which is a randomly
        chosen integer between 2 and m-1
    d -- the inverse of e modulo eulerphi(p*q),
        as an integer between 2 and m-1
    n -- the product p*q.
Examples:
>>> p = random_prime(20); q = random_prime(20)
>>> print p, q
37999414403893878907L 25910385856444296437L #rand
>>> e, d, n = rsa_init(p, q)
>>> e
5 #rand
>>> d
787663591619054108576589014764921103213L #rand
>>> n
984579489523817635784646068716489554359L #rand
"""
m = (p-1)*(q-1)
e = 3
while gcd(e, m) != 1: e += 1
d = inversemod(e, m)
return e, d, p*q

```

In Listing 7.3.6, we compute $m = \varphi(pq)$, find a random encryption exponent that is coprime to m , and compute the inverse of the encryption exponent modulo m .

Listing 7.3.7 (Encrypt Using RSA).

```

def rsa_encrypt(plain_text, e, n):
    """
    Encrypt plain_text using the encrypt
    exponent e and modulus n.
    Input:
        plain_text -- arbitrary string
        e -- an integer, the encryption exponent
        n -- an integer, the modulus
    Output:
        str -- the encrypted cipher text
    Examples:
    >>> e = 1413636032234706267861856804566528506075
    >>> n = 2109029637390047474920932660992586706589
    """

```

```

>>> rsa_encrypt("Run Nikita!", e, n)
[78151883112572478169375308975376279129L] #rand
>>> rsa_encrypt("Run Nikita!", e, n)
[1136438061748322881798487546474756875373L] #rand
"""
plain = str_to_numlist(plain_text, n)
return [powermod(x, e, n) for x in plain]

```

Listing 7.3.7 defines `rsa_encrypt`, which converts a plain text message to a list of integers, then returns the e th powers of those integers modulo n , where e is the encryption exponent.

Listing 7.3.8 (Decrypt Using RSA).

```

def rsa_decrypt(cipher, d, n):
    """
    Decrypt the cipher_text using the decryption
    exponent d and modulus n.
    Input:
        cipher_text -- list of integers output
                    by rsa_encrypt
    Output:
        str -- the unencrypted plain text
    Examples:
    >>> d = 938164637865370078346033914094246201579
    >>> n = 2109029637390047474920932660992586706589
    >>> msg1 = [1071099761433836971832061585353925961069]
    >>> msg2 = [1336506586627416245118258421225335020977]
    >>> rsa_decrypt(msg1, d, n)
    'Run Nikita!'
    >>> rsa_decrypt(msg2, d, n)
    'Run Nikita!'
    """
    plain = [powermod(x, d, n) for x in cipher]
    return numlist_to_str(plain, n)

```

In Listing 7.3.8 we define `rsa_decrypt`, which raises each input integer to the power of d modulo n , then converts the resulting list of integers back to a string.

7.4 Quadratic Reciprocity

The main algorithmic ideas in Chapter 4 are computation of the Legendre symbol, and an algorithm for finding square roots in $\mathbf{Z}/p\mathbf{Z}$.

7.4.1 Computing the Legendre Symbol

Corollary 4.2.2 provides a simple and efficient algorithm to compute $\left(\frac{a}{p}\right)$, which we implement below.

Listing 7.4.1 (Legendre Symbol).

```
def legendre(a, p):
    """
    Returns the Legendre symbol a over p, where
    p is an odd prime.
    Input:
        a -- an integer
        p -- an odd prime (primality not checked)
    Output:
        int: -1 if a is not a square mod p,
            0 if gcd(a,p) is not 1
            1 if a is a square mod p.
    Examples:
    >>> legendre(2, 5)
    -1
    >>> legendre(3, 3)
    0
    >>> legendre(7, 2003)
    -1
    """
    assert p%2 == 1, "p must be an odd prime."
    b = powermod(a, (p-1)/2, p)
    if b == 1: return 1
    elif b == p-1: return -1
    return 0
```

7.4.2 Finding Square Roots

In this section we implement the algorithm of Section 4.5 for finding square roots of integers modulo p .

Listing 7.4.2 (Square Root Modulo).

```
def sqrtmod(a, p):
    """
    Returns a square root of a modulo p.
    Input:
        a -- an integer that is a perfect
            square modulo p (this is checked)
        p -- a prime
```

Output:

```

    int -- a square root of a, as an integer
           between 0 and p-1.
Examples:
>>> sqrtmod(4, 5)           # p == 1 (mod 4)
3                            #rand
>>> sqrtmod(13, 23)        # p == 3 (mod 4)
6                            #rand
>>> sqrtmod(997, 7304723089) # p == 1 (mod 4)
761044645L                  #rand
"""
a %= p
if p == 2: return a
assert legendre(a, p) == 1, "a must be a square mod p."
if p%4 == 3: return powermod(a, (p+1)/4, p)

def mul(x, y): # multiplication in R # (1)
    return ((x[0]*y[0] + a*y[1]*x[1]) % p, \
            (x[0]*y[1] + x[1]*y[0]) % p)
def pow(x, n): # exponentiation in R # (2)
    ans = (1,0)
    xpow = x
    while n != 0:
        if n%2 != 0: ans = mul(ans, xpow)
        xpow = mul(xpow, xpow)
        n /= 2
    return ans

while True:
    z = randrange(2,p)
    u, v = pow((1,z), (p-1)/2)
    if v != 0:
        vinv = inversemod(v, p)
        for x in [-u*vinv, (1-u)*vinv, (-1-u)*vinv]:
            if (x*x)%p == a: return x%p
        assert False, "Bug in sqrtmod."

```

The implementation above follows the algorithm in Section 4.5 closely. In lines (1) and (2) we define the functions `mul` and `pow` for multiplying two elements of the ring R of Section 4.5, where elements are represented as pairs of integers modulo p . Notice that Python supports definition of a function inside another function. Also, notice that the `pow` function defined starting at line (2) is very similar to `powermod` defined in Listing 7.2.5.

7.5 Continued Fractions

The main algorithms of Chapter 5 involve evaluating the value of a continued fraction as in Section 5.1, and computing continued fractions of floating point numbers as described in Section 5.2.1. We implement these algorithms, and also implement a simple function for writing a number as a sum of two squares.

The function in Listing 7.5.1 computes the partial convergents of a continued fraction as in Proposition 5.1.9.

Listing 7.5.1 (Convergents of Continued Fraction).

```
def convergents(v):
    """
    Returns the partial convergents of the continued
    fraction v.
    Input:
        v -- list of integers [a0, a1, a2, ..., am]
    Output:
        list -- list [(p0,q0), (p1,q1), ...]
              of pairs (pm,qm) such that the mth
              convergent of v is pm/qm.
    Examples:
    >>> convergents([1, 2])
    [(1, 1), (3, 2)]
    >>> convergents([3, 7, 15, 1, 292])
    [(3, 1), (22, 7), (333, 106), (355, 113), (103993, 33102)]
    """
    w = [(0,1), (1,0)]
    for n in range(len(v)):
        pn = v[n]*w[n+1][0] + w[n][0]
        qn = v[n]*w[n+1][1] + w[n][1]
        w.append((pn, qn))
    del w[0]; del w[0] # remove first entries of w
    return w
```

In Listing 7.5.2 we define `contfrac_rat`, which computes the continued fraction of an arbitrary rational number, using an algorithm derived from the proof of Proposition 5.1.9. Notice that we give the rational number as input by giving its numerator and denominator, since Python has no native type for rational numbers (it is not difficult to define such a type using Python classes, but we will not do so here, since in this chapter we do no nontrivial arithmetic with rational numbers). Notice that the definition of `contfrac_rat` below is almost the same as that of `gcd` in Listing 7.1.1, except that we keep track of the partial quotients.

Listing 7.5.2 (Continued Fraction of Rational).

