Advanced Computational Number Theory

N1MA9W11

Karim Belabas

Sep 1^{st} 2013

Contents

List of Algorithms

Foreword

This text is a set of notes for the graduate course on *Advanced computational number theory*, which I gave in Bordeaux in 2005–2007, 2010 and 2013. This is work in progress, the last lecture is still missing, and the ones included may contain mistakes. So use at your own risk ! Please send any corrections to `Karim.Belabas@math.u-bordeaux1.fr`

The course uses classical and modern factorization algorithms to present important ideas and techniques in computational number theory. We will cover the reduction of $\mathbb{Z}$-modules and lattices (the LLL algorithm), factorization of univariate polynomials over finite fields, the rationals and the complex numbers, then primality testing (up to the Elliptic Curve Primality Proving algorithm) and integer factorization (up to the Number Field Sieve), as well as some basic algebraic number theory (maximal orders, class groups and units of number fields).

The emphasis is on important ideas throughout, and asymptotically fast methods, certainly not programming efficiency. Many tricks must be implemented before the algorithms that we will study become really practical. For instance, many variants will compete to achieve a given result, each with a certain range of input sizes on which it will be optimal, in a given environment. Hence, a good implementation should provide all of them, as well as finely tuned thresholds to decide which method to use. We shall ignore such concerns and blissfully lose constant or logarithmic factors when technical details would obscure our main point.

Very good references covering about the same material are Gerhard & von zur Gathen [24] for chapters 1 – 3, Cohen [6] for chapters 2, 4 and 5, and Crandall & Pomerance [9] for Chapter 4.

Happy reading!

Chapter 1

Introduction

1.1 Basic definitions

1.1.1 Algorithms

We call *algorithm* a computer program, meant to solve all instances of a problem: on a given *input*, it must produce a (correct) *output* in a finite number of steps. Precise definitions would require us to define a computational model, to specify what a *problem* or a *computer program* is. Check out Papadimitriou [18] for a good exposition of the necessary formalism.

We classify algorithms in two categories: the *deterministic* ones, not allowed the use of a random generator; the *randomized* (or probabilistic) ones, where we do allow them. The distinction is important because we have no true source of randomness for actual computer programs, we may only simulate it. Also for a given input, a randomized algorithm may perform differently each time it is run, possibly very badly. This is the price to pay for better practical characteristics *on average.*

As stated above, an algorithm must return a correct answer. For randomized computations that may return a wrong answer, hopefully with small probability, we will use the word *method* instead of algorithm. Obviously a deterministic computation returning a wrong answer will be of little use, and does not deserve a specific name.

1.1.2 Complexity (generalities)

Running the program on a fixed input expends some resources: time, space (or memory), random bits, processors... A *complexity estimate* is a function $f : \mathbb{N} \to \mathbb{R}^+$ such that for all instances of *input size* $\leqslant s$ of the given problem, we can guarantee that less than $f(s)$ units of resources are used during the computation. Unless mentioned otherwise the resource we are interested in is the running time, and algorithms are deterministic. The input size must be defined suitably, and meaningfully, for each particular problem. We will see examples shortly.

We use the following notations from analysis, whose meaning is slightly non-standard, hence made precise below:

- we write $f(s) = O\big(g(s)\big)$ if there exist absolute constants s_0 and $C > 0$ such that $|f(s)| \leqslant Cg(s)$ for all $s > s_0$; C is called an *implied* constant. In the algorithms we will study, the constants s_0 and C will be *effective*, meaning that we can in principle give an actual numerical value for them: this helps setting algorithmic thresholds.

- if T is a list of parameters for the problem, not containing s, we write $f = O_T(g)$ if the constants C and s_0 above are allowed to depend on the values of the parameters in T. For instance the notation $f(s) = O_\varepsilon(s^{1+\varepsilon})$ for all $\varepsilon > 0$ means that for every fixed $\varepsilon > 0$, there exist constants $s_0(\varepsilon)$ and $C(\varepsilon)$ such that $f(s) \leqslant C(\varepsilon)s^{1+\varepsilon}$ for all $s > s_0(\varepsilon)$.

- We use interchangeably

  - $f \ll g$ and $f = O(g)$,
  - $f \ll_T g$ and $f = O_T(g)$,

  usually for typographical reasons.

- In this course, we are not interested in precise asymptotic orders and use the "Soft-Oh" notation, $\widetilde{O}(f) := O(f) \times (\log f)^{O(1)}$, in order to hide logarithmic factors. Note that a function that is $O_\varepsilon(s^{1+\varepsilon})$ for all $\varepsilon > 0$ may or may not be $\widetilde{O}(s)$, for instance $f(s) = s\exp(\sqrt{\log s})$ is not.

The following vocabulary is standard, where s is the problem size and $f(s)$ a running time estimate:

- if $f(s) = O(s)$, the cost is *linear*. (The algorithm is *linear-time*.)

- if $f(s) = O(s^d)$ for some d, the cost is *polynomial*. (The algorithm is *polynomial-time*.)

