Algorithmic Number Theory
MSRI Publications
Volume **44**, 2008

Basic algorithms in number theory

JOE BUHLER AND STAN WAGON

1. Introduction

Our subject combines the ancient charms of number theory with the modern fascination with algorithmic thinking. Newcomers to the field can appreciate this conjunction by studying the many elementary pearls in the subject. The aim here is to describe a few of these gems with the combined goals of providing background for subsequent articles in this volume, and luring the reader into pursuing full-length treatments of the subject, such as [Bach and Shallit 1996; Bressoud and Wagon 2000; Cohen 1993; Crandall and Pomerance 2005; Knuth 1981; von zur Gathen and Gerhard 2003; Shoup 2005].

Many details will be left to the reader, and we will assume that he or she knows (or can look up) basic facts from number theory, algebra, and elementary programming.

We tend to focus more on the mathematics and less on the sometimes fascinating algorithmic details. However, the subject is grounded in, and motivated by, examples; one can learn interesting and surprising things by actually implementing algorithms in number theory. Implementing almost any of the algorithms here in a modern programming language isn't too hard; we encourage budding

number theorists to follow the venerable tradition of their predecessors: write programs and think carefully about the output.

2. Algorithmic complexity

Algorithms take input and produce output. The *complexity* of an algorithm A is a function $C_A(n)$, defined to be the maximum, over all input I of size at most n, of the cost of running A on input I. Cost is often measured in terms of the number of "elemental operations" that the algorithm performs and is intended, in suitable contexts, to approximate the running time of actual computer programs implementing these algorithms.

A formalization of these ideas requires precise definitions for "algorithm," "input," "output," "cost," "elemental operation," and so on. We will give none.

Instead, we consider a series of number-theoretic algorithms and discuss their complexity from a fairly naive point of view. Fortunately, this informal and intuitive approach is usually sufficient for purposes of algorithmic number theory. More precise foundations can be found in many texts on theoretical computer science or algorithmic complexity such as [Garey and Johnson 1979; Hopcroft and Ullman 1979; Kozen 2006].

The first problem arises in elementary school.

PROBLEM 1. MULTIPLICATION: Given integers x and y, find their product xy.

From the algorithmic perspective, the problem is woefully underspecified. We interpret it in the following natural (but by no means only possible) way. An algorithm that solves MULTIPLICATION takes two strings of symbols as input and writes a string of symbols as its output. The input strings are base b representation of integers x and y, where $b > 1$ is fixed, and in practice one might expect $b = 2$, 10, 2^{32}, or 2^{64}. The algorithm follows a well-defined procedure in which the next step is determined by the current state of the computation; one might imagine a program written in an idealized form of your favorite computer language that has access to unlimited memory. Its output string represents the base-b representation of the product xy.

The natural notion of the size of an integer x is the total number of symbols (base-b digits) in the input, perhaps augmented by a small constant to allow for delimiting the integer and specifying its sign. For definiteness, we define the base-b size of x to be

$$\text{size}_b(x) := 1 + \lceil \log_b(1 + |x|) \rceil,$$

where $\log_b$ is the logarithm to the base b, and $\lceil u \rceil$ is the ceiling of u — the smallest integer greater than or equal to u.

The size of an integer x is $O(\log |x|)$, where $g(x) = O(f(x))$ is a shorthand statement saying that g is in the class of functions such that there is a constant C with $|g(x)| \le C\,|f(x)|$ for sufficiently large x. Note that $O(\log_a x) = O(\log_b x)$ for $a, b > 1$. In particular, if we are interested in complexity only up to a constant factor the choice of $b > 1$ is irrelevant.

The usual elementary school multiplication algorithm uses $O(n^2)$ digit operations to multiply two input integers of size n. More precisely, if x and y have size n, then approximately n^2 digit-sized multiplications and n additions of n-digit intermediate products are required. Since adding n-digit integers takes time $O(n)$, the overall complexity of multiplying n-digit integers using this algorithm is $O(n^2) + n \cdot O(n) = O(n^2)$. Notice that the O-notation gracefully summarizes an upper bound on the running time of the algorithm — the complexity $O(n^2)$ is independent of the base b, the precise details of measuring the size of an integer, the definition of size of two inputs (as the maximum of the two integer inputs, or the total of their sizes), and so on.

An algorithm A is said to take *polynomial time* if its complexity $C_A(n)$ is $O(n^k)$ for some integer k. Although this is a flexible definition, with unclear relevance to computational practice, it has proved to be remarkably robust. In fact, it is sometimes reasonable to take "polynomial time" as synonymous with "efficient," and in any case the notion has proved useful in both theory and practice.

Once it is known that a problem can be solved in polynomial time, it is interesting to find the smallest possible exponent k. Several improvements to the $O(n^2)$ multiplication algorithm are known, and the current state of the art is a striking algorithm of Schönhage [1971] that takes time $O(n \log n \log\log n)$ to multiply two n-digit integers. This is sometimes written inexactly as $O(n^{1+\varepsilon})$, where ε denotes an arbitrarily small positive number. Note that $O(n)$ is an obvious lower bound since the input has size $O(n)$ and in order to multiply integers it is necessary to read them. Algorithms that use the Schönhage algorithm, or related ones, are said to use *fast arithmetic*, and algorithms that are close to obvious lower bounds are sometimes said to be *asymptotically fast*. Many such algebraic and arithmetic algorithms are known (see [Bernstein 2008] for examples), and they are becoming increasingly important in computational practice.

The elemental operations above act on single digits (i.e., bits if $b = 2$), and the resulting notion of complexity is sometimes called *bit complexity*. In other contexts it might be more useful to assume that any arithmetic operation takes constant time on integers of arbitrary size; this might be appropriate, for example, if all integers are known to fit into a single computer word. When complexity of an algorithm is defined by counting arithmetic operations, the

results is said to be the *arithmetic complexity* of the algorithm. In this model the cost of a single multiplication is $O(1)$, reminding us that complexity estimates depends dramatically on the underlying assumptions.

PROBLEM 2. EXPONENTIATION: Given x and a nonnegative integer n, compute x^n.

Again, the problem is underspecified as it stands. We will assume that x is an element of a set that has a well-defined operation (associative with an identity element) that is written multiplicatively; moreover, we will measure cost as the number of such operations required to compute x^n on input x and n. The size of the input will be taken to be the size of the integer n.

Although x^{16} can be computed with 15 multiplications in an obvious way, it is faster to compute it by 4 squarings. More generally, the binary expansion $n = \sum a_i 2^i$, with $a_i \in \{0, 1\}$, implies that

$$x^n = x^{a_0}(x^2)^{a_1}(x^4)^{a_2}\cdots \tag{2-1}$$

which suggests a clever way to interleave multiplications and squarings:

RIGHT-TO-LEFT EXPONENTIATION

Input: x as above, and a nonnegative integer n
Output: x^n
1. $y := 1$
2. While $n > 0$
   if n is odd, $y := xy$ // a_i is 1
   $x := x^2$, $n := \lfloor n/2 \rfloor$
3. Return y

Here ":=" denotes assignment of values to variables, "//" indicates a comment, "1" denotes the identity for the operation, and the floor $\lfloor u \rfloor$ is the largest integer less than or equal to u. The correctness of the algorithm is reasonably clear from equation (2-1) since x^{2^k} is multiplied into y if and only if the kth bit a_k of the binary expansion of n is nonzero. This can be proved more formally by showing by induction that at the beginning of Step 2, $X^N = x^n y$ holds, where X and N denote the initial values of the variables x and n. When n is 0 equation says that $X^N = y$, so that y is the desired power.

The usual inductive definition of $\mathrm{Exp}(x, n) := x^n$ gives an obvious recursive algorithm:

$$\mathrm{Exp}(x,n) = \begin{cases} 1 & \text{if } n = 0, \\ \mathrm{Exp}(x^2, n/2) & \text{if } n > 0 \text{ is even}, \\ x \cdot \mathrm{Exp}(x^2, (n-1)/2) & \text{if } n \text{ is odd.} \end{cases} \tag{2-2}$$

Experienced programmers often implement recursive versions of algorithms because of their elegance and obvious correctness, and when necessary convert them to equivalent, and perhaps faster, iterative (nonrecursive) algorithms. If this is done to the recursive program the result is to Right-to-Left algorithm above.

Curiously, if the inductive definition is replaced by the mathematically equivalent algorithm in which squaring follows the recursive calls,

$$\mathrm{Exp}(x,n) = \begin{cases} 1 & \text{if } n = 0, \\ \mathrm{Exp}(x,n/2)^2 & \text{if } n > 0 \text{ is even}, \\ x \cdot \mathrm{Exp}(x,(n-1)/2)^2 & \text{if } n \text{ is odd}, \end{cases} \tag{2-3}$$

then the corresponding iterative algorithm is genuinely different.

LEFT-TO-RIGHT EXPONENTIATION

Input: x, a nonnegative integer n, a power of two $m = 2^a$ such that $m/2 \le n < m$

Output: x^n

1. $y := 1$
2. While $m > 1$
   $m := \lfloor m/2 \rfloor,\ y := y^2$
   If $n \ge m$ then $y := xy,\ n := n - m$
3. Return y

Correctness follows inductively by proving that at the beginning of Step 2, $n < m$ and $y^m x^n = x^N$. In contrast to the earlier algorithm, this version consumes the bits a_i in the binary expansion of n starting with the leftmost (most significant) bit.

The complexity of any of the versions of this algorithm (collectively called EXP in the sequel) is $O(\log n)$ since the number of operations is bounded by $2 \cdot \mathrm{size}_2(n)$. As will be seen, this remarkable efficiency has numerous applications in algorithmic number theory. Note that the naive idea of computing x^n by repeatedly multiplying by x takes time $O(n)$, which is exponential in the input size.

REMARK 1. In a specific but important practical case the left-to-right version of EXP is better than the right-to-left version. Suppose that our operation is "multiplying modulo N" and that x is small relative to N. Then multiplying by the original x is likely to take less time than modular multiplication by an arbitrary integer X in the range $0 \le X < N$. The left-to-right version preserves the original x (though the squarings involve arbitrary integers), whereas the right-to-left version modifies x and hence performs almost all operations on arbitrary elements. In other words, with a different computational model (bit

complexity, with the specific underlying operation "multiply modulo N", and x small) the left-to-right algorithm, either recursive or iterative, is significantly better than right-to-left exponentiation.

REMARK 2. If the underlying operation is multiplication of integers, the bit complexity of computing x^n is exponential, since the output has size that is exponential in the input size $\log n$. Any algorithm will be inefficient, illustrating yet again the dependence on the underlying computational model.

This discussion of calculating powers barely scratches the surface of a great deal of theoretical and practical work. The overwhelming importance of exponentiation has led to many significant practical improvements; perhaps the most basic is to replace the base-2 expansion of n with a base-b expansion for b a small power of 2. On the theoretical side, there are interesting results on finding the absolutely smallest possible number of operations required to compute x^n [Knuth 1981].

PROBLEM 3. GCD: Given positive integers a and b, find the largest integer that is a divisor of both a and b.

The greatest common divisor (GCD) is denoted $\gcd(a, b)$. Perhaps the most famous number-theoretic algorithm of all is due to Euclid.

EUCLID'S ALGORITHM
Input: Positive integers a and b
Output: $\gcd(a, b)$
While $b > 0$
    $\{a, b\} := \{b, a \bmod b\}$
Return a

Here $r = a \bmod b$ is the remainder when a is divided by b, i.e., the unique integer r, $0 \le r < b$, such that there is a q with $a = qb + r$. The simultaneous assignment statement $\{a, b\} = \{b, a \bmod b\}$ could be implemented in a more prosaic programming language by something along the lines of the three statements $temp := b$, $b = a \bmod b$, $a = temp$. The correctness of the algorithm can be verified by showing that the GCD of a and b doesn't change at each step of the algorithm, and that when a becomes divisible by b, then b is the GCD.

Just as with multiplication, the remainder $a \bmod b$ can be found in time $\log^2(\max(a, b))$ with straightforward algorithms, and time $\log^{1+\varepsilon} \max(a, b)$ for any positive ε if asymptotically fast algorithms are used. (Here $\log^r x$ is shorthand for $(\log x)^r$.) It isn't too hard to work out that $a > b$ after one step of the algorithm, and then the smallest number is halved in (at most) two steps of the algorithm. This means that the number of times that the loop is executed (the number of remainder operations) is bounded by $O(\log \max(a, b))$. Thus the

algorithm has complexity $O(k^3)$ on k-bit input, or $O(k^{2+\varepsilon})$ if fast arithmetic is used. We will have much much more to say about Euclid's algorithm later.

If $\gcd(a, b) = 1$ then a and b are said to be *relatively prime*, or *coprime*.

PROBLEM 4. PRIMALITY: Given a positive integer $n > 1$, is n is a prime?

The is an example of a *decision problem*, for which the output is either "yes" or "no."

Perhaps the most straightforward algorithm for PRIMALITY is the trial division method: for $d = 2, 3, \ldots$, test whether n is divisible by d. If n is divisible by some $d \leq \sqrt{n}$ then n is composite; if n is not divisible by any $d \leq \sqrt{n}$ then n is prime. The time required is $O(\sqrt{n})$, which is an exponential function of the size of n, and this algorithm is impractical for even moderately large n.

Fermat's Little Theorem says that if n is a prime and a is coprime to n then $a^{n-1} \equiv 1 \bmod n$, which implies that the order of a in $(\mathbb{Z}/n\mathbb{Z})^*$) divides $n-1$. The condition $a^{n-1} \equiv 1 \bmod n$ is easy to check, using EXP in $(\mathbb{Z}/n\mathbb{Z})^*$.

REMARK 3. We take the opportunity to remind the reader that two integers are congruent modulo n if their difference is divisible by n, that $\mathbb{Z}/n\mathbb{Z}$ denotes the set of n classes under this equivalence relation, and that $\mathbb{Z}/n\mathbb{Z}$ is a ring under the natural addition and multiplication operations induced from operations on the integers. Moreover, $(\mathbb{Z}/n\mathbb{Z})^*$ denotes the group of units in this ring, i.e., the set of congruence classes containing integers that are coprime to n, under the operation of multiplication. This group has $\phi(n)$ elements, where $\phi(n)$ is the Euler-phi function — the number of positive integers less than n that are coprime to n. Finally, we let $a \bmod n$ denote the class of $\mathbb{Z}/n\mathbb{Z}$ that contains a. We will tolerate the conflict with earlier usage because the meaning can be disambiguated from context: if $a \bmod n$ is an integer then the remainder is intended, and if $a \bmod n$ lies in $\mathbb{Z}/n\mathbb{Z}$ then the congruence class is intended.