```

def contfrac_rat( numer, denom ):
    """
    Returns the continued fraction of the rational
    number numer/denom.
    Input:
        numer -- an integer
        denom -- a positive integer coprime to num
    Output
        list -- the continued fraction [a0, a1, ..., am]
                of the rational number num/denom.
    Examples:
    >>> contfrac_rat(3, 2)
    [1, 2]
    >>> contfrac_rat(103993, 33102)
    [3, 7, 15, 1, 292]
    """
    assert denom > 0, "denom must be positive"
    a = numer; b = denom
    v = []
    while b != 0:
        v.append(a/b)
        (a, b) = (b, a%b)
    return v

```

Listing 7.5.3 contains an implementation of the continued fraction procedure from Section 5.2.1. Suppose x is a floating point number input to Python (i.e., a C double, i.e., a number possibly in scientific notation like on a hand calculator). We compute terms a_n of the continued fraction expansion of x along with the partial convergents p_n/q_n , until the difference $p_n/q_n - x$ is 0 to the precision of a Python float.

Listing 7.5.3 (Continued Fraction of Floating Point Number).

```

def contfrac_float(x):
    """
    Returns the continued fraction of the floating
    point number x, computed using the continued
    fraction procedure, and the sequence of partial
    convergents.
    Input:
        x -- a floating point number (decimal)
    Output:
        list -- the continued fraction [a0, a1, ...]
                obtained by applying the continued

```

```

        fraction procedure to x to the
        precision of this computer.
    list -- the list [(p0,q0), (p1,q1), ...]
           of pairs (pm,qm) such that the mth
           convergent of continued fraction
           is pm/qm.

Examples:
>>> v, w = contfrac_float(3.14159); print v
[3, 7, 15, 1, 25, 1, 7, 4]
>>> v, w = contfrac_float(2.718); print v
[2, 1, 2, 1, 1, 4, 1, 12]
>>> contfrac_float(0.3)
([0, 3, 2, 1], [(0, 1), (1, 3), (2, 7), (3, 10)])
"""

v = []
w = [(0,1), (1,0)] # keep track of convergents
start = x
while True:
    a = int(x) # (1)
    v.append(a)
    n = len(v)-1
    pn = v[n]*w[n+1][0] + w[n][0]
    qn = v[n]*w[n+1][1] + w[n][1]
    w.append((pn, qn))
    x -= a
    if abs(start - float(pn)/float(qn)) == 0: # (2)
        del w[0]; del w[0] # (3)
        return v, w
    x = 1/x

```

In line (1) we use the `int` command to coerce x into an `int`, which has the affect of computing $\lfloor x \rfloor$. In line (2) the command `float(qn)` results in a float, so that the quotient `float(pn)/float(qn)` is a float that approximates the rational number p_n/q_n . If we had instead written `pn/qn` in line (2), then `pn/qn` would always be an integer, which is not what we want. In line (3) we delete the first two entries of the list `w`, which are the partial convergents 0 and ∞ .

Remark 7.5.4. The Python module `gmpy` supports arbitrary precision arithmetic with floating point numbers. It does not come standard with Python, but can be downloaded from <http://gmpy.sourceforge.net/>. You could modify `contfrac_float` to use `gmpy`, and compute the continued fraction expansion of floating point numbers with many digits.

Listing 7.5.5 contains an implementation of an algorithm based on the proof of Theorem 5.6.1 for quickly writing a prime $p \equiv 1 \pmod{4}$ as a sum of two integer squares, even if the prime is huge (hundreds of digits).

Listing 7.5.5 (Write Prime as Sum of Two Squares).

```
def sum_of_two_squares(p):
    """
    Uses continued fractions to efficiently compute
    a representation of the prime p as a sum of
    two squares.  The prime p must be 1 modulo 4.
    Input:
        p -- a prime congruent 1 modulo 4.
    Output:
        integers a, b such that p is a*a + b*b
    Examples:
    >>> sum_of_two_squares(5)
    (1, 2)
    >>> sum_of_two_squares(389)
    (10, 17)
    >>> sum_of_two_squares(86295641057493119033)
    (789006548L, 9255976973L)
    """
    assert p%4 == 1, "p must be 1 modulo 4"
    r = sqrtmod(-1, p) # (1)
    v = contfrac_rat(-r, p) # (2)
    n = int(sqrt(p))
    for a, b in convergents(v): # (3)
        c = r*b + p*a # (4)
        if -n <= c and c <= n: return (abs(b),abs(c))
    assert False, "Bug in sum_of_two_squares." # (5)
```

The code in Listing 7.5.5 combines several functions defined earlier in this chapter. In line (1) we call the `sqrtmod` function of Listing 7.4.2 in the case $p \equiv 1 \pmod{4}$, which was the difficult case for finding square roots that uses a non-deterministic algorithm. In line (2) we use compute the continued fraction of the rational number $-r/p$, and in line (3) we iterate over the convergents of this continued fraction. When the c from line (4) satisfies the appropriate bound, we have found our sum-of-two-squares representation. The proof of Theorem 5.6.1 guarantees that there will be such a c and that line (5) will never be reached.

7.6 Elliptic Curves

The fundamental algorithms that we described in Chapter 6 are arithmetic of points on elliptic curve, the Pollard $(p - 1)$ and elliptic curve integer factorization methods, and the the ElGamal elliptic curve cryptosystem. In this section we implement each of these algorithms for elliptic curves over $\mathbf{Z}/p\mathbf{Z}$, and finish with an investigation of the associative law on an elliptic curve.

7.6.1 Arithmetic

Each elliptic curve function takes as first input an elliptic curve $y^2 = x^3 + ax + b$ over $\mathbf{Z}/p\mathbf{Z}$, which we represent by a triple (a, b, p) . We represent points on an elliptic curve in Python as a pair (x, y) , with $0 \leq x, y < p$ or as the string "Identity". The functions in Listings 7.6.1 and 7.6.2 implement the group law (Algorithm 6.1.1) and computation of mP for possibly large m .

Listing 7.6.1 (Elliptic Curve Group Law).

```
def ellcurve_add(E, P1, P2):
    """
    Returns the sum of P1 and P2 on the elliptic
    curve E.
    Input:
        E -- an elliptic curve over Z/pZ, given by a
            triple of integers (a, b, p), with p odd.
        P1 --a pair of integers (x, y) or the
            string "Identity".
        P2 -- same type as P1
    Output:
        R -- same type as P1
    Examples:
    >>> E = (1, 0, 7) # y**2 = x**3 + x over Z/7Z
    >>> P1 = (1, 3); P2 = (3, 3)
    >>> ellcurve_add(E, P1, P2)
    (3, 4)
    >>> ellcurve_add(E, P1, (1, 4))
    'Identity'
    >>> ellcurve_add(E, "Identity", P2)
    (3, 3)
    """
    a, b, p = E
    assert p > 2, "p must be odd."
    if P1 == "Identity": return P2
    if P2 == "Identity": return P1
```

```

x1, y1 = P1; x2, y2 = P2
x1 %= p; y1 %= p; x2 %= p; y2 %= p
if x1 == x2 and y1 == p-y2: return "Identity"
if P1 == P2:
    if y1 == 0: return "Identity"
    lam = (3*x1**2+a) * inversemod(2*y1,p)
else:
    lam = (y1 - y2) * inversemod(x1 - x2, p)
x3 = lam**2 - x1 - x2
y3 = -lam*x3 - y1 + lam*x1
return (x3%p, y3%p)

```

Listing 7.6.2 (Computing a Multiple of a Point).

```

def ellcurve_mul(E, m, P):
    """
    Returns the multiple m*P of the point P on
    the elliptic curve E.
    Input:
        E -- an elliptic curve over Z/pZ, given by a
            triple (a, b, p).
        m -- an integer
        P -- a pair of integers (x, y) or the
            string "Identity"
    Output:
        A pair of integers or the string "Identity".
    Examples:
    >>> E = (1, 0, 7)
    >>> P = (1, 3)
    >>> ellcurve_mul(E, 5, P)
    (1, 3)
    >>> ellcurve_mul(E, 9999, P)
    (1, 4)
    """
    assert m >= 0, "m must be nonnegative."
    power = P
    mP = "Identity"
    while m != 0:
        if m%2 != 0: mP = ellcurve_add(E, mP, power)
        power = ellcurve_add(E, power, power)
        m /= 2
    return mP

```

7.6.2 Integer Factorization

In Listing 7.6.3 we implement Algorithm 6.2.2 for computing the least common multiple of all integers up to some bound.

Listing 7.6.3 (Least Common Multiple of Numbers).

