Computational Number Theory

April 4, 2017

Acknowledgements

These lecture notes are based on those from an Oxford course given by James McKee, and are reproduced here with his kind permission. They contain examples due to Richard Pinch. Parts of the notes are based on material from the book by Pohst and Zassenhaus. While I take full responsibility for the current contents of these notes, considerable thanks are due to Richard and, in particular, to James.

I am anxious to be informed of any misprints or other errors that are spotted. Please email them to me at

`rhb maths.ox.ac.uk`

Particular thanks are due to Eoin Byrne, who typeset a large part of these notes, and to Michelle Kovesi, whose meticulous reading of the notes has eliminated numerous misprints.

Roger Heath-Brown
April, 2013

Books

Much of the material is covered in Cohen's *A course in computational algebraic number theory* (Springer, GTM 138). However although the material is in many cases the same, the treatment in these notes is often somewhat different.

Good general books include Knuth's *The art of computer programming*, especially volume 2, and *The design and analysis of computer algorithms*, by Aho, Hopcroft and Ullmann.

More specific sources include:-

Continued fractions

Many books on elementary number theory, such as Davenport (*The higher arithmetic*), Hardy & Wright, Baker, Rose, ...

Elliptic curves

Koblitz, *A course in number theory and cryptography* (Springer, GTM 114)

Lattices

Pohst and Zassenhaus, *Algorithmic algebraic number theory* (CUP)

Smart, *The algorithmic resolution of Diophantine equations* (CUP, LMSST 41)

Factorisation

Riesel, *Prime numbers and computer methods for factorization* (Birkhäuser)

1 Background material

1.1 Polynomial time

Let f, g be two functions from $\mathbb{N}^r$ to $\mathbb{R}_{\geq 0}$. We write

$$f = O(g)$$

if there exist constants A, B such that

$$f(n_1, \ldots, n_r) \leq Ag(n_1, \ldots, n_r)$$

whenever $n_1, \ldots, n_r \geq B$.

For example, the number of digits in the base b representation of a positive integer n is

$$O(\max(1, \tfrac{\log n}{\log b})).$$

We shall often be interested in estimating running times for algorithms. We may have input parameters $n_1, \ldots, n_r$, and then the running time (i,e., the number of elementary operations—what operations we consider to be elementary may depend on context) will be some function of $n_1, \ldots, n_r$, say $f(n_1, \ldots, n_r)$. A precise expression for $f(n_1, \ldots, n_r)$ may be hard to determine, and may be too complicated to provide any illumination. Often one is satisfied with finding a 'nice' function $g(n_1, \ldots, n_r)$ such that $f = O(g)$.

A frequently-met special case is where the input to an algorithm consists of a single positive integer n. Let $f(n)$ be the running time of the algorithm. If

$$f(n) = O((\log n)^d)$$

for some constant d, then we say that the algorithm runs in **polynomial time**. More generally, if b is the number of bits in the input, and there is a constant d such that an algorithm runs in time $O(b^d)$, then we say that the algorithm runs in polynomial time. If $d = 0, 1, 2, 3, \ldots$, then we say that the algorithm runs in constant time, linear time, quadratic time, cubic time,....

In practice, the implied constants A, B in the O-notation are important. A quadratic time algorithm will run more quickly than a (genuinely) cubic time algorithm for large enough values of the input, but for specific values of the input, the asymptotic behaviour is irrelevant.

Basic arithmetic operations on the integers, $+, -, \times, \div$ (finding quotient and remainder), are all polynomial-time, even when using naive methods. Hence

arithmetic in $\mathbb{Z}/n\mathbb{Z}$ can be done in polynomial time (for division, where possible, one computes a multiplicative inverse using Euclid's algorithm: see immediately below).

1.2 Euclid's Algorithm

Any positive integer n can be written uniquely (up to reordering the factors) as a product of primes. If

$$a = \prod_{i=1}^{r} p_i^{\alpha_i},$$

$$b = \prod_{i=1}^{r} p_i^{\beta_i},$$

where $p_1, \ldots, p_r$ are distinct primes, then the greatest common divisor of a and b, $\gcd(a, b)$ is given by

$$\gcd(a, b) = \prod_{i=1}^{r} p_i^{\min(\alpha_i, \beta_i)}.$$

It is the greatest integer dividing both a and b.

Computing $\gcd(a, b)$ by finding the prime factorisations of a and b can be slow (we shall be looking at this problem in some detail later). Can we do better?

If $|a| \geq |b| > 0$, then we can write

$$a = qb + r$$

with

$$|r| \leq \frac{|b|}{2}$$

(q is the nearest integer to $\frac{a}{b}$). Any common divisor of a and b divides r (and b). Any common divisor of b and r divides a (and b). Therefore $\gcd(a, b) = \gcd(b, r)$. This gives a recursive algorithm for computing $\gcd(a, b)$.

Euclid's algorithm
Input: integers a and b.
Output: $\gcd(a, b)$.

Step 1 If $|a| < |b|$, then swap a and b.

Step 2 If $b = 0$, then STOP with output $|a|$.

Step 3 Compute q, r such that $a = qb + r$, with $|r| \leq \frac{1}{2}|b|$. Replace a by b, b by r, and go to Step 2.

Each application of Step 3 reduces $|b|$ by a factor of at least 2. Hence we loop $O(\log(\min|a|, |b|))$ times, and we see that Euclid's algorithm runs in polynomial time.

There are many variants, some particularly convenient for use with binary computers. See Cohen §1.3, especially Algorithm 1.3.5.

If $d = \gcd(a, b)$, then there exist integers x, y such that $d = xa + yb$. Euclid's algorithm can be extended to find x, y. In the following, all triples (x, y, z) satisfy $xa + yb = z$.

Extended Euclid
Input: integers a and b.
Output: integers x and y such that $xa + yb = \gcd(a, b)$.

Step 1 If $|a| < |b|$, then swap a and b. Let $A = (1, 0, a), B = (0, 1, b)$.

Step 2 If $b = 0$, then STOP with output $A = (x, y, \gcd(a, b))$.

Step 3 Compute q, r such that $a = qb + r$, with $|r| \leq \frac{1}{2}|b|$. Copy A into A_{old}. Replace A by B, replace B by $A_{old} - qB$, and go to Step 2.

This algorithm can be used to compute inverses in $\mathbb{Z}/n\mathbb{Z}$ in polynomial time. Given $a \in \mathbb{Z}$, we compute x, y such that $xa+yn = \gcd(a, n)$. If $\gcd(a, n) > 1$, then a is not invertible mod n. If $\gcd(a, n) = 1$, then x is the desired inverse of a mod n.

1.3 Modular exponentiation

Computing $a^m (\text{mod } n)$ can be done by performing $m - 1$ multiplications $(\text{mod } n)$. We can do much better as follows.

Write

$$m = 2^s + 2^t + \cdots$$

where $s > t > \ldots$ (i.e., compute the binary representation of m). Then compute the list

$$a, a^2, a^4, a^8, \ldots, a^{2^s}$$

(all mod n) where each term is the square of the previous one. Multiply together all those a^{2^i} for which 2^i appears in the binary representation of m, to get

$$a^{2^s} a^{2^t} \cdots = a^{2^s + 2^t + \cdots} = a^m.$$

We have thus computed $a^m \pmod n$ in $O(\log m)$ multiplications.

The above idea is implemented more efficiently (without having to store either the binary expansion of m or the relevant a^{2^i}) by the following algorithm.

Modular exponentiation: $a^m \pmod n$

Step 1 Let $x = m, y = a, z = 1$.

Step 2 If x is odd, then multiply z by $y \pmod n$, and then subtract 1 from x.

Step 3 If $x = 0$, then STOP with output z.

Step 4 Square $y \pmod n$, divide x by 2, and go to Step 2.

See Cohen §1.2 for variants.

1.4 Continued fractions: basic definitions

Let θ be any real number. Put $a_0 = \lfloor \theta \rfloor$ (the largest integer not greater than θ). If $a_0 \neq \theta$, then we can write $\theta = a_0 + \frac{1}{\theta_1}$, where $\theta_1 > 1$, and we put $a_1 = \lfloor \theta_1 \rfloor$. If $a_1 \neq \theta_1$, then we can write $\theta_1 = a_1 + \frac{1}{\theta_2}$, where $\theta_2 > 1$, and we put $a_2 = \lfloor \theta_2 \rfloor$. This process can be continued indefinitely, unless $a_n = \theta_n$ for some n. Note that $a_1, a_2, \ldots$ are all positive integers, although a_0 might be negative or zero. This process is the **continued fraction process**, and the a_i are known as the **partial quotients** of θ.

If the process terminates, then we have

$$\begin{aligned}\theta &= a_0 + \frac{1}{\theta_1} \\ &= a_0 + \frac{1}{a_1 + \frac{1}{\theta_2}} \\ &\vdots \\ &= a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \frac{1}{a_n}}}}}\end{aligned}$$

We then write

$$\theta = [a_0, a_1, \ldots, a_n].$$

We also use this notation when the a_i are not necessarily integers.

If the continued fraction process does not terminate, then we write

$$\theta = [a_0, a_1, a_2, \ldots],$$

and for any n we then have

$$\theta = [a_0, a_1, a_2, \ldots, a_n, \theta_{n+1}],$$

where $a_0, \ldots, a_n$ are integers, but θ_{n+1} is not.

If we set

$$\frac{p_n}{q_n} = [a_0, \ldots, a_n],$$

where $\gcd(p_n, q_n) = 1$, then we call $\frac{p_n}{q_n}$ the nth **convergent** to θ. We shall see that

$$\frac{p_n}{q_n} \to \theta \text{ as } n \to \infty.$$

1.5 Continued fractions: a recurrence relation for the convergents

Let $a_0, a_1, a_2, \ldots$ be a sequence of integers, with $a_i > 0$ when $i > 0$. Define p_n, q_n by

$$p_0 = a_0,\ q_0 = 1,\ p_1 = a_0 a_1 + 1,\ q_1 = a_1,$$

$$p_n = a_n p_{n-1} + p_{n-2},\ q_n = a_n q_{n-1} + q_{n-2},\ \text{for } n \geq 2.$$

Then:

(a) $p_n q_{n+1} - p_{n+1} q_n = (-1)^{n+1}$;

(b) $\gcd(p_n, q_n) = 1$;

(c) $\frac{p_n}{q_n} = [a_0, \ldots, a_n]$;

(d) If the a_i are produced by applying the continued fraction process to θ, then $\frac{p_n}{q_n}$ is the nth convergent to θ, and

$$\theta = \frac{p_n \theta_{n+1} + p_{n-1}}{q_n \theta_{n+1} + q_{n-1}}.$$

Proof
(a) We use induction on n. We have $p_0 q_1 - p_1 q_0 = a_0 a_1 - a_0 a_1 - 1 = -1$, so the result holds for $n = 0$. Suppose that the result holds for $n = m - 1$, and consider the case $n = m$. We have, using the recurrence relation,

$$\begin{aligned} p_m q_{m+1} - p_{m+1} q_m &= p_m(a_{m+1} q_m + q_{m-1}) - (a_{m+1} p_m + p_{m-1}) q_m \\ &= p_m q_{m-1} - p_{m-1} q_m \\ &= -(-1)^m \\ &= (-1)^{m+1}, \end{aligned}$$

so the result holds for $n = m$.
(b) This is immediate from (a).
The remark in (d) that p_n/q_n is the nth convergent to θ follows immediately from (c). We use induction on n to prove the rest of (d) along with (c), remembering that (c) does not require *a priori* that the a_i are produced by the continued fraction process. First note that

$$\frac{p_1}{q_1} = a_0 + \frac{1}{a_1} = [a_0, a_1].$$

Also

$$\begin{aligned} \frac{p_1 \theta_2 + p_0}{q_1 \theta_2 + q_0} &= \frac{(a_0 a_1 + 1)\theta_2 + a_0}{a_1 \theta_2 + 1} \\ &= a_0 + \frac{\theta_2}{a_1 \theta_2 + 1} \\ &= a_0 + \frac{1}{a_1 + \frac{1}{\theta_2}} \\ &= \theta, \end{aligned}$$

so the result holds for $n = 1$. Suppose that the result holds for $n = m - 1$, and consider the case $n = m$. We have

$$\begin{aligned}
[a_0, \ldots, a_m] &= \frac{p_{m-1}a_m + p_{m-2}}{q_{m-1}a_m + q_{m-2}}, \text{ using (d), with } n = m-1 \\
&\qquad\qquad \text{and } \theta = [a_0, \ldots, a_m] \\
&= \frac{p_m}{q_m}, \text{ which is (c), with } n = m.
\end{aligned}$$

To establish (d) with $n = m$, note that

$$\begin{aligned}
\theta &= [a_0, \ldots, a_m, \theta_{m+1}] \\
&= [a_0, \ldots, a_{m-1}, a_m + \frac{1}{\theta_{m+1}}] \\
&= \frac{p_{m-1}(a_m + \frac{1}{\theta_{m+1}}) + p_{m-2}}{q_{m-1}(a_m + \frac{1}{\theta_{m+1}}) + q_{m-2}}, \text{ using (d), with } n = m-1 \\
&= \frac{p_m + \frac{p_{m-1}}{\theta_{m+1}}}{q_m + \frac{q_{m-1}}{\theta_{m+1}}}, \text{ using the recurrence relations} \\
&= \frac{p_m\theta_{m+1} + p_{m-1}}{q_m\theta_{m+1} + q_{m-1}}, \text{ which is (d), with } n = m.
\end{aligned}$$

1.6 Continued fractions: some properties of the convergents

For parts (a) to (d), we suppose that the continued fraction process does not terminate.