- if $f(s) = O(\exp(s^d))$, the cost is *subexponential* if $d < 1$ and *exponential* otherwise.

Our general understanding is that a polynomial algorithm is good (scales nicely as the problem size increases), subexponential is not that bad, and exponential just will not work. Admittedly this is because we shall only be tackling relatively easy problems, for which it makes no doubt that there exists an algorithm to solve them. So we can afford to be picky.

We will hardly touch the more difficult subject of *lower bounds* and complexity classes: it is easier to produce and study an explicit algorithm than to prove that *any* algorithm solving the problem must expend at least that many resources. However, note that an algorithm that runs in linear time is heuristically best possible: it simply means that we perform an operation on each bit of the input. If we can do better, then our encoding of the input is redundant and we are studying the wrong problem.

1.1.3 Algebraic and bit complexity

We fix a base ring R throughout this section. Our algorithms are presented as functions accepting certain *inputs* (satisfying some preconditions), and returning certain *ouputs*. Functions consist of numbered steps, executed sequentially; we allow

- standard flow control instructions : *for* or *while* loops, *if/then/else* tests, a *return* statement (that aborts the function and returns the specified output).

- assignments to a countable set of named registers — which could be denoted $(r_i)_{i\in\mathbb{N}}$ but we will use more descriptive names : $x \leftarrow 1$ (the previous value of x is lost, unless it has first been copied to another register). We allow assignments of arbitrary elements of R^n to these registers for arbitrary $n \geqslant 1$; the *size* of such a register is n by definition. For notational convenience, we abuse this notation by allowing multidimensional arrays (in particular matrices) and multivariate polynomials over R, with the understanding that we could emulate this in the previous notation by using a cumbersome specific coding system. This is not innocent : we have implicitly chosen a *dense* representation for compound objects, where all entries must be explicitly enumerated, instead of a *sparse* representation, where some clever encoding is used to save space (for instance, store only the non-zero entries).

- arithmetic operations in R $(+, -, \times)$.

Definition 1.1 (Algebraic complexity, over R)**.** The space complexity of a given run of our algorithm is the maximum of the total size of all registers used as the steps are executed. The time complexity is the total number of arithmetic operations performed in R.

Exercise 1.2. Given $A, B \in R[X]$ of size $\leqslant s$ (degree $< s$), write an algorithm that computes

1. their sum $C := A + B$ in time $O(s)$.

2. their product $C := A \times B$ in time $O(s^2)$.

In both cases, what is the space complexity ?

This simple model depends on the fixed base ring R, that needs to be specified. It is usually easy to estimate algebraic complexities. The major drawback is that, unless the ring R is finite, time and space complexity have no obvious relation to actual running times and memory use in a real computer. If R is $\mathbb{Z}$ for instance, we want to take into account the fact that integers with many digits are more costly to manipulate.

Assume that all positive integers are represented by a string of binary digits (or bits). We define the size $s(N)$ of the integer $N > 0$ as the number of bits needed to encode N. Writing

$$N = \sum_{i=0}^{s-1} n_i 2^i, \quad n_i \in \{0, 1\},$$

we obtain $s(N) = \lfloor \log_2(N) \rfloor + 1 = \lceil \log_2(N+1) \rceil \sim \log_2 N$ as $N \to \infty$. We may represent both positive and negative numbers by specifying that the leading bit (0 or 1) encodes the sign (± 1) of the integer. This adds one bit to the representation and we still have $s(N) \sim \log_2 N$. This leads to the model of *bit complexity*:

Definition 1.3 (Bit complexity)**.** We allow the same operations as in the model of algebraic complexity for $R = \mathbb{Z}$, and make the following changes.

- The size of an element $n \in \mathbb{Z}$ is $\log_2(|n| + 1) \sim \log_2 |n|$ as $n \to \infty$. Note that $|n| = 2^s - 1$. As before, the space complexity is the total size of all registers used, but big integers use up more space.

- Let a, b be two integers of size $\leqslant s$:
  - the time complexity of $a + b$ is $O(s)$.
  - the time complexity of $a \times b$ is denoted $\mathsf{M}(s)$.

In properly defined general computational models (Turing or RAM machines for instance), one can *prove* that $\mathsf{M}(s) = O(s^2)$ (schoolbook arithmetic) or $\mathsf{M}(s) = \widetilde{O}(s)$ (using the Schönhage-Strassen algorithm and DFT-based multiplication). The formal difficulty is that we must reduce everything to bits and bit operations; in particular operations on loop indices are no different from operations in $R = \mathbb{Z}$, sizes of integers are themselves integers, etc. Our "axiomatic" approach avoids these difficulties, but one must be aware that our axioms can become theorems provided we start at a more fundamental level.

Unless mentioned otherwise, we will use this latter model (Bit complexity) to estimate complexities.