In the favorable circumstance in which the prime factorization of $n-1$ is known, Fermat's Little Theorem can be turned on its head to give a proof of primality.

THEOREM 5. If a and n are integers such that $a^{n-1} \equiv 1 \bmod n$, and $a^{(n-1)/q} \not\equiv 1 \bmod n$ for all prime divisors q of $n-1$, then n is prime.

PROOF. As noted above, the congruence $a^{n-1} \equiv 1 \bmod n$ implies that the order of $a \bmod n$ in $(\mathbb{Z}/n\mathbb{Z})^*$, which we will denote by $\mathrm{ord}_n(a)$, is a divisor of $n-1$. Any proper divisor of $n-1$ is a divisor of $(n-1)/q$ for some prime q. The second condition of the theorem says that $\mathrm{ord}(a)$ does not divide $(n-1)/q$ for any q, and we conclude that $\mathrm{ord}(a) = n-1$. Thus $(\mathbb{Z}/n\mathbb{Z})^*$ has $n-1$ elements and n is a prime, as claimed. □

A generalization of this theorem due to Pocklington says that only a partial factorization of $n-1$ is necessary: if m is a divisor of $n-1$ with $m > \sqrt{n}$, then

n is a prime if $a^{n-1} \equiv 1 \bmod n$, and $a^{(n-1)/q} \not\equiv 1 \bmod n$ for prime divisors q of m. Loosely, this says that primality is easy to test if $n-1$ is half-factored.

Unfortunately, this does not give an efficient primality test: For large n no algorithm is known that efficiently factors or half-factors $n-1$.

As a first try, observe that if n is composite then a^{n-1} should be, intuitively, a random integer modulo n. Thus we could choose several random a and report "probable prime" when Fermat's congruence $a^{n-1} \equiv 1 \bmod n$ holds for all a, and "composite" if it fails for any one of them. Unfortunately, there are positive composite integers called Carmichael numbers (e.g., $n = 561 = 3 \cdot 11 \cdot 17$; see [Crandall and Pomerance 2005]) such that the congruence holds for all a that are relatively prime to n.

As a second try, we "take the square root" of the Fermat congruence. To explain this it is convenient to review Legendre and Jacobi symbols.

An integer a is a *quadratic residue* modulo an odd prime p if it is a nonzero square, i.e., if there is an x (not divisible by p) such that $x^2 \equiv a \bmod p$. Nonsquares are said to be *quadratic nonresidues*. This information is encoded in the *Legendre symbol*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p \text{ divides } a, \\ 1 & \text{if } x \text{ is a quadratic residue mod } p, \\ -1 & \text{if } x \text{ is a quadratic nonresidue mod } p. \end{cases}$$

Euler's Criterion gives an explicit congruence for the Legendre symbol

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \bmod p, \tag{2-4}$$

from which the symbol can be computed efficiently using EXP in $(\mathbb{Z}/p\mathbb{Z})^*$. The *Jacobi symbol* generalizes the Legendre symbol and is defined, for an integer a and a positive odd positive integer b, by reverting to the Legendre symbol when b is an odd prime, and enforcing multiplicativity in the denominator; if $b = \prod b^{e_p}$ is the factorization of b into prime powers, then

$$\left(\frac{a}{b}\right) = \prod_p \left(\frac{a}{p}\right)^{e_p}.$$

The Jacobi symbol is multiplicative in both the numerator and denominator, depends only on $a \bmod b$, and obeys the famous law of quadratic reciprocity

$$\left(\frac{a}{b}\right)\left(\frac{b}{a}\right) = (-1)^{((a-1)(b-1))/4}, \quad a, b \text{ odd}.$$

Moreover, two "supplementary laws" hold:

$$\left(\frac{-1}{b}\right) = (-1)^{(b-1)/2}, \qquad \left(\frac{2}{b}\right) = (-1)^{(b^2-1)/8}.$$

This leads to a natural recursive algorithm, using the identities

$$\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right),$$
$$\left(\frac{a}{b}\right) = (-1)^{(b^2-1)/8}\left(\frac{a/2}{b}\right),$$
$$\left(\frac{a}{b}\right) = (-1)^{(a-1)(b-1)/4}\left(\frac{b}{a}\right),$$

applicable, respectively, when $a < 0$ or $a \geq b$, a is even, or a is odd. The crucial point is that this is an efficient algorithm even if the factorization of b is unknown. An actual implementation of this resembles Euclid's algorithm augmented with some bookkeeping.

REMARK 4. Let $n = F_k := 2^{2^k} + 1$ be the kth Fermat number. The reader may enjoy using Theorem 5 and quadratic reciprocity to show that if $n = F_k$ and $3^{(n-1)/2} \equiv -1 \bmod n$ then n is prime, and to prove that if n is prime then the congruence holds.

Now we return to the problem of giving an efficient algorithm for primality. We use Euler's congruence

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \bmod p$$

together with one of the seminal ideas of twentieth century computer science: it can be beneficial to allow algorithms to make random moves!

A *probabilistic* (*randomized*) algorithm extends our earlier implicit notion of an algorithm in that such an algorithm is allowed to flip a coin as needed and make its next move depending on the result. Typically a coin flip is deemed to cost one unit of running time. It is possible to model this more formally by thinking of the sequence of random bits $b \in \{0, 1\}$ as a second input string to the algorithm. Saying that an algorithm has a property with probability p means that it has the property for a fraction p of the possible auxiliary input strings (coin flips). For example, if the algorithm returns a correct answer for two-thirds of all possible input bit strings, then we say that the probability of correctness is $p = 2/3$.

It may seem worse than useless to allow algorithms to make random moves unrelated to the problem at hand but, as we will see, this additional capability can be surprisingly powerful.

REMARK 5. Probabilistic algorithms are ubiquitous. In circumstances in which it is necessary to emphasize that an algorithm is not probabilistic it will be referred to as a *deterministic* algorithm.

The following primality test is a famous example of a probabilistic algorithm.

SOLOVAY–STRASSEN PRIMALITY TEST

Input: A positive integer k, an odd integer $n > 1$
Output: "Prime" or "Composite"
1. For $i = 1, 2, \ldots, k$
    Choose a randomly from $\{1, 2, \ldots, n-1\}$
    If $\left(\frac{a}{n}\right) = 0$ or $\left(\frac{a}{n}\right) \neq a^{(n-1)/2} \bmod n$ then
        Output "Composite" and halt.
2. Output "Prime"

REMARK 6. The algorithm chooses a uniformly random element of a finite set. The reader may enjoy the puzzle of figuring out how to do this using coin flips, i.e., events of probability $1/2$. The goal is to simulate an arbitrary probability, and to do so in an efficient manner, e.g., to use about $\log n$ coin flips on average to choose a uniformly random element from an n-element set.

There are different flavors of probabilistic algorithms, according to whether the output (or some possible outputs) are true or just highly likely to be true, whether running times are bounds, or merely expected running times, etc. The following theorem clarifies this in the case of Solovay–Strassen, e.g., showing that the answer "Composite" is always true, the answer "Prime" is highly likely to be true, and the running time is polynomial in the input size and absolute value of the logarithm of the error probability.

THEOREM 6. The Solovay–Strassen algorithm returns "Prime" if n is prime, and returns "Composite" with probability at least $1 - 1/2^k$ if n is composite. Its expected running time is bounded by a polynomial in $\log n$ and k.

PROOF. (Sketch.) The GCD and exponentiation steps can be done it at most time $O(\log^3 n)$, so each iteration takes polynomial time. The only aspect of the running time that depends on the input bits is the choice of a random element of the $(\mathbb{Z}/n\mathbb{Z})^*$.

If n is prime, then Euler's congruence implies that the test returns "Prime."

If n is composite then the algorithm will return "Composite" unless each of k random choices of a lies in the subgroup

$$E(n) := \left\{a \in (\mathbb{Z}/n\mathbb{Z})^* : \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \bmod n\right\}$$

of the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^*$, sometimes called the group of "Euler liars" for the composite number n. The coin flips done by the algorithm allow it to choose a uniformly random element of $(\mathbb{Z}/n\mathbb{Z})^*$. It can be shown [Bach and Shallit 1996] that $|E(n)| \leq \phi(n)/2$, so that the chance of a composite number passing all k of these tests is at most $1 - 1/2^k$. □

The technique of repeating a test to quickly drive the probability of bad behavior to near zero is common in practical probabilistic algorithms. For example, the probability of failure in one round of Solovay–Strassen is at most $1/2$, and that means that the probability of failure after k independent tests is at most $1/2^k$, so assuring the failure probability of at most ε requires $\log(1/\varepsilon)$ tests.

The Jacobi symbol compositeness test inside the Solovay–Strassen algorithm can be improved. If n is odd, a is coprime to n, and $n-1 = 2^k r$, where r is odd, let $A := a^r \bmod n$. If n is prime then, by Fermat's Theorem, $A^{2^k} \equiv 1 \bmod n$. Since the only square roots of 1 in a field are ± 1, we know that if n is prime then either $A = 1$, or else -1 occurs in the sequence $A, A^2, A^4, \ldots, A^{2^k} = 1 \bmod n$. If this happens we say that n is an strong probable prime to the base a.

If this does not happen, i.e., either $A \neq 1 \bmod n$ and $A^{2^j} \neq -1 \bmod n$ for $0 \leq j < k$, then n is definitely composite, and we say that a is a *witness* of n's compositeness.

The obvious generalization of Solovay–Strassen using this test is to choose a series of a's, and see whether one is a witness of n's compositeness or n is a strong probable prime to all of those bases. This is sometimes called the strong Fermat test.

Here are several nontrivial results related to the above ideas; for a discussion and details see [Crandall and Pomerance 2005] and [Damgård et al. 1993].

(i) If an odd n is composite then at least $3/4$ of the a's coprime to n are witnesses to n being composite.

(ii) If a famous conjecture in number theory is true — the Extended Riemann Hypothesis (ERH) — then any odd composite n has a witness that is less than $2\log^2 n$. In particular, if we assume the ERH then there is a polynomial-time primality test (this also applies to the Solovay–Strassen test).

(iii) Let n be an odd integer chosen uniformly randomly between 2^k and 2^{k+1}, and let a be chosen randomly among integers between 1 and n that are coprime to n. If n is a strong probable prime to the base a then the probability that a is composite is less than $k^2/4^{\sqrt{k}-2}$. Thus for large n a single probable prime test can give high confidence of primality.

These ideas can provide overwhelming evidence that a given integer is prime, but no rigorous proof. If we want absolute proof, the story changes. Many proof techniques have been considered over the years, and this continues to be an active area for practical work. The central theoretical question is whether primality can be proved in polynomial time, and this was settled dramatically in 2002 when Manindra Agrawal, Neeraj Kayal, and Nitin Saxena [Agrawal et al. 2004] discovered a deterministic polynomial-time algorithm. For a description of this, and algorithms currently used in practice, see [Schoof 2008b].

PROBLEM 7. QUADRATIC NONRESIDUES: Given an odd prime p, find a quadratic nonresidue modulo p.

This simple problem illustrates stark differences between theory and practice, and deterministic and probabilistic algorithms.

If p is an odd prime there are $(p-1)/2$ quadratic residues and $(p-1)/2$ nonresidues mod p, so if a mod p is nonzero, then a has a 50/50 chance of being a nonresidue. Moreover, quadratic residuosity/nonresiduosity is easy to establish by calculating a Legendre symbol. Thus there is an obvious probabilistic algorithm: repeatedly choose a at random until a nonresidue is found. With overwhelming likelihood a nonresidue will be found quickly, and thus quadratic nonresidues can be found quickly with a probabilistic polynomial-time algorithm.

However, no deterministic polynomial-time algorithm is known. Nothing seems to be better than testing $a = 2, 3, \ldots$ until arriving at a nonresidue. There are heuristic grounds to think that there is a nonresidue $a = O(\log^{1+\varepsilon} p)$. However, the best that can be proved is that one finds a nonresidue $a = O(p^{1/4})$. The simple-minded (deterministic) algorithm for finding a nonresidue could take exponential time. It is known that if the Extended Riemann Hypothesis is true, then there is a nonresidue $a < 2\log^2 p = O(\log^2 p)$ [Bach 1990].

PROBLEM 8. FACTORING: Given a positive integer $n > 1$, find a proper divisor m of n, i.e., a divisor m such that $1 < m < n$.

Factoring appears to be much harder than primality, both in theory and practice. Trial division again gives an obvious algorithm that is impractical unless n has a small divisor. The problem has fascinated mathematicians for centuries, and a vast menagerie of algorithms are known. Details of two of the most important current algorithms are described elsewhere in this volume [Poonen 2008; Stevenhagen 2008b]. Both algorithms require the use of sophisticated mathematical ideas: one requires the use of elliptic curves, and the other relies extensively on algebraic number theory.

We now describe a striking factoring algorithm, called the Pollard ρ algorithm, due to John Pollard [1978].

Let n be a composite integer that is not a prime power. (It is easy to check whether or not n is a perfect power by taking sufficiently accurate kth roots for $2 \le k \le \log_2 n$.) Let

$$f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}, \quad x \mapsto f(x) = x^2 + 1,$$

and let

$$f^k(x) = f(f(\cdots f(x)\cdots))$$

denote the kth iterate of f applied to x. The Pollard ρ algorithm is:

POLLARD ρ FACTORING ALGORITHM

Input: A composite $n > 1$, not a prime power
Output: A nontrivial factor of n
1. Choose a random $x \in \{0, 1, \ldots, n-1\}$
2. For $i := 1, 2, \ldots$
   $g := \gcd(f^i(x), f^{2i}(x))$
   If $g = 1$, go to the next i
   If $1 < g < n$ then output g and halt
   If $g = n$ then go back to Step 1 and choose another x

What could possibly be going on here? The birthday paradox in elementary probability theory says that a collection of 23 or more people is more likely than not to have two people with the same birthday. More generally, if elements are chosen randomly from a set of size n, with replacement, a repeat is likely after $O(\sqrt{n})$ choices [Knuth 1981].