(a) θ lies between $\frac{p_n}{q_n}$ and $\frac{p_{n+1}}{q_{n+1}}$.
Proof $\theta = [a_0, \ldots, a_n, \theta_{n+1}] = [a_0, \ldots, a_n + \frac{1}{\theta_{n+1}}]$, where $0 < \frac{1}{\theta_{n+1}} \leq \frac{1}{a_{n+1}}$, so θ lies between $[a_0, \ldots, a_n]$ and $[a_0, \ldots, a_n + \frac{1}{a_{n+1}}] = [a_0, \ldots, a_n, a_{n+1}]$.

(b) $|\theta - \frac{p_n}{q_n}| \leq \frac{1}{q_n q_{n+1}}$.
Proof From (a), $|\theta - \frac{p_n}{q_n}| \leq |\frac{p_n}{q_n} - \frac{p_{n+1}}{q_{n+1}}| = \frac{1}{q_n q_{n+1}}$, using 1.5(a).

(c) $q_{n+2} \geq 2q_n, p_{n+2} \geq 2p_n \quad (n \geq 1)$
Proof Immediate from the recurrence relations.

(d) $\frac{p_n}{q_n} \to \theta$ as $n \to \infty$.
Proof Immediate from (b) and (c).

(e) The continued fraction process terminates if and only if θ is rational.
Proof The 'only if' part is clear. Conversely, suppose that $\theta = \frac{a}{b}$ is rational, and that the process does not terminate. Then taking n such that $q_{n+1} > b$ gives $|\theta - \frac{p_n}{q_n}| \geq \frac{1}{bq_n} > \frac{1}{q_n q_{n+1}}$, contradicting (b).

Note that 1.5(a) could be used to compute inverses mod n. To compute the inverse of a mod n, we compute convergents to $\frac{a}{n}$. By (e), we eventually reach $p_r = a, q_r = n$, provided that $\gcd(a, n) = 1$. By 1.5(a), we then have $p_{r-1}n - aq_{r-1} = (-1)^{r+1}$, so that q_{r-1} is, up to choice of sign, the desired inverse. This method is equivalent to (a variant of) Euclid's algorithm. From (c), we have $r = O(\log n)$.

The continued fraction process gives us a sequence of rational approximations to any irrational number θ. These approximations are rather good, indeed they are the 'best possible' in a sense made precise below.

Examples

(a)

$$\theta = \tfrac{16}{9}, \quad a_0 = 1, \quad \theta = 1 + \tfrac{7}{9}.$$

$$\theta_1 = \tfrac{9}{7}, \quad a_1 = 1, \quad \theta_1 = 1 + \tfrac{2}{7}.$$

$$\theta_2 = \tfrac{7}{2}, \quad a_2 = 3, \quad \theta_2 = 3 + \tfrac{1}{2}.$$

$$\theta_3 = 2 \quad a_3 = 2 \qquad \theta_3 = 2$$

$\frac{16}{9} = [1, 1, 3, 2]$

(Compare with Euclid's algorithm.)

$\frac{p_0}{q_0} = \frac{1}{1}$

$\frac{p_1}{q_1} = 1 + \frac{1}{1} = \frac{2}{1}$

$\frac{p_2}{q_2} = 1 + \frac{1}{1+\frac{1}{3}} = 1 + \frac{3}{4} = \frac{7}{4}$

$\frac{p_3}{q_3} = 1 + \frac{1}{1+\frac{1}{3+\frac{1}{2}}} = 1 + \frac{1}{1+\frac{2}{7}} = 1 + \frac{7}{9} = \frac{16}{9}$

(Check the properties of the convergents proved above. Remark that we

have computed the inverse of 16 mod 9, and of 9 mod 16.)

(b) $\theta = \sqrt{19} = [4, 2, 1, 3, 1, 2, 8, 2, 1, 3, 1, 2, 8, \ldots]$
(For irrational numbers, the partial quotients are often mysterious. Two exceptions are quadratic irrationals, and certain functions of e.)

(c) $e = [2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, \ldots]$

1.7 Continued fractions: closeness of the approximations

Throughout this section, we suppose that θ is irrational, so that the continued fraction process does not terminate.

(a) $|q_n\theta - p_n|$ decreases as n increases.

Proof $\theta = \frac{p_n\theta_{n+1}+p_{n-1}}{q_n\theta_{n+1}+q_{n-1}}$, hence, using 1.5(a),

$$\begin{aligned}
|q_n\theta - p_n| &= |\frac{q_np_n\theta_{n+1} + q_np_{n-1} - p_nq_n\theta_{n+1} - p_nq_{n-1}}{q_n\theta_{n+1} + q_{n-1}}| \\
&= \frac{1}{q_n\theta_{n+1} + q_{n-1}} \\
&< \frac{1}{q_n + q_{n-1}} \\
&= \frac{1}{(a_n + 1)q_{n-1} + q_{n-2}} \\
&< \frac{1}{q_{n-1}\theta_n + q_{n-2}} \\
&= |q_{n-1}\theta - p_{n-1}|.
\end{aligned}$$

(b) The convergents give successively closer approximations to θ.

Proof This is a weaker statement than (a).

(c)
$$\frac{1}{(a_{n+1} + 2)q_n^2} < |\theta - \frac{p_n}{q_n}| < \frac{1}{a_{n+1}q_n^2} \leq \frac{1}{q_n^2}.$$

Proof $a_{n+1}q_n^2 < q_n^2\theta_{n+1} + q_nq_{n-1} < (a_{n+1}+2)q_n^2$. Now use $|\theta - p_n/q_n| = 1/(q_n^2\theta_{n+1} + q_nq_{n-1})$, as in the proof of (a).

(d) If p and q are integers with $0 < q < q_{n+1}$, then $|q\theta - p| \geq |q_n\theta - p_n|$. (In this sense, continued fraction approximations are "best possible".)

Proof From 1.5(a), we can find integers u, v such that

$$\begin{aligned} p &= up_n + vp_{n+1}, \\ q &= uq_n + vq_{n+1}. \end{aligned}$$

Thus $u \neq 0$ $(0 < q < q_{n+1})$. If $v \neq 0$, then u and v cannot both be negative $(0 < q)$ nor both positive $(q < q_{n+1})$, so they have opposite signs. Since $q_n\theta - p_n$ and $q_{n+1}\theta - p_{n+1}$ also have opposite signs, we have

$$|q\theta - p| = |u(q_n\theta - p_n) + v(q_{n+1}\theta - p_{n+1})| \geq |q_n\theta - p_n|,$$

as required.

(e) If p, q are positive integers with $|\theta - p/q| < 1/(2q^2)$, then p/q is a convergent to θ.

Proof Take n such that $q_n \leq q < q_{n+1}$ (one can clearly do this if θ is irrational; as an exercise, check the rational case). Then

$$\begin{aligned} |p/q - p_n/q_n| &\leq |\theta - p/q| + |\theta - p_n/q_n| \\ &= q^{-1}|q\theta - p| + q_n^{-1}|q_n\theta - p_n| \\ &\leq (1/q + 1/q_n)|q\theta - p| \quad \text{by (d)} \\ &< (2/q_n).(1/2q) \\ &= 1/qq_n. \end{aligned}$$

Hence $|p/q - p_n/q_n| = 0$.

Thus we see that all continued fraction approximations are "good", and that all "sufficiently good" rational approximations arise via the continued fraction process.

1.8 Continued fractions: quadratic irrationals

The partial quotients of θ are ultimately periodic if and only if θ is a quadratic irrational. (For a proof see any of the standard texts.) Moreover, the continued fraction expansion of $\sqrt{n}$ can be computed using only integer arithmetic,

and relatively low precision.

Example
$\lfloor\sqrt{19}\rfloor = 4$

$\sqrt{19} = 4 + (\sqrt{19} - 4)$

$\frac{1}{\sqrt{19}-4} = \frac{\sqrt{19}+4}{19-16} = \frac{\sqrt{19}+4}{3}$

$\lfloor\frac{\sqrt{19}+4}{3}\rfloor = \lfloor\frac{4+4}{3}\rfloor$
(Exercise: $\lfloor\frac{a+b}{c}\rfloor = \lfloor\frac{\lfloor a\rfloor+b}{c}\rfloor$ for any positive integers b, c.)

$\frac{\sqrt{19}+4}{3} = 2 + \frac{\sqrt{19}-2}{3}$

$\frac{3}{\sqrt{19}-2} = \frac{3(\sqrt{19}+2)}{15} = \frac{\sqrt{19}+2}{5}$
(Exercise: the denominator always divides the norm of the numerator, which keeps the numbers small.)

1.9 The Legendre symbol

If n is an integer with $n \geq 2$, then $(\mathbb{Z}/n\mathbb{Z})^*$ denotes the multiplicative group of those residue classes of integers mod n which are relatively prime to n. The order of this group is $\phi(n)$, where ϕ is **Euler's totient function**.

Let p be prime, greater than 2. Then $\phi(p) = p - 1$. Moreover $(\mathbb{Z}/p\mathbb{Z})^*$ is then cyclic. The subset of invertible squares mod p is a subgroup of order $(p-1)/2$. Indeed if g is a generator of $(\mathbb{Z}/p\mathbb{Z})^*$, then the squares are precisely the even powers of g, and the non-squares are the odd powers of g.

If b is an integer, and p is an odd prime, then the **Legendre symbol**, $(\frac{b}{p})$, is defined by

$$\begin{aligned}(\tfrac{b}{p}) &= 0 \text{ if } p \text{ divides } b;\\ (\tfrac{b}{p}) &= 1 \text{ if } b \text{ is a non-zero square mod } p;\\ (\tfrac{b}{p}) &= - \text{ if } b \text{ is a not a square mod } p.\end{aligned}$$

If $(\frac{b}{p}) = 1$, then b is called a **quadratic residue** mod p; if $(\frac{b}{p}) = -1$, then b is called a **quadratic non-residue** mod p.

The Legendre symbol $(\frac{b}{p})$ can be computed in $O(p)$ operations in $(\mathbb{Z}/p\mathbb{Z})^*$, simply by testing whether $a^2 = b \pmod p$ for $a = 1, 2, \ldots, p-1$. We can do much better by using **Euler's criterion**:

$$(\tfrac{b}{p}) = b^{(p-1)/2} (\text{mod } p).$$

This is trivial if p divides b. Otherwise note that $g^r = 1 (\text{mod } p)$ if and only if $p-1$ divides r, and deduce that $b^{(p-1)/2} = 1 (\text{mod } p)$ if and only if b is an even power of g, and so, equivalently, if and only if b is a non-zero square mod p. If b is not a square mod p then $b^{(p-1)/2}$ is a non-trivial square-root of $1 (\text{mod } p)$, and so must be -1. Using the modular exponentiation algorithm of §1.3 we can compute $(\frac{b}{p})$ via Euler's criterion in $O(\log p)$ operations in $(\mathbb{Z}/p\mathbb{Z})^*$.

1.10 The Jacobi symbol

If n is composite, then the structure of $(\mathbb{Z}/n\mathbb{Z})^*$ is more complicated than when n is prime. The group is usually not cyclic. In general, far fewer than half the elements are squares. We shall extend the Legendre symbol $(\frac{b}{n})$ to include odd composite n, when $(\frac{b}{n})$ is called the **Jacobi symbol**. It will still be true that $(\frac{b}{n}) = 1$ for all b which are squares mod n (and relatively prime to n) but $(\frac{b}{n})$ will equal 1 for some non-squares also. Indeed $(\frac{b}{n})$ will equal 1 for exactly half the elements of $(\mathbb{Z}/n\mathbb{Z})^*$.

If $n = \prod_i p_i^{e_i}$, where the p_i are distinct odd primes, we define the Jacobi symbol by

$$(\tfrac{b}{n}) = \prod_i (\tfrac{b}{p_i})^{e_i}$$

(with $(\frac{b}{1}) = 1$), where the expression on the right is a product of Legendre symbols.

The Jacobi symbol satisfies the following properties (where m, n are odd positive integers, and a, b are arbitrary integers).

(a) $(\frac{b}{n}) = (\frac{b \bmod n}{n})$;

(b) $(\frac{b}{mn}) = (\frac{b}{m})(\frac{b}{n})$;

(c) $(\frac{ab}{n}) = (\frac{a}{n})(\frac{b}{n})$;

(d) $(\frac{-1}{n}) = (-1)^{(n-1)/2}$;

(e) $(\frac{2}{n}) = (-1)^{(n^2-1)/8}$;

(f) $(\frac{m}{n}) = (\frac{n}{m})(-1)^{(m-1)(n-1)/4}$.