```
def lcm_to(B):
    """
    Returns the least common multiple of all
    integers up to B.
    Input:
        B -- an integer
    Output:
        an integer
    Examples:
    >>> lcm_to(5)
    60
    >>> lcm_to(20)
    232792560
    >>> lcm_to(100)
    69720375229712477164533808935312303556800L
    """
    ans = 1
    logB = log(B)
    for p in primes(B):
        ans *= p**int(logB/log(p))
    return ans
```

Next we implement Pollard's $p - 1$ method, as in Algorithm 6.2.3. We use only the bases $a = 2, 3$, but you could change this to use more bases by modifying the for loop in Listing 7.6.4.

Listing 7.6.4 (Pollard).

```
def pollard(N, m):
    """
    Use Pollard's (p-1)-method to try to find a
    nontrivial divisor of N.
    Input:
        N -- a positive integer
        m -- a positive integer, the least common
            multiple of the integers up to some
            bound, computed using lcm_to.
    Output:
        int -- an integer divisor of n
    Examples:
```

```

>>> pollard(5917, lcm_to(5))
61
>>> pollard(779167, lcm_to(5))
779167
>>> pollard(779167, lcm_to(15))
2003L
>>> pollard(187, lcm_to(15))
11
>>> n = random_prime(5)*random_prime(5)*random_prime(5)
>>> pollard(n, lcm_to(100))
315873129119929L      #rand
>>> pollard(n, lcm_to(1000))
3672986071L         #rand
"""
for a in [2, 3]:
    x = powermod(a, m, N) - 1
    g = gcd(x, N)
    if g != 1 and g != N:
        return g
return N

```

In order to implement the elliptic curve method and also in our upcoming elliptic curve cryptography implementation, it will be useful to define the function `randcurve` of Listing 7.6.5, which computes a random elliptic curve over $\mathbf{Z}/p\mathbf{Z}$ and a point on it. For simplicity, `randcurve` always returns a curve of the form $y^2 = x^3 + ax + 1$, and the point $P = (0, 1)$. As an exercise you could change this function to return a more general curve, and find a random point by choosing a random x , then incrementing it until $x^3 + ax + 1$ is a perfect square.

Listing 7.6.5 (Random Elliptic Curve).

```

def randcurve(p):
    """
    Construct a somewhat random elliptic curve
    over  $\mathbf{Z}/p\mathbf{Z}$  and a random point on that curve.
    Input:
        p -- a positive integer
    Output:
        tuple -- a triple  $E = (a, b, p)$ 
        P -- a tuple  $(x, y)$  on  $E$ 
    Examples:
    >>> p = random_prime(20); p
    17758176404715800329L      #rand
    >>> E, P = randcurve(p)
    >>> print E

```

```
(15299007531923218813L, 1, 17758176404715800329L) #rand
>>> print P
(0, 1)
"""
assert p > 2, "p must be > 2."
a = randrange(p)
while gcd(4*a**3 + 27, p) != 1:
    a = randrange(p)
return (a, 1, p), (0,1)
```

In Listing 7.6.6, we implement the elliptic curve factorization method.

Listing 7.6.6 (Elliptic Curve Factorization Method).

```
def elliptic_curve_method(N, m, tries=5):
    """
    Use the elliptic curve method to try to find a
    nontrivial divisor of N.
    Input:
        N -- a positive integer
        m -- a positive integer, the least common
            multiple of the integers up to some
            bound, computed using lcm_to.
        tries -- a positive integer, the number of
            different elliptic curves to try
    Output:
        int -- a divisor of n
    Examples:
    >>> elliptic_curve_method(5959, lcm_to(20))
    59L #rand
    >>> elliptic_curve_method(10007*20011, lcm_to(100))
    10007L #rand
    >>> p = random_prime(9); q = random_prime(9)
    >>> n = p*q; n
    117775675640754751L #rand
    >>> elliptic_curve_method(n, lcm_to(100))
    117775675640754751L #rand
    >>> elliptic_curve_method(n, lcm_to(500))
    117775675640754751L #rand
    """
    for _ in range(tries): # (1)
        E, P = randcurve(N) # (2)
        try: # (3)
            Q = ellcurve_mul(E, m, P) # (4)
        except ZeroDivisionError, x: # (5)
            g = gcd(x[0], N) # (6)
```

```

        if g != 1 or g != N: return g      # (7)
    return N

```

In line (1) the underscore means that the for loop iterates `tries` times, but that no variable is “wasted” recording which iteration we are in. In line (2) we compute a random elliptic curve and point on it. The elliptic curve method works by assuming N is prime, doing a certain computation, on an elliptic curve over $\mathbf{Z}/N\mathbf{Z}$, and detecting if something goes wrong. Python contains a mechanism called exception handling, which leads to a very simple implementation of the elliptic curve method, that uses the elliptic curve functions that we have already defined. The `try` statement in line (3) means that the code in line (4) should be executed, and if the `ZeroDivisionError` exception is raised, then the code in lines (6) and (7) should be executed, but not otherwise. Recall that in the definition of `inversemod` from Listing 7.2.2, when the inverse could not be computed, we raised a `ZeroDivisionError`, which included the offending pair (a, n) . Thus when computing mP , if at any point it is not possible to invert a number modulo N , we jump to line (6), compute a gcd with N , and hopefully split N .

7.6.3 ElGamal Elliptic Curve Cryptosystem

Listing 7.6.7 defines a function that creates an ElGamal cryptosystem over $\mathbf{Z}/p\mathbf{Z}$. This is simplified from what one would do in actual practice. One would use a more general random elliptic curve and point than we do in `elgamal_init`, and count the number of points on it using the Schoof-Elkies-Atkin algorithm, then repeat this procedure if the number of points is not a prime or a prime times a small number, or is p , $p - 1$, or $p + 1$. Since implementing Schoof-Elkies-Atkin is beyond the scope of this book, we have not included this crucial step.

Listing 7.6.7 (Initialize ElGamal).

```

def elgamal_init(p):
    """
    Constructs an ElGamal cryptosystem over  $\mathbf{Z}/p\mathbf{Z}$ , by
    choosing a random elliptic curve  $E$  over  $\mathbf{Z}/p\mathbf{Z}$ , a
    point  $B$  in  $E(\mathbf{Z}/p\mathbf{Z})$ , and a random integer  $n$ . This
    function returns the public key as a 4-tuple
     $(E, B, n*B)$  and the private key  $n$ .
    Input:
        p -- a prime number
    Output:
        tuple -- the public key as a 3-tuple
                 $(E, B, n*B)$ , where  $E = (a, b, p)$  is an

```

```

        elliptic curve over  $Z/pZ$ ,  $B = (x, y)$  is
        a point on  $E$ , and  $n*B = (x',y')$  is
        the sum of  $B$  with itself  $n$  times.
    int -- the private key, which is the pair  $(E, n)$ 
Examples:
>>> p = random_prime(20); p
17758176404715800329L    #rand
>>> public, private = elgamal_init(p)
>>> print "E =", public[0]
E = (15299007531923218813L, 1, 17758176404715800329L)    #rand
>>> print "B =", public[1]
B = (0, 1)
>>> print "nB =", public[2]
nB = (5619048157825840473L, 151469105238517573L)    #rand
>>> print "n =", private[1]
n = 12608319787599446459    #rand
"""
E, B = randcurve(p)
n = randrange(2,p)
nB = ellcurve_mul(E, n, B)
return (E, B, nB), (E, n)

```

In Listing 7.6.8 we define `elgamal_encrypt`, which encrypts a message using the ElGamal cryptosystem on an elliptic curve.

Listing 7.6.8 (Encrypt Using ElGamal).