1.2 Examples

Example 1.4. Problem: given $N \in \mathbb{Z}_{>0}$, as a string of bits, factor N as a product of primes. The naive approach to solve the problem is to check successively for $d = 2, 3, \dots, \left\lfloor \sqrt{N} \right\rfloor$, whether $d \mid N$. The bit complexity model is appropriate and $s(N) \sim \log_2 N$. The algorithm requires $\sqrt{N}$ divisions in the worst case (when N is in fact prime) and even if each of these took unit time, $\sqrt{N} = \sqrt{2^s - 1} \sim 2^{s/2}$ is exponential in s.

- The current best *proven* algorithm for integer factorization runs in randomized time $\exp \widetilde{O}(s^{1/2})$.

- The current best algorithm in practice (the number field sieve, NFS) is *conjectured* to run in randomized time $\exp \widetilde{O}(s^{1/3})$. It "routinely" factors general integers having 200 decimal digits.

Both algorithms require subexponential space, of the same order of magnitude as their run-time : $\exp \widetilde{O}(s^{1/2})$ and $\exp \widetilde{O}(s^{1/3})$, respectively.

Example 1.5. A special case of the above: given N, check whether it is a prime number. This can be done in polynomial time $\widetilde{O}(s^{10.5})$ (a landmark result by Agrawal, Kayal and Saxena [1]), improved to $O_\varepsilon(s^{6+\varepsilon})$ by Lenstra and Pomerance. Curriously enough, the implied constant in Agrawal, Kayal and Saxena's $O(s^{10.5})$ is effective, whereas in Lenstra and Pomerance's stronger result, it is not — at least not unless ε is large enough.

The current best algorithm in practice (FastECPP) is not known to be polynomial time; it is conjectured to run in randomized time $\widetilde{O}(s^4)$. It "routinely" proves the primality of general integers having 25000 decimal digits. When the algorithm stops, it in fact produces a *primality certificate*, which is a short proof for the primality of N, which can be checked (easily and independently) in time $\widetilde{O}(s^3)$.

If the Generalized Riemann Hypothesis (GRH) is true, we have much simpler deterministic algorithms that run in time $\widetilde{O}(s^4)$, based for instance on Rabin–Miller's compositeness test. This yields a *conditional* algorithm (correct only if we assume some unproven hypothesis), contrary to the previous examples (ECPP, NFS) where the output is correct and we only conjecture something about the running time. To my knowledge, none of these GRH variants produce useful certificates.

Example 1.6. Given $T \in \mathbb{F}_2[X]$, as a vector of elements in $\mathbb{F}_2$, factor T into irreducible factors. In this case, using either the bit complexity or the algebraic complexity model with $R = \mathbb{F}_2$ yields the same formal results; writing

$$T = \sum_{i=0}^{s-1} \varepsilon_i X^i, \quad \varepsilon_i \in \mathbb{F}_2,$$

we have $s(T) = \deg T + 1$. This is an analog of our first factorization problem over $\mathbb{Z}$, in fact much simpler. The running time of the best algorithm so far is $\widetilde{O}(s^2)$, essentially the same as a naive multiplication!

Example 1.7. Given $T \in \mathbb{F}_q[X]$, the problem is again to factor T; $\mathbb{F}_q$ is given as $\mathbb{F}_p[Y]/(D)$, with $\deg D = \log_p q$. The bit complexity model yields $s(T) = (\deg T + 1)\log_2 q$. Justification for the size chosen: an element A in $\mathbb{F}_p[Y]/(D)$ is encoded as a polynomial of degree ($< \deg D$), whose coefficients are integers in $[0, p-1]$: the total size is $\deg D \log_2 p = \log_2 q$.

The fastest known algorithm for this still runs in time $\widetilde{O}(s^2)$, but it is randomized: no deterministic polynomial time algorithm is known at present. More precisely, the best such algorithms have running time polynomial in n, but exponential in $\log q$. Hence, small finite fields are easy, even if we insist on deterministic methods. Since this cannot be expressed in terms of the single parameter $s(T)$ above, we shall keep as many parameters as needed in complexity estimates (here n and $\log q$), in order to describe such interesting features.

In fact, this is still not precise enough. Fields of small *characteristic* are easy, so we should keep three parameters: the degree n, the characteristic p, and the extension degree e such that $q = p^e$. Then deterministic algorithms exist which are polynomial in e and n, but they are exponential in $\log p$.

Example 1.8. Given an odd prime p, we want to find a quadratic non-residue in $\mathbb{F}_p$, i.e. an $a \in \mathbb{F}_p$ such that $\chi(a) = -1$, where χ is the Legendre character. As size for an instance, we choose the size of p: $s = \lfloor \log_2(p) \rfloor + 1$.

To solve the problem, we can for instance pick a random $a \in \mathbb{F}_p^*$ or try for $a = 1, 2, 3, 4 \ldots$ (Granted, we can avoid 1 and 4.) The probabilistic approach is easy to analyze: since there are as many squares as non-squares in $\mathbb{F}_p^*$, namely $(p-1)/2$, an element a chosen uniformly at random is a solution with probability $1/2$. Note that the uncertainty only concerns the running time: the result is guaranteed by the computation of the Legendre symbol.