All of these follow readily from properties of the Legendre symbol. For the Legendre symbol, formula (f) is the celebrated law of **quadratic reciprocity**, first proved by Gauss. We take these properties on trust, and use them to remark that the Jacobi symbol $(\frac{b}{n})$ can be computed in $O(\log n)$ operations on $\mathbb{Z}/n\mathbb{Z}$ without needing to know the prime factorization of n. Indeed by (a) we can suppose that $0 \leq b < n$, and then we can use (c) and (e) to remove any factors of 2 from b, to reduce to the case in which b and n are positive odd integers; then (f) reduces the computation to that of $(\frac{n}{b})$. One may check that after two such reductions the value of "n" is at least halved. Thus after $O(\log n)$ reductions we will be left with either $(\frac{1}{n})$ or $(\frac{0}{n})$, which have values 1 and 0 respectively. The reader will observe that this process is somewhat similar to the "binary gcd algorithm".

1.11 Square-roots modulo a prime

We start with a particularly simple special case. If $p = 3 \pmod 4$ we may write $p = 2s + 1$ with s odd. Then $x = b^{(s+1)/2}$ will be a square root of $b \bmod p$ whenever b is a quadratic residue of p. To see this we note that $b^s = b^{(p-1)/2} = 1 \pmod p$, by Euler's criterion, so that

$$x^2 = b^{s+1} = b \pmod p.$$

In general we have to work a bit harder. Suppose again that we are trying to find a square-root of $b \pmod p$, where b is a quadratic residue of p. Assume we have an integer z which is a quadratic non-residue modulo p. (It is not quite obvious, but in fact the above algorithm for $p = 3 \pmod 4$ corresponds to taking $z = -1$.) Write $p - 1 = 2^r s$, where s is odd, and set

$$y = z^s \pmod p.$$

Then run the following procedure.

Step 1. Initialize by setting $w = b^{(s+1)/2} \pmod p$ and $n = r$.

Step 2. Check whether $w^2 = b \pmod p$, and if so set $x = w$ and stop.

Step 3. Compute $(w^2 b^{-1})^{2^{t-1}}$ modulo p for $t = n-1, n-2, \ldots$ until one reaches a value which is -1 modulo p.

Step 4. Set $w_0 = wy^{2^{r-t-1}} \pmod p$.

Step 5. Replace w by w_0 and n by $n - 1$, and return to Step 2.

Of course the parameter n plays no role in the algorithm — it is just a convenience for the proof. To check that this works we will show that we always have $(w^2b^{-1})^{2^{n-1}} = 1 \pmod{p}$ at Step 2. In particular, if we ever get down to $n = 1$ we will have $w^2 = b \pmod{p}$. We prove this claim by induction. It is certainly true at the outset, when $w = b^{(s+1)/2} \pmod{p}$ and $n = r$, since we then have

$$(w^2b^{-1})^{2^{n-1}} = (b^s)^{2^{r-1}} = b^{(p-1)/2} = 1 \pmod{p},$$

by Euler's criterion. Then, provided that $(w^2b^{-1})^{2^{n-1}} = 1 \pmod{p}$ at Step 2, when we compute the value in Step 3 for $t = n - 1$ we will have $n \geq 2$ and

$$\{(w^2b^{-1})^{2^{t-1}}\}^2 = (w^2b^{-1})^{2^{n-1}} = 1 \pmod{p},$$

so that

$$(w^2b^{-1})^{2^{t-1}} = \pm 1 \pmod{p}.$$

Thus either $t = n - 1$ is satisfactory or $(w^2b^{-1})^{2^{n-2}} = 1 \pmod{p}$. In this latter case we must have $n \geq 3$, and we can repeat the argument. Hence, provided that $(w^2b^{-1})^{2^{n-1}} = 1 \pmod{p}$ at Step 2, there will always be a satisfactory value of t in Step 3. We can never reach the stage at which the decreasing sequence of exponents produces $(w^2b^{-1})^{2^0} = 1 \pmod{p}$, since this is equivalent to the stopping condition $w^2 = b \pmod{p}$ in Step 2.

To complete the proof of the claim it is now enough to check that

$$(w_0^2b^{-1})^{2^{n-2}} = 1 \pmod{p}$$

at Step 4. We will in fact show that

$$(w_0^2b^{-1})^{2^{t-1}} = 1 \pmod{p},$$

which is sufficient, since $t \leq n - 1$. However

$$(w_0^2b^{-1})^{2^{t-1}} = (w^2b^{-1}y^{2^{r-t}})^{2^{t-1}} = (w^2b^{-1})^{2^{t-1}}y^{2^{r-1}} \pmod{p}.$$

We arranged in Step 3 that $(w^2b^{-1})^{2^{t-1}} = -1 \pmod{p}$. Thus it is enough to observe that

$$y^{2^{r-1}} = z^{2^{r-1}s} = z^{(p-1)/2} = -1 \pmod{p}$$

by Euler's criterion, since z was assumed to be a quadratic non-residue.

There remains the problem of finding a quadratic non-residue z of p. This is easy in practice, since one can choose z at random and have a 50% chance that $(\frac{z}{p}) = -1$. Thus the expected number of trials before finding a suitable z is 2. This produces a **probabilistic algorithm**. However to give a **deterministic algorithm**, which is guaranteed to succeed without any random choices, is another matter entirely.

1.12 Elliptic curves over finite fields

The ground field
Let $p > 3$ be a prime. We shall work throughout over the field with p elements, $\mathbb{F}_p$. One can easily generalize things to work over an arbitrary finite field, including characteristics 2 and 3, but prime fields (the integers mod p) are more familiar, and the formulae are simpler when the characteristic is at least 5. Moreover, some results are easier to formulate when the number of elements in the field is a prime, rather than a prime power.

Definition of an elliptic curve
An **elliptic curve** over $\mathbb{F}_p$ is the set of points $(x, y) \in \mathbb{F}_p^2$ satisfying an equation of the form

$$y^2 = x^3 + ax + b$$

(where $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \neq 0$), together with "the point at infinity" denoted by $\mathbf{O}$.

Examples
Take $p = 5$. then

$$y^2 = x^3 + 3x$$

defines an elliptic curve over $\mathbb{F}_5$ (with $a = 3$, $b = 0$ and $4.3^3 + 27.0^2 \neq 0$), having 10 points namely

$$\mathbf{O}, (0,0), (1,2), (1,3), (2,2), (3,1), (3,4), (4,1), (4,4).$$

Similarly

$$y^2 = x^3 + 2x$$

gives an elliptic curve over $\mathbb{F}_5$ with just two points $\mathbf{O}$ and $(0, 0)$.

The number of points
For a given value of x the number of $y \in \mathbb{F}_p$ for which $y^2 = x^3 + ax + b$ could be 0, or 1, or 2 (the second case arising when $x^3 + ax + b = 0$). The number of values of y will be

$$1 + \left(\frac{x^3 + ax + b}{p}\right)$$

in each case, so that, including the point at infinity, the number of points on the elliptic curve will be

$$1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{x^3 + ax + b}{p}\right)\right) = p + 1 + \sum_{x \in \mathbb{F}_p} \left(\frac{x^3 + ax + b}{p}\right) = p + 1 + \varepsilon,$$

say. If the values of the Legendre symbol were $+1$ and -1 at random one would expect ε to show quite a bit of cancellation. In fact there is the following result.

Hasse's Theorem (1933)
For any elliptic curve over $\mathbb{F}_p$ one has $|\varepsilon| \leq 2\sqrt{p}$. Thus if the number of points on the curve (including the point at infinity) is N then

$$|N - (p+1)| \leq 2\sqrt{p}.$$

Examples If $p = 5$, then we have $|\varepsilon| \leq 4$, so that $2 \leq N \leq 10$. In fact all possibilities occur:

ε	N	Equation
-4	2	$y^2 = x^3 + 2x$
-3	3	$y^2 = x^3 + 4x + 2$
-2	4	$y^2 = x^3 + x$
-1	5	$y^2 = x^3 + 3x + 2$
0	6	$y^2 = x^3 + 1$
1	7	$y^2 = x^3 + 2x + 2$
2	8	$y^2 = x^3 + 4x$
3	9	$y^2 = x^3 + x + 1$
4	10	$y^2 = x^3 + 3x$

It is useful to know not only that the number of points is close to $p+1$, but also that there are "many" possibilities for the number of points. It can be shown that every number of points in Hasse's range is possible.

The group law
An abelian group law, written additively, is defined on the set of points of an elliptic curve E by

- $\mathbf{O}$ is the identity;
- $P + Q + R = \mathbf{O}$ if and only if there is a line meeting E at P, Q and R, counted properly. (Any vertical line $x =$constant passes through $\mathbf{O}$, and if $P = Q$ the line will be tangent at P.)

Inverses
We have $-(x, y) = (x, -y)$. For if $P = (x, y)$ is on E then so is $Q = (x, -y)$ and the line through Q and Q is vertical and so passes through $\mathbf{O}$. Hence $P + Q + \mathbf{O} = \mathbf{O}$, and so $Q = -P$. Note that this argument, suitably interpreted, remains correct even when $y = 0$.

P+Q
The line through P and Q (or the tangent at P if $P = Q$) meets E at a third point, R say. then $P + Q + R = \mathbf{O}$, whence $P + Q = -R$. So to add P and Q we take the third point of intersection of the line PQ with E, and change the sign of its y coordinate.

Explicit formulae

(i) $-\mathbf{O} = \mathbf{O}$; $\mathbf{O} + \mathbf{O} = \mathbf{O}$.

(ii) $(-x, y) = (x, -y)$; $(x, y) + (x, -y) = \mathbf{O}$.

(iii) If (x_1, y_1) and (x_2, y_2) are on E and $x_1 \neq x_2$, then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ where

$$x_3 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)^2 - x_1 - x_2$$

and

$$y_3 = -y_1 + \left(\frac{y_1 - y_2}{x_1 - x_2}\right)(x_1 - x_3).$$

(iv) If (x_1, y_1) is on E and $y_1 \neq 0$, then $2(x_1, y_1) = (x_3, y_3)$ where

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$$

and

$$y_3 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3).$$

Is this an abelian group law?
Addition is plainly commutative, since the line through P and Q is the same as the line through Q and P; we have an identity element, and inverses exist. Thus all the axioms for an abelian group are satisfied apart possibly from associativity. This can in principle be checked from the above formulae, separating several special cases, or one can refer to the various textbooks.

An example
Consider $E : y^2 = x^3 + 3x$ over $\mathbb{F}_5$. Suppose we wish to add $P = (0, 0)$ and $Q = (1, 2)$, which both lie on E. The line through P and Q is $y = 2x$. If (u, v) lies both on E and on this line we will have $v^2 = u^3 + 3u$ and $v = 2u$, so that $(2u)^2 = u^3 + 3u$. This cubic has roots $u = 0, 1$ and 3 over $\mathbb{F}_5$, corresponding to $v = 0, 2$ and 1 respectively. Thus the line meets E at $P = (0, 0)$, $Q = (1, 2)$ and $(3, 1)$. It follows that $P + Q = -(3, 1) = (3, -1)$.
To compute $2Q$ for example, we need the tangent line at Q. This passes through $Q = (1, 2)$ and has slope dy/dx. Since

$$2y\frac{dy}{dx} = \frac{d}{dx}(x^3 + 3x) = 3x^2 + 3$$

we have $4(dy/dx) = 6$, so that the slope of the tangent line at Q will be -1. The tangent line is therefore $y = 4x + 3$, which meets E at Q, with multiplicity 2, and at $(4, 4)$, via a calculation analogous to that used above for $P + Q$. It follows that $2Q = -(4, 4) = (4, -4) = (4, 1)$.

Points of order two Points of order two are easy to spot. If $2(x, y) = \mathbf{O}$ then $(x, y) = -(x, y) = (x, -y)$, which holds if and only if $y = 0$. (It was for reasons such as this that we insisted at the outset that $p \geq 5$.)

Examples of groups
Take $p = 5$ and consider the curves $E_1 : y^2 = x^3 + x + 2$ and $E_2 : y^2 = x^3 + x$. Both curves have four points

$$E_1 : \ \mathbf{O}, (1, 2), (1, 3), (4, 0),$$

$$E_2 : \ \mathbf{O}, (0, 0), (2, 0), (3, 0),$$

so in each case we have an abelian group of order 4, which can only be C_4 or $C_2 \times C_2$. For E_1 there is exactly one point of order 2, so the group must be C_4, while for C_2 there are three points of order 2 and the group is $C_2 \times C_2$.

Possible group types
For an elliptic curve over a finite field, the group is either cyclic or a product of two cyclic groups. If $C_m \times C_m$ is a subgroup then $p = 1 (\text{mod } m)$. Thus, for example, if an elliptic curve over $\mathbb{F}_5$ has 9 points the group must be C_9.

Computing multiples of points
To compute kP for large k we can use a repeated doubling technique. Find $2P$, $4P$, $8P$,... successively, and add together whichever of these correspond to the binary expansion of k. Thus kP can be found in $O(\log k)$ steps. The process mimics modular exponentiation closely, and can be rearranged so as to avoid storing all the various points $2^r P$.

2 Lattices

2.1 Some definitions

Throughout this section k and n will be positive integers with $k \leq n$.