Let p be an (unknown) prime divisor of n. Evidence suggests that $f^k(x)$ mod p is indistinguishable from a random sequence. Assuming that this is the case, the birthday paradox implies that the sequence repeats a value after $O(\sqrt{p})$ steps, after which time it cycles. Thus the sequence of values mod p has the form

$$y_1, \ldots, y_m, y_{m+1}, \ldots, y_{m+k} = y_m, y_{m+k+1} = y_{m+1}, \ldots.$$

A little calculation shows that if i is the smallest multiple of k that exceeds m then $y_i = y_{2i}$. (This elegant idea is usually referred to as Floyd's cycle-finding algorithm, and it enables the algorithm to use only a very small amount of memory.) In the context of the Pollard ρ algorithm this means that p divides $\gcd(f^i(x), f^{2i}(x))$. Thus the GCDs in the algorithm will sooner or later be divisible by p. One catastrophe that can happen is that, by some strange coincidence, all primes dividing n happen to divide the GCD at the same time, but this is unlikely in practice.

The complexity of this algorithm is $O(n^{1/4})$ in the worst case that n is the product of two roughly equal primes. The Pollard ρ algorithm does have the virtue that it finds smaller factors sooner, so that it is reasonable to apply the algorithm to a composite number when there is no knowledge of the size of the factors. Many further details and optimizations of this charming algorithm and variants can be found in [Cohen 1993; Knuth 1981; Teske 2001].

In the last 25 years, a number of algorithms have been proposed for factoring in subexponential time, i.e., in time less than $O(n^\varepsilon)$ for all $\varepsilon > 0$. All are probabilistic, and most rely, for this favorable complexity estimate, on highly plausible, but as yet unproved, assumptions. The conjectured run times are sometimes said to be the *heuristic complexity* of these algorithms.

To describe the complexity of these algorithms, let

$$L_n[a;c] = \exp\big((c+o(1))(\log n)^a(\log\log n)^{1-a}\big).$$

where $o(1)$ denotes a term that goes to 0 as n goes to infinity. This function interpolates between polynomial time, for $a=0$, and exponential time, for $a=1$, and its peculiar shape arises from formulas for the density of smooth numbers [Granville 2008] (a number is "y-smooth" if all of its prime factors are less than or equal to y).

The first of the two new algorithms described elsewhere in this volume is the Elliptic Curve Method (ECM), due to Hendrik Lenstra [1987]; see also [Poonen 2008]. The algorithm relies on the theory of elliptic curves over finite fields and some eminently believable heuristics concerning them; the (heuristic) complexity of this probabilistic algorithm is $L[1/2;1]$. The ECM shares the advantage, with trial division and the Pollard ρ algorithm, that it finds smaller factors faster. A number of other factoring algorithms are known that have complexity $L[1/2;1]$. An algorithm based on class groups of quadratic fields [Lenstra and Pomerance 1992] is "rigorous" in the sense that its $L[1/2;1]$ complexity does not rely on any heuristics.

A few years after the ECM appeared, John Pollard described an algorithm for factoring numbers of the form $n = a\cdot b^k + c$ for k large. Shortly afterwards, this was generalized to arbitrary positive integers n. This algorithm is a successor to the so-called Quadratic Sieve algorithm (QS), and is called the Number Field Sieve (NFS); its heuristic complexity is $L_n[1/3; 4/3^{2/3}] = L_n[1/3; 1.92\ldots]$. The unproved assertions on which this complexity is based are natural statements concerning the proportion of polynomials values that are smooth; these assertions seem true in practice but their proof seems beyond current techniques. The basic idea of this so-called *index-calculus* algorithm is to find smooth integers, and smooth elements of algebraic number fields, by sieving, and then solve a system of linear equations over the field with two elements. The speed of the algorithm in practice derives from the fact that the basic sieving operation can be implemented efficiently on modern computers. Details can be found in [Crandall and Pomerance 2005; Lenstra and Lenstra 1993; Stevenhagen 2008b].

PROBLEM 9. DISCRETE LOGARITHMS: Given an element x of a finite cyclic group G and a generator g of that group, find a nonnegative integer k such that $x = g^k$.

The problem is not precise until the representation of elements of, and the operation in, the finite group is made explicit. The difficulty of the discrete logarithm problem (DLOG) depends on the size of the group and on its specific representation. The additive group $G = \mathbb{Z}/n\mathbb{Z} = \{0,1,2,\ldots,n-1\}$ is cyclic, and the discrete logarithm problem in G is easy with the usual representation of

elements. The discrete logarithm problem for the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ is hard, and its difficulty is comparable to the difficulty of factoring an integer of the size of p. There is a Pollard ρ algorithm for solving this DLOG problem [Pomerance 2008; Teske 2001], but the best currently known algorithm for this group is a sub-exponential index calculus algorithm closely related to the NFS factoring algorithm, with heuristic complexity $L_p[1/3; 4\cdot 3^{2/3}]$; see [Pomerance 2008; Schirokauer 2008].

REMARK 7. Interest in the DLOG problem has been stimulated by cryptographic applications, most notably the famous Diffie–Hellman protocol: If A and B want to have a public conversation which ends with them sharing a secret that no onlooker could reasonably discover, then they start by (publicly) agreeing on a suitable cyclic group G of order n together with a generator g of that group. Then A chooses a random integer $a < n$, and communicates g^a to B, in public. Similarly B chooses a random integer b and transmits g^b to A over a public channel. They can then each privately compute a joint secret

$$s = (g^a)^b = (g^b)^a.$$

The most obvious way for an eavesdropper to defeat this is to intercept g^a, solve the implicit DLOG problem to find a, to intercept g^b and compute $s = (g^b)^a$. More generally, the eavesdropper can succeed if he or she can find g^{ab} knowing g, g^a, and g^b. No efficient algorithm to do this is known if G is a large cyclic subgroup of prime order in $(\mathbb{Z}/p\mathbb{Z})^*$ for some prime p, or for a large cyclic subgroup G of prime order of the group of points $E(\mathbb{F}_p)$ on a suitable elliptic curve E over a finite field $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$. The representation of a cyclic group in the group of points of an elliptic curve seems particularly opaque, and in this case no sub-exponential discrete logarithm algorithms are known at all. For details on these cases, see [Pomerance 2008; Poonen 2008; Schirokauer 2008].

The difficulty of the abstract discrete logarithm problem in a group is dominated by the difficulty of the problem in the largest cyclic subgroup of prime order [Pomerance 2008]. This explains why in the cases of the groups $(\mathbb{Z}/p\mathbb{Z})^*$ and $E(\mathbb{F}_p)$ above, one usually considers cyclic subgroups of prime order.

The largest cyclic subgroup of prime order in a cyclic group G of order 2^n has order 2, and there is a particularly efficient DLOG algorithm for G. Let g be a generator of G. There is a chain of subgroups

$$1 = G_0 \subset G_1 \subset G_2 \subset \cdots \subset G_{n-1} \subset G_n = G,$$

where G_m has 2^m elements. To find the logarithm of $a \in G$ with respect to g, note that $a^{2^{n-1}}$ is of order 1 or 2. In the first case, a lies in G_{n-1}. In the second case, ag lies in the subgroup. In either case we then use recursion.

PROBLEM 10. SQUARE ROOTS MODULO A PRIME: Given an odd prime p and a quadratic residue a, find an x such that $x^2 \equiv a \bmod p$.

We start by showing how to efficiently reduce this problem to QUADRATIC NONRESIDUES. This means that modular square roots can be found efficiently once a quadratic nonresidue is known. Let a be a quadratic nonresidue. Write $p-1 = 2^t q$, where q is odd. The element a^q lies in the subgroup G of $(\mathbb{Z}/p\mathbb{Z})^*$ of order 2^t, and $b = g^q$ is a generator of that group. By the observation above on 2-power cyclic groups, the discrete logarithm in the cyclic 2-group G is easy and we can efficiently find a k such that

$$a^q = b^k.$$

Note that k is even since $a^{(p-1)/2} \equiv 1 \bmod p$. Simple calculation shows that $x = a^{(p-q)/2} b^{k/2}$ is a square root of a:

$$x^2 = a^{(p-q)} b^k = a^{(p-q)} a^q = a^p \equiv a \bmod p.$$

The running time of the this procedure depends on the power of 2 dividing $p-1$.

The conclusion is that square roots modulo p can be found efficiently if quadratic nonresidues can be found efficiently; using this idea, taking square roots modulo a prime is easy in practice, and easy in theory if probabilistic algorithms are allowed.

More recently, René Schoof [1985] discovered a deterministic polynomial-time algorithm for finding square roots of a fixed integer a modulo a prime, using elliptic curves. In practice, this algorithm is not competitive with the probabilistic algorithms above since the exponent on $\log p$ is large. However, Schoof's paper has had an enormous impact on the study of elliptic curves over finite fields [Poonen 2008], since it also pioneered new techniques for finding the order of the group $E(\mathbb{F}_p)$ for large p.

PROBLEM 11. MODULAR SQUARE ROOTS: Given an integer n and an integer a, determine whether a is a square modulo n, and find an x such that $x^2 \equiv a \bmod n$ if x exists.

If the prime factorization of n is known, then the algorithm above can be combined with Hensel's Lemma and the Chinese Remainder Theorem (both discussed later) to find a square root of a modulo n. However, the working algorithmic number theorist is often confronted with integers n whose prime factorization is unknown. In this case, no efficient modular square root algorithm is known.

In fact, more is true: MODULAR SQUARE ROOTS is equivalent, in a sense to be made precise shortly, to FACTORING. This says that in addition to the above fact — that factoring n enables the square root problem to be solved easily —

it is also the case that an algorithm for MODULAR SQUARE ROOTS enables n to be factored efficiently. To factor n, first check that it is odd and not a perfect power. Then choose a random y and apply the hypothetical MODULAR SQUARE ROOTS algorithm to $a = y^2 \bmod n$ to get an x such that

$$x^2 \equiv y^2 \bmod n.$$

Any common divisor of x and y is a factor of n, so assume that x and y have no common divisor larger than 1. If p is a factor of n then p divides $x^2 - y^2 = (x-y)(x+y)$. In addition, it divides exactly one of the factors $x-y$ or $x+y$ (if it divides both, it would divide their sum $2x$ and their difference $2y$).

If y is random, then any odd prime that divides $x^2 - y^2$ has a 50/50 chance of dividing $x+y$ or $x-y$, and any two primes should presumably (i.e., heuristically) have a 50/50 chance of dividing different factors. In that case, greatest common divisor $\gcd(x-y, n)$ will be a proper factor of n. If this fails, try again by choosing another random y. After k choices, the probability that n remains unfactored is 2^{-k}.

Thus FACTORING and MODULAR SQUARE ROOTS are in practice equivalent in difficulty.

REMARK 8. Replacing square roots by eth roots for $e > 2$ leads to a problem closely related to the RSA cryptosystem, perhaps the most famous of all public-key cryptographic systems. Let $n = pq$ be the product of two large primes, and $G = (\mathbb{Z}/n\mathbb{Z})^*$. The RSA cryptosystem uses exponentiation as a mixing transformation on G. A message of arbitrary length is broken a sequence of elements of G, and each element is encrypted separately. If $e > 1$ is an integer relatively prime to $(p-1)(q-1)$ then the encryption map $E: G \to G$ defined by

$$E(x \bmod n) = x^e \bmod n$$

is a bijection that can be computed efficiently using EXP. The decryption mapping is $D(y) = y^d$ where the integer d is defined by the congruence

$$ed \equiv 1 \bmod (p-1)(q-1).$$

The decryption exponent d can be found efficiently, if the factorization of n is known, using the Extended Euclidean Algorithm described in the next section. If n and e are chosen suitably, then finding $D(y)$ without knowing p and q requires the solution of a modular eth roots problem. It is plausible that breaking this system is no easier than factoring.

PROBLEM 12. BOUNDED MODULAR SQUARE ROOTS: Given an integer n and integers a and b, determine whether there is an integer x such that $x < b$ and $x^2 \equiv a \bmod n$.

It may be peculiar to ask for square roots below a given bound, but this additional condition turns out to make the problem much more difficult than MODULAR SQUARE ROOTS, assuming the widely believed $P \neq NP$ conjecture. Specifically, BOUNDED MODULAR SQUARE ROOTS is known to be NP-complete.

Briefly, a decision problem is in P if there is a polynomial-time algorithm for it. A decision problem is in NP if there is an algorithm $f(x, y)$ in P that takes an instance x of a problem and a "certificate" y as input, such that for every "yes" instance x there is a certificate y such that $f(x, y)$ returns "yes." BOUNDED MODULAR SQUARE ROOTS is certainly in NP, as are almost all of the algorithms that we have considered. Indeed, given an instance (a, b, n), the integer x is a certificate: the verifications that $x < b$ and $x^2 \equiv a \bmod n$ can be done in polynomial time. Membership in NP doesn't imply that finding the certificate is easy, but merely that it exists; if certificates could be found easily the problem would be in P.

Finally, a decision problem is NP-complete if it at least as hard as any other problem in NP, in the sense that any other problem in NP can be reduced to it. The notion of reduction needs to be defined precisely, and involves ideas similar to the equivalence of FACTORING and MODULAR SQUARE ROOTS sketched above. For details see [Garey and Johnson 1979].

A discussion of the complexity of number-theoretic problems would be incomplete without mentioning a problem for which no algorithm whatsoever exists.

PROBLEM 13. DIOPHANTINE EQUATIONS: Given a polynomial $f(x_1, \ldots, x_n)$ with integral coefficients in n variables, is there n-tuple of integers x such that $f(x) = 0$?

A famous result of Yuri Matijasevic, building on work of Julia Robinson, Martin Davis, and Hilary Putnam shows that this is an undecidable problem [Matiyasevich 1993; Davis 1973]. Although the problem might be easy for a specific f, there is no algorithm (efficient or otherwise) that takes f as input and always determines whether $f(x) = 0$ is solvable in integers.