The second, deterministic, approach is much harder to analyze. This can be done by comparing the Dirichlet L-function $L(\chi, s)$ and Riemann's ζ function, but we must now use a deep conjecture in number theory: the Generalized Riemann Hypothesis[1], or GRH for short, which states, for a given L-function, that $\mathrm{Re}(s) > 1/2 \Rightarrow L(s) \neq 0$.

Theorem 1.9 (Bach [3]). *Assume that GRH holds for ζ and $L(\chi, \cdot)$. Then there exists a quadratic non-residue in $\mathbb{F}_p$ having a representative less than $2(\log p)^2$.*

[1] The problem is easy to understand: if $\chi(1) = \chi(2) = \cdots = \chi(x) = 1$, then $S(x) = \sum_{n \leqslant x} \chi(n) = \sum_{n \leqslant x} 1$. An elementary estimate such as $|S(x)| \leqslant \sqrt{p}\log p$ (Polyá-Vinogradov) proves that $x \ll \sqrt{p}\log p$. Proving the expected estimate $S(x) = o(x)$ for "short character sums", say for $x = p^{1/10}$, is a notoriously hard problem in analytic number theory, which becomes trivial under GRH.

This time, contrary to the conditional primality test algorithm in Example 1.5, the result produced by the algorithm is correct whether GRH is true or false. The only conditional part concerns the running time analysis. But the underlying idea is the same, we shall explain this later in more detail (Theorem 4.19).

1.3 Randomized algorithms

In the above, we ignored a very natural question: how do we measure the cost of a randomized algorithm? In this case we talk about *expected costs*: for a *fixed* input i, we average the cost over all possible runs of the program. More precisely, assume that the (finite or infinite) set of possible runs S_i is equipped with a probability measure p, and the cost of a given run $a \in S_i$ is $f_i(a)$, then the expected cost for this input is

$$\mathbb{E}(f_i) = \sum_{a \in S_i} p(a) f_i(a).$$

The expected cost for a size s is the max of these expectations over all i of size less than s. (Often, $\mathbb{E}(f_i)$ does not depend on i, only on its size.) In simple cases, S_i is finite and the possible runs are equidistributed, $p(a) = 1/\#S_i$ for all $a \in S_i$.

Note that, for us, a randomized algorithm must return a correct result, the uncertainty only affects the running time: computer science slang calls this a *Las Vegas* method. There is a weaker notion, which we will not need but is useful in fast linear algebra for instance, where the running time is precisely bounded but, averaging over possible runs of the program, we have a probability less than $p < 1$ of giving a wrong result (*Monte Carlo* method). We can assume that p is as small as we please by running the program multiple times: running the program k times, the probability of error is now less than p^k, assuming that the successive runs are *independent*. As before, this is not strictly true since, on a physical machine, we do not know how to run truly independent instances of a program, but we will ignore this problem in our model.

Even weaker, consider Fermat's primality test, i.e. given an integer $n > 2$ we compute $d \equiv 2^{n-1} \pmod{n}$; if $d \not\equiv 1 \pmod{n}$ then n is not a prime number and we answer NO; otherwise, we know nothing for sure but n may still be prime, so we answer MAYBE and try again. We can try and devise a randomized algorithm by checking that $a^{n-1} \equiv 1 \pmod{n}$ for some $1 \leqslant a \leqslant n$ chosen uniformly at random. But we run into trouble for some inputs: a Carmichael number n is a non-prime integer satisfying $a^n \equiv a \pmod{n}$ for all $a \in \mathbb{Z}$. There are infinitely many such integers (a difficult result of Alford, Granville and Pomerance [2]), $561 = 3 \times 11 \times 17$ being the smallest one. So if n is a Carmichael number, we get the answer MAYBE for all a coprime to n; the probability of getting this answer is

$$\frac{\phi(n)}{n} = \prod_{p|n}\left(1 - \frac{1}{p}\right),$$

which does not look bounded away from 1, for instance if the number of distinct prime divisors of n remains bounded. (I do not know whether infinitely many such Carmichael numbers exist.) This does not look like a good algorithm! Note the importance of the sample space chosen: the algorithm is bad because for *some fixed inputs* (Carmichael numbers, an infinite set) it behaves badly, independently of our random choices. A good algorithm must behave nicely on a sizeable portion of all possible random choices *for all inputs*. We could have defined a weaker notion of randomized algorithm which behaves badly on a small proportion of our inputs, i.e. for most input distributions we would be fine.

1.4 Some principles

In this section we list some basic principles of efficient algorithmic design, with special emphasis on techniques relevant to computational number theory.