A **lattice** L, of **rank** (or **dimension**) k in $\mathbb{R}^n$ is a $\mathbb{Z}$-module spanned by k linearly independent vectors in $\mathbb{R}^n$ (linearly independent over $\mathbb{R}$). Thus

$$L = \{\sum_{i=1}^{k} x_i b_i : x_i \in \mathbb{Z}\},$$

where $b_1, \ldots, b_k$ are linearly independent column vectors in $\mathbb{R}^n$. Any such linearly independent spanning set for L is called a **basis**.

If $c_1, \ldots, c_k$ is another basis, then there exists a $k \times k$ integer matrix U, with determinant ± 1, such that

$$(c_1, \ldots, c_k) = (b_1, \ldots, b_k)U, \quad (*)$$

where $(c_1, \ldots, c_k)$ and$(b_1, \ldots, b_k)$ are $n \times k$ matrices.

The determinant of L, denoted $\Delta(L)$ or $\det(L)$, is defined by

$$\Delta(L) = \det(B^t B)^{1/2}$$

where $B = (b_1, \ldots, b_k)$. The matrix $B^t B$ will be a positive-definite symmetric matrix, and hence will have positive determinant. It takes the form

$$\left(b_i^t b_j\right)_{i,j \leq k}$$

with entries being the scalar products of the basis vectors. One sees from (*) that the definition of $\Delta(L)$ is independent of the choice of basis. Moreover in the special case $k = n$ we have

$$\Delta(L) = |\det(b_1, \ldots, b_n)| = |\det(B)|.$$

In fact $\Delta(L)$ is the k-dimensional volume of the **fundamental parallelotope**

$$\{\sum_{i=1}^{k} x_i b_i : 0 < x_i \leq 1\}.$$

If $L_1 \subseteq L_2$ we say that L_1 is a **sublattice** of L_2. (Here the corresponding values of n must be the same, but we might have $k_1 < k_2$.)

Example Let $k = n = 2$ and take

$$b_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

Then

$$\Delta(L) = \left| \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} \right| = 2.$$

The symmetric matrix $Q = B^t B$ may be used to define a positive definite quadratic form

$$Q(x, y) = (x\ y) B^t B \begin{pmatrix} x \\ y \end{pmatrix} = (x\ y) \begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 2x^2 + 4xy + 4y^2.$$

In general any positive definite quadratic form arises from a suitable lattice in this way. Note that $Q(x_1, \ldots, x_k) = || \sum_{i=1}^{k} x_i b_i ||^2$, where $||\ ||$ is the standard Euclidean norm on $\mathbb{R}^n$.

Instead of the basis b_1, b_2 we could use

$$c_1 = \begin{pmatrix} 7 \\ 19 \end{pmatrix}, \quad c_2 = \begin{pmatrix} 6 \\ 16 \end{pmatrix},$$

which is also a basis for L. However in many situations we want to find a basis in which the vectors are as short as possible.

2.2 The shortest non-zero lattice vector

A lattice L has a well-defined minimal length among non-zero vectors. The *square* of this length is usually denoted by $M_1(L)$. To prove this assertion we consider the quadratic form Q associated to L. Since Q is positive definite one may complete the square successively to produce

$$\begin{aligned} Q(x_1, \ldots, x_k) = a_1(x_1 + \lambda_{12} x_2 + \ldots + \lambda_{1k} x_k)^2 \\ + a_2(x_2 + \lambda_{23} x_3 + \ldots + \lambda_{2k} x_k)^2 + \ldots + a_k x_k^2. \end{aligned}$$

The coefficients a_j will all be strictly positive since Q is positive definite. Then the vector b_1 has $||b_1||^2 = Q(1, 0, \ldots, 0) = a_1$. We claim that there are only finitely many lattice vectors b with $||b||^2 \leq a_1$, or in other words, only finitely many integer vectors $(x_1, \ldots, x_k)$ with $Q(x_1, \ldots, x_k) \leq a_1$. This condition is equivalent to

$$a_1(x_1 + \lambda_{12} x_2 + \ldots + \lambda_{1k} x_k)^2 + a_2(x_2 + \lambda_{23} x_3 + \ldots + \lambda_{2k} x_k)^2 + \ldots + a_k x_k^2 \leq a_1.$$

Thus we have $a_k x_k^2 \leq a_1$, which shows that there are finitely many possibilities for x_k. Then $a_{k-1}(x_{k-1} + \lambda_{k-1,k} x_k)^2 + a_k x_k^2 \leq a_1$, so that there are finitely many possibilities for x_{k-1}, and so on.

Example
Continuing the example before, we have

$$Q(x, y) = 2x^2 + 4xy + 4y^2 = 2(x + y)^2 + 2y^2,$$

with $||b_1||^2 = a_1 = 2$. If $2(x + y)^2 + 2y^2 \leq 2$ then $y^2 \leq 1$ so that $y = 0$ or ± 1. Trying these values of y we find that $Q(x, y) \leq 2$ for integers x, y precisely when $(x, y) = (0, 0), (1, 0), (-1, 0), (-1, 1)$ or $(1, -1)$. Since we want to have a non-zero vector the first of these is disallowed, leaving 4 vectors whose length is the minimal value $\sqrt{2}$, namely $b_1, -b_1, -b_1 + b_2$ and $b_1 - b_2$. In particular $M_1(L) = 2$.

In general, in higher dimensions, this process is horribly inefficient, particularly if one starts with a basis that consists of vectors that are much larger than necessary. Thus we shall describe, later in this section, a good algorithm (the "LLL" algorithm) for finding a tolerably good basis, which can be used as a starting point for the above process. Indeed in many cases the LLL basis is already good enough for the required application.

2.3 Hermite's theorem

Hermite's theorem states that for each $k \in \mathbb{N}$ there is a constant μ_k such that

$$M_1(L)^k \leq \mu_k \Delta(L)^2$$

for every lattice L of rank k.

Sketch proof
Since we may restrict attention to the k-dimensional subspace of $\mathbb{R}^n$ in which L lies it suffices to suppose that $k = n$. Consider the fundamental parallelepiped

$$P = \{\sum_{i=1}^{k} x_i b_i : 0 < x_i \leq 1\}.$$

For different lattice points x the translates $P + x$ are disjoint, and their union is the whole of $\mathbb{R}^k$. Thus in a large cube C the number of lattice points is approximately $\mathrm{Vol}(C)/\mathrm{Vol}(P) = \mathrm{Vol}(C)/\Delta(L)$. This approximation gets better and better the larger the cube we use.

On the other hand, consider the sphere

$$S := \{x \in \mathbb{R}^k : ||x|| < \frac{1}{2}\sqrt{M_1(L)}\}.$$

We claim that the translates $S+x$ of these are also disjoint, for distinct lattice vectors $x \in L$. If this were not the case we would have two different vectors x, x' a distance less than $\sqrt{M_1(L)}$ apart. This would produce a non-zero vector $x - x' \in L$ of length less than $\sqrt{M_1(L)}$, which is impossible.

Since the translates of S are disjoint, the number of them which can lie in a large cube C is at most $\mathrm{Vol}(C)/\mathrm{Vol}(S)$. However, since there is (essentially) one such translate for every lattice point in C we deduce that $\mathrm{Vol}(C)/\Delta(L)$ is (more or less) at most $\mathrm{Vol}(C)/\mathrm{Vol}(S)$. Again the approximation gets better and better the larger the cube we use. We deduce that $\Delta(L)^{-1} \leq \mathrm{Vol}(S)^{-1}$. If we write $\mu_k^{-1/2}$ for the volume of the sphere of radius $1/2$ we have $\mathrm{Vol}(S) = M_1(L)^{k/2}\mu_k^{-1/2}$, and the result follows.

This argument does not give the best possible value for μ_k. Finding the optimal values is a well-known problem, which has been solved only for $k \leq 8$.

2.4 An important corollary

Corollary For any lattice L there is a real number $c(L) > 0$ such that $\Delta(L_1) \geq c(L)$ for every sublattice L_1 of L, irrespective of the rank of L_1.

Proof Suppose that L_1 is a sublattice of L and that L_1 has rank h. Then $M_1(L) \leq M_1(L_1)$, and $M_1(L_1) \leq \mu_h^{1/h}\Delta(L_1)^{2/h}$. Thus

$$\Delta(L_1) \geq M_1(L)^{h/2}\mu_h^{-1/2},$$

so that if L has rank k we can take $c(L) = \min_{h \leq k} M_1(L)^{h/2}\mu_h^{-1/2}$.

2.5 The Gram–Schmidt process

The Gram–Schmidt process deals with vector spaces over $\mathbb{R}$, and converts an arbitrary basis into an orthogonal one. In our situation we do not need to produce an orthonormal basis (with vectors of length 1) but merely one in which any two different basis vectors are orthogonal. To fix our notation let us recall how this is done.

Given linearly independent vectors $b_1, \ldots, b_k$ in $\mathbb{R}^n$, one defines $b_1^* = b_1$ and

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad (2 \leq i \leq k)$$

with

$$\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \quad (1 \leq j \leq i-1).$$

Then $b_1^*, \ldots, b_k^*$ will be orthogonal, and will span the same vector space as $b_1, \ldots, b_k$. However they will not span the same *lattices*, unless by fluke all the numbers μ_{ij} are integers.

2.6 LLL-reduced bases

We now begin the study of the reduction process discovered by Lenstra, Lenstra and Lovász. A lattice basis $b_1, \ldots, b_k$ is said to be LLL-reduced if the associated Gram–Schmidt basis and associated coefficients μ_{ij} satisfy

(1) $|\mu_{ij}| \leq \frac{1}{2}$ for $1 \leq j < i \leq k$; and

(2) $||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2$ for $2 \leq i \leq k$.

Motivation
Condition (1) says that $b_1, \ldots, b_k$ are "almost orthogonal", in the sense that the Gram–Schmidt coefficients μ_{ij} are small.

Condition (2) restricts the relative sizes of the basis vectors. Using orthogonality one can see that (2) is equivalent to

$$||b_i^*||^2 \geq (\tfrac{3}{4} - \mu_{i,i-1}^2) ||b_{i-1}^*||^2 \quad (2 \leq i \leq k).$$

Thus the lengths $||b_i^*||$ form an "almost-increasing" sequence.

The precise fraction $\frac{3}{4}$ which appears in (2) is not crucial. Any value strictly between $\frac{1}{4}$ and 1 could be used, but $\frac{3}{4}$ is a good compromise. Larger values would give better bases, but at the expense of longer running times.

2.7 Properties of LLL-reduced bases

Throughout this section suppose that $b_1, \ldots, b_k$ is an LLL-reduced basis for L, with associated Gram–Schmidt coefficients μ_{ij}. Then

(a) We have

$$||b_j||^2 \leq 2^{i-1} ||b_i^*||^2 \leq 2^{i-1} ||b_i||^2$$

for $1 \leq j \leq i \leq k$.

(b) $\Delta(L) \leq \prod_{i=1}^{k} ||b_i|| \leq 2^{k(k-1)/4} \Delta(L)$.

(c) $||b_1|| \leq 2^{(k-1)/4}\Delta(L)^{1/k}$.

(d) For every non-zero vector x in L we have $||b_1|| \leq c||x||$ where

$$c = \max_{1\leq i\leq k} \frac{||b_1||}{||b_i^*||} \leq 2^{(k-1)/2}.$$

Remarks
From (c) one sees that

$$M_1(L)^k \leq ||b_1||^{2k} \leq 2^{k(k-1)/2}\Delta(L)^2,$$

giving us a form of Hermite's theorem, with the (rather bad) explicit value $\mu_k = 2^{k(k-1)/2}$. Notice also that (d) shows that $||b_1||$ is close to being a minimal-length vector in the lattice. It is larger than minimal by at worst the constant factor c.

Proofs
(a) We have

$$||b_i^*||^2 \geq (\tfrac{3}{4} - \mu_{i,i-1}^2)||b_{i-1}^*||^2 \geq \tfrac{1}{2}||b_{i-1}^*||^2 \ \ (2 \leq i \leq k).$$

Hence $||b_j^*||^2 \leq 2^{i-j}||b_i^*||^2$ for $1 \leq j \leq i \leq k$ by induction. Thus

$$\begin{aligned} ||b_i||^2 &= ||b_i^*||^2 + \sum_{j=1}^{i-1} \mu_{ij}^2||b_j^*||^2 \quad (**) \\ &\leq \left(1 + \sum_{j=1}^{i-1} 2^{i-j-2}\right)||b_i^*||^2 \\ &= (1 + \tfrac{1}{4}(2^i - 2))||b_i^*||^2 \\ &\leq 2^{i-1}||b_i^*||^2, \end{aligned}$$

so that

$$||b_j||^2 \leq 2^{j-1}||b_j^*||^2 \leq 2^{j-1+i-j}||b_i^*||^2 = 2^{i-1}||b_i^*||^2.$$

The inequality (a) then follows, using (**).