```

def elgamal_encrypt(plain_text, public_key):
    """
    Encrypt a message using the ElGamal cryptosystem
    with given public_key = (E, B, n*B).
    Input:
        plain_text -- a string
        public_key -- a triple (E, B, n*B), as output
                    by elgamal_init.
    Output:
        list -- a list of pairs of points on E that
                represent the encrypted message
    Examples:
    >>> public, private = elgamal_init(random_prime(20))
    >>> elgamal_encrypt("RUN", public)
    [(6004308617723068486L, 15578511190582849677L), \ #rand
     (7064405129585539806L, 8318592816457841619L))]    #rand
    """
    E, B, nB = public_key
    a, b, p = E

```

```

assert p > 10000, "p must be at least 10000."
v = [1000*x for x in \
      str_to_numlist(plain_text, p/1000)]      # (1)
cipher = []
for x in v:
    while not legendre(x**3+a*x+b, p)==1:      # (2)
        x = (x+1)%p
    y = sqrtmod(x**3+a*x+b, p)                 # (3)
    P = (x,y)
    r = randrange(1,p)
    encrypted = (ellcurve_mul(E, r, B), \
                 ellcurve_add(E, P, ellcurve_mul(E,r,nB)))
    cipher.append(encrypted)
return cipher

```

In line (1) we encode the plain text message as a sequence of integers that are all 0 modulo 1000. It would be nice if the integers returned by `str_to_numlist` were the x -coordinates of points on the elliptic curve E , but typically only half the $x \in \mathbf{Z}/p\mathbf{Z}$ will actually be x -coordinates of points on E . Thus we multiply the integers returned by `str_to_numlist`, and 1 to them in line (2) until they are the x -coordinates of points on E . Note that since half the elements of $\mathbf{Z}/p\mathbf{Z}$ are perfect squares, we should only have to add 1 very few times to obtain a perfect square. The rest of the Listing 7.6.8 is a straightforward implementation of ElGamal as described in Section 6.3.2.

In Listing 7.6.9 we give the corresponding decryption routine, which takes into account the way we encoded integers as points on E .

Listing 7.6.9 (Decrypt Using ElGamal).

```

def elgamal_decrypt(cipher_text, private_key):
    """
    Encrypt a message using the ElGamal cryptosystem
    with given public_key = (E, B, n*B).
    Input:
        cipher_text -- list of pairs of points on E output
                       by elgamal_encrypt.
    Output:
        str -- the unencrypted plain text
    Examples:
    >>> public, private = elgamal_init(random_prime(20))
    >>> v = elgamal_encrypt("TOP SECRET MESSAGE!", public)
    >>> print elgamal_decrypt(v, private)
    TOP SECRET MESSAGE!
    """
    E, n = private_key

```

```

p = E[2]
plain = []
for rB, P_plus_rnB in cipher_text:
    nrB = ellcurve_mul(E, n, rB)
    minus_nrB = (nrB[0], -nrB[1])
    P = ellcurve_add(E, minus_nrB, P_plus_rnB)
    plain.append(P[0]/1000)
return numlist_to_str(plain, p/1000)

```

7.6.4 Associativity of the Group Law

Theorem 7.6.10. *Suppose that P_1 , P_2 , and P_3 are points on an elliptic curve E over a field K . If the additions in $(P_1 + P_2) + P_3$ and $P_1 + (P_2 + P_3)$ are all given by step 4 of Algorithm 6.1.1 with $\lambda = (y_1 - y_2)/(x_1 - x_2)$, then the x -coordinates of $(P_1 + P_2) + P_3$ and $P_1 + (P_2 + P_3)$ are equal.*

One could give similar proofs in the cases not covered by the theorem (and hence obtain full associative law), but we will be content with this one case, since the proofs of the other cases are similar.

Proof. We combine some algebra with a computer computation. Write $P_i = (x_i, y_i)$ for $i = 1, 2, 3$. For any i, j , set $\lambda_{ij} = (y_i - y_j)/(x_i - x_j)$ and $\nu_{ij} = y_i - \lambda_{ij}x_i$. Then letting $P_4 = (x_4, y_4) = P_1 + P_2$ and $P_5 = (x_5, y_5) = P_2 + P_3$, we have by our hypothesis on the P_i and Algorithm 6.1.1 that

$$P_4 = (\lambda_{12}^2 - x_1 - x_2, -\lambda_{12}x_4 - \nu_{12})$$

and

$$P_5 = (\lambda_{23}^2 - x_2 - x_3, -\lambda_{23}x_5 - \nu_{23}).$$

We use the formula from Algorithm 6.1.1 and a computation to verify that the x -coordinates of $P_3 + P_4$ are the same. We have

$$x(P_3 + P_4) = \lambda_{34}^2 - x_3 - x_4 = \frac{(y_3 - y_4)^2}{(x_3 - x_4)^2} - (x_3 + x_4)$$

and

$$x(P_1 + P_5) = \lambda_{15}^2 - x_1 - x_5 = \frac{(y_1 - y_5)^2}{(x_1 - x_5)^2} - (x_1 + x_5).$$

Equating and multiplying through by denominators, we see that to verify that the x coordinates are the same we must show that the following equality holds:

$$\begin{aligned} (x_1 - x_5)^2((y_3 - y_4)^2 - (x_3 + x_4)(x_3 - x_4)^2) \\ = (x_3 - x_4)^2((y_1 - y_5)^2 - (x_1 + x_5)(x_1 - x_5)^2) \end{aligned}$$

To prove that this equality holds for all points on elliptic curves, it suffices to show that it holds in the fraction field F of the ring

$$R = \mathbf{Z}[x_1, x_2, x_3, y_1, y_2, y_3, a, b]/I,$$

where $x_1, x_2, x_3, y_1, y_2, y_3, a, b$ are indeterminates, and I is the ideal generated by $y_i^2 - (x_i^3 + ax_i + b)$ for $i = 1, 2, 3$. Note that we expressed x_4 and x_5 in terms of the x_i and y_i for $i \leq 3$, so we may view x_4 and x_5 as elements of F . Listing 7.6.12 contains a program that implements arithmetic in the ring R , the field F , and verifies that the equality needed does hold. For a discussion of how the program works, see the remarks after Listing 7.6.12. When run this program outputs true, which shows that the x -coordinates are correct, as claimed. \square

Remark 7.6.11. If the program below were implemented in a compiled language (e.g., C++) it would be longer, but it would also likely be faster, as Python is not a good language from an efficiency point of view for implementing basic arithmetic in a ring.

Listing 7.6.12 (Verify Associativity of Elliptic Curve Group Law).

```
# The variable order is x1, x2, x3, y1, y2, y3, a, b
class Poly:
    def __init__(self, d):
        self.v = dict(d)
    def __cmp__(self, other):
        self.normalize(); other.normalize()
        if self.v == other.v: return 0
        return -1

    def __add__(self, other):
        w = Poly(self.v)
        for m in other.monomials():
            w[m] += other[m]
        return w
    def __sub__(self, other):
        w = Poly(self.v)
        for m in other.monomials():
            w[m] -= other[m]
        return w
    def __mul__(self, other):
        if len(self.v) == 0 or len(other.v) == 0:
            return Poly([])
        m1 = self.monomials(); m2 = other.monomials()
        r = Poly([])
        for m1 in self.monomials():
```

```

        for m2 in other.monomials():
            z = [m1[i] + m2[i] for i in range(8)]
            r[z] += self[m1]*other[m2]
    return r
def __neg__(self):
    v = {}
    for m in self.v.keys():
        v[m] = -self.v[m]
    return Poly(v)
def __div__(self, other):
    return Frac(self, other)

def __getitem__(self, m):
    m = tuple(m)
    if not self.v.has_key(m): self.v[m] = 0
    return self.v[m]
def __setitem__(self, m, c):
    self.v[tuple(m)] = c
def __delitem__(self, m):
    del self.v[tuple(m)]

def monomials(self):
    return self.v.keys()
def normalize(self):
    while True:
        finished = True
        for m in self.monomials():
            if self[m] == 0:
                del self[m]
                continue
            for i in range(3):
                if m[3+i] >= 2:
                    finished = False
                    nx0 = list(m); nx0[3+i] -= 2;
                    nx0[7] += 1
                    nx1 = list(m); nx1[3+i] -= 2;
                    nx1[i] += 1; nx1[6] += 1
                    nx3 = list(m); nx3[3+i] -= 2;
                    nx3[i] += 3
                    c = self[m]
                    del self[m]
                    self[nx0] += c;
                    self[nx1] += c;
                    self[nx3] += c
        # end for

```