1.4.1 Arithmetic is hard, Linear Algebra is easy.

Because of this, we translate as much arithmetic as possible into linear algebra. Let us consider a prototypical example. Given $a, b \in \mathbb{Z}_{>0}$ we have to compute $\gcd(a, b)$. The arithmetic approach is to use the equality

$$\gcd(a,b) = \prod p^{\min(v_p(a), v_p(b))}, \quad \text{where} \quad a = \prod p^{v_p(a)}, \quad b = \prod p^{v_p(b)}.$$

This requires factoring a and b, a hard problem over $\mathbb{Z}$.

Now consider the Euclidean Algorithm. If $a = bq + r$, where $q, r \in \mathbb{Z}$, then $\gcd(a, b) = \gcd(b, r)$, and we may as well use Euclidean division to guarantee that $0 \leqslant r < b$ is small. This is obviously a linear operation: write it as

$$\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix} = \begin{pmatrix} b & r \end{pmatrix}$$

Iterating, the pair (a, b) is eventually replaced by $(\gcd(a, b), 0)$ and we have:

$$\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -q_k \end{pmatrix} = \begin{pmatrix} \gcd(a,b) & 0 \end{pmatrix}.$$

This approach only uses Euclidean divisions, and conceptually is a sequence of matrix multiplication. Both are fast and easy operations, a much better method! Note that multiplying out the 2×2 matrices as they are computed, we obtain a Bezout relation.

$$\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} u & u' \\ v & v' \end{pmatrix} = \begin{pmatrix} \gcd(a,b) & 0 \end{pmatrix}.$$

Exercise 1.10. Use the above to solve in $x, y \in \mathbb{Z}$ an equation of the form $ax + by = c$, where $a, b, c \in \mathbb{Z}$. Generalize the idea and explain how to solve in integers a linear system with integer coefficients. More generally, solve in $(x_1, \dots, x_n) \in \mathbb{Z}^n$ the linear system of congruences

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \equiv \begin{pmatrix} b_1 & \pmod{M_1} \\ & \vdots \\ b_m & \pmod{M_m} \end{pmatrix}$$

1.4.2 Be Lazy

Avoid all work that is not absolutely necessary, and defer any costly computation until it is impossible to avoid it. Besides the general techniques in software engineering like optimizing the inner loops or precomputing as much as possible (with appropriate data types for fast retrieval), some techniques are typical in computer algebra:

- symbolic computation: work with formal symbols like π, $2^{10^{10}}$, etc. instead of working with their (exact or approximate) decimal expansions,

- sparse representations: to compute $(X^n+1)^2 = X^{2n}+2X^n+1$ we need much fewer than $O(n^2)$ operations. Structured matrices are another example.

- approximate computation: instead of computing x in $\mathbb{Z}$ compute $\hat{x}$ which is sufficiently close to x that we can reconstruct x from $\hat{x}$. For instance, we may take $\hat{x} \in \mathbb{Z}/N\mathbb{Z}$ or a decimal approximation. The point is to avoid *intermediate expressions swell*: if the final result is small, we can reconstruct it. Of course, this does not work if the result is huge! (In fact, it may still save time: we often can compute and recombine local information faster than we would compute directly a global result.)

The last point is deeper and most important. Let us see how this works on a few concrete examples.

Example: the determinant

Let $A \in M_n(\mathbb{Z})$, we want to compute the determinant of A. We have the trivial bound

$$|\det A| \leqslant n! \, \|A\|_\infty^n := B(A),$$

so the size of the integer $\det A$ is bounded by $n \log n + n \log \|A\|_\infty$. The obvious approach to solve the problem is Gauss pivoting which is expensive: it introduces rational coefficients, whose size increases after each matrix operation. Possible solutions are to compute $\det A$ as

1. a floating point number (round final result to $\mathbb{Z}$): no explosion but stability problems. In order to be able to round the result to $\mathbb{Z}$, one must compute with at least $n \log n + n \log \|A\|_\infty$ digits of accuracy. But this will likely not be enough due to round-off errors in intermediate computations, inherent to the floating point model.

2. an element in $\mathbb{Z}/p\mathbb{Z}$. If $p > 2B(A)$ is prime, compute $\det \bar{A}$ in $M_n(\mathbb{F}_p)$ which is equal to $\det A \pmod{p}$. Now there is a single integer x in that class with $|x| \leqslant B(A)$. Problem: how to find a large prime p ? The prime number theorem tells us a random integer in $[2^k, 2^{k+1}[$ is prime with probability $1/(k \log 2)$, and we can quickly test for compositeness (see §4.1.3). (By the way, why do we pick p prime instead of some random large integer ?)

3. an element in $\mathbb{Z}/p_1\mathbb{Z} \times \cdots \times \mathbb{Z}/p_k\mathbb{Z}$, for some distinct primes whose product is $> 2B(A)$. Compute the determinant of A modulo p_k for each k, say $\det(A) \equiv a_k \pmod{p_k}$ and then use the *Chinese Remainder Theorem* to solve the congruences $x \equiv a_k \pmod{p_k}$, $k = 1, 2, \ldots, n$. This turns out to be the most efficient approach.

Exercise 1.11. Make the last two algorithms precise and evaluate their costs.