3. Euclid's algorithm

Euclid's algorithm, given above, has been an extraordinarily fertile source of algorithmic ideas, and can be viewed as a special case of famous modern algorithms, including Hermite normal form algorithms, lattice basis reduction algorithms, and Gröbner basis algorithms.

The GCD of integers a and b is the unique nonnegative integer d such that $d\mathbb{Z} = a\mathbb{Z} + b\mathbb{Z}$. Here $d\mathbb{Z}$ denotes the principal ideal in the ring of integers consisting of all multiples of d, and $a\mathbb{Z} + b\mathbb{Z} := \{ax + by : x, y \in \mathbb{Z}\}$. To prove

that d exists, it suffices to consider nonzero a and b, in which case d can be taken to be the least positive element of $a\mathbb{Z} + b\mathbb{Z}$. Indeed, if $z = ax + by$ is an arbitrary element of $a\mathbb{Z} + b\mathbb{Z}$ then $z = qd + r$ where $r = z \bmod d$ so that $0 \le r < d$. Since $r = z - qd$ is a nonnegative element of $a\mathbb{Z} + b\mathbb{Z}$, it follows that $r = 0$, and z is a multiple of d as claimed.

This shows that for nonzero a and b the GCD $d = \gcd(a, b)$ is the smallest positive integral linear combination of a and b. In addition, d is the largest integer that divides both a and b and it is the only positive divisor that is divisible by all other divisors. Moreover, it is described in terms of the prime factorizations of positive a and b by

$$\gcd(a, b) = \prod_p p^{\min(a_p, b_p)}, \quad \text{if } a = \prod_p p^{a_p}, \quad b = \prod_p p^{b_p}.$$

The Greeks would have been more comfortable with the following geometric definition of the GCD: if $a > b > 0$ consider a rectangle of width a and height b. Remove a b-by-b square from one end of the rectangle. Continue removing maximal subsquares (squares with one side being a side of the rectangle) until the remaining rectangle is a square. The side length of the remaining square is the "common unit of measure" of a and b, i.e., their GCD.

For instance, if $a = 73$ and $b = 31$, then the removal of two maximal subsquares leaves an 11-by-31 rectangle, the removal of two further maximal squares leaves an 11-by-9 rectangle, the removal of one maximal subsquare leaves a 2-by-9 rectangle, the removal of four maximal subsquares leaves a 2-by-1 rectangle, and the removal of one maximal subsquare leaves a unit square.

This procedure makes sense for arbitrary real numbers, leading to the notion of a continued fraction; this process terminates for an a-by-b rectangle if and only if a/b is a rational number.

3.1. Extended Euclidean algorithm. In many applications of the Euclidean algorithm the identity $a\mathbb{Z} + b\mathbb{Z} = d\mathbb{Z}$ needs to be made explicit; i.e., in addition to finding d, both x and y must be found. To do this it suffices to augment Euclid's algorithm.

EXTENDED EUCLIDEAN ALGORITHM (EEA)

Input: Positive integers a and b

Output: x, y, z where $z = \gcd(a, b)$ and $z = ax + by$

$\{X, Y, Z\} := \{1, 0, a\}$

$\{x, y, z\} := \{0, 1, b\}$

While $z > 0$

$\quad q := \lfloor Z/z \rfloor$

$\quad \{X, Y, Z, x, y, z\} := \{x, y, z, X - qx, Y - qy, Z - qz\}$

Return X, Y, Z

Simple algebra shows that at every step

$$aX + bY = Z, \qquad ax + by = z, \tag{3-1}$$

so that triples (X, Y, Z) encode elements of the ideal $a\mathbb{Z} + b\mathbb{Z}$. Moreover, $Z - qz = Z \bmod z$ so that the Z-values recapitulate the ordinary Euclidean algorithm.

The execution of the algorithm on input $a = 73$ and $b = 31$ is summarized in the following table, where each row describes the state of the computation just after q is computed.

X	Y	Z	x	y	z	q
1	0	73	0	1	31	2
0	1	31	1	−2	11	2
1	−2	11	−2	5	9	1
−2	5	9	3	−7	2	4
3	−7	2	−14	33	1	2
−14	33	1	31	−73	0	

Thus $\gcd(73, 31) = 1$ and $(-14) \cdot 73 + 33 \cdot 31 = 1$.

The reader should notice that from the third row onwards X and Y have opposite signs, and their values increase in absolute value in successive rows. It isn't hard to verify that (for positive a and b) this is always the case.

What is the running time of this algorithm? It can be shown that the number of arithmetic operations required is linear in the input size. The idea of the argument is as follows. The maximal number of iterations are required when the quotients q is always 1. An induction argument shows that if this is the case and n iterations of the loop are required then $a \geq F_{n+2},\ b \geq F_{n+1}$, where F_n denotes the nth Fibonacci number ($F_1 = F_2 = 1,\ F_{n+1} = F_n + F_{n-1}$). On the other hand, the Euclidean algorithm starting with $a = F_{n+1},\ b = F_n$ takes n steps. Using $F_n = O(\phi^n),\ \phi = (1 + \sqrt{5})/2$, it follows that $n = O(\log a)$, as desired. A careful accounting of the bit complexity, using the fact that the size of the integers decreases during the course of the algorithm, shows that the bit complexity is $O(\log^2 a)$.

On occasion it is useful to formulate the Euclidean algorithm in terms of 2-by-2 matrices. Associate the matrix

$$M := \begin{bmatrix} X & Y \\ x & y \end{bmatrix}.$$

to a row X, Y, Z, x, y, z, q. Then by (3-1)

$$M \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} Z \\ z \end{bmatrix}.$$

The matrix M' of the next row is

$$M' = \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix} M.$$

Iterating this in the specific calculation above gives

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 73 \\ 31 \end{bmatrix}, \tag{3-2}$$

so that the Euclidean algorithm can be interpreted as a sequence of 2-by-2 matrix multiplications by matrices of a special form.

All of the ideas in the EEA apply to any commutative ring in which division with "smaller" remainder exists. For instance, the ring $F[x]$ of polynomials over a field F admits a division algorithm for which the remainder polynomial has smaller degree than the divisor. The ring $\mathbb{Z}[i] := \{a + bi : a, b \in \mathbb{Z}\}$ of Gaussian integers inside the complex numbers has this property if size is measured as the absolute value, and we take the quotient in the ring to be the result of taking the quotient in the complex numbers and rounding to the nearest Gaussian integer.

One important application of the EEA is to find inverses in $(\mathbb{Z}/n\mathbb{Z})^*$. If $\gcd(a, n) = 1$ the EEA can be used to find x, y such that $ax + ny = 1$, which implies that $ax \equiv 1 \bmod n$ and $(a \bmod n)^{-1} = x \bmod n$.

3.2. Continued fractions. The *partial quotients* of the continued fraction expansion of a real number α are the terms of the (finite or infinite) sequence $a_0, a_1, \ldots$ defined as follows.

CONTINUED FRACTION OF A REAL NUMBER

Input: A real number α
Output: A sequence a_i of integers
$\alpha_0 := \alpha$
For $i := 0, 1, \ldots$
    Output $a_i := \lfloor \alpha_i \rfloor$
    Halt if $\alpha_i = a_i$
    $\alpha_{i+1} := 1/(\alpha_i - a_i)$

EXAMPLE 1. If $\alpha = 73/31$ the partial quotients are $2, 2, 1, 4, 2$. If $\alpha = \sqrt{11}$ the partial quotients are ultimately periodic: $3, 3, 6, 3, 6, 3, 6, \ldots$.

The reader should verify that this procedure terminates if and only if α is a rational number.

Although written as if it was an algorithm, this is more of a mathematical construct than an algorithm for several reasons: (a) it may fail to terminate, (b) real numbers cannot be represented in finite terms suitable for a computer, and (c) testing real numbers for equality is (algorithmically) problematic.

The relationship between α_i and α_{i+1} can be written as

$$\alpha_i = a_i + \frac{1}{\alpha_{i+1}}. \tag{3-3}$$

Iterating this for $\alpha = 73/31$ gives the finite continued fraction

$$\frac{73}{31} = 2 + \cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{4+\cfrac{1}{2}}}} = [2; 2, 1, 4, 2],$$

where $[a_0; a_1, \ldots, a_n]$ denotes the value of a finite continued fraction with partial quotients a_i. The values of the successive partial continued fractions of a real number are called *convergents* of the continued fraction; e.g., the convergents of the continued fraction for $73/31$ are

$$[2] = 2, \quad [2;2] = \frac{9}{2}, \quad [2;2,1] = \frac{11}{9}, \quad [2;2,1,4] = \frac{31}{11}, \quad [2;2,1,4,2] = \frac{73}{31}.$$

Periodic continued fractions represent quadratic irrational numbers $a + b\sqrt{d}$, where a, b, d are rational numbers (and d is a nonsquare). For instance, if α denotes the purely periodic continued fraction $[3; 6, 3, 6, 3, 6, \ldots]$ then

$$\alpha = 3 + \cfrac{1}{6+\cfrac{1}{\alpha}}\,.$$

Clearing fractions gives $2\alpha^2 - 6\alpha - 1 = 0$, so that $\alpha > 0$ implies that $\alpha = (3 + \sqrt{11})/2$.

PROPOSITION 14. Let $a_0, a_1, \ldots$ be a sequence of real numbers with $a_i > 0$ for $i > 0$. Define a real number $[a_0; a_1, \ldots, a_n]$ recursively by

$$[a_0;] = a_0, \qquad [a_0; a_1, \ldots, a_n, a_{n+1}] = [a_0; a_1, \ldots, a_n + 1/a_{n+1}].$$

Finally, define sequences x_i, y_i recursively by

$$\begin{aligned} x_{-1} &= 0, \quad x_0 = 1, \quad x_{n+1} = a_{n+1}x_n + x_{n-1}, \quad n \geq 0, \\ y_{-1} &= 1, \quad y_0 = a_0, \quad y_{n+1} = a_{n+1}y_n + y_{n-1}, \quad n \geq 0. \end{aligned} \tag{3-4}$$

Then for nonnegative n,

$$y_n x_{n-1} - y_{n-1}x_n = (-1)^{n-1}, \qquad y_n/x_n = [a_0; a_1, \ldots, a_n].$$

Moreover, if the a_i are the partial quotients of the continued fraction of a real number α then the integers x_n and y_n are coprime, and

$$\alpha = \frac{y_n\alpha_{n+1} + y_{n-1}}{x_n\alpha_{n+1} + x_{n-1}}. \tag{3-5}$$

REMARK. Several notations are commonly used for convergents; the choice here is dictated by a geometric interpretation to be given shortly.

PROOF. The coprimality of x_n and y_n follows once $x_n y_{n-1} - x_{n-1} y_n = (-1)^n$ is proved. This and all of the other assertions follow from straightforward induction arguments. For instance, assuming that $y_n/x_n = [a_0; a_1, \ldots, a_n]$ for a given n and arbitrary a_i we have

$$\begin{aligned}\frac{y_{n+1}}{x_{n+1}} &= \frac{a_{n+1}y_n + y_{n-1}}{a_{n+1}x_n + x_{n-1}} = \frac{a_{n+1}(a_n y_{n-1} + y_{n-2}) + y_{n-1}}{a_{n+1}(a_n x_{n-1} + x_{n-2}) + x_{n-1}} \\ &= \frac{(a_n + 1/a_{n+1})y_{n-1} + y_{n-2}}{(a_n + 1/a_{n+1})x_{n-1} + x_{n-2}} \\ &= [a_0; a_1, \ldots, a_n + 1/a_{n+1}] = [a_0; a_1, \ldots, a_n, a_{n+1}]. \qquad \square\end{aligned}$$

As with the Euclidean algorithm, it is sometimes convenient to formulate continued fractions in terms of 2-by-2 matrices, e.g., defining

$$M_n := \begin{bmatrix} a_n & 1 \\ 1 & 0 \end{bmatrix}, \qquad P_n := \begin{bmatrix} y_n & y_{n-1} \\ x_n & x_{n-1} \end{bmatrix} \tag{3-6}$$

and observing that (3-4) implies that

$$P_n = M_0 M_1 \cdots M_n. \tag{3-7}$$

A careful look at the convergents of the continued fraction of $73/31$ reveals that they recapitulate the Euclidean algorithm! This is easy to verify using

$$\begin{bmatrix} a & 1 \\ 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & -a \end{bmatrix}.$$

Indeed, multiplying the relation $\left[\begin{smallmatrix} 73 \\ 31 \end{smallmatrix}\right] = P_4 \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ repeatedly on the left by the M_k^{-1} gives the earlier EEA formula (3-2); moreover, this process can be reversed to show that the EEA yields a continued fraction.

Since $\det(M_k) = -1$ and $\det(P_n) = y_n x_{n-1} - x_n y_{n-1}$ the product formula (3-7) gives another proof of the first statement of the proposition:

$$y_n x_{n-1} - y_{n-1} x_n = (-1)^{n+1}. \tag{3-8}$$

3.3. Rational approximation. Let $\alpha, \alpha_n, a_n, y_n, x_n$ be as in the previous section. We wish to quantify the sense in which y_n/x_n is a good approximation to α. Dividing (3-8) by $x_n x_{n-1}$ gives

$$\frac{y_n}{x_n} - \frac{y_{n-1}}{x_{n-1}} = \frac{(-1)^{n+1}}{x_n x_{n-1}}.$$

Iterating this identity and using $y_0/x_0 = a_0$ gives the useful formula

$$\frac{y_n}{x_n} = a_0 + \frac{1}{x_0x_1} - \frac{1}{x_1x_2} + \cdots + (-1)^n \frac{1}{x_nx_{n+1}}. \tag{3-9}$$

Since the $x_n = a_n x_{n-1} + x_{n-2}$ are strictly increasing it follows that the sequence of convergents y_n/x_n converges.