```

        # end for
        if finished: return
    # end while

one = Poly({(0,0,0,0,0,0,0,0):1}) # (9)

class Frac: # (10)
    def __init__(self, num, denom=one):
        self.num = num; self.denom = denom
    def __cmp__(self, other): # (11)
        if self.num * other.denom == self.denom * other.num:
            return 0
        return -1

    def __add__(self, other): # (12)
        return Frac(self.num*other.denom + \
                    self.denom*other.num,
                    self.denom*other.denom)
    def __sub__(self, other):
        return Frac(self.num*other.denom - \
                    self.denom*other.num,
                    self.denom*other.denom)
    def __mul__(self, other):
        return Frac(self.num*other.num, \
                    self.denom*other.denom)
    def __div__(self, other):
        return Frac(self.num*other.denom, \
                    self.denom*other.num)
    def __neg__(self):
        return Frac(-self.num,self.denom)

def var(i): # (14)
    v = [0,0,0,0,0,0,0,0]; v[i]=1;
    return Frac(Poly({tuple(v):1}))

def prove_associative(): # (15)
    x1 = var(0); x2 = var(1); x3 = var(2)
    y1 = var(3); y2 = var(4); y3 = var(5)
    a = var(6); b = var(7)

    lambda12 = (y1 - y2)/(x1 - x2)
    x4 = lambda12*lambda12 - x1 - x2
    nu12 = y1 - lambda12*x1
    y4 = -lambda12*x4 - nu12
    lambda23 = (y2 - y3)/(x2 - x3)

```

```

x5      = lambda23*lambda23 - x2 - x3
nu23    = y2 - lambda23*x2
y5      = -lambda23*x5 - nu23
s1 = (x1 - x5)*(x1 - x5)*((y3 - y4)*(y3 - y4) \
      - (x3 + x4)*(x3 - x4)*(x3 - x4))
s2 = (x3 - x4)*(x3 - x4)*((y1 - y5)*(y1 - y5) \
      - (x1 + x5)*(x1 - x5)*(x1 - x5))
print "Associative?"
print s1 == s2                                # (17)

```

A class is a new Python datatype, and Listing 7.6.12 makes use of them. In line (1) we begin the definition of a class called `Poly`, which will implement arithmetic in the ring R of the proof of the Theorem 7.6.10. We view an element of R as a set of pairs (c_i, m_i) , where $c_i \in \mathbf{Z}$ and m_i is a monomial in variables $x_1, x_2, x_3, y_1, y_2, y_3, a, b$. In Python we represent a monomial by an 8-tuple of integers, for example, we represent $x_1 x_2 x_3^5 y_1 y_2 y_3 a^3 b$ by $(1, 1, 5, 1, 1, 1, 3, 1)$. Python has a data structure called a dictionary, which is a mapping from an (almost) arbitrary finite set of Python objects to any Python objects; in Python we represent an element $f \in R$ as a dictionary viewed as a mapping from the nonzero monomials in f to the coefficients of those monomials.

In line (2) we initialize a `Poly` from a dictionary `d`. In line (3) we compare two `Poly`'s for equality after normalizing them in line (4) by applying the relations $y_i^2 = x_i^3 + ax_i + b$. In the block beginning at line (5), we define the basic arithmetic operations on elements of R . In the block beginning at line (6), we define some functions that are useful for accessing coefficients of elements of R . The `monomials` function of line (7) returns the monomials in an element of R . The `normalize` function of line (8) reduces elements of R modulo the relations $y_i^2 = x_i^3 + ax_i + b$. A normalized element has the property that no exponent of any y_i is bigger than 1. In line (9) we define the element $1 \in R$, which will be useful later.

Line (10) begins the definition of the `Frac` class, which represents elements of the fraction field F of R . The comparison function of line (11) decides whether two elements of F are equal by cross multiplying and checking equality. The arithmetic in the block starting with line (12) is the standard arithmetic in any fraction field.

The `var` function of line (14) defines a function such that `var(i)` returns the i th element of the following list: $x_1, x_2, x_3, y_1, y_2, y_3, a, b$, where x_1 is the 0th element, etc.

Finally the function `prove_associative`, which starts at line (15), is where all the action is. Here we define the variables x_i, y_i, a, b , and input the formulas appearing in the proof of Theorem 7.6.10. Running `prove_associative` completes the computation for Theorem 7.6.10.