The modular approaches in the above exercise are called *homomorphic imaging schemes*. In order to avoid expensive computations, even when the result is large, first map to cheaper rings, compute there, then come back.

Example: gcd in $\mathbb{Z}[X]$

Let A, B be two monic polynomials in $\mathbb{Z}[X]$ each of degree 100, program the Euclidean algorithm over $\mathbb{Q}[X]$ with A, B as inputs. In general they are coprime, but polynomials appearing in the intermediate steps will be horrible. Bounding the bit complexity of this naive algorithm is non-trivial.

As before let us reduce the problem to the case when the polynomials are in $\mathbb{F}_p[X]$ by computing modulo a suitable prime p: we can compute the gcd of $\bar{A}$ and $\bar{B}$ in $\mathbb{F}_p[X]$ and then reconstruct the gcd of the given two polynomials in $\mathbb{Z}[X]$.

To explain what is a "suitable prime", we recall some properties of the resultant, which is not as nice as a gcd, but more generally available (see [14, §V.10] for details).

Definition 1.12. Let R be a commutative ring with unity, P, Q two polynomials in $R[X]$ of degree m and n respectively. The *resultant* of P and Q, denoted by

$\mathrm{Res}(P,Q)$, is the determinant of the $(n+m)\times(n+m)$ Sylvester matrix,

$$\begin{pmatrix} p_n & \dots & 0 & q_m & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_0 & & p_n & q_0 & & q_m \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & p_0 & 0 & \dots & q_0 \end{pmatrix},$$

where $P(X)=\sum_{i=0}^{n} p_i X^i$, $Q(X)=\sum_{i=0}^{m} q_i X^i$.

Lemma 1.13. *There exists $U, V \in R[X]$ such that $PU+QV=\mathrm{Res}(P,Q)$. We may choose $\deg U < \deg Q$, $\deg V < \deg P$.*

Proof. Exercise. Try to understand the matrix above as representing an R-linear map from $R[X]_{\deg<m}\times R[X]_{\deg<n}$ to $R[X]_{\deg<n+m}$. □

Lemma 1.14. *If R is an integral domain, $\mathrm{Res}(A,B)=0$ if and only if A and B have a common root in the algebraic closure of the field of fractions $K=\mathrm{Frac}(R)$. In other words,*

$$\mathrm{Res}(A,B)=0 \Longleftrightarrow \gcd_{K[X]}(A,B)\neq 1.$$

Corollary 1.15. *Let $\delta=\gcd_{\mathbb{Z}[X]}(A,B)$, p a prime number not dividing the leading coefficients of A and B, and let $\bar{A},\bar{B}$ denote the canonical projections from $\mathbb{Z}[X]$ to $\mathbb{F}_p[X]$. Then*

$$\gcd(\bar{A},\bar{B})=\overline{\gcd(A,B)}$$

if and only if p does not divide $\mathrm{Res}(A/\delta,B/\delta)$.

The asumption on the leading coefficients ensures that $\deg(A)=\deg(\bar{A})$ and $\deg(B)=\deg(\bar{B})$ so that the Sylvester matrices over $\mathbb{Z}$ and $\mathbb{F}_p$ are formed in a similar way. (It can be weakened : the result remains true if p does not divide $\gcd(\mathrm{lc}(A),\mathrm{lc}(B))$.) Any such p not dividing the integer $\mathrm{Res}(A/\delta,B/\delta)$, which is non-zero (why?), is suitable.

Exercise 1.16. 1. Bound the smallest such p in terms of n, m, $\|A\|_\infty$, $\|B\|_\infty$.

2. Assume that $\|\delta\|_\infty \leqslant C$, $p>2C$ is prime and that we have computed $\bar{\delta}=\gcd(\bar{A},\bar{B})$. Prove that reconstructing δ is easy if p is suitable.

3. To test the latter condition *without* computing the resultant (which is not possible since we do not know δ yet), prove the following first: if the prime p does not divide the leading coefficient of A and B, and $P\in\mathbb{Z}[X]$ divides A and B is such that $\bar{P}=\gcd(\bar{A},\bar{B})$ in $\mathbb{F}_p[X]$, then $P=\gcd(A,B)$ in $\mathbb{Z}[X]$.

4. Work out a similar approach using several small primes instead of a single big one.

Working out the solution to the exercise, then incorporating fast arithmetic, we eventually obtain

Theorem 1.17. *Let $f, g \in \mathbb{Z}[X]$ of degree less than n and sup-norm less than A. Their gcd can be computed in time $\widetilde{O}(n^2 + n \log A)$.*

Proof. See [24, §6.8 and §11.1] □

1.4.3 Divide and conquer

Most of the asymptotically fast algorithms have a recursive formulation. We shall analyze them using the following lemma.

Lemma 1.18. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ such that*

- $f(x) \leqslant af(x/b) + cx$, *for some* $a > 0$, $b > 1$ *and* $c \in \mathbb{R}$,
- $f(x) \leqslant 1$, *if* $x < x_0$.