THEOREM 15. The sequence y_n/x_n converges to α. For even n the convergents increase and approach α from below; for odd n the convergents decrease and approach α from above. Both $y_n/x_n - \alpha$ and $y_n - x_n\alpha$ alternate in sign, and decrease in absolute value. Moreover,

$$\frac{1}{x_{n+2}} < |y_n - \alpha x_n| < \frac{1}{x_{n+1}}.$$

PROOF. From (3-5),

$$y_n - \alpha x_n = y_n - \frac{x_n(y_n\alpha_{n+1} + y_{n-1})}{x_n\alpha_{n+1} + x_{n-1}} = \frac{x_{n-1}y_n - x_n y_{n-1}}{x_n\alpha_{n+1} + x_{n-1}} = \frac{(-1)^{n+1}}{x_n\alpha_{n+1} + x_{n-1}}.$$

From $a_{n+1} = \lfloor \alpha_{n+1} \rfloor < \alpha_{n+1} < a_{n+1} + 1$ we get

$$\begin{aligned} x_{n+1} &= x_n a_{n+1} + x_{n-1} \\ &< x_n\alpha_{n+1} + x_{n-1} < x_n(a_{n+1} + 1) + x_{n-1} = x_{n+1} + x_n \le x_{n+2}, \end{aligned}$$

and the desired inequalities in the theorem follow by taking reciprocals. It follows immediately that $\lim y_n/x_n = \alpha$ and the other statements follow from (3-9) and basic facts about alternating series. □

COROLLARY 16. Any convergent y_n/x_n in the continued fraction for α satisfies $|y_n/x_n - \alpha| < 1/x_n^2$.

It is convenient to formalize the notion of a good approximation to α by rational numbers. If (x, y) is a point in the XY-plane, then the distance from (x, y) to the line $Y = \alpha X$ along the vertical line $X = x$ is $|y - \alpha x|$. Say that (x, y), $x > 0$, is a *best approximator* to α if x and y are coprime and (x, y) has the smallest vertical distance to $Y = \alpha X$ among all integer points with denominator at most x, i.e.,

$$|y - \alpha x| < |v - u\alpha| \quad \text{for all integers } u, v \text{ with } 0 < u < x.$$

The following theorem says that being a best approximator is equivalent to being a convergent, and it gives an explicit inequality for a rational number y/x that is equivalent to being a convergent.

THEOREM 17. Let α be an irrational real number and x, y a pair of coprime integers with $x > 0$. Then the following are equivalent:

(a) y/x is a convergent to α.

(b) If x' is a multiplicative inverse of y mod x (in the sense that $yx' \equiv 1 \bmod x$ and $1 \le x' < x$), then

$$\frac{-1}{x(2x-x')} < \frac{y}{x} - \alpha < \frac{1}{x(x+x')}. \tag{3-10}$$

(c) y/x is a best approximator to α.

COROLLARY 18. If y/x is a rational number such that $\left|\frac{y}{x} - \alpha\right| < \frac{1}{2x^2}$, then y/x is a convergent to α.

PROOF OF THE COROLLARY. Without loss of generality, x and y are coprime. If x' is as in the theorem, then $1 \le x' < x$, giving $2x - x' < 2x$ and $x + x' < 2x$. Multiplying by x and taking reciprocals gives

$$\frac{-1}{x(2x-x')} < \frac{-1}{2x^2} < \frac{y}{x} - \alpha < \frac{1}{2x^2} < \frac{1}{x(x+x')},$$

and the corollary follows immediately from the theorem. □

The convergents in the continued fraction of α determine lattice points (i.e., points with integer coordinates) in the plane, and they are unusually close to the line $Y = \alpha X$. The convergents alternate back and forth over the line, and each point (x, y) is closer to the line than all points with smaller X. Since any best approximator y_n/x_n comes from a convergent, it follows that the open parallelogram $\{(x, y) : 0 < x < x_{n+1},\ |y - \alpha x| < |y_n - \alpha x_n|\}$ contains no lattice points other than the origin.

If you are standing at the origin of the plane looking in the direction of $Y = \alpha X$, then the lattice points $(x, y) \in \mathbb{Z}^2$ that are closest to the line occur alternately on both sides of the line, and become very close to the line. It is difficult to illustrate this graphically because the convergents approximate so well that they quickly appear to lie on the line. In Figure 1 an attempt has been made to convey this by greatly distorting the scale on the y-axis; specifically, a point at distance d from the line $Y = \alpha X$ is displayed at a distance that is proportional to $d^{2/5}$. In that figure, α is the golden ratio $\phi = [1; 1, 1, 1, \ldots]$, and the coordinates of the convergents (F_n, F_{n+1}) are consecutive Fibonacci numbers. (The figure suggests a fact, brought to our attention by Bill Casselman, which the reader may enjoy proving: The even convergents (x_n, y_n) are the vertices on the convex hull of the integer lattice points that lie below the line $Y = \alpha X$, and the odd convergents are the convex hull of the integer lattice points above the line.)

Before proving Theorem 17, it is convenient to present a lemma.

LEMMA 19. Let x, x', y, y' be integers with $yx' - y'x = \pm 1$, and let α lie in between y/x and y'/x'. Then there are no lattice points in the interior of the parallelogram bounded by the lines $X = x$, $X = x'$, the line through (x, y) with slope α, and the line through (x', y') with slope α.

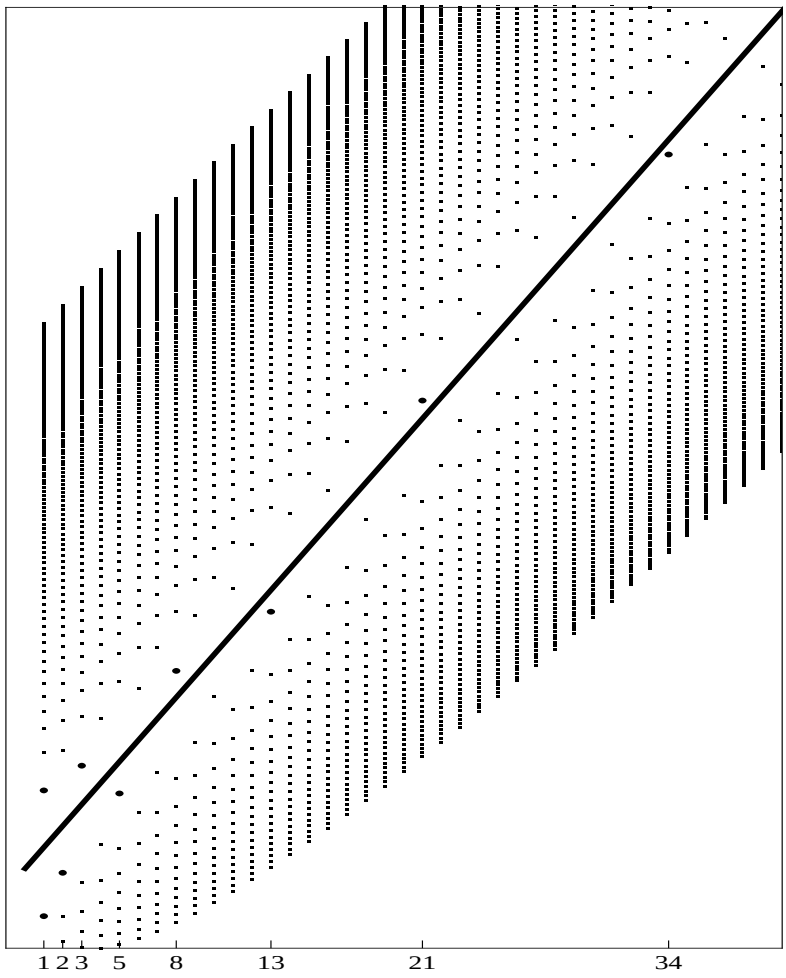

Figure 1. Approximations to the golden ratio ϕ.

PROOF OF THE LEMMA. To fix the ideas, assume that $0 < x' < x$ and $y' - \alpha x' < 0 < y - \alpha x$ (any other case is similar). Let (u, v) be a lattice point with $x' < u < x$. Then (u, v) is in the stated parallelogram if and only if

$$y' - \alpha x < v - \alpha u < y - \alpha x. \tag{3-11}$$

The condition $yx' - y'x = \pm 1$ implies that there are integers r, s such that

$$u = rx' + sx, \qquad v = ry' + sy.$$

The first equation implies that r and s are nonzero and have opposite sign; subtracting α times the first equation from the second gives

$$v - \alpha u = r(y' - \alpha x') + s(y - \alpha x).$$

Since $y-\alpha x$ and $y'-\alpha x'$ have opposite sign, the terms on the right hand side have the same sign. This implies that $|v-\alpha u| \geq \max(|y-\alpha x|, |y'-\alpha x'|)$ which contradicts (3-11), showing that (u, v) does not lie in the parallelogram and proving the Lemma. □

PROOF OF THEOREM 17. (a) ⇒ (b), i.e., convergents satisfy the inequalities (3-10): Assume that $y/x = y_n/x_n$ is a convergent, and that y_{n-1}/x_{n-1} is the preceding convergent. If n is odd then $yx_{n-1} - y_{n-1}x = 1$ and $x' = x_{n-1}$. From Theorem 15,

$$0 < y-\alpha x = \frac{1}{x_n\alpha_{n+1}+x_{n-1}} < \frac{1}{x_n+x_{n-1}}$$

and the inequalities (3-10)

$$\frac{-1}{x(2x-x')} < 0 < \frac{y}{x}-\alpha < \frac{1}{x(x+x')}$$

follow. The case for even n is similar except that $y/x-\alpha$ is negative, $yx_{n-1} - y_{n-1}x = -1$ requires us to take $x' = x-y$, and the left inequality in (b) is the nontrivial one.

(b) ⇒ (c), the inequalities (3-10) imply that y/x is a best approximator: Assume that $x > 0$, x and y are coprime, and y/x satisfies the inequalities. Let (u, v) be a supposedly better approximator than (x, y) so that $0 < u \leq x$. There are now two cases. First, suppose that $y-\alpha x > 0$. Choose x' and y' such that $yx'-xy' = 1$. Subtracting $y/x-y'/x' = 1/xx'$ from $y/x-\alpha < 1/(x(x+x'))$ gives

$$\frac{y'}{x'}-\alpha < \frac{1}{x(x+x')}-\frac{1}{xx'} = \frac{-1}{(x+x')x}\ , \tag{3-12}$$

showing that $y'-x'\alpha$ is negative. The (proof of the) Lemma now shows that an inequality of the form (3-11) is impossible, which implies that (x, y) is a best approximator.

The case $y-\alpha x < 0$ is similar and will be left to the reader.

(c) ⇒ (a): We show that if (u, v) isn't a convergent then v/u isn't a best approximator. If (u, v) isn't a convergent there is an n such that $x' = x_{n-1} < u \leq x = x_n$. Setting $y' = y_{n-1}$, $y = y_n$, the Lemma now applies, and (u, v) isn't a best approximator unless $u = x_n$ in which case v/u isn't a best approximator unless $v/u = y_n/x_n$. □

4. Modular polynomial equations

The goal of this section is to consider the problem of solving polynomial congruences $f(x) \equiv 0 \bmod n$, i.e., to finding roots of polynomials in $(\mathbb{Z}/n\mathbb{Z})[X]$.

We consider the successively more general cases in which n is a prime, prime power, and arbitrary positive integer.

4.1. Equations modulo primes. The problem of finding integer solutions to polynomial equations modulo a prime p is (essentially) the problem of finding roots to polynomials

$$f(X) \in \mathbb{F}_p[X]$$

with coefficients in the field with p elements. Fermat's Little Theorem say that every element of $\mathbb{F}_p$ is a root of the polynomial $X^p - X$; comparing degrees and leading coefficients shows that

$$X^p - X = \prod_{a \in \mathbb{F}_p} (X - a) = \prod_{a=0}^{p-1} (X - a) \ \in \mathbb{F}_p[X].$$

It follows that if $f(X) \in \mathbb{F}_p[X]$ is an arbitrary polynomial, the gcd of $f(X)$ and $X^p - X$ is the product of all linear factors $X - a$, where a is a root of f.

Similarly the roots of $X^{(p-1)/2} = 1$ are the quadratic residues modulo p, so

$$X^{(p-1)/2} - 1 = \prod_{a \in R_p} (X - a) \in \mathbb{F}_p[X]\,,$$

where R_p denotes the set of quadratic residues modulo p. This suggests the following elegant algorithm for finding all roots of a polynomial in $\mathbb{F}_p[X]$.

CANTOR–ZASSENHAUS (f, p)

Input: A prime p and a polynomial $f \in \mathbb{F}_p[X]$
Output: A list of the roots of f
1. $f := \gcd(f, X^p - X)$
2. If f has degree 1
   Return the root of f
3. Choose b at random in $\mathbb{F}_p$
   $g := \gcd(f(X), (X + b)^{(p-1)/2} - 1)$
   If $0 < \deg(g) < \deg(f)$ then
      Return CANTOR–ZASSENHAUS(g,p)
         $\cup$ CANTOR–ZASSENHAUS$(f/g,p)$
   Else, go to Step 3

The correctness follows from the earlier remarks — at Step 3, $f(X)$ is a product of distinct linear factors $X - a$, and then g is the product of all $X - a$ such that $a + b$ is a quadratic residue. If b is random then we would expect about half of the linear factors of f to divide g.

Moreover, the algorithm takes (probabilistic) polynomial time since the input size for a (general) polynomial over $\mathbb{F}_p$ is $n \log p$ and all operations take time bounded by a polynomial in n and $\log p$.

4.2. Equations modulo prime powers. Next we consider prime powers. The key idea is due to Kurt Hensel and gives an explicit construction that applies in sufficiently favorable situations to allow us to "lift" a solution to $f(X) \equiv 0 \bmod p^n$ to a solution modulo p^{2n}.

The idea is illustrated cleanly by the example of square roots modulo prime powers. Suppose that we have found a square root of $a \bmod p$, i.e., and integer x such that $x^2 \equiv a \bmod p$. Let $y = (x^2+a)/(2x) \bmod p^2$. (Since $2x$ is invertible mod p it is invertible mod p^2.) Some algebraic juggling shows that $y^2 \equiv a \bmod p^2$. The formula for y comes from Newton's Lemma from calculus:

$$y = x - \frac{f(x)}{f'(x)} = x - \frac{x^2-a}{2x} = \frac{x^2+a}{2x}.$$

In any event, a root mod p of $f(X) = X^2 - a \bmod p$ has been lifted to a root mod p^2. The general story is as follows.