(b) Let B be the matrix with columns $b_1, \ldots, b_k$ and let B^* be the matrix with columns $b_1^*, \ldots, b_k^*$. Then $B = B^*J$ where J is the upper triangular matrix

$$J = \begin{pmatrix} 1 & \mu_{2,1} & \mu_{3,1} & \cdots & \mu_{k,1} \\ 0 & 1 & \mu_{2,3} & \cdots & \mu_{k,2} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \mu_{k,k-1} \\ 0 & \cdots & \cdots & \cdots & 1 \end{pmatrix}.$$

Then $\Delta(L)^2 = \det(B^tB) = \det((B^*)^tB^*)$, since $\det(J) = 1$. However the vectors b_i^* are orthogonal, so that $(B^*)^tB^*$ is a diagonal matrix, with diagonal entries $(b_i^*)b_i^* = ||b_i^*||^2$. We therefore conclude that $\Delta(L)^2 = \prod_{i=1}^k ||b_i^*||^2$. However (**) yields $||b_i^*|| \leq ||b_i||$, so that

$$\begin{aligned}
\Delta(L) &\leq \prod_{i=1}^k ||b_i|| \quad \text{(which is the first inequality in (b))} \\
&\leq \prod_{i=1}^k 2^{(i-1)/2}||b_i^*|| \quad \text{(using (a) with } j = i\text{)} \\
&= 2^{k(k-1)/4} \prod_{i=1}^k ||b_i^*|| \\
&= 2^{k(k-1)/4}\Delta(L).
\end{aligned}$$

Remark The same argument shows that $\prod_{i=1}^k ||c_i|| \geq \Delta(L)$ for any basis $c_1, \ldots, c_k$ of L, whether LLL-reduced or not.

(c) Set $j = 1$ in (a) and take the product for $1 \leq i \leq k$ to get

$$||b_1||^{2k} \leq \prod_{i=1}^k 2^{i-1}||b_i^*||^2 = 2^{k(k-1)/2}\Delta(L)^2,$$

and the required inequality follows.

(d) We have $||b_1||^2 \leq c^2||b_i^*||^2$ for $1 \leq i \leq k$, by definition of c. Any non-zero x in L may be written as $\sum_{i=1}^k r_i b_i$ with $r_i \in \mathbb{Z}$, and also as $\sum_{i=1}^k r_i^* b_i^*$ with $r_i^* \in \mathbb{R}$. Choose the largest index i such that $r_i \neq 0$ and call it $i = h$. By construction of the Gram–Schmidt basis one has $b_h^* - b_h \in \langle b_1, \ldots, b_{h-1}\rangle$, and also $b_i^* \in \langle b_1, \ldots, b_{h-1}\rangle$ for $1 \leq i \leq h-1$. It follows that $(r_h - r_h^*)b_h \in \langle b_1, \ldots, b_{h-1}\rangle$, and since the the vectors b_i are linearly independent this shows that $r_h^* = r_h$. However r_h is a non-zero integer so that $||r_h^*|| \geq 1$. We therefore have

$$||x||^2 = \sum_{i=1}^k (r_i^*)^2||b_i^*||^2 \geq (r_h^*)^2||b_h^*||^2 \geq ||b_h^*||^2 \geq ||b_1||^2/c^2$$

as required.

2.8 The LLL reduction algorithm

Suppose that we are given a basis $b_1, \ldots, b_k$ of a lattice L, and we wish to find a reduced basis, as defined by the conditions (1) and (2) of §2.6. The first

thing that we do is to compute the associated Gram–Schmidt basis $b_1^*, ..., b_k^*$ and the constants μ_{ij}. Let $m = 2$. (We'll see the significance of m later.) If $k = 1$ then we are finished. Otherwise we do the following:

Step A
Reduce $|\mu_{m,m-1}|$ to at most $\frac{1}{2}$, by adding a suitable multiple of b_{m-1} to b_m.

For example, take

$$b_1 = \begin{pmatrix} 7 \\ 19 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 6 \\ 16 \end{pmatrix}.$$

Then $b_1^* = b_1$ and

$$\mu_{2,1} = \frac{1}{49 + 361} \begin{pmatrix} 7 & 19 \end{pmatrix} \begin{pmatrix} 6 \\ 16 \end{pmatrix} = \frac{346}{410} = \frac{173}{205} > \frac{1}{2}.$$

So our basis is not yet LLL-reduced. Moreover

$$b_2^* = \begin{pmatrix} 6 \\ 16 \end{pmatrix} - \frac{173}{205} \begin{pmatrix} 7 \\ 19 \end{pmatrix} = -\frac{1}{205} \begin{pmatrix} 19 \\ -7 \end{pmatrix}.$$

To carry out Step A, we replace b_2 by

$$b_2 - b_1 = \begin{pmatrix} -1 \\ -3 \end{pmatrix}.$$

Note that b_1^* and b_2^* are unchanged.

Step B
If condition (2) holds, with $i = m$, then go to Step C.
Otherwise, swap b_{m-1} and b_m; and make appropriate modifications to the Gram–Schmidt basis, and then to the μ_{ij}.
If $m > 2$ replace m by $m - 1$.
Return to Step A.

Note that condition (2) requires us only to know the lengths of the b_i^*, so one needs only to keep track of their current lengths, and of the μ_{ij}. If a swap is performed in Step B, then the lengths of b_{m-1}^* and b_m^* are interchanged, but no others, and the μ_{ij} may change for $i \geq m - 1$.

Step C
For $j = m - 2, m - 3, \ldots, 1$, reduce $|\mu_{m,j}|$ to at most $\frac{1}{2}$, by adding a suitable multiple of b_j to b_m.

Increase m by 1. If $m = k + 1$ stop, and otherwise return to Step A.

The significance of the parameter m is that whenever we go to Step A, the vectors $b_1, \ldots, b_{m-1}$ are an LLL-reduced basis for the lattice which they span. Initially m is set to 2, and the aim is to reach $m = k + 1$. When going from Step B to Step A, the value of m decreases, unless it is already equal to 2. In contrast, when going from Step C to Step A the value of m increases. The reader will find it helpful to draw a flowchart for the process!!

Since the value of m can go down as well as up it is not obvious that the algorithm terminates. To investigate this we look closely at the way any changes to the basis affect various sublattices. For each $i = 1, \ldots, k$ we define L_i as the sublattice of L spanned by $b_1, \ldots, b_i$.

The first thing to notice is that neither Step A nor Step C change any of the lattices L_i. Consider what happens in Step B. If a swap is performed the only lattice which changes is L_{m-1}. The new value of b^*_{m-1} is

$$\text{old}\, b^*_m + \mu_{m,m-1}.\text{old}\, b^*_{m-1}.$$

The swap is performed only when condition (2) fails, and we see that this implies that the process multiplies $||b^*_{m-1}||^2$ by a factor which is at most $\frac{3}{4}$. However we have

$$\begin{aligned} \Delta(L_{m-1})^2 &= \det((b_1 \ldots b_{m-1})^t (b_1 \ldots b_{m-1})) \\ &= \det((b^*_1 \ldots b^*_{m-1})^t (b^*_1 \ldots b^*_{m-1})) \\ &= \prod_{i=1}^{m} ||b^*_i||^2. \end{aligned}$$

(For the second equality one needs to remember that the matrix relating a basis to its Gram–Schmidt orthogonalization is upper triangular, with 1's on the diagonal.) This equality shows that a swap at Step B has the effect of reducing $\Delta(L_{m-1})^2$ by a factor of at most $\frac{3}{4}$, while leaving $\Delta(L_i)^2$ fixed for all $i \neq m - 1$. However the corollary in §2.4 shows that these $\Delta(L_i)$ are all bounded below by a positive constant, so the algorithm must eventually terminate.

2.9 An example

As before, we consider the lattice spanned by

$$b_1 = \begin{pmatrix} 7 \\ 19 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 6 \\ 16 \end{pmatrix}.$$

Then $\Delta(L_1)^2 = 49 + 361 = 410$, while

$$\Delta(L_2) = \Delta(L) = \left| \det \begin{pmatrix} 7 & 6 \\ 19 & 16 \end{pmatrix} \right| = 2.$$

Property (c) in §2.7 shows that an LLL-reduced basis would have

$$||b_1||^2 \leq 2^{(k-1)/4} \Delta(L)^{1/k} = 2^{1/4}.2^{1/2} = 2^{3/4},$$

so that our basis is very far from reduced.

We compute

$$b_1^* = b_1, \quad \mu_{2,1} = \frac{173}{205}, \quad b_2^* = \frac{1}{205} \begin{pmatrix} 19 \\ -7 \end{pmatrix}$$

and go to Step A, with $m = 2$.

Step A: Replace b_2 by

$$b_2 - b_1 = \begin{pmatrix} -1 \\ -3 \end{pmatrix},$$

so that $\mu_{2,1}$ is replaced by $\frac{-7-57}{410} = \frac{-64}{410}$, which has absolute value at most $\frac{1}{2}$.

Step B: Condition (2) fails (it must do, since we know that $||b_1||$ is not small enough), so we swap b_1 and b_2 to get

$$b_1 = \begin{pmatrix} -1 \\ -3 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 7 \\ 19 \end{pmatrix}.$$

Observe that we now have $\Delta(L_1)^2 = 10 < \frac{3}{4} \cdot 410$. We return to Step A, still with $m = 2$.

Step A: This time we replace b_2 by

$$b_2 + 6b_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

and go to Step B.

Step B: Condition (2) still fails, unsurprisingly since $||b_1||$ is still too large. So we swap the basis vectors to reach

$$b_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad b_2 = \begin{pmatrix} -1 \\ -3 \end{pmatrix}.$$

Again $\Delta(L_1)^2$ has been reduced, and one checks that $\Delta(L_1)^2 = 2 < \frac{3}{4} \cdot 10$. We return to Step A.

Step A: Replace b_2 by

$$b_2 + 2b_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

and go to Step B.

Step B: At last condition (2) is satisfied, so we go to step C.

Step C: Increase m to 3, and stop, returning

$$b_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

2.10 The knapsack problem

We are given a knapsack which can carry a limited weight and we have a given set of objects of known weights. We want to pack as much into the knapsack as possible.

Example The given weights are $1, 2, 4, 8, 16, 32, 64$ and the knapsack can hold weight 12. There is a unique optimal packing, with weights 4 and 8. Indeed for a knapsack of any given capacity there is a unique optimal packing, which is easy to spot.

In general, if the weights $w_1, \ldots, w_k$ are in "superincreasing order", such that

$$w_i > \sum_{j=1}^{i-1} w_j \quad \text{for } i = 2, \ldots, k,$$

then the knapsack problem is easy: one uses the **greedy algorithm**, packing the heaviest weight possible, then the next heaviest possible, and so on.

However in general the knapsack problem is hard. Indeed it is known to be NP-complete. Given that it is *provably* hard, cryptographers have tried

to devise coding systems based on knapsack problems. Unfortunately, while we know that the knapsack problem in general is hard, it is not so easy to produce, and use, particular examples of hard knapsack problems.

2.11 A knapsack public-key cryptosystem

Alice wants to send a secret message to Bob. Bob chooses some secret (but easy) knapsack weights $w_1, \ldots, w_k$ (a superincreasing sequence, for example). He chooses two (secret) coprime integers N, e with $N > \sum_1^k w_i$. He computes knapsack weights $h_1, \ldots, h_k$ satisfying

$$h_i = ew_i (\text{mod } N).$$

There will be infinitely many choices satisfying these conditions and Bob tries to make his weights correspond to a hard knapsack problem. Bob makes public his weights $h_1, \ldots, h_k$.

To send Bob a message consisting of k binary digits $x_1, \ldots, x_k$ Alice computes

$$M = \sum_{i=1}^{k} x_i h_i$$

and sends M to Bob.

To decrypt the message Bob finds d such that $de = 1 (\text{mod } N)$ and computes

$$\begin{aligned} dM &= \sum_{i=1}^{k} x_i (dh_i) \\ &= \sum_{i=1}^{k} x_i w_i (\text{mod } N) \\ &= \sum_{i=1}^{k} x_i w_i \quad (\text{since } N > \sum w_i). \end{aligned}$$

Bob can easily recover $x_1, \ldots, x_k$ since the weights $w_1, \ldots, w_k$ correspond to an easy knapsack problem.

The operations required to implement this system are extremely simple — much easier than in RSA or in discrete logarithm cryptosystems.

Suppose the evil Eve hacks into the system and intercepts the message M sent by Alice to Bob. She knows the weights $h_1, \ldots, h_k$ which Bob had published, but does not know e, d or N. If she wants to recover $x_1, \ldots, x_k$ she will have to solve the hard knapsack problem $\sum x_i h_i = M$.

Example Choose easy weights $w_i = 3^i$ for $1 \leq i \leq 5$, and take $N = 400 (> 3 + 9 + 27 + 81 + 243 = 363)$. Pick $e = 147$ say, and compute (easily, using the Euclidean algorithm) $d = 283$. Choose (hard ?) knapsack weights $h_i = 147 w_i (\text{mod } 400)$ as $h_1 = 41$, $h_2 = 123$, $h_3 = 369$, $h_4 = 307$, and $h_5 = 121$. These are the weights that are made public.

Alice wants to send us the message $1, 0, 0, 1, 1$, which she encodes as $M = 41 + 307 + 121 = 469$.