Then

$$f(x) \ll \begin{cases} x^{\log_b a} & \text{if } a > b, \\ x \log x & \text{if } a = b, \\ x & \text{if } a < b. \end{cases}$$

The hypotheses are understood as follows: $f(x)$ is a cost function for a given size x; we cut our original problem in a independent subproblems which are b times smaller, then recombine the partial solution into a global one in linear time $O(x)$. The favorable situation is $a \leqslant b$, where the subproblems have balanced sizes and do not proliferate, yielding $\widetilde{O}(x)$ time, essentially linear.

Proof. Let $x \geqslant x_0$, et let $\ell := \lceil \log_b(x/x_0) \rceil$ be the smallest integer such that $x/b^\ell \leqslant x_0$. We have

$$\begin{aligned} f(x) &\leqslant af(x/b) + cx \leqslant a\big(af(x/b^2) + cx/b\big) + cx \\ &\leqslant \ldots \leqslant a^\ell f(x/b^\ell) + cx \sum_{i=0}^{\ell-1} (a/b)^i. \end{aligned}$$

By the choice of ℓ, we have $f(x/b^\ell) \leqslant 1$; further, $\ell \leqslant \log_b x + 1$, thus for all $y \geqslant 1$,

$$y^\ell \leqslant y \times y^{\log_b x} = yx^{\log_b y}.$$

We can bound $a^\ell \leqslant ax^{\log_b a}$; how we bound the geometric sum depends on the value of $r = a/b$:

- if $r > 1$, the sum is $(r^\ell - 1)/(r-1)$, and $r^\ell \leqslant r x^{\log_b(a/b)} = O(x^{\log_b a - 1})$;
- if $r = 1$, the sum is $\ell \leqslant 1 + \log_b x$;
- if $r < 1$, the sum is $(r^\ell - 1)/(r-1) < 1/(1-r)$.

□

Corollary 1.19. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ such that*

- *$f(x) \leqslant af(x/b) + cx^r$, for some $r > 0$, $a > 0$, $b > 1$ and $c \in \mathbb{R}$,*
- *$f(x) \leqslant 1$, if $x < x_0$. Then*

$$f(x) \ll \begin{cases} x^{\log_b a} & \text{if } a > b^r, \\ x^r \log x & \text{if } a = b^r, \\ x^r & \text{if } a < b^r. \end{cases}$$

Proof. Let $g(x) = f(x^{1/r})$ and apply the Lemma to g. □

Let us start with a simple example (mergesort), we shall study a much more involved one (the Fast Fourier Transform) in the next section.

Example: mergesort. We want to sort n numbers in increasing order using only comparisons. We denote by $f(n)$ the number of comparisons required.

1. Sort the first half; cost: $f(n/2)$.
2. Sort the second half; cost: $f(n/2)$.
3. Merge the two sorted sublists; cost $O(n)$ (why?)

Whence $f(x) \leqslant 2f(x/2) + O(x)$, and the lemma implies that $f(x) = O(x \log x)$.

Exercise 1.20. Using the asymptotic estimate $\log n! \sim n \log n$, prove that this order of magnitude is optimal : by using only comparisons, it is not possible to sort n integers in $o(n \log n)$ operations.

1.5 The Fast Fourier transform (FFT)

We investigate in this section a very important application of the divide and conquer paradigm: the Fast Fourier Transform algorithm, or FFT, which is the basis of most (asymptotically) fast multiplication algorithms.

1.5.1 When $\omega_n \in K$

Let $n = 2^k$ be a power of 2 and let K be a field containing an n-th primitive root ω of 1. In other words, $\omega \in K^*$ has multiplicative order exactly n. Let $T(X) = a_0 + a_1X + \cdots + a_{n-1}X^{n-1}$, with $(a_0, \ldots, a_{n-1}) \in K^n$. We define:

$$\mathcal{F}(T, \omega) := (T(\omega^0), \ldots, T(\omega^{n-1})) \in K^n$$

the discrete Fourier transform of T, and by abuse of notation.

$$\mathcal{F}\big((a_0, \ldots, a_{n-1}), \omega\big) := \mathcal{F}(a_0 + a_1X + \cdots + a_{n-1}X^{n-1}, \omega).$$

Our problem is to compute $\mathcal{F}(T, \omega)$. Note that $T(\omega^\ell) = \sum_{0 \leqslant t < n} a_t\omega^{\ell t}$, a discrete version of $\hat{f}(\ell) = \int f(t)e^{it\ell}\,dt$: we are computing the Discrete Fourier Transform (DFT) of $t \mapsto a_t$. Define the even and odd parts of T:

$$T_0(X^2) := \frac{T(X) + T(-X)}{2}, \quad XT_1(X^2) := \frac{T(X) - T(-X)}{2}$$

so that $T(X) = T_0(X^2) + XT_1(X^2)$. In particular, for all i, we have

$$T(\omega^i) = T_0((\omega^2)^i) + \omega^i T_1((\omega^2)^i).$$

Since $\deg(T_0), \deg(T_1) \leqslant n/2$ and $\operatorname{ord}(\omega^2) = n/2$, this leads to a simple recursive algorithm:

Algorithm 1. (Fast) Discrete Fourier Transform

Input: R a commutative ring; $T \in R[X]$, $\deg T < n = 2^k$, a primitive root of unity ω of order n in R. The vector $(1, \omega, \ldots, \omega^{n-1})$ is precomputed.