THEOREM 20 (HENSEL'S LEMMA). Let p be a prime, $f(x) \in \mathbb{Z}[X]$ a polynomial with integer coefficients, and $a \in \mathbb{Z}$ an integer such that

$$f(a) \equiv 0 \bmod p^k, \qquad f'(a) \not\equiv 0 \bmod p\ .$$

Then $b \equiv a - f(a)/f'(a) \bmod p^{2k}$ is the unique integer modulo p^{2k} that satisfies

$$f(b) \equiv 0 \bmod p^{2k}, \qquad b \equiv a \bmod p^k.$$

REMARK 9. Since $u = f'(a)$ is relatively prime to p it is relatively prime to any power of p and has an inverse modulo that power, computable with the EEA. Thus the division in the formula in the theorem makes sense. Inverses modulo high powers can be found using Newton's method (for rational functions): find the inverse of u modulo prime powers by finding roots of $f(X) = u - 1/X$. Specifically, if $au \equiv 1 \bmod p^k$ then $bu \equiv 1 \bmod p^{2k}$, where $b = a - f(a)/f'(a) = a(2-au)$.

PROOF. Replacing $f(X)$ by $f(X+a)$, it suffices to prove the result when $a = 0$. Thus we are trying to find a root of $f(X) \equiv 0 \bmod p^{2k}$, given

$$f(X) = c_0 + c_1 X + c_2 X^2 + \cdots, \quad c_0 = f(0),\ c_1 = f'(0).$$

By hypothesis, $c_0 \equiv 0 \bmod p^k$ and $c_1 \not\equiv 0 \bmod p$. It follows that $b = -c_0/c_1$ is the unique integer modulo p^{2k} such that $b \equiv 0 \bmod p^k$ and $f(b) \equiv 0 \bmod p^{2k}$, finishing the proof. □

The reader will find it instructive to find a square root of, say, -11 modulo 3^8.

The elementary form of Hensel's Lemma above is equivalent to a more elegant and general statement in the p-adic numbers. Since p-adic numbers (and, more generally, non-archimedean local fields) arise later in this volume, we digress to explain this idea. See [Cassels 1986], [Koblitz 1984], and [Serre 1973] for a much more thorough treatment.

Fix a prime p and a constant ρ such that $0 < \rho < 1$. An absolute value can be defined on the rational numbers $\mathbb{Q}$ by

$$|x|_p = \rho^n,$$

where $x = p^n y,\ n = v_p(x)$ is the unique integer (positive or negative) such that y is a rational number whose numerator and denominator are coprime to p. By convention, $|0|_p = 0$.

Under this absolute value, powers of p are "small." The absolute value satisfies the "non-archimedean" inequality $|x + y|_p \le \max(|x|_p, |y|_p)$, which is stronger than the triangle inequality.

The absolute value $|x|_p$ gives the rational numbers $\mathbb{Q}$ the structure of a metric space by setting the distance between x and y to be $|x-y|_p$. The completion $\mathbb{Q}_p$ of this metric space is called the *p-adic numbers*, and it can be shown that the p-adic numbers are a locally compact topological field under natural extensions of the field operations and metric to the completion. The field is easily seen to be independent of the choice of ρ, though in some circumstances in algebraic number theory it is convenient to make the choice $\rho = 1/p$.

A nonzero element a of $\mathbb{Q}_p$ can be represented concretely as a "Laurent series in p,", i.e.,

$$a = a_k p^k + a_{k+1} p^{k+1} + a_{k+1} p^{k+2} + \cdots,$$

where k is an integer, the digits a_n are integers chosen in the range $0 \le a_n < p$, and $a_k \neq 0$. Moreover, $v_p(a) = k$ and $|a| = \rho^k$. Arithmetic operations are easy to visualize in this Laurent representation, at least approximately, by thinking of p^n as being very small if n is very large (analogous in some ways to the usual realization of real numbers as decimal expansions). The subset for which $v_p(a) \ge 0$ is the unit disk

$$\mathbb{Z}_p = \left\{ \sum_{n \ge 0} a_n p^n \right\} = \{x \in \mathbb{Q}_p : |x|_p \le 1\},$$

which is a subring of $\mathbb{Q}_p$ and is called the ring of *p-adic integers*.

The p-adic numbers are a bit peculiar when first encountered, but it turns out that they beautifully capture the idea of calculating "modulo arbitrarily large

powers of p." The reader might enjoy proving that

$$-1 = 1 + 2 + 2^2 + 2^3 + \cdots \quad \text{in } \mathbb{Q}_2,$$

and the fact that infinite series in $\mathbb{Q}_p$ converge if and only if their terms go to zero.

Suppose that we are interested in roots of $f(X) \in \mathbb{Z}_p[X]$. (By multiplying by a power of p the problem of finding roots for $f(X) \in \mathbb{Q}_p[X]$ readily reduces to $f(X) \in \mathbb{Z}_p[X]$.) Finding roots $x = a_0 + a_1 p + a_2 p^2 + \cdots$ to $f(x) = 0$ amounts to solving "$f(x) \equiv 0$ modulo p^∞," and Hensel's Lemma can be translated to the statement that for $x \in \mathbb{Z}/p\mathbb{Z}$ with $|f(x)|_p < \varepsilon$ and $|f'(x)|_p = 1$, there is a unique $y \in \mathbb{Z}/p\mathbb{Z}$ such that

$$f(y) = 0, \qquad |y - x| < \varepsilon.$$

A more general form of Hensel's Lemma, which can be proved along the lines above, says that if $|f(x)| < |f'(x)|^2$ then there is a unique $y \in \mathbb{Z}_p$ such that $f(y) = 0,\ |y - x| \le |f(x)/f'(x)^2|$.

The problem of finding roots is a special case of finding factors of a polynomial, and an even more general form of Hensel's Lemma guarantees that approximate factors can, in suitable circumstances, be lifted to p-adic factors.

If f is a polynomial with p-adic integer coefficients then let $f \bmod p$ denote the polynomial obtained by looking at the coefficients modulo p.

THEOREM 21. Assume that $f \in \mathbb{Z}_p[X]$ factors modulo p as $f \bmod p = gh \in (\mathbb{Z}/p\mathbb{Z})[x]$ and g and h are coprime in $(\mathbb{Z}/p\mathbb{Z})[x]$. Then there are unique G and H in $\mathbb{Z}_p[X]$ such that $f = GH$ and G and H lift g and h in the sense that $G \bmod p = g$ and $H \bmod p = h$.

We sketch a proof of the theorem. Since p-adic integers are approximated arbitrarily closely by (ordinary) integers, it suffices to prove a statement about integer polynomials: if $f \in \mathbb{Z}[x]$ satisfies $f \equiv gh \bmod p$ where g and h are coprime modulo p (i.e., the elements that they determine in $\mathbb{F}_p[X]$ are coprime), then for arbitrarily large n there are integer polynomials G, H that are congruent to g, h, respectively, modulo p and satisfy $f \equiv GH \bmod p^n$.

To do this it turns out to be essential to simultaneously lift a certificate of the fact that g and h are relatively prime. Assume that we are given polynomials f, g, h, r, s with integer coefficients such that

$$f \equiv gh \bmod p^k, \qquad rg + sh \equiv 1 \bmod p^k,$$

where $\deg(r) < \deg(h)$, $\deg(s) < \deg(g)$; our goal is to find G, H, R, S satisfying a similar congruence modulo p^{2k}. Note that the hypotheses of the theorem imply that r and s exist for $k = 1$, using the Euclidean Algorithm for polynomials over $\mathbb{F}_p$. The pair r, s is said to be a *coprimality certificate* of g and $h \bmod p^k$.

Before proving the theorem it is useful to show how to use a coprimality certificate mod p^k to represent any polynomial as a linear combination of g and h modulo p^k.

LEMMA 22. If $rg+sh \equiv 1 \bmod p^k$, then for all $u \in \mathbb{Z}[X]$ there are $A, B \in \mathbb{Z}[X]$ such that

$$Ag + Bh \equiv u \bmod p^k.$$

If $\deg(u) < \deg(g) + \deg(h)$ then we can take $\deg(A) < \deg(h)$, $\deg(B) < \deg(g)$.

PROOF. Multiplying the assumed equation by u gives $u \equiv rgu+shu = (ru)g + (su)h \bmod p^k$ so the first statement is immediate. To verify the assertion about degrees, we have to work harder. Let A be the remainder when ru is divided by h, i.e., $ru = q_1h + A$, $\deg(A) < \deg(h)$. Similarly, let B be the remainder when su is divided by g, i.e., $su = q_2g + B$, $\deg(B) < \deg(g)$. Then

$$\begin{aligned} Ag + Bh &= (ru - q_1h)g + (su - q_2g)h \\ &= (rg + sh)u - (q_1 + q_2)gh \equiv u + Qgh \bmod p^k, \end{aligned}$$

where $Q = -q_1 - q_2$. Since gh is monic of degree $\deg(g) + \deg(h)$ and all other terms in the congruence have degree strictly less than $\deg(g) + \deg(h)$ modulo p^k, it follows that $Q \equiv 0 \bmod p^k$, finishing the proof. □

Now we prove the theorem by showing how to lift the factorization, and lift the certificate, i.e., find the desired G, H, R, S given g, h, r, s.

To lift the factorization write $f = gh + p^k u$, and use Lemma 22 to find A, B with $u \equiv Ag + Bh \bmod p^k$; it is straightforward to check that $G = g + p^k B$, and $H = h + p^k A$ satisfy $f \equiv GH \bmod p^{2k}$.

To lift the certificate $rg + sh = 1 \bmod p^k$, write $rg + sh = 1 + p^k u$, use Lemma 22 to find $u \equiv Ag+Bh \bmod p^k$, and check that $RG+SH \equiv 1 \bmod p^{2k}$ where $R = r + p^k A$ and $S = s + p^k B$.

4.3. Chinese Remainder Theorem. Finally, we come to the case of solving polynomial equations modulo arbitrary positive integers n, given that we can solve the equations modulo the prime power divisors of n. The Chinese Remainder Theorem gives an immediate way for doing this, by giving an explicit recipe for using solutions $f(x_1) \equiv 0 \bmod n_1$, $f(x_2) \equiv 0 \bmod n_2$ to produce an x in $\mathbb{Z}/(n_1n_2)\mathbb{Z}$ such that $f(x) \equiv 0 \bmod n_1n_2$ when the n_i are coprime.

THEOREM 23 (CHINESE REMAINDER THEOREM). If m and n are coprime, and a and b are arbitrary integers, then there is an integer c such that

$$c \equiv a \bmod m, \qquad c \equiv b \bmod n.$$

Any two such integers c are congruent modulo mn.

PROOF. Apply the EEA to m and n to find x and y such that $mx + ny = 1$. The integer

$$c = any + bmx$$

is obviously congruent to a mod m and b mod n.

If c' also satisfies these congruences then $d = c - c'$ is divisible by the coprime integers m and n, and hence by mn. Thus c and c' are congruent modulo mn as claimed. □

This can be stated more algebraically as follows.

COROLLARY 24. If $\gcd(m, n) = 1$ then the rings $\mathbb{Z}/(mn)\mathbb{Z}$ and $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ are isomorphic via the map

$$x \bmod mn \mapsto (x \bmod m, x \bmod n).$$

Also, $\phi(mn) = \phi(m)\phi(n)$ where ϕ is the Euler ϕ-function.

Indeed, the first statement in the Chinese Remainder Theorem above says that the map is surjective, and the second says that it is injective. Units in the direct product of two rings are pairs (u, v) where u and v are units in the respective rings, so the multiplicativity of φ follows from the first statement.

As promised, the theorem shows how to combine modular solutions to polynomial equations: if $f(a) \equiv 0 \bmod m$ and $f(b) \equiv 0 \bmod n$ then apply the Chinese Remainder Theorem to find a c such that $c \equiv a \bmod m$ and $c \equiv b \bmod n$; these congruences imply that $f(c) \equiv 0 \bmod mn$. By induction, this technique generalizes to more than 2 moduli: if $n_1, n_2, \ldots, n_k$ are pairwise coprime, and a_i are given then there is an integer x, unique modulo the product of the n_i, such that $x \equiv a_i \bmod n_i$ for $1 \le i \le k$.

In fact, there are two natural inductive proofs. The product of k moduli $n = \prod n_i$ could be reduced to the $k = 2$ case by $n = n_1 m$, where m is the product of the n_i for $i > 1$, or $n = m_1 m_2$ where each m_i is the product of approximately half of the n_i. The latter method is significantly more efficient in practice: with reasonable assumptions, the first method takes $O(k^2)$ arithmetic operations, and the second takes $O(k \log k)$ arithmetic operations.

5. Quadratic extensions

A quadratic extension K of a field F is a field containing F such that the dimension of K as an F-vector space is 2. This means that K is isomorphic to $F[X]/(f(X))$ where $f(X) \in F[X]$ is an irreducible quadratic polynomial. More concretely, $K = \{a + b\alpha : a, b \in F\}$ where $\alpha \in K$ satisfies an irreducible quadratic equation with coefficients in F. Finally, if $F = \mathbb{Q}$ then any quadratic extension is (isomorphic to) a subset of the complex numbers of the form $\mathbb{Q}(\sqrt{d}) = \{a + b\sqrt{d}\}$, where d is a nonsquare in $\mathbb{Q}$.

In this section we consider four algorithms that are either illuminated by, or directly apply to, quadratic field extensions.

5.1. Cipolla. Let p be an odd prime. A natural and direct algorithm, originally due to Cipolla [1902], finds square roots modulo p by taking a single exponentiation in a quadratic extension of $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. Unlike the modular square root algorithm presented earlier, it has a running time that is independent of the power of 2 that divides $p-1$.

A polynomial $f(X) = X^2 - aX + b \in \mathbb{F}_p[X]$ is irreducible if and only if its discriminant $D := a^2 - 4b$ is a quadratic nonresidue. Let $\alpha = [X]$ be the coset of X in $K := \mathbb{F}_p[X]/(f(X))$ so that K can be viewed as the set of linear polynomials $\{a + b\alpha : a, b \in \mathbb{F}_p\}$ added in the usual way, and multiplied using the identity $\alpha^2 = a\alpha - b$.