```
>>> prove_associative()
```

Associative?
True

7.7 Exercises

- 7.1 (a) Let $y = 10000$. Compute $\pi(y) = \#\{\text{primes } p \leq y\}$.
 (b) The prime number theorem implies $\pi(x)$ is asymptotic to $\frac{x}{\log(x)}$.
 How close is $\pi(y)$ to $y/\log(y)$, where y is as in (a)?
- 7.2 Design an analogue of the `trial_division` function of Listing 7.1.3 that uses a sequence `dif` of length longer than 8, so it skips integers not coprime to 210 (see the discussion after Listing 7.1.3).
- 7.3 Compute the last two digits of 3^{45} .
- 7.4 Find the integer a such that $0 \leq a < 113$ and

$$102^{70} + 1 \equiv a^{37} \pmod{113}.$$

- 7.5 Find the proportion of primes $p < 1000$ such that 2 is a primitive root modulo p .
- 7.6 Find a prime p such that the smallest primitive root modulo p is 37.
- 7.7 You and Nikita wish to agree on a secret key using the Diffie-Hellman key exchange. Nikita announces that $p = 3793$ and $g = 7$. Nikita secretly chooses a number $n < p$ and tells you that $g^n \equiv 454 \pmod{p}$. You choose the random number $m = 1208$. What is the secret key?
- 7.8 You see Michael and Nikita agree on a secret key using the Diffie-Hellman key exchange. Michael and Nikita choose $p = 97$ and $g = 5$. Nikita chooses a random number n and tells Michael that $g^n \equiv 3 \pmod{97}$, and Michael chooses a random number m and tells Nikita that $g^m \equiv 7 \pmod{97}$. Brute force crack their code: What is the secret key that Nikita and Michael agree upon? What is n ? What is m ?
- 7.9 In this problem, you will “crack” an RSA cryptosystem. What is the secret decoding number d for the RSA cryptosystem with public key $(n, e) = (5352381469067, 4240501142039)$?
- 7.10 Nikita creates an RSA cryptosystem with public key

$$(n, e) = (1433811615146881, 329222149569169).$$

In the following two problems, show the steps you take to factor n . (Don't simply factor n directly using a computer.)

- (a) Somehow you discover that $d = 116439879930113$. Show how to use the probabilistic algorithm of Section 3.3.3 to use d to factor n .
- (b) In part (a) you found that the factors p and q of n are very close. Show how to use the Fermat factorization method of Section 3.3.2 to factor n .
- 7.11 Compute the p_n and q_n for the continued fractions $[-3, 1, 1, 1, 1, 3]$ and $[0, 2, 4, 1, 8, 2]$. Check that the propositions in Section 5.1.1 hold.
- 7.12 A theorem of Hurwitz (1891) asserts that for any irrational number x , there exists infinitely many rational numbers a/b such that

$$\left| x - \frac{a}{b} \right| < \frac{1}{\sqrt{5}b^2}.$$

Take $x = e$, and obtain four rational numbers that satisfy this inequality.

- 7.13 Which of the following numbers is a sum of two squares? Express those that are as a sum of two squares.

−389, 12345, 729, 1729, 5809961789

- 7.14 (a) Show that the set of numbers $59 + 1 \pm s$ for $s \leq 15$ contains 14 numbers that are B -power smooth for $B = 20$.
- (b) Find the proportion of primes p in the interval from 10^{12} and $10^{12} + 1000$ such that $p - 1$ is $B = 10^5$ power-smooth.

Answers and Hints

1. Prime Numbers

2. They are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.
3. Emulate the proof of Proposition 1.2.4.

2. The Ring of Integers Modulo n

1. They are 5, 13, 3, and 8.
2. For example $x = 22$, $y = -39$.
3. Hint: Use the binomial theorem and prove that if $r \geq 1$ then p divides $\binom{p}{r}$.
6. For example, $S_1 = \{0, 1, 2, 3, 4, 5, 6\}$, $S_2 = \{1, 3, 5, 7, 11, 13, 23\}$, $S_3 = \{0, 2, 4, 6, 8, 10, 12\}$, and $S_4 = \{2, 3, 5, 7, 11, 13, 29\}$. In each we find S_i by listing the first seven numbers satisfying the i th condition, then adjusted the last number if necessary so that the reductions would be distinct modulo 7.
7. An integer is divisible by 5 if and only if the last digits is 0 or 5. An integer is divisible by 9 if and only if the sum of the digits is divisible by 9. An integer is divisible by 11 if and only if the alternating sum of the digits is divisible by 11.
8. Hint for part (a): Use the divisibility rule you found in Exercise 1.7.

9. 71

10. 8

11. As explained on page 22, we know that $\mathbf{Z}/n\mathbf{Z}$ is a ring for any n . Thus to show that $\mathbf{Z}/p\mathbf{Z}$ is a field it suffices to show that every nonzero element $\bar{a} \in \mathbf{Z}/p\mathbf{Z}$ has an inverse. Lift a to an element $a \in \mathbf{Z}$, and set $b = p$ in Proposition 2.3.1. Because p is prime, $\gcd(a, p) = 1$, so there exists x, y such that $ax + py = 1$. Reducing this equality modulo p proves that \bar{a} has an inverse $x \pmod{p}$. Alternative one could argue just like after Definition 2.1.10 that $\bar{a}^m = 1$ for some m , so some power of \bar{a} is the inverse of \bar{a} .

12. 302

14. Only for $n = 1, 2$. If $n > 2$, then n is either divisible by an odd prime p or 4. If $4 \mid n$, then $2^e - 2^{e-1}$ divides $\varphi(n)$ for some $e \geq 2$, so $\varphi(n)$ is even. If an odd p divides n , then the even number $p^e - p^{e-1}$ divides $\varphi(n)$ for some $e \geq 1$.

15. The map ψ is a homomorphism since both reduction maps

$$\mathbf{Z}/mn\mathbf{Z} \rightarrow \mathbf{Z}/m\mathbf{Z} \quad \text{and} \quad \mathbf{Z}/mn\mathbf{Z} \rightarrow \mathbf{Z}/n\mathbf{Z}$$

are homomorphisms. It is injective because if $a \in \mathbf{Z}$ is such that $\psi(a) = 0$, then $m \mid a$ and $n \mid a$, so $mn \mid a$ (since m and n are coprime), so $a \equiv 0 \pmod{mn}$. The cardinality of $\mathbf{Z}/mn\mathbf{Z}$ is mn and the cardinality of the product $\mathbf{Z}/m\mathbf{Z} \times \mathbf{Z}/n\mathbf{Z}$ is also mn , so ψ must be an isomorphism. The units $(\mathbf{Z}/mn\mathbf{Z})^*$ are thus in bijection with the units $(\mathbf{Z}/m\mathbf{Z})^* \times (\mathbf{Z}/n\mathbf{Z})^*$.

For the second part of the exercise, let $g = \gcd(m, n)$ and set $a = mn/g$. Then $a \not\equiv 0 \pmod{mn}$, but $m \mid a$ and $n \mid a$, so $a \in \ker(\psi)$.

16. We express the question as a system of linear equations modulo various numbers, and use the Chinese remainder theorem. Let x be the number of books. The problem asserts that

$$x \equiv 6 \pmod{7}$$

$$x \equiv 2 \pmod{6}$$

$$x \equiv 1 \pmod{5}$$

$$x \equiv 0 \pmod{4}$$

Applying CRT to the first pair of equations we find that $x \equiv 20 \pmod{42}$. Applying CRT to this equation and the third we find that $x \equiv 146 \pmod{210}$. Since 146 is not divisible by 4, we add multiples of 210 to 146 until we find the first x that is divisible by 4. The first multiple works, and we find that the aspiring mathematicians have 356 math books.

17. Note that $p = 3$ works, since $11 = 3^2 + 2$ is prime. Now suppose $p \neq 3$ is any prime such that p and $p^2 + 2$ are both prime. We must have $p \equiv 1 \pmod{3}$ or $p \equiv 2 \pmod{3}$. Then $p^2 \equiv 1 \pmod{3}$, so $p^2 + 2 \equiv 0 \pmod{3}$, which contradicts the fact that $p^2 + 2$ is prime.
18. For (a) $n = 1, 2$, see solution to Exercise 2.14. For (b), yes there are many such examples. For example, $m = 2, n = 4$.
19. By repeated application of multiplicativity and Equation (2.2.2) on page 29, we see that if $n = \prod_i p_i^{e_i}$ is the prime factorization of n , then

$$\varphi(n) = \prod_i (p_i^{e_i} - p_i^{e_i-1}) = \prod_i p_i^{e_i-1} \cdot \prod_i (p_i - 1).$$

20. 1, 6, 29, 34
21. Let $g = \gcd(12n+1, 30n+2)$. Then $g \mid 30n+2-2 \cdot (12n+1) = 6n$. For the same reason g also divides $12n+1-2 \cdot (6n) = 1$, so $g = 1$, as claimed.
24. There is no primitive root modulo 8, since $(\mathbf{Z}/8\mathbf{Z})^*$ has order 4, but every element of $(\mathbf{Z}/8\mathbf{Z})^*$ has order 2. Prove that if ζ is a primitive root modulo 2^n , for $n \geq 3$, then the reduction of $\zeta \pmod{8}$ is a primitive root, a contradiction.
25. 2 is a primitive root modulo 125.
26. Let $\prod_{i=1}^m p_i^{e_i}$ be the prime factorization of n . Slightly generalizing Exercise 15 we see that

$$(\mathbf{Z}/n\mathbf{Z})^* \cong \prod (\mathbf{Z}/p_i^{e_i}\mathbf{Z})^*.$$

Thus $(\mathbf{Z}/n\mathbf{Z})^*$ is cyclic if and only if the product $(\mathbf{Z}/p_i^{e_i}\mathbf{Z})^*$ is cyclic. If $8 \mid n$, then there is no chance $(\mathbf{Z}/n\mathbf{Z})^*$ is cyclic, so assume $8 \nmid n$. Then by Exercise 2.25 each group $(\mathbf{Z}/p_i^{e_i}\mathbf{Z})^*$ is itself cyclic. A product of cyclic groups is cyclic if and only if the orders of the factors in the product are coprime (this follows from Exercise 2.15). Thus $(\mathbf{Z}/n\mathbf{Z})^*$ is cyclic if and only if the numbers $p_i(p_i - 1)$, for $i = 1, \dots, m$ are pairwise coprime. Since $p_i - 1$ is even, there can be at most one odd prime in the factorization of n , and we see that $(\mathbf{Z}/n\mathbf{Z})^*$ is cyclic if and only if n is an odd prime power, twice an odd prime power, or $n = 4$.

3. Public-Key Cryptography

1. The best case is that each letter is A. Then the question is to find the largest n such that $1 + 27 + \dots + 27^n \leq 10^{20}$. By computing

$\log_{27}(10^{20})$, we see that $27^{13} < 10^{20}$ and $27^{14} > 10^{20}$. Thus $n \leq 13$, and since $1 + 27 + \cdots + 27^{n-1} < 27^n$, and $2 \cdot 27^{13} < 10^{20}$, it follows that $n = 13$.

2. This is not secure, since it is just equivalent to a “Caesar Cipher”, that is a permutation of the letters of the alphabet, which is well-known to be easily broken using a frequency analysis.
3. If we can compute the polynomial

$$f = (x-p)(x-q)(x-r) = x^3 - (p+q+r)x^2 + (pq+pr+qr)x - pqr,$$

then we can factor n by finding the roots of f , e.g., using Newton’s method (or Cardona’s formula for the roots of a cubic). Because p, q, r , are distinct odd primes we have

$$\varphi(n) = (p-1)(q-1)(r-1) = pqr - (pq+pr+qr) + p+q+r,$$

and

$$\sigma(n) = 1 + (p+q+r) + (pq+pr+qr) + pqr.$$

Since we know n , $\varphi(n)$, and $\sigma(n)$, we know

$$\begin{aligned} \sigma(n) - 1 - n &= (p+q+r) + (pq+pr+qr), \quad \text{and} \\ \varphi(n) - n &= (p+q+r) - (pq+pr+qr). \end{aligned}$$

We can thus compute both $p+q+r$ and $pq+pr+qr$, hence deduce f and find p, q, r .

4. Quadratic Reciprocity

1. They are all 1, -1 , 0, and 1.
2. By Proposition 4.3.3 the value of $\left(\frac{3}{p}\right)$ depends only on the reduction $\pm p \pmod{12}$. List enough primes p such that the $\pm p$ reduce to 1, 5, 7, 11 modulo 12 and verify that the asserted formula holds for each of them.
6. Since $p = 2^{13} - 1$ is prime there are either two solutions or no solutions to $x^2 \equiv 5 \pmod{p}$, and we can decide which using quadratic reciprocity. We have

$$\left(\frac{5}{p}\right) = (-1)^{(p-1)/2 \cdot (5-1)/2} \left(\frac{p}{5}\right) = \left(\frac{p}{5}\right),$$

so there are two solutions if and only if $p = 2^{13} - 1$ is $\pm 1 \pmod{5}$. In fact $p \equiv 1 \pmod{5}$, so there are two solutions.

7. We have $4^{48} = 2^{96}$. By Fermat’s Little Theorem $2^{96} = 1$, so $x = 1$.

8. For (a) take $a = 19$ and $n = 20$. We found this example using the Chinese remainder theorem applied to $4 \pmod{5}$ and $3 \pmod{4}$, and used that $\left(\frac{19}{20}\right) = \left(\frac{19}{5}\right) \cdot \left(\frac{19}{4}\right) = (-1)(-1) = 1$, yet 19 is not a square modulo either 5 or 4, so is certainly not a square modulo 20.
9. Hint: First reduce to the case that $6k - 1$ is prime, by using that if p and q are primes not of the form $6k - 1$, then neither is their product. If $p = 6k - 1$ divides $n^2 + n + 1$, it divides $4n^2 + 4n + 4 = (2n + 1)^2 + 3$, so -3 is a quadratic residue modulo p . Now use quadratic reciprocity to show that -3 is not a quadratic residue modulo p .

5. Continued Fractions

9. Suppose $n = x^2 + y^2$, with $x, y \in \mathbf{Q}$. Let d be such that $dx, dy \in \mathbf{Z}$. Then $d^2n = (dx)^2 + (dy)^2$ is a sum of two integer squares, so by Theorem 5.6.1 if $p \mid d^2n$ and $p \equiv 3 \pmod{4}$, then $\text{ord}_p(d^2n)$ is even. We have $\text{ord}_p(d^2n)$ is even if and only if $\text{ord}_p(n)$ is even, so Theorem 5.6.1 implies that n is also a sum of two squares.
11. The squares modulo 8 are 0, 1, 4, so a sum of two squares reduces modulo 8 to one of 0, 1, 2, 4 or 5. Four consecutive integers that are sums of squares would reduce to four consecutive integers in the set $\{0, 1, 2, 4, 5\}$, which is impossible.

6. Elliptic Curves

1. The second point of intersection is $(129/100, 383/1000)$.
2. The group is cyclic of order 9, generated by $(4, 2)$. The elements of $E(K)$ are
- $$\{\mathcal{O}, (4, 2), (3, 4), (2, 4), (0, 4), (0, 1), (2, 1), (3, 1), (4, 3)\}.$$
3. In part (a) the pattern is that $N_p = p + 1$. For part (b), a hint is that when $p \equiv 2 \pmod{3}$, the map $x \mapsto x^3$ on $(\mathbf{Z}/p\mathbf{Z})^*$ is an automorphism, so $x \mapsto x^3 + 1$ is a bijection. Now use what you learned about squares in $\mathbf{Z}/p\mathbf{Z}$ from Chapter 4.
4. For all sufficiently large real x , the equation $y^2 = x^3 + ax + b$ has a real solution y . Thus the group $E(\mathbf{R})$ is not countable, since \mathbf{R} is not countable. But any finitely generated group is countable.
5. In a course on abstract algebra one often proves the nontrivial fact that every subgroup of a finitely generated abelian group is finitely generated. In particular, the torsion subgroup G_{tor} is finitely generated. However, a finitely generated abelian torsion group is finite.

6. Hint: Multiply both sides of $y^2 = x^3 + ax + b$ by a power of a common denominator, and “absorb” powers into x and y .
7. Hint: see Exercise 4.5.

7. Computational Number Theory

All code below assume that the Python functions from Chapter 7 have been defined.