To decrypt this we compute $283 \times 469 = 327 (\text{mod } 400)$, and solve the easy problem $327 = 1.3^1 + 0.3^2 + 0, 3^3 + 1.3^4 + 1.3^5$ to recover the message.

2.12 Breaking the knapsack system with LLL

If we are lucky, we may be able to use LLL to break the knapsack cryptosystem. We want to find $x_i \in \{0, 1\}$ such that $\sum_1^k x_i h_i = M$.

Consider the lattice of rank $k + 1$ generated by the basis

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ h_1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ h_2 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \\ h_k \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ -M \end{pmatrix}.$$

A solution to $\sum_1^k x_i h_i = M$ will give a (very) short lattice vector

$$\begin{pmatrix} x_1 \\ \vdots \\ x_k \\ 0 \end{pmatrix}$$

which LLL may be able to find.

For the example of the previous section, starting with

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 41 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 123 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 369 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 307 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 121 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -469 \end{pmatrix}$$

the LLL algorithm produces the reduced basis

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -2 \\ 1 \\ 0 \\ -2 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ -2 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ -2 \\ -5 \\ -2 \\ 3 \\ 3 \end{pmatrix},$$

and the first of these does indeed recover the message M.

(Incidentally, the squared lengths of these reduced basis vectors are, in order, 3, 6, 7, 10, 8, 51. Generally we expect the lengths to be in increasing order, but it does not always happen, as this example illustrates.)

3 Factorisation of integers

3.1 Overview, trial division, and Fermat's method

The problem Given an integer $n > 1$ we want to find a prime factor p of n.

If we have an algorithm to do this, we can iterate it on n/p to find the complete prime factorisation of n. From now on we will assume that n is not prime, since there are good, fast, (polynomial) algorithms for primality testing — an interesting topic which we shall not go into here.

A solution: trial division
Try dividing n by 2, 3, 5, 7, ... until a factor is found. One may either test only primes as possible factors, which requires one to have a way (maybe a table) to detect primes, or one can try all odd divisors. Given that n is composite, it has a factor $p \leq \sqrt{n}$, so at most $\sqrt{n}$ divisions are necessary.

Trial division is an important benchmark against which to measure other algorithms. It is a very poor algorithm for numbers which have been specially constructed to be hard to factor, by multiplying together two large primes for example. On the other hand, for large numbers n produced randomly, trial division has an excellent chance of finding a prime divisor very quickly. Indeed for 50% of large random integers, trial division finds a factor at the very first attempt! Thus for randomly produced integers trial division up to some cut-off point may be a sensible first move.

Fermat's method
Try to solve $n = x^2 - y^2$ for integers x, y, so that $n = (x - y)(x + y)$ gives a factorization of n. Usually one searches using x, starting at $x_0 = \lceil \sqrt{n} \rceil$ and trying $x_0, x_0 + 1, x_0 + 2, \ldots$. If $n = pq$ with p small one needs to reach $x = (p + q)/2$, which will be much larger than $\sqrt{n}$. So the method will be much worse than trial division. However if p and q are close the method can be more efficient. For example, with $n = 971609 = 809 \times 1201$ one starts at $x_0 = 986$, and has to try values of x until one reaches $(p+q)/2 = 1005$; thus there are 20 trials to perform.
Fermat's method can be refined, for example by using congruences to restrict the values of x to be tried. In this case $n = 1 (\text{mod } 8)$, so that if $x^2 - y^2 = n$ the number $x^2 - 1$ will be a square (mod 8). Thus x cannot be even. Similarly $n = 2 (\text{mod } 3)$ so that $x^2 - 2$ must be a square (mod 3). This can only happen when $3 \mid x$. These consideration reduce the values of x to be tried to $x = 987, 993, 999, 1005, 1011, \ldots$, of which the fourth value is successful.

Fermat's method is now of historical interest only, but many modern approaches try to express n as a difference of squares by much more subtle methods.

3.2 The remainder of the course

For the rest of the course we will take a look at a few of the many factorisation methods that have been published. Some are true algorithms, while others have a probabilistic and/or heuristic element to them. Although our focus will be on running times, the reader should also think about the difficulties in implementation, the possibility of parallelisation, and questions of storage requirements, for example.

Most of the mathematical prerequisites have been prepared, but we will have an interlude on smooth numbers when we discuss factor-base methods.

3.3 Euler's method and its more recent variants

Euler observed that if, for a given d, one can find two different representations of n as $x^2 + dy^2$, say as

$$n = x_0^2 + dy_0^2 = x_1^2 + dy_1^2,$$

then $n \mid (y_1x_0)^2 - (y_0x_1)^2 = (y_1x_0 - y_0x_1)(y_1x_0 + y_0x_1)$. Unless we are unlucky, and one can give precise conditions under which this happens, n will divide neither of the individual factors $y_1x_0 - y_0x_1$ or $y_1x_0 + y_0x_1$, so that we will obtain a non-trivial divisor of n by computing $\gcd(n, y_1x_0 - y_0x_1)$.

For a given choice of d one would check possible values $y \leq \sqrt{n/d}$ to see whether $n - dy^2$ is a square. Although a large value of d decreases the amount of work to be done, it also decreases the chance that n will have one, let alone two, representations as $x^2 + dy^2$.

3.4 Lehmer & Lehmer's variant (1974)

Suppose that n is a product of just two prime factors. (Trial division up to $n^{1/3}$ will reduce us to this case.) Then one can guarantee that one of the following ten forms will split n via Euler's method:-

$$x^2 + y^2,\ x^2 + 2y^2,\ x^2 - 2y^2,\ x^2 + 3y^2,\ x^2 - 3y^2,$$

$$x^2 + 6y^2,\ x^2 - 6y^2,\ 3x^2 - y^2,\ 6x^2 - y^2,\ 2x^2 + 3y^2.$$

For the indefinite forms, it suffices to look at values x, y of size $O(\sqrt{n})$. In fact, for a specific n, one can reduce the number of forms to be tried to 3, by congruence considerations.

One can reduce the possible values of x, y to be tried using congruences, in the same way as with Fermat's method. At the time it was devised this method was one of the fastest in practice(!)

3.5 McKee's method

In this adaptation of Euler's method one uses a large value of d.

Step 1 Check that n is not a square, or a higher power. If it is we have found a factor. Otherwise choose $x_0 = \lfloor \sqrt{n - n^{2/3}} \rfloor$ and $d = n - x_0^2$. Then $n = x_0^2 + d$, with d approximately $n^{2/3}$.

Step 2 Check n for factors up to $(4d/3)^{1/4} = O(n^{1/6})$, using trial division; and stop if you find one.

Step 3 For each integer a in the interval $[\sqrt{d/12}, \sqrt{4d/3}]$, search for solutions to $an = x^2 + dy^2$ with $x, y \in \mathbb{N}$ and $y^2 \neq a$. It can be proved (though we shall not do it here) that if n is composite, and the algorithm reaches this stage, then there always will be such a solution.

For good procedures to search for solutions x, y see below.

Step 4 Having found solutions $n = x_0^2 + d,\ an = x_1^2 + dy_1^2$ it follows that $n \mid (y_1 x_0)^2 - x_1^2$. It is then not hard to check that $\gcd(n, y_1 x_0 - x_1)$ is a non-trivial factor of n.

One can show that a suitable implementation of this will run in $O(n^{1/3+\varepsilon})$ steps, for any small fixed $\varepsilon > 0$. In practice it is better to take d somewhat smaller than $n^{2/3}$. With d around $n^{1/2}$ one gets an algorithm with theoretical running time $O(n^{3/8+\varepsilon})$, which soon beats trial division.

To search for solutions of $an = x^2 + dy^2$ in Step 3 one could first factor d via trial division, and find a quadratic non-residue for each prime factor p. The algorithm of §1.11 may then be used to solve $x^2 = an (\mathrm{mod}\ p)$ for each such p, and then one can combine the solutions using the Chinese Remainder

Theorem. All this becomes far easier if we can choose d to be prime in the first place.

Example
Take $n = 1082154235955237$

Step 1 $n = 32896084^2 + 1893420181 = x_0^2 + d$, say, where in fact d is prime.
Step 2 Trial division up to 233 is a relatively tiny amount of work, and finds no prime divisors.
Step 3 Search over values for a between 12562 and 50245, finding that

$$43036n = 2591866961^2 + d.145101^2.$$

Note that with these values of d and a the congruence $x^2 = an (\text{mod } d)$ has only 7 solutions which need checking, in the range $x < \sqrt{an}$.
Step 4 $n = 12345701 \times 87654337$.

3.6 Modern speed-ups of Fermat's method

Instead of looking for solutions of $x^2 - y^2 = n$ one can try $x^2 - ny^2 = z^2$, in effect making n a difference of two rational squares, rather than integer ones. There are variants in which one examines $x^2 - any^2 = z^2$ for some small a, on the basis that if one finds prime factor of an which is suitably large (we hope), this will divide n.

There are two approaches in the literature, one is to make $x^2 - ny^2$ small, in the hope that this will increase the chance of $x^2 - ny^2$ being square. This leads to the Continued Fraction Method, and to SQUFOF (the SQUare FOrm Factoring method). The second method keeps both x and y small, and is due to McKee (1999).

These methods are expected to run in time $O(n^{1/4+\varepsilon})$, though we currently cannot prove this. In practice they are soon better than trial division.

The Continued Fraction and SQUFOF methods

Take x/y to be a continued fraction approximation to $\sqrt{n}$. Then $|x/y - \sqrt{n}| < y^{-2}$, so that

$$|x/y + \sqrt{n}| < y^{-2} + 2\sqrt{n} \leq 1 + 2\sqrt{n}.$$

Hence

$$|x^2/y^2 - n| < (1 + 2\sqrt{n})/y^2,$$

giving us $|x^2 - ny^2| < 1 + 2\sqrt{n}$. Thus the chance that $x^2 - ny^2$ is a square is $O(n^{-1/4})$. So one runs the continued fraction algorithm for $\sqrt{n}$, keeping track only of the value of $x (\mathrm{mod}\ n)$. (We have $x = p_k$ in the notation of §§4–8, and the numbers p_k grow exponentially; but fortunately it is enough to work with $x (\mathrm{mod}\ n)$.) If $x = x' (\mathrm{mod}\ n)$ with $0 \le x' < n$ then we have $x'^2 = r (\mathrm{mod}\ n)$ with a remainder r which is guaranteed to have $r < 1 + 2\sqrt{n}$. If we have $r = z^2$ then $\gcd(x' - z, n) = \gcd(x - z, n)$ will be a factor of n (if we are lucky).

SQUFOF (see Cohen for details) was an important algorithm in its day, and in effect runs through the continued fraction algorithm, but without computing x or y.

McKee's 1999 method

Suppose that n is odd and composite, and set $b = \lceil \sqrt{n} \rceil$. Define a quadratic form

$$Q(x, y) = (x + by)^2 - ny^2 = x^2 + 2bxy + (b^2 - n)y^2,$$

so that the coefficients are $O(\sqrt{n})$. We will seek integers x, y, z such that $Q(x, y) = z^2$, in the hope that $\gcd(x + by - z, n)$ is a non-trivial factor of n. Note that $y = 1$ corresponds to Fermat's method.
McKee's method relies on the following lemma.

Lemma
Suppose that $n = pq$ with $q > p > 2n^{1/4}$, and choose a positive integer $T \le n^{1/2} - n^{1/4}$. Then there are at least T distinct integer triples (x, y, z) such that $Q(x, y) = z$ and

(1) y is even and $2 \le y \le n^{1/4} + 2(T - 1)$,

(2) $y|x| < 2T^{1/4}\sqrt{n}$,

(3) $|z| < (T^2 - 1)\sqrt{n}$.

Moreover $\gcd(x + by - z, n)$ is a non-trivial factor of n in each case.

Proof
The proof consists of unexciting technical details, which you might want to

skip!
For each integer $t = 0, \dots, T-1$ let

$$r = \lfloor \sqrt{q/p} \rfloor < \sqrt{q/p} \leq \sqrt{(n^{3/4}/2)/(2n^{1/4})} = n^{1/4}/2$$

and put $y = 2(r+t)$, so that (1) holds. Take

$$z = q - (r+t)^2 p \quad \text{and} \quad x = q + p(r+t)^2 - by$$

so that

$$Q(x,y) = (x+by)^2 - ny^2 = (q+p(r+t)^2)^2 - 4pq(r+t)^2 = (q-(p(r+t)^2)^2 = z^2.$$

Moreover

$$z \leq q - r^2 p \leq q - (\sqrt{q/p} - 1)^2 p = 2\sqrt{qp} - p < 2\sqrt{n}$$

and

$$\begin{aligned} -z &\leq -q + (\sqrt{q/p} + T - 1)^2 p = 2(T-1)\sqrt{qp} + (T-1)^2 p \\ &\leq 2(T-1)\sqrt{n} + (T-1)^2\sqrt{n} = (T^2-1)\sqrt{n}, \end{aligned}$$

which verifies (3). For (2) we first suppose that $x \geq 0$. In this case we have

$$2bxy \leq (x+by)^2 - b^2 y^2 = (n-b^2)y^2 + z^2 \leq z^2 \leq T^4 n$$

by (3), on recalling that b was defined to be $\lceil \sqrt{n} \rceil$. Thus we have

$$xy \leq T^4 n/2b \leq T^4\sqrt{n}$$

for $x \geq 0$. If $x < 0$ we put $r = \sqrt{q/p} - \eta$, with $0 \leq \eta < 1$. Then if we set $b = \sqrt{n} - \delta$ with $0 \leq \delta < 1$ we may calculate that

$$x = q + p(r+t)^2 - by = p(t-\eta)^2 + 2\eta\delta - 2\delta\sqrt{q/p} - 2\delta t \geq -2\delta(\sqrt{q/p} + T).$$

If $x < 0$ this yields

$$|x| \leq 2\sqrt{q/p} + 2T \leq 2p^{-1}\sqrt{n} + 2T \leq n^{1/4} + 2T,$$

since $p > 2n^{1/4}$. On the other hand

$$y = 2(r+t) \leq 2(\sqrt{q/p} + T) \leq n^{1/4} + 2T$$

by the same argument, so that

$$y|x| \leq (n^{1/4} + 2T)^2 \leq 2T^4\sqrt{n}$$

if $n \geq 1000$, say. This completes the proof of (2).
Finally we observe that $x + by - z = 2p(r+t)^2$. This is divisible by p, but not by q, since $0 < q + r < n^{1/4} + T \leq \sqrt{n}$.