In any field that contains $\mathbb{F}_p$ the map $\varphi(x) = x^p$ is a field automorphism of K, sometimes called the Frobenius map. Indeed, it is obvious that φ is multiplicative, and $\varphi(x + y) = (x + y)^p = x^p + y^p$ follows from the fact that the interior binomial coefficients are divisible by p. Moreover $\varphi(x) = x$ if and only if $x \in \mathbb{F}_p$, since every element of $\mathbb{F}_p$ is a root of the polynomial $x^p - x$, and a polynomial of degree p can have at most p roots in a field.

CIPOLLA'S MODULAR SQUARE ROOT ALGORITHM

Input: An odd prime p and a quadratic residue $b \in \mathbb{F}_p$
Output: $u \in \mathbb{F}_p$ such that $u^2 = b$
1. Select random a until $a^2 - 4b$ is a nonresidue modulo p
2. Return $\alpha^{(p+1)/2}$, where $\alpha :=$ a root of $x^2 - ax + b$

Once an a has been found in Step 1, then $f(X) = X^2 - aX + b$ is irreducible, and the above notation applies. Apply the Frobenius automorphism φ to the equation $\alpha^2 - a\alpha + b = 0$ to find that $\beta := \alpha^p$ is also a root of f, so that $f(X) = (X - \alpha)(X - \beta)$. Comparing coefficients gives

$$b = \alpha \cdot \beta = \alpha \cdot \alpha^p = \alpha^{(p+1)}$$

so that $\alpha^{(p+1)/2}$ is a square root of b, proving correctness of the algorithm. Although the exponentiation is done in K, the final result lies in $\mathbb{F}_p$.

EXAMPLE 2. Let $p = 37$, $a = 34$. Easy Legendre symbol calculations show that 34 is a quadratic residue modulo 37, and $b = 1$ produces a nonresidue in Step 1, so that $f(x) = x^2 - x + 34$ is irreducible. We compute $\alpha^{(p+1)/2} = \alpha^{19}$ using EXP; at each step of the calculation we replace α^2 by $\alpha - 34 = \alpha + 3$ and reduce

the coefficients modulo 37. The following table summarizes the calculation:

k	α^k
2	$\alpha + 3$
4	$\alpha^2 + 6\alpha + 9 = 7\alpha + 12$
8	$49\alpha^2 + 168\alpha + 144 = 32\alpha + 32 = -5(\alpha + 1)$
9	$-5(\alpha^2 + \alpha) = -5(2\alpha + 3)$
18	$25(4\alpha^2 + 12\alpha + 9) = 30\alpha + 7$
19	$30\alpha^2 + 7\alpha = 16$

As predicted, the final result lies in the field with 37 elements, and $16^2 \equiv 34 \bmod 97$.

5.2. Lucas–Lehmer. Theorem 5 gives a method for verifying primality of n when the factorization of $n-1$ is known. A similar primality test, attributed to Lucas and Lehmer, enables the primality of n to be established efficiently if the prime factorization of $n+1$ is known.

Let $f(X) = X^2 - aX + b \in \mathbb{Z}[x]$ be an irreducible polynomial with integer coefficients, $D := a^2 - 4b$ its discriminant, and $R := \mathbb{Z}[X]/(f(X))$. Let $\alpha := [X]$ and $\beta := a - \alpha$ be the two roots of f in R.

For $k \geq 0$ define $V_k = \alpha^k + \beta^k \in R$. Both sequences $\{\alpha^k\}$ and $\{\beta^k\}$ satisfy the recursion

$$s_{k+1} = as_k - bs_{k-1},$$

so their sum V_k also does, by linearity. Since $V_0 = 2$ and $V_1 = a$ are integers it follows that all of the V_k are integers. More elaborate "doubling" recursions are easy to discover, allowing the V_k to be computed in $O(\log k)$ arithmetic operations; these are equivalent to using EXP to calculate high powers of a matrix, noticing that

$$\begin{bmatrix} V_{k+2} & V_{k+1} \\ V_{k+1} & V_k \end{bmatrix} = \begin{bmatrix} a & -b \\ 1 & 0 \end{bmatrix}^k \begin{bmatrix} V_2 & V_1 \\ V_1 & V_0 \end{bmatrix}.$$

THEOREM 25. Let n be an odd positive integer, and a, b integers such that $D = a^2 - 4b$ satisfies

$$\left(\frac{D}{n}\right) = -1.$$

Define V_k as above, and let $m = (n+1)/2$. If $V_m \equiv 0 \bmod n$ and $\gcd(V_{m/q}, n) = 1$ for all prime divisors q of m, then n is prime.

PROOF. Let p be a prime divisor of n with

$$\left(\frac{D}{p}\right) = -1.$$

Working modulo p, i.e., in

$$K := R/pR = \mathbb{Z}[x]/(p, f(x)) = \mathbb{F}_p[x]/(\overline{f}(x)),$$

we see that $\overline{f}(x) \in \mathbb{F}_p[x]$ is irreducible so that the ideas of the preceding section apply. Since the roots α and β of an irreducible quadratic polynomial are nonzero, we can define

$$u = \beta/\alpha = \alpha^p/\alpha = \alpha^{p-1} \in K.$$

The equation $V_m \equiv 0 \bmod p$ implies that $\alpha^m = -\beta^m$, i.e., $u^m = -1$ and $u^{2m} = 1$. Similarly, the equations $V_{m/q} \not\equiv 0 \bmod p$ imply that $u^{2m/q} \neq 1$ for all prime divisors q of m.

Therefore u has order $2m = n + 1$ in K. Since

$$u^{p+1} = \left(\alpha^{p-1}\right)^{p+1} = \alpha^{p^2-1} = 1,$$

it follows that $p + 1$ is divisible by $n + 1$, which implies that $n = p$, i.e., that n is a prime as claimed. □

REMARK 10. As in the earlier primality test relying on the factorization of $n-1$, it can be shown that it suffices to restrict attention to primes q that divide a factor $F > \sqrt{n}$ of $n + 1$.

5.3. Units in quadratic fields. If d is a rational number that is not a perfect square, then the quadratic extension $F = \mathbb{Q}(\sqrt{d}) = \{a+b\sqrt{d} : a, b \in \mathbb{Q}\}$ over $\mathbb{Q}$ is unchanged if we replace d by de^2. We will from now on assume that d is a squarefree integer, i.e., not divisible by any perfect square $n^2 > 1$. If $d < 0$ then $\mathbb{Q}(\sqrt{d})$ is said to be an *imaginary* quadratic field, and if $d > 0$ $\mathbb{Q}(\sqrt{d})$ is said to be a *real* quadratic field.

A *number field* is a field having finite degree over $\mathbb{Q}$. These fields are the core subject matter of algebraic number theory, and are of critical importance in several subsequent articles in this volume, e.g., [Stevenhagen 2008a; 2008b]. Quadratic fields $\mathbb{Q}(\sqrt{d})$ already illustrate many of the ideas that come up in studying number fields in general, and are a vast and fertile area of study.

Let $F = \mathbb{Q}(\sqrt{d})$. The *ring of integers* $\mathcal{O}_F$ in F is the set of $v \in F$ that are roots of a monic polynomial with integer coefficients; this ring plays the role, in F, that the integers do in $\mathbb{Q}$.

PROPOSITION 26. With the above notation, $\mathcal{O}_F = \mathbb{Z}[\omega]$ where

$$\omega = \begin{cases} \sqrt{d} & \text{if } d \equiv 2 \text{ or } 3 \mod 4, \\ (1+\sqrt{d})/2 & \text{if } d \equiv 1 \bmod 4. \end{cases}$$

Thus $\mathcal{O}_F = \mathbb{Z}[\sqrt{d}]$ if $d \equiv 2$ or $3 \mod 4$, and

$$\mathcal{O}_F = \mathbb{Z}[\omega] = \left\{ \frac{x+y\sqrt{d}}{2} : x, y \in \mathbb{Z}, x \equiv y \bmod 2 \right\}$$

if $d \equiv 1 \mod 4$.

The key idea of the proof is that if $v = x + y\sqrt{d} \in F$ is an algebraic integer then it satisfies the equation $X^2 - 2xX + (x^2 - dy^2) = 0$, which implies that x and therefore y are either integers or half-integers; we leave the details to the reader.

The algorithmically inclined reader will observe that the ring of integers is an unfriendly object. For instance, we have been assuming in this discussion that the integer d is squarefree. In fact, it is difficult to detect or verify whether a large integer is squarefree, and therefore impossible to find the ring of integers in $F = \mathbb{Q}(\sqrt{d})$ for general d. See [Stevenhagen 2008a] for a discussion of how algorithmic number theorists deal with this issue for arbitrary number fields.

We now turn our attention to finding units in rings of integers of quadratic fields.

The only units (elements with multiplicative inverses) in $\mathbb{Z}$ are ± 1. In order to find units in quadratic fields we introduce some standard terminology.

- If $v = x + y\sqrt{d} \in F$ then $v' = x - y\sqrt{d}$ is the *conjugate* of v.
- The *norm* of $v = x + y\sqrt{d} \in F$ is

$$N(v) = vv' = (x + y\sqrt{d})(x - y\sqrt{d}) = x^2 - dy^2 \in \mathbb{Q}.$$

The map $u \mapsto u'$ is easily checked to be an automorphism, and $(uv)' = u'v'$ implies that the norm map is multiplicative: $N(uv) = N(u)N(v)$.

Note that if $d \equiv 2$ or $3 \mod 4$ then $N(x + y\omega) = (x + y\omega)(x - y\omega) = x^2 - dy^2$, while if $d \equiv 1 \mod 4$ and $\omega = (1 + \sqrt{d})/2$ then

$$N(x + y\omega) = (x + y\omega)(x + y\omega') = x^2 - xy - \frac{d-1}{4}y^2.$$

LEMMA 27. An element u in $\mathcal{O}_F$ is a unit if and only if $N(u) = \pm 1$.

PROOF. If $N(u) = \pm 1$ then $u(\pm u') = 1$ and u is a unit. If v is a unit then there is a $u \in \mathcal{O}_F$ such that $uv = 1$. Taking the norm and using the multiplicativity of the norm gives $N(u)N(v) = 1$, so that $N(u)$ is a unit in $\mathbb{Z}$, i.e., $N(u) = \pm 1$. □

In imaginary quadratic fields there are only finitely many units, and they are easy to determine by considering the explicit form of the norm mapping given above; the details of are again left to the reader, and the result is:

THEOREM 28. Units in an imaginary quadratic field $F = \mathbb{Q}(\sqrt{d})$, $d < 0$, are:

$$\mathcal{O}_F^* = \begin{cases} \{\pm 1, \pm i\} & \text{if } d = -1, \\ \{\pm 1, \pm\omega, \pm\omega^2\} & \text{if } d = -3, \\ \{\pm 1\} & \text{if } d < -3. \end{cases}$$

The units of the ring $\mathbb{Z}[\sqrt{d}]$ are $\{\pm 1\}$, unless $d = -1$.

Finding units in (the ring of integers of) real quadratic fields is considerably more interesting. If $d \equiv 2$ or $3 \mod 4$ then $\mathcal{O}_F = \mathbb{Z}[\sqrt{d}]$ and finding units is equivalent to solving Pell's equation [Lenstra 2008]

$$x^2 - dy^2 = \pm 1.$$

If $d \equiv 1 \mod 4$ then units $u = x + y\omega$ correspond to solutions to $x^2 - xy - y^2(d-1)/4 = \pm 1$. After multiplying by 4 and completing the square this is equivalent to integer solutions to $X^2 - dY^2 = \pm 4$.

In either case, the problem of finding units reduces to solving Pell's equation or a variant. This can be done by finding the continued fraction of ω.

THEOREM 29. If $F = \mathbb{Q}(\sqrt{d})$, $d > 0$, and $u = x - y\omega$ is a unit in $\mathcal{O}_F^*$ with x and y positive, then x/y is a convergent in the continued fraction expansion of ω.

If $u = x + y\omega$ is a unit then $-u$, u', and $-u'$ are all units. If u is a unit, then one of $\pm u, \pm u'$ has the form $x - y\omega$ with x, y positive, so the restriction to positive x, y is unimportant: any unit can be obtained from a convergent by changing the sign and/or taking a conjugate.

PROOF. Assume that $u = x - y\omega$ is a unit with x and y positive. If $d \equiv 2$ or $3 \mod 4$, then $x^2 - dy^2 = \pm 1$. Moreover $d \geq 2$ and

$$\frac{x}{y} + \sqrt{d} = \sqrt{d \pm \frac{1}{y^2}} + \sqrt{d} \geq 1 + \sqrt{2} > 2. \tag{5-1}$$

From this equation and $(x/y - \sqrt{d})(x/y + \sqrt{d}) = \pm 1/y^2$ we get

$$\left| \frac{x}{y} - \sqrt{d} \right| < \frac{1}{2y^2},$$

so that, by Corollary 18, x/y is a convergent in the continued fraction of ω.

The case $d \equiv 1 \mod 4$ is similar, but slightly more involved. If $u = x - y\omega$ is a unit then $N(u) = x^2 - xy - (d-1)y^2/4 = \pm 1$. Solving the quadratic equation $(x/y)^2 - (x/y) - (d-1)/4 \pm 1/y^2 = 0$ for x/y and doing some algebra shows that

$$\frac{x}{y} - \omega' \geq 2, \tag{5-2}$$

except possibly in the case $d = 5$, $y = 1$. Indeed, this inequality is equivalent to

$$\sqrt{d \pm 4/y^2} + \sqrt{d} \geq 4,$$

which is easy to verify if $d > 5$ (so that $d \geq 13$), or if $y \geq 2$. If either of these hold, proceed as above: $(x/y - \omega)(x/y - \omega') = \pm 1/y^2$ and (5-2) imply that $|x/y - \omega| < 1/2y^2$ and the theorem follows as before. If $d = 5$ and $y = 1$ then the theorem can be checked directly: both $2/1$ and $1/1$ are convergents. □

There is much more to say about continued fractions of arbitrary quadratic irrationals. We enumerate some of the most important facts.

(i) The continued fraction of a real number α is periodic if and only if α is a quadratic irrational.

(ii) The continued fraction of a quadratic irrational α is purely periodic if and only if $\alpha > 1$ and $-1 < \alpha' < 0$.