```
1. >>> len(primes(10000))
    1229
    >>> 10000/log(10000)
    1085.73620476
```

```
3. >>> powermod(3,45,100)
    43
```

4. First raise both sides of the equation to the power of the multiplicative inverse of 37 modulo $112 = \varphi(113)$, which is 109 to get $a \equiv (102^{70} + 1)^{109} \pmod{113}$. We then evaluate this and obtain $a = 60$.

```
>>> inversemod(37, 112)
    109
>>> powermod(102, 70, 113)
    98
>>> powermod(99, 109, 113)
    60
```

5. Using the following program we see that the number 2 is a primitive root 67 out of 168 times (about 40 percent).

```
>>> P = primes(1000)
>>> Q = [p for p in P if primitive_root(p) == 2]
>>> print len(Q), len(P)
    67 168
```

6. The first such prime is 36721.

```
>>> P = primes(50000)
>>> Q = [primitive_root(p) for p in P]
>>> Q.index(37)
    3893
>>> P[3893]
    36721
```

7. 2156, since the secret key is $g^{nm} \equiv 454^m \equiv 2156$.

8. To break the system, we need to find n such that $5^n \equiv 3 \pmod{97}$. The following program does this finds $n = 70$, and similarly one finds that $m = 31$. The secret key is $5^{70 \cdot 31} \equiv 44 \pmod{97}$.

```
>>> for n in range(97):
...     if powermod(5,n,97)==3: print n
70
```

9. We factor n and compute $\varphi(n)$ then the inverse d of e modulo $\varphi(n)$.

```
>>> factor(5352381469067)
[(141307, 1), (37877681L, 1)]
>>> d=inversemod(4240501142039, (141307-1)*(37877681-1))
>>> d
5195621988839L
```

11.

```
>>> convergents([-3,1,1,1,1,3])
[(-3, 1), (-2, 1), (-5, 2), (-7, 3), \
(-12, 5), (-43, 18)]
>>> convergents([0,2,4,1,8,2])
[(0, 1), (1, 2), (4, 9), (5, 11), \
(44, 97), (93, 205)]
```

12. The following code outputs the first 8 examples. First we import the math library, in order to compute a decimal approximation to e . Then we compute terms of the continued fraction of e along with the partial convergents. Finally we print only those partial convergents that satisfy the Hurwitz inequality.

```
>>> import math
>>> e = math.exp(1)
>>> v, convs = contfrac_float(e)
>>> [(a,b) for a, b in convs if \
abs(e - a*1.0/b) < 1/(math.sqrt(5)*b**2)]
[(3, 1), (19, 7), (193, 71), (2721, 1001), \
(49171, 18089), (1084483, 398959), \
(28245729, 10391023), (325368125, 119696244)]
```

13. -389 is not a sum of two squares because it is negative. 12345 is not because 3 exactly divides it. $729 = 3^6 = (3^3)^2 + 0^2$. The number 5809961789 is prime and equals $51542^2 + 56155^2$.

```
>>> factor(12345)
[(3, 1), (5, 1), (823, 1)]
>>> factor(729)
[(3, 6)]
>>> factor(5809961789)
```

```

[(5809961789L, 1)]
>>> 5809961789 % 4
1L
>>> sum_of_two_squares(5809961789)
(51542L, 56155L)

```

14. We use the following program. The computation of P_s takes a few seconds, since our implementation of `factor` is not very efficient.