The clever bit in the algorithm, which is adaptable to other problems, is as follows. We are looking for a solution to $Q(x,y) = z^2$ with the variables in certain ranges given by the lemma. Given a range $X < \ell \leq 2X$, the probability that z will have a prime factor ℓ in this range is of order 1 in $\log X$. So if we take T a bit bigger than $\log X$, there is a pretty good chance that there will be a solution in which z has a prime divisor $\ell \in (X, 2X]$. So we run through the various possible p, looking for solutions in which $\ell \mid z$.
How can we solve $Q(x,y) = 0 (\mathrm{mod}\ \ell^2)$? We begin by finding the solutions x_0 of $Q(x_0, 1) = 0 (\mathrm{mod}\ \ell^2)$. There will normally be either two solutions or none. Then $x = x_0 y (\mathrm{mod}\ \ell^2)$ for one of the solutions, so that $x = x_0 y - \mu\ell^2$, say. However if we choose $P \geq 2T^2 n^{1/4}$ then

$$\left| \frac{x_0}{\ell^2} - \frac{\mu}{y} \right| = \left| \frac{x}{\ell^2 y} \right| = \frac{2|x|y}{\ell^2} \frac{1}{2y^2} \leq \frac{4T^4\sqrt{n}}{P^2} \frac{1}{2y^2} \leq \frac{1}{2y^2}$$

by part (2) of the lemma, and this means that μ/y will be one of the convergents in the continued fraction expansion of x_0/ℓ^2.
Hence for each prime ℓ we will be able to find any solution with $\ell \mid z$ in polynomial time. Taking P to be of order $T^2 n^{1/4}$ then gives us a running time $O(n^{1/4+\varepsilon})$, but without a complete guarantee of success.

3.7 Other $O(n^{1/4+\varepsilon})$ algorithms

There are in the literature a large number of algorithms with expected running time $O(n^{1/4+\varepsilon})$. Indeed the fastest known algorithms which are both deterministic and provably correct have this running time. (They are therefore "better" than those described above, being both deterministic and provably correct.) One of these, due to Strassen (1977) merely computes $\lfloor\sqrt{n}\rfloor!$ modulo n in a very efficient manner. Taking the gcd with n gives a factor of n.

One of the simplest algorithms is Pollard's ρ-algorithm.
Step 1 Choose an integer n other than 0 or -2, and define $f(x) = x^2 + n$. Set $a = b = 2$.
Step 2 Compute $a' = f(a) (\mathrm{mod}\ n)$ and $b' = f(f(b)) (\mathrm{mod}\ n)$. Find the gcd of $|a' - b'|$ and n. If the value is n give up. If the value is between 1 and n we have found a proper factor of n.
Step 3 If the gcd is 1 replace a by a' and b by b', and return to Step 2.

Example (From Wikipedia)
Take $n = 8051$ and $f(x) = x^2 + 1$. Then

(i) a=2 and $b = 2$ become 5 and 26, and $\gcd(21, 8051) = 1$.

(ii) a=5 and $b = 26$ become 26 and 7474 (mod 8051), and $\gcd(7448, 8051) = 1$.

(iii) a=26 and $b = 7474$ become 677 and 871 (mod 8051), and $\gcd(194, 8051) = 97$.

After k steps we have $a = f^k(2) (\mathrm{mod}\ n)$ and $b = f^{2k}(2) (\mathrm{mod}\ n)$. If p is a prime factor of n we expect the values $f^k(2) (\mathrm{mod}\ p)$ to form a random sequence, which eventually has a repetition, so that $f^k(2) = f^j(2) (\mathrm{mod}\ p)$ with $k < j$. As in the birthday paradox, we expect this to happen with $k, j = O(\sqrt{p})$. Write $\ell = j - k$, so that the sequence has an initial section followed by a cycle of period ℓ. (Pictorially one can think of this as forming a letter "rho" as in the name of the algorithm.) Now, if $h = \ell\lceil k/\ell \rceil$ then $h \geq k$ and $f^h(2) = f^{2h}(2)$. Thus after h steps we will have $p \mid a - b$. We expect $h = 0(\sqrt{p})$. Moreover we will have $n \mid a - b$ only if the cycle lengths for all other prime factors of n happen to divide h.

Thus failures of the algorithm may occur, but are uncommon. Moreover the expected running time is $O(n^{\varepsilon}\sqrt{p})$ where p is the smallest prime factor of n. This is certainly $O(n^{1/4+\varepsilon})$. For comparison, trial division takes time roughly $O(p)$, while some of the other algorithms discussed have running times $O(n^{1/4+\varepsilon})$ even when one of the prime factors is small.

3.8 Factor bases

There are many situations where "factor bases" are useful, but we will consider here the problem of finding integers such that $x^2 = y^2 (\mathrm{mod}\ n)$, in the hope of using this to factor n. The basic idea is to find many congruences $x_i^2 = y_i (\mathrm{mod}\ n)$, with $1 \leq i \leq K$, say, and then to look for a subset of indices I such that $\prod_{i \in I} y_i$ is a square y^2. We will then be able to take $x = \prod_{i \in I} x_i$.

Example (From Koblitz)
Take $n = 1829$. We have

$$42^2 = -65, \ \ 43^2 = 20, \ \ 61^2 = 63, \ \ 85^2 = -91, \ \ \text{all (mod 1829)}.$$

The product of the four right-hand sides is $2^2 3^2 5^2 7^2 13^2 = 2730^2$. On the other hand $42.43.61.85 = 1459 (\mathrm{mod}\ 1829)$, so that $1459^2 = 2730^2 (\mathrm{mod}\ 1829)$. We

therefore compute $\gcd(2730 - 1459, 1829) = \gcd(1271, 1829) = 31$, giving a factor of 1829.

Generally, how do we find a product of the y_i's which is a square? A **factor base** is a set B of relatively small primes, together with the "prime" -1. For a simple model we will use the primes p up to a certain bound b. We then generate lots of pairs (x, y) such that $x^2 = y(\text{mod } n)$, and keep only those, (x_i, y_i) (with $1 \leq i \leq N$ say), for which y_i is a product of factors in B. We can then write

$$y_i = \prod_{p \in B} p^{e_{p,i}},$$

and in order to make $\prod_{i \in I} y_i$ a square it suffices to find exponents $f_i = 0$ or 1, such that the simultaneous congruences

$$\sum_{i=1}^{N} f_i e_{p,i} = 0(\text{mod } 2) \quad (\forall p \in B)$$

hold. This is a set of equations over $\mathbb{F}_2$ which can be solved by Gaussian elimination. We will have a non-trivial solution if $N > \#B$, so we will need to have more suitable relations than primes in the factor base.

One feature of factor base methods is clear. There will be a non-trivial memory requirement, since we will have to store the various pairs (x_i, y_i) and manipulate the matrix of exponents $e_{p,i}$. In contrast, the algorithms in the course before this point have had relatively minor memory requirements.

Large-prime variants
There are many variations on this line of attack, but one important one retains pairs (x, y) for which $y = y'p$ with y' a product of primes in the factor base and p a medium size prime — outside the factor base but not too enormously large. If two such pairs (x, y) are found which have the same extra factor p, then we can eliminate p, modulo squares, by multiplying the two relations.

3.9 Smooth numbers

An understanding of "smooth numbers" is essential for the analysis of factor base methods. If y is a positive real number we say that $n \in \mathbb{N}$ is y-smooth, if all prime factors of n have size at most y. Thus 105 is 8-smooth, but not 6-smooth.

Definition $\psi(x,y)$ is the number of y-smooth integers $n \le x$.
For example, the 3-smooth integers up to 20 are 1,2,3,4,6,8,9,12,16,18, so that $\psi(20,3) = 10$.

We will only be interested in the situation in which $2 \le y < x$. The ratio $\log x/\log y$ plays a crucial role in understanding $\psi(x,y)$ and, following standard practice, we define

$$u = \frac{\log x}{\log y}$$

and introduce the **Dickman function** $\rho(u)$.

Facts (which we will use without proof)
(i) There is a function $\rho(u)$ such

$$\psi(x,y) = \rho(u)x + O(x/\log y) \quad (x \ge y \ge 2).$$

The "main term" $\rho(u)x$ may be smaller than the "error term" $O(x/\log y)$ unless u is fairly small. However one can prove formulae with better error terms. The function $\rho(u)$ satisfies $\rho(u) = 1$ for $0 \le u \le 1$ and otherwise may be defined by the recursive relation

$$\rho(u) = \rho(k) - \int_k^u \frac{\rho(t-1)}{t} dt,$$

where $k = \lfloor u \rfloor$. In particular we have $\rho(u) = 1 - \log u$ when $1 \le u \le 2$.

(ii) We have $\rho(u) \le (k!)^{-1}$, where $k = \lfloor u \rfloor$. A good approximation (we will not make this precise) is to take $\rho(u)$ "=" u^{-u} for large u.

We will verify statement (i) in the case $1 \le u \le 2$.

Lemma (Buchstab's formula)
For $x \ge 1$ and $0 < y \le z$ we have

$$\psi(x,y) = \psi(x,z) - \sum_{y < p \le z} \psi(x/p, p).$$

(In such sums p always runs over primes.)

Proof For any $w \le x$, each w-smooth integer $n > 1$ can be written uniquely as $n = mp$ where $p \le w$ and m is p-smooth, by taking p as the largest prime factor of n. Since 1 is w-smooth we have

$$\psi(x,w) = 1 + \sum_{p \le w} \psi(x/p, p).$$

The lemma follows on comparing the cases $w = y$ and $w = z$.

Proposition
We have

$$\psi(x, y) = (1 - \log u)x + O(x/\log x)$$

for $1 \leq u \leq 2$.

Proof We have $\psi(w, z) = \lfloor w \rfloor$ when $z \geq w$. Applying Buchstab with $z = x$ and $\sqrt{x} \leq y \leq x$ gives

$$\begin{aligned}
\psi(x, y) &= \psi(x, x) - \sum_{y < p \leq x} \psi(x/p, p) \\
&= \lfloor x \rfloor - \sum_{y < p \leq x} \lfloor x/p \rfloor \ \text{(since } p > x/p \text{ for such } p) \\
&= x\left(1 - \sum_{y < p \leq x} 1/p\right) + O\left(\sum_{p \leq x} 1\right) \\
&= x(1 - \log\log x + \log\log y) + O(x/\log x) \\
&= x(1 - \log u) + O(x/\log x).
\end{aligned}$$

In the penultimate step we used the facts that

$$\sum_{p \leq t} 1/p = \log\log t + C + O(1/\log t)$$

for a certain numerical constant C (analogous to the Euler constant) and that the number of primes up to x is $O(x/\log x)$.

3.10 Smooth square factoring

We are now ready to describe a simple factor base factoring method. It is very far from being the best such method.

Step 1 Pick a smoothness bound b and let $B = \{p \leq b : p \text{ prime}\}$.

Step 2 Pick elements $x \in \mathbb{Z}/n\mathbb{Z}$ at random, and see if the least residue of $x^2 (\text{mod } n)$ is b-smooth, using trial division. If it is, record its prime factorization. Repeat until more than $\#B$ such values have been found.

Step 3 With the equations

$$x_i^2 = \prod_{p \in B} p^{e_{p,i}} (\text{mod } n)$$

use Gaussian elimination over $\mathbb{F}_2$ to find a set of indices I such that

$$\sum_{i \in I} e_{p,i} = 0 (\mathrm{mod}\ 2)$$

for each $p \in B$, whence

$$\prod_{i \in I} \prod_{p \in B} p^{e_{p,i}}$$

is a square, y^2 say. Then $x^2 = y^2 (\mathrm{mod}\ n)$, where

$$x = \prod_{i \in I} x_i.$$

Check whether $\gcd(x - y, n)$ is a proper factor of n, and if not try again!