(iii) The continued fraction of a quadratic irrational α can be computed entirely with integer arithmetic; α_k has the form $(P_k + \sqrt{d})/Q_k$ for integers P_k, Q_k, where Q_k divides $d - P_k^2$, and these integers are determined by the recursions

$$a_k = \lfloor \alpha_k \rfloor, \quad P_{k+1} = a_k Q_k - P_k, \quad Q_{k+1} = \frac{d - P_{k+1}^2}{Q_k}.$$

(iv) The period of the continued fraction of ω is $O(\sqrt{d} \log d)$. The continued fractions of $\sqrt{d}$ and $(1 + \sqrt{d})/2$ have the respective shapes

$$[a; \overline{P, 2a}], \qquad [a, \overline{P, 2a - 1}],$$

where a is a positive integer, the sequence $P = a_1, a_2, \ldots, a_2, a_1$ is a palindrome, and the a_i are less than a.

(v) If the period of the continued fraction is r, then the $(r-1)$st convergent corresponds to a so-called fundamental unit u, and any other unit of the form $\pm u^n$ for integers n; i.e.,

$$\mathcal{O}_F^* = \{\pm u^n : n \in \mathbb{Z}\} \simeq \mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}.$$

(vi) The index of $\mathbb{Z}[\sqrt{d}]^*$ in $\mathcal{O}_F^*$ is 1 or 3.

(vii) The fundamental generating unit of $\mathbb{Q}(\sqrt{d})$ has norm -1 if and only if P has even length. If this is the case, the units of norm one are of the form $\pm u^{2k}$.

5.4. The Smith–Cornacchia algorithm. Throughout this section integers d and n satisfy $0 < d < n$, and we consider integer solutions x, y to

$$x^2 + dy^2 = n. \tag{5-3}$$

Note that the problem of solving this equation is equivalent to

$$(x + y\sqrt{-d})(x - y\sqrt{-d}) = n,$$

i.e., finding elements of $\mathbb{Z}[\sqrt{-d}]$ of norm n. Fix d and let $R = \mathbb{Z}[\sqrt{-d}]$.

A coprime solution x, y is said to be *primitive*. Any coprime solution determines a square root $t = x/y \bmod n$ of $-d$ modulo n; indeed,

$$t^2 + d = (x^2 + dy^2)/y^2 \equiv 0 \bmod n;$$

the unique such t with $1 \le t < n$ is sometimes called the *signature* of the solution.

A signature determines a map from R to $\mathbb{Z}/n\mathbb{Z}$ defined by taking $\sqrt{-d}$ to t. The kernel of this ring homomorphism is the principal ideal

$$I_t = (x + y\sqrt{-d})R = nR + (t + \sqrt{-d})R.$$

The Smith–Cornacchia algorithm has two steps: determine all potential signatures t, and use the Euclidean algorithm to determine which of the ideals $I_t := nR + (t + \sqrt{-d})R$ are principal.

Given a primitive solution (x, y) to $x^2 + dy^2 = n$ with signature t define an integer z by

$$z = \frac{ty - x}{n},$$

so that $x = ty - nz$. Dividing the inequality

$$|2xy| \le x^2 + y^2 \le x^2 + dy^2 = n$$

by $2ny^2$ gives

$$\left|\frac{x}{ny}\right| = \left|\frac{t}{n} - \frac{z}{y}\right| \le \frac{1}{2y^2}.$$

Corollary 18 implies that z/y is a convergent of the (finite) continued fraction of t/n.

Thus to solve (5-3) it suffices to compute the convergents in the continued fraction of t/n and see whether any of the denominators give valid solutions y in the equation $x^2 + dy^2 = n$.

In fact, it is slightly simpler to merely keep track of the remainders in the Euclidean algorithm applied to t and n; indeed $|x|$ is a remainder, and the equation $x^2 + dy^2 = n$ can be solved for y. Since $x \le \sqrt{n}$, the remainders have to be calculated at least to the point that they drop below $\sqrt{n}$.

EXAMPLE 3. If $d = 5$ and $n = 12829$ then we first have to find a square root t of -5 modulo 12829. Since n is a prime, the square root is unique up to sign; in fact, Cipolla's algorithm gives $t = \pm 3705$ without difficulty. The sequence of remainders in the Euclidean algorithm is $12829, 3705, 1714, 277, 52, 17, 1$. The first x below $\sqrt{12829}$ works:

$$x = 52, \qquad y = \sqrt{(n - x^2)/5} = 45, \qquad 52^2 + 5 \cdot 45^2 = 12829.$$

THEOREM 30. If a primitive solution to $x^2 + dy^2 = n$ has signature t then $|x|$ is the largest (i.e., first encountered) remainder that is less than $\sqrt{n}$ in the Euclidean algorithm applied to n and t. Moreover, (x, y) is (essentially) the unique solution with signature t: if (u, v) is another such solution then $(x, y) = \pm(u, v)$ if $d > 1$, and if $d = 1$ reversal might occur, i.e., $(x, y) = \pm(u, v)$ or $(x, y) = \pm(v, u)$.

PROOF. (See also [Nitaj 1995; Schoof 1995].) Let x be the first remainder less than $\sqrt{n}$. Assume that the equation has some solution (u, v) with signature t: $u^2 + dv^2 = n$, $u \equiv ty \bmod n$. Assume that $u \neq \pm x$. Then $|u|$ is a subsequent term in the remainder sequence for t and n. By the Euclidean algorithm, there are z, z' such that $x = yt - zn$ and $u = vt - z'n$. The remainders decrease in absolute value, so $x > u$, and the coefficients of t increase in absolute value, so $|y| < |v|$. We know that $x^2 + dy^2 \equiv (t^2 + d)y^2 \equiv 0 \bmod n$. Moreover,

$$x^2 + dy^2 < n + dv^2 = n + d \cdot \frac{n - dv^2}{d} < 2n.$$

Since we also know that $x^2 + dy^2$ is divisible by n it follows that $x^2 + dy^2 = n$.

Two different solutions x, y and u, v determine generators $x + y\sqrt{-d}$ and $u + v\sqrt{-d}$ of I_t and hence there is a unit a in R such that $x + y\sqrt{-d} = a(u + v\sqrt{-d})$. If $d > 1$ the only units are ± 1. If $d = 1$ then $a = \pm i$ is possible, and these account for all cases in the theorem. □

The algorithm that implements this is obvious.

SMITH–CORNACCHIA ALGORITHM

Input: Relatively prime positive integers d and n

Output: All primitive solutions to $x^2 + dy^2 = n$ (possibly none)

1. Find all positive solutions (less than n) to $t^2 + d \equiv 0 \bmod n$
2. For each solution t, find the first remainder x less than $\sqrt{n}$ in the Euclidean algorithm applied to n and t; if $y := \sqrt{(n - x^2)/d}$ is an integer, output (x, y)

The second step of the algorithm is efficient. Unfortunately, the first step isn't in general, since it requires a modular square root. However, in the special case that n is a prime, this step can be done efficiently by a probabilistic algorithm.

EXAMPLE 4. The special case $d = 1$ is interesting as it is the classic problem of writing an integer as the sum of two squares. By Euler's criterion, -1 is a quadratic residue modulo an odd prime p if and only if p is congruent to 1 modulo 4, and this algorithm gives a constructive proof of Euler's result that these are exactly the primes that are the sum of two squares. Moreover, it shows that if z is a Gaussian integer then the factorization of z in $\mathbb{Z}[i]$ reduces to the problem of factoring $N(z)$ in the integers. See [Bressoud and Wagon 2000] for further details.

References

[Agrawal et al. 2004] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P", *Ann. of Math.* (2) **160**:2 (2004), 781–793.

[Bach 1990] E. Bach, "Explicit bounds for primality testing and related problems", *Math. Comp.* **55**:191 (1990), 355–380.

[Bach and Shallit 1996] E. Bach and J. Shallit, *Algorithmic number theory, I: Efficient algorithms*, MIT Press, Cambridge, MA, 1996.

[Bernstein 2008] D. Bernstein, "Fast multiplication and its applications", pp. 325–384 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Bressoud and Wagon 2000] D. Bressoud and S. Wagon, *A course in computational number theory*, Key College, Emeryville, CA, 2000.

[Cassels 1986] J. W. S. Cassels, *Local fields*, London Mathematical Society Student Texts **3**, Cambridge University Press, Cambridge, 1986.

[Cipolla 1902] M. Cipolla, "La determinazione assintotica dell'n^{imo} numero primo", *Rend. Accad. Sci. Fis. Mat. Napoli* **8** (1902), 132–166.

[Cohen 1993] H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics **138**, Springer, Berlin, 1993.

[Cox 1989] D. A. Cox, *Primes of the form $x^2 + ny^2$: Fermat, class field theory and complex multiplication*, Wiley, New York, 1989.

[Crandall and Pomerance 2005] R. Crandall and C. Pomerance, *Prime numbers: A computational perspective*, 2nd ed., Springer, New York, 2005.

[Damgård et al. 1993] I. Damgård, P. Landrock, and C. Pomerance, "Average case error estimates for the strong probable prime test", *Math. Comp.* **61**:203 (1993), 177–194.

[Davis 1973] M. Davis, "Hilbert's tenth problem is unsolvable", *Amer. Math. Monthly* **80** (1973), 233–269.

[Flajolet and Vallée 1998] P. Flajolet and B. Vallée, "Continued fraction algorithms, functional operators, and structure constants", *Theoret. Comput. Sci.* **194**:1-2 (1998), 1–34.

[Garey and Johnson 1979] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco, 1979.

[von zur Gathen and Gerhard 2003] J. von zur Gathen and J. Gerhard, *Modern computer algebra*, 2nd ed., Cambridge University Press, Cambridge, 2003.

[Granville 2008] A. Granville, "Smooth numbers: computational theory and beyond", pp. 267–323 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Hopcroft and Ullman 1979] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, MA, 1979.

[Knuth 1981] D. E. Knuth, *The art of computer programming, II: Seminumerical algorithms*, 2nd ed., Addison-Wesley, Reading, MA, 1981.

[Koblitz 1984] N. Koblitz, *p-adic numbers, p-adic analysis, and zeta-functions*, 2nd ed., Graduate Texts in Mathematics **58**, Springer, New York, 1984.

[Kozen 2006] D. C. Kozen, *Theory of computation*, Springer, London, 2006.

[Lenstra 1987] H. W. Lenstra, Jr., "Factoring integers with elliptic curves", *Ann. of Math.* (2) **126**:3 (1987), 649–673.

[Lenstra 2008] H. W. Lenstra, Jr., "Solving the Pell equation", pp. 1–23 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Lenstra and Lenstra 1993] A. K. Lenstra and H. W. Lenstra, Jr. (editors), *The development of the number field sieve*, Lecture Notes in Mathematics **1554**, Springer, Berlin, 1993.

[Lenstra and Pomerance 1992] H. W. Lenstra, Jr. and C. Pomerance, "A rigorous time bound for factoring integers", *J. Amer. Math. Soc.* **5**:3 (1992), 483–516.

[Matiyasevich 1993] Y. V. Matiyasevich, *Hilbert's tenth problem*, MIT Press, Cambridge, MA, 1993.

[Nitaj 1995] A. Nitaj, "L'algorithme de Cornacchia", *Exposition. Math.* **13**:4 (1995), 358–365.

[Pollard 1978] J. M. Pollard, "Monte Carlo methods for index computation (mod p)", *Math. Comp.* **32**:143 (1978), 918–924.

[Pomerance 2008] C. Pomerance, "Elementary thoughts on discrete logarithms", pp. 385–396 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Poonen 2008] B. Poonen, "Elliptic curves", pp. 183–207 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[van der Poorten 1986] A. J. van der Poorten, "An introduction to continued fractions", pp. 99–138 in *Diophantine analysis* (Kensington, 1985), edited by J. H. Loxton and A. J. van der Poorten, London Math. Soc. Lecture Note Ser. **109**, Cambridge Univ. Press, Cambridge, 1986.

[Schirokauer 2008] O. Schirokauer, "The impact of the number field sieve on the discrete logarithm problem in finite fields", pp. 397–420 in *Surveys in algorithmic*

number theory, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Schönhage 1971] A. Schönhage, "Schnelle Berechnung von Kettenbruchentwicklungen", *Acta Informatica* **1** (1971), 139–144.

[Schoof 1985] R. Schoof, "Elliptic curves over finite fields and the computation of square roots mod p", *Math. Comp.* **44**:170 (1985), 483–494.

[Schoof 1995] R. Schoof, "Counting points on elliptic curves over finite fields", *J. Théor. Nombres Bordeaux* **7**:1 (1995), 219–254.

[Schoof 2008a] R. Schoof, "Computing Arakelov class groups", pp. 447–495 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Schoof 2008b] R. Schoof, "Four primality testing algorithms", pp. 101–125 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Serre 1973] J.-P. Serre, *A course in arithmetic*, Graduate Texts in Mathematics **7**, Springer, New York, 1973.

[Shor 1997] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Comput.* **26**:5 (1997), 1484–1509.

[Shoup 2005] V. Shoup, *A computational introduction to number theory and algebra*, Cambridge University Press, Cambridge, 2005.

[Stevenhagen 2008a] P. Stevenhagen, "The arithmetic of number rings", pp. 209–266 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Stevenhagen 2008b] P. Stevenhagen, "The number field sieve", pp. 83–100 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

[Teske 2001] E. Teske, "Square-root algorithms for the discrete logarithm problem", pp. 283–301 in *Public-key cryptography and computational number theory* (Warsaw, 2000), edited by K. Alster et al., de Gruyter, Berlin, 2001.

[Yui and Top 2008] N. Yui and J. Top, "Congruent number problems in dimension one and two", pp. 613–639 in *Surveys in algorithmic number theory*, edited by J. P. Buhler and P. Stevenhagen, Math. Sci. Res. Inst. Publ. **44**, Cambridge University Press, New York, 2008.

JOE BUHLER
CENTER FOR COMMUNICATIONS RESEARCH
SAN DIEGO, CA, 92121
jpb@reed.edu

STAN WAGON
MACALESTER COLLEGE
ST. PAUL, MN 55105
wagon@macalester.edu