```

>>> N = [60 + s for s in range(-15,16)]
>>> def is_powersmooth(B, x):
...     for p, e in factor(x):
...         if p**e > B: return False
...     return True
>>> Ns = [x for x in N if is_powersmooth(20, x)]
>>> print len(Ns), len(N), len(Ns)*1.0/len(N)
14 31 0.451612903226
>>> P = [x for x in range(10**12, 10**12+1000)\
         if miller_rabin(x)]
>>> Ps = [x for x in P if \
         is_powersmooth(10000, x-1)]
>>> print len(Ps), len(P), len(Ps)*1.0/len(P)
2 37 0.0540540540541

```

References

- [ACD⁺99] K. Aardal, S. Cavallar, B. Dodson, A. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C.&C. Putnam, and P. Zimmermann, *Factorization of a 512-bit RSA key using the Number Field Sieve*, <http://www.loria.fr/~zimmerma/records/RSA155> (1999).
- [AGP94] W. R. Alford, Andrew Granville, and Carl Pomerance, *There are infinitely many Carmichael numbers*, *Ann. of Math. (2)* **139** (1994), no. 3, 703–722. MR 95k:11114
- [AKS02] M. Agrawal, N. Kayal, and N. Saxena, *PRIMES is in P*, to appear in *Annals of Math.*,
<http://www.cse.iitk.ac.in/users/manindra/primalty.ps> (2002).
- [BS76] Leonard E. Baum and Melvin M. Sweet, *Continued fractions of algebraic power series in characteristic 2*, *Ann. of Math. (2)* **103** (1976), no. 3, 593–610. MR 53 #13127
- [Bur89] D.M. Burton, *Elementary number theory*, second ed., W. C. Brown Publishers, Dubuque, IA, 1989. MR 90e:11001
- [Cal] C. Caldwell, *The Largest Known Primes*,
<http://www.utm.edu/research/primes/largest.html>.

- [Cer] Certicom, *The certicom ECC challenge*,
[http://www.certicom.com/
index.php?action=res,ecc_challenge](http://www.certicom.com/index.php?action=res,ecc_challenge).
- [Cla] Clay Mathematics Institute, *Millennium prize problems*,
http://www.claymath.org/millennium_prize_problems/.
- [Coh] H. Cohn, *A short proof of the continued fraction expansion of e* ,
<http://research.microsoft.com/~cohn/publications.html>.
- [Coh93] H. Cohen, *A course in computational algebraic number theory*,
Graduate Texts in Mathematics, vol. 138, Springer-Verlag, Berlin,
1993. MR 94i:11105
- [Con97] John H. Conway, *The sensual (quadratic) form*, Carus Mathematical Monographs, vol. 26, Mathematical Association of America, Washington, DC, 1997, With the assistance of Francis Y. C. Fung. MR 98k:11035
- [CP01] R. Crandall and C. Pomerance, *Prime numbers*, Springer-Verlag, New York, 2001, A computational perspective. MR 2002a:11007
- [Cre] J. E. Cremona, *mwrnk (computer software)*,
<http://www.maths.nott.ac.uk/personal/jec/ftp/progs/>.
- [Cre97] ———, *Algorithms for modular elliptic curves*, second ed., Cambridge University Press, Cambridge, 1997.
- [Dav99] H. Davenport, *The higher arithmetic*, seventh ed., Cambridge University Press, Cambridge, 1999, An introduction to the theory of numbers, Chapter VIII by J. H. Davenport. MR 2000k:11002
- [DH76] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Trans. Information Theory **IT-22** (1976), no. 6, 644–654. MR 55 #10141
- [Eul85] Leonhard Euler, *An essay on continued fractions*, Math. Systems Theory **18** (1985), no. 4, 295–328, Translated from the Latin by B. F. Wyman and M. F. Wyman. MR 87d:01011b
- [FT93] A. Fröhlich and M. J. Taylor, *Algebraic number theory*, Cambridge University Press, Cambridge, 1993. MR 94d:11078
- [GS02] X. Gourdon and P. Sebah, *The $\pi(x)$ project*,
[http://numbers.computation.free.fr/constants/primes/
pix/pixproject.html](http://numbers.computation.free.fr/constants/primes/pix/pixproject.html).
- [Guy94] R. K. Guy, *Unsolved problems in number theory*, second ed., Springer-Verlag, New York, 1994, Unsolved Problems in Intuitive Mathematics, I. MR 96e:11002

- [Hoo67] C. Hooley, *On Artin's conjecture*, J. Reine Angew. Math. **225** (1967), 209–220. MR 34 #7445
- [HW79] G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, fifth ed., The Clarendon Press Oxford University Press, New York, 1979. MR 81i:10002
- [IBM01] IBM, *IBM's Test-Tube Quantum Computer Makes History*, http://www.research.ibm.com/resources/news/20011219_quantum.shtml.
- [IR90] K. Ireland and M. Rosen, *A classical introduction to modern number theory*, second ed., Springer-Verlag, New York, 1990. MR 92e:11001
- [Khi63] A. Ya. Khintchine, *Continued fractions*, Translated by Peter Wynn, P. Noordhoff Ltd., Groningen, 1963. MR 28 #5038
- [Knu97] Donald E. Knuth, *The art of computer programming*, third ed., Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, 1997, Volume 1: Fundamental algorithms, Addison-Wesley Series in Computer Science and Information Processing.
- [Knu98] ———, *The art of computer programming. Vol. 2*, second ed., Addison-Wesley Publishing Co., Reading, Mass., 1998, Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing. MR 83i:68003
- [Kob84] N. Koblitz, *Introduction to elliptic curves and modular forms*, Graduate Texts in Mathematics, vol. 97, Springer-Verlag, New York, 1984. MR 86c:11040
- [Leh14] D. N. Lehmer, *List of primes numbers from 1 to 10,006,721*, Carnegie Institution Washington, D.C. (1914).
- [Lem] F. Lemmermeyer, *Proofs of the Quadratic Reciprocity Law*, <http://www.rzuser.uni-heidelberg.de/~hb3/rchrono.html>.
- [Len87] H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2) **126** (1987), no. 3, 649–673. MR 89g:11125
- [LL93] A. K. Lenstra and H. W. Lenstra, Jr. (eds.), *The development of the number field sieve*, Lecture Notes in Mathematics, vol. 1554, Springer-Verlag, Berlin, 1993. MR 96m:11116
- [LMG⁺01] Vandersypen L. M., Steffen M., Breyta G., Yannoni C. S., Shorwood M. H., and Chuang I. L., *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance*, Nature **414** (2001), no. 6866, 883–887.

- [LT72] S. Lang and H. Trotter, *Continued fractions for some algebraic numbers*, J. Reine Angew. Math. **255** (1972), 112–134; addendum, *ibid.* **267** (1974), 219–220; MR **50** #2086. MR **46** #5258
- [LT74] ———, *Addendum to: “Continued fractions for some algebraic numbers”* (J. Reine Angew. Math. **255** (1972), 112–134), J. Reine Angew. Math. **267** (1974), 219–220. MR **50** #2086
- [Mor93] P. Moree, *A note on Artin’s conjecture*, Simon Stevin **67** (1993), no. 3-4, 255–257. MR **95e**:11106
- [NZM91] I. Niven, H. S. Zuckerman, and H. L. Montgomery, *An introduction to the theory of numbers*, fifth ed., John Wiley & Sons Inc., New York, 1991. MR **91i**:11001
- [Old70] C. D. Olds, *The Simple Continued Fraction Expression of e* , Amer. Math. Monthly **77** (1970), 968–974.
- [Per57] O. Perron, *Die Lehre von den Kettenbrüchen. Dritte, verbesserte und erweiterte Aufl. Bd. II. Analytisch-funktionentheoretische Kettenbrüche*, B. G. Teubner Verlagsgesellschaft, Stuttgart, 1957. MR **19**,25c
- [Ros] Guido van Rossum, *Python*, <http://www.python.org>.
- [RSA] RSA, *The New RSA Factoring Challenge*, <http://www.rsasecurity.com/rsalabs/challenges/factoring>.
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM **21** (1978), no. 2, 120–126. MR **83m**:94003
- [Sho97] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), no. 5, 1484–1509. MR **98i**:11108
- [Sil86] J. H. Silverman, *The arithmetic of elliptic curves*, Graduate Texts in Mathematics, vol. 106, Springer-Verlag, New York, 1986. MR **87g**:11070
- [Sin99] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, Doubleday, 1999.
- [Slo] N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences*, <http://www.research.att.com/~njas/sequences/>.
- [ST92] J. H. Silverman and J. Tate, *Rational points on elliptic curves*, Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1992. MR **93g**:11003

- [Wal48] H. S. Wall, *Analytic Theory of Continued Fractions*, D. Van Nostrand Company, Inc., New York, N. Y., 1948. MR 10,32d
- [Wei03] E. W. Weisstein, *RSA-576 Factored*,
<http://mathworld.wolfram.com/news/2003-12-05/rsa/>.
- [Wil00] A. J. Wiles, *The Birch and Swinnerton-Dyer Conjecture*,
http://www.claymath.org/prize_problems/birchsd.htm.