What size b should one use? We expect to need $O(\#B) = O(b/\log b)$ good values of x_i, and the number of attempts before getting a b-smooth value by chance will be $O(1/\rho(u)) = O(u^u)$, say. Thus Step 2 can be expected to take something like $O(bu^u)$ steps, where $u = \log n / \log b$. Taking $b = \exp(\sqrt{(\log n)(\log\log n)/2})$ minimizes this, very roughly. Then, after allowing for Step 3, which will need something like $O((\#B)^3)$ steps, we see that the overall running time might be about

$$\exp(c\sqrt{(\log n)(\log\log n)}) \quad (*)$$

for some constant c ($= 3/\sqrt{2}$ in this calculation).

It is important to note that this grows more slowly than any fixed power n^ε.

One obvious way to improve matters is to pick values of x for which x^2 has only a small residue modulo n, which will increase its chance of being b-smooth. All such variants end up with an expected running time of the shape (*), but with various values for the constant c.

3.11 Pollard's $p - 1$ method

We aim to factor an odd composite n. Let p be the least prime dividing n. Pollard's $p - 1$ method (1974) is likely to succeed if $p - 1$ happens to be b-smooth for some relatively small b.

Step 1 Pick a smoothness bound b, in the hope that $p - 1$ will be b-smooth.

Step 2 For each prime $q \leq b$ define $r(q)$ such that $q^{r(q)} \leq \sqrt{n} < q^{r(q)+1}$, and set

$$k = \prod_{q \leq b} q^{r(q)}.$$

Hence if $p-1$ is b-smooth we will have $p-1|k$.
Step 3 Pick an integer a and check that it is coprime to n. If it is not coprime we find a factor of n. Compute $a^k (\mathrm{mod}\ n)$ by the repeated squaring technique, and find $\gcd(a^k - 1, n)$. If $p-1$ is b-smooth then $p-1|k$, whence $a^k = 1 (\mathrm{mod}\ p)$, and hence $p \mid \gcd(a^k - 1, n)$.
Step 4 If $\gcd(a^k - 1, n)$ provides a non-trivial factor of n then stop. If the gcd is n then we can try $a^{k'}$ for various divisors k' of k. If the gcd is 1 we can try increasing b.

How long does the algorithm take? Computing $a^k (\mathrm{mod}\ n)$ requires $O(\log k)$ multiplications modulo n, with

$$\log k = \sum_{q \leq b} r(q) \log q \leq \pi(b) \log \sqrt{n} = O(b \log n).$$

This step dominates the running time, which is therefore $O(bn^{\varepsilon})$. And what is the chance of success? From Fact (ii) in subsection 3.9 this is about u^{-u} for large u. So if we tried $b = n^{1/4}$ then, in the worst case with p around $\sqrt{n}$, the probability is $1 - \log 2 = 0.31\ldots$. Thus we get an algorithm taking time $O(n^{1/4+\varepsilon})$, which succeeds in about three-tenths of cases.

3.12 The elliptic curve method

The problem with Pollard's $p-1$ method is that it can only work when $p-1$ is smooth. In Lenstra's Elliptic Curve Method (1987) we work in the group of points on a curve modulo p, and follow an analogous procedure. This time we are likely to succeed if the group order is smooth. But the difference is that now, if we are unsuccessful, we can try other curves until we hit on one whose order is smooth.

Step 1 Pick a smoothness bound b. The hope will be that we can find an elliptic curve such that the number of points over $\mathbb{Z}/p\mathbb{Z}$ (for some prime factor p of n) is b-smooth.
Step 2 Let $k = \prod_{q \leq b,} q^{r(q)}$, where q runs over primes and

$$q^{r(q)} \leq n^{1/2} + 2n^{1/4} + 1 < q^{r(q)+1}.$$

If $p \leq \sqrt{n}$ and N_p is the number of points on an elliptic curve mod p, then Hasse's bound implies that $N_p \leq n^{1/2} + 2n^{1/4} + 1$. So we will have $N_p \mid k$ if N_p happens to be b-smooth.
Step 3 Pick an integer pair (x_0, y_0) modulo n at random, and take E as the curve $y^2 = x^3 + x + y_0^2 - x_0^3 - x_0$. Then $P = (x_0, y_0)$ lies on E. Compute kP by the repeated squaring method, working modulo n. If $kP = \mathbf{0}$ modulo p for some $p \mid n$, then this process will produce a denominator which cannot be inverted modulo p, since the point at infinity is, in effect, the point $(0/0, 1/0)$. Thus, when we remove denominators in the calculation modulo n, by finding their inverses via the Euclidean algorithm, we will encounter a gcd which is greater than 1. This will produce a divisor of n.

How to choose b
Given b, the time taken to test each curve is $O(b(\log n)^c)$ for some constant c, just as for the $p-1$ method. The probability of success is about u^{-u}, where $u = (\log p)/(\log b)$. This assumes that all numbers of points are equally likely. (But they are not.) Heuristically the running time is then expected to be $O(u^u b(\log n)^c)$ steps. This is minimized by taking b to be about $\exp(\sqrt{(\log p)(\log\log p)/2})$, giving a running time

$$O(\exp(\sqrt{(1+\varepsilon)(\log n)(\log\log n)}).$$

Of course, we cannot choose b to depend on p, since p is unknown. Thus in practice we might assume p to be of size $\sqrt{n}$. Alternatively we can try an increasing succession of values of b, so that eventually we will reach a value close to $\exp(\sqrt{(\log p)(\log\log p)/2})$.

Variants of this method were used successfully to factor the Fermat numbers F_{10} and F_{11}.

The Elliptic Curve Method is one of most widely used techniques for "everyday" factoring. It is ultimately slower than the Number Field Sieve, for example. However it has the attractive feature that the time taken is less when n has a smaller prime factor, just as with the Pollard ρ-method. This makes it particularly good for "naturally occurring" factorization problems, such as the Fermat numbers mentioned above.

3.13 The quadratic sieve

A simple way to look for x^2 with a smooth remainder modulo n is to take $x_0 = \lceil \sqrt{n} \rceil$ and try successive values $x = x_0, x_0 + 1, x_0 + 2, \ldots$ If $x = x_0 + t$ with t not too large then the remainder is $x^2 - n$, which will be of order $t\sqrt{n}$. This is more like $\sqrt{n}$ in size than n, and so will be more likely to be smooth than a random integer in $(0, n]$. If q is a prime divisor of $x^2 - n$ then n must be a quadratic residue of q, so that only half the primes need to be included in the factor base.

Rather than test $x^2 - n$ for smoothness by trial division the quadratic sieve uses a different technique to "sieve out" non-smooth values. In order to do this the factor base is extended to include powers of primes q^e up to b, and not just primes themselves. Then, to search values $L < x \leq U$ looking for cases in which $x^2 - n$ is b-smooth, we proceed as follows. We produce an array indexed by the integers x in our interval. Then, for each prime power q^e in our factor base we compute the solutions (usually two) of $x^2 = n (\mathrm{mod}\ q^e)$, and call them a and b. We then go through the array in steps of length q^e adding (an approximation to) $\log q$ for every $x = a (\mathrm{mod}\ q^e)$. We repeat this for integers $x = b (\mathrm{mod}\ q^e)$, and then move on to the next prime power. This is the **sieving** process.

Once we are done, we check though the contents of the array, looking for places where the value is (approximately) $\log(x^2 - n)$, in which case $x^2 - n$ must be composed of prime factors $q \leq b$. We can then decompose these smooth values into their prime factorization by trial division, but this step need only be done for the few numbers we actually want, rather than for every single value $x^2 - n$.

A crude estimate for the time requirement of the sieve process uses the fact that there are at most $2(1 + (U - L)/q^e)$ values of x to be visited for each prime power q^e. Thus the time taken is

$$O\left(\sum_{q^e \leq b} (1 + (U - L)/q^e)\right) = O(b) + O((U - L)\log b).$$

For comparison, trial division for takes time $O(b)$ for each value of x, yielding a far worse total $O((U - L)b)$.

3.14 The number field sieve

The number field sieve was introduced by Pollard in 1988. It is the fastest algorithm currently available. However its complexity means that it is a substantial task to implement it. A number n has to be very large before the number field sieve is faster than other methods. We will describe it in its simplest form.

Choose two monic irreducible polynomials $f_1, f_2 \in \mathbb{Z}[x]$ with smallish coefficients, and a common root, m say, modulo n. We want the degrees to be small, but not too small. One easy way to achieve this is to begin by picking a degree d, asymptotically of size $(\log n)^{1/3}(\log\log n)^{-1/3}$ (but this might be 5 or 6 in practice), and to take m about $n^{1/d}$ or just under. Now write n in base m as

$$n = m^d + a_{d-1}m^{d-1} + \ldots + a_0,$$

so that $0 \leq a_i < m$. We then take $f_1(x) = x^d + a_{d-1}x^{d-1} + \ldots + a_0$ and $f_2(x) = x - m$. A random polynomial is almost certainly irreducible, but if we are unlucky with f_1 we can try another m.

Consider the number field $\mathbb{Q}(\theta)$ where θ is a root of f_1. Then $\mathbb{Z}[\theta]$ is a subring of the ring of integers for $\mathbb{Q}(\theta)$. Unfortunately:-

(i) $\mathbb{Z}[\theta]$ may not be the full ring of integers;

(ii) Even if it is, it may not be a unique factorization domain; and

(iii) The unit group is infinite, if $d \geq 3$.

We will ignore these problems in this brief description!

We now have two ring homomorphisms:

$$\phi_1 : \mathbb{Z}[\theta] \to \mathbb{Z}/n\mathbb{Z} \quad \text{given by} \quad \phi_1 : \theta \mapsto \overline{m} = m + n\mathbb{Z},$$

and

$$\phi_2 : \mathbb{Z} \to \mathbb{Z}/n\mathbb{Z} \quad \text{given by} \quad \phi_2 : k \mapsto \overline{k} = k + n\mathbb{Z}.$$

We aim to find a set I of pairs (a, b) of coprime integers, such that $\prod_{(a,b)\in I}(a - \theta b)$ is a square α^2 in the ring $\mathbb{Z}[\theta]$, and also $\prod_{(a,b)\in I}(a - bm)$ is a square k^2 in $\mathbb{Z}$. If we can do this then

$$\phi_1(\beta)^2 = \prod_{(a,b)\in I} \phi_1(a - b\theta) = \prod_{(a,b)\in I} (\overline{a} - \overline{b}\overline{m}) = \prod_{(a,b)\in I} \phi_2(a - bm) = \phi_2(k)^2.$$

This produces two integer squares whose difference is divisible by n, and we can hope that this will enable us to split n.

To find a suitable set I we choose factor bases for $\mathbb{Z}[\theta]$ and $\mathbb{Z}$. For the former, the factor base will consist of a basis for the units, together with a set of prime ideals with norm below some smoothness bound. We then try lots of pairs (a, b), looking for cases where both $a - b\theta$ and $a - bm$ are smooth in their respective senses, and hence can be decomposed into "primes" from the respective factor bases. One can sieve, as with the quadratic sieve, to speed this process.
Having gathered enough relations, in which both $a - b\theta$ and $a - bm$ are smooth, we use linear algebra over $\mathbb{F}_2$ to find a subset of the pairs such that the products $\prod_{(a,b)\in I}(a - \theta b)$ and $\prod_{(a,b)\in I}(a - mb)$ are both squares.

The whole process is rather like the smooth square factoring algorithm of Section 3.10, or the quadratic sieve, except that it is done over a number field. The technical problems are considerable, but in the calculation of the expected running time everything depends on the size of m, rather than n (if d is chosen around $(\log n)^{1/3}(\log\log n)^{-1/3}$). The outcome is that one has an expected running time of

$$O(\exp(c(\log n)^{1/3}(\log\log n)^{2/3}))$$

which is asymptotically better than any other current method.

We conclude with a simple example, courtesy of Richard Pinch.
$n = 84101 = 290^2 + 1$.

Take $d = 2$, $m = 290$, $f_1(x) = x^2 + 1$, $f_2(x) = x - 290$.

Then our number field is $\mathbb{Q}(i)$, and the ring we work in is $\mathbb{Z}[i]$ which fortunately is the full ring of integers, is a Unique Factorization Domain, and has only the units ± 1 and $\pm i$.

Taking $a = -38$, $b = -1$ we have $a - ib = -38 + i = -(2 + i)(4 - i)^2$, and $a - mb = 2^2.3^2.7$
Taking $a = -22$, $b = -19$, we have $a - ib = -22 + 19i = -(2 + i)(3 - 2i)^2$ and $a - mb = 2^4.7^3$

It follows that

$$(-38 + i)(-22 + 19i) = \big((2 + i)(4 - i)(3 - 2i)\big)^2 = (31 - 12i)^2$$

and

$$(-38+m)(-22+19m) = 2^6.3^2.7^4 = 1176^2.$$

Then $\phi_1(31-12i) = \overline{31-22m} = -\overline{3449}$ and $\phi_2(1176) = \overline{1176}$, whence $3449^2 = 1176^2 (\text{mod } 84101)$.

We therefore calculate $\gcd(3449-1176, 84101)$, giving us a value 2273, and we find that $84101 = 2273 \times 37$.