Output: $\mathcal{F}(T, \omega)$.

```
1: If $n < 1$, return the 1-dimensional vector containing the scalar $T$.
2: Compute $(a_0, \ldots, a_{n/2-1}) = \mathcal{F}(T_0, \omega^2)$, calling ourselves recursively.
3: Compute $(b_0, \ldots, b_{n/2-1}) = \mathcal{F}(T_1, \omega^2)$.
4: Return $(a_i + \omega^i b_i)_{i<n}$.   {Extend $a_i$ and $b_i$ by periodicity: $a_{n/2+i} := a_i$.}
```

We denote by $f(n)$ the number of basic operations in R, of type $+, \times$ (in fact, multiplication by a power of ω), required to compute $\mathcal{F}(T, \omega)$. Again we have $f(x) \leqslant 2f(x/2) + O(x)$, hence $f(x) = O(x \log x)$. This almost linear cost is a large improvement over the obvious approach where we evaluate successively the $T(\omega^i)$: in the latter case we obtain each individual $T(\alpha)$ in time $O(n)$ using the identity

$$T(\alpha) = a_0 + \alpha(a_1 + \alpha(\ldots(a_{n-2} + a_{n-1}\alpha)\ldots))$$

(Horner's scheme), yielding an $O(n^2)$ algorithm for the naive DFT.

Whatever algorithm we use to compute it, the DFT has a very nice property: it is just as easy to invert as to compute.

Lemma 1.21. *Let R be a commutative ring containing a root of unity ω of order n such that $(1-\omega^\ell)$ is invertible for all ℓ not divisible by n. For all $T \in R[X]$, $\deg T < n$, we have*

$$\mathcal{F}(\mathcal{F}(T,\omega),\omega^{-1}) = nT.$$

Proof.

$$\sum_{i<n} T(\omega^i)\omega^{-ki} = \sum_{i,j<n} a_j \omega^{i(j-k)} = na_k.$$

Indeed, the terms for $j=k$ add up to na_k; for $j \neq k$, use

$$\sum_{i<n} \omega^{i\ell} = 0 \quad \text{when } \ell \not\equiv 0 \pmod{n}.$$

(Multiply by the unit $(1-\omega^\ell)$.) □

Note that the technical condition $(1-\omega^\ell) \in R^*$ is automatic if K is a field since ω has order $n \Rightarrow \omega^\ell \neq 1$. The following exercise presents another situation where the condition is automatically satisfied :

Exercise 1.22. For $k \in \mathbb{Z}_{>0}$, write Φ_k for the k-th cyclotomic polynomial. Let R be a commutative ring, $n = p^k$ a power of a prime p. Let further $\omega \in R^*$ be of order n, such that $\Phi_n(\omega) = 0$. (This is automatic if R is a domain, but need not be true in general.) Using the factorization

$$\sum_{i<n} X^i = \prod_{j=1}^{k} \Phi_{p^j}(X) = \prod_{j=1}^{k} \Phi_p(X^{p^{j-1}}),$$

show that we still have

$$\sum_{i<n} \omega^{i\ell} = 0 \quad \text{when } \ell \not\equiv 0 \pmod{n},$$

so the conclusion of the Lemma still holds for $T \in R[X]$.

Corollary 1.23. *Let R be a commutative ring of characteristic $\neq 2$, and let m be a power of 2. Let ω be the class of X in $D = R[X]/(X^m+1)$. Then ω has order $2m$ and $(1-\omega^\ell)$ is invertible in D for all ℓ not divisible by $2m$.*

Proof. If m is a power of 2, then $\Phi_{2m} = X^m + 1$. □

Exercise 1.24. For p a fixed small prime, R and ω as in the previous exercise, write

$$T(X) = T_0(X^p) + XT_1(X^p) + \cdots + X^{p-1}T_{p-1}(X^p),$$

where the T_i have degree $\leqslant n/p$. Devise a p-adic DFT algorithm using $O_p(n \log n)$ operations in R.

We now come to a most important application.

Algorithm 2. Fast multiplication in $K[X]$, $\omega_n \in K$

```
Input: $S, T \in K[X]$, where $\deg S, \deg T < \frac{n}{2}$.
Output: $n \times S \times T$.
Compute $\omega^2, \dots, \omega^{n-1}$.
Compute $\mathcal{F}(S, \omega) = (a_0, \dots, a_{n-1})$.
Compute $\mathcal{F}(T, \omega) = (b_0, \dots, b_{n-1})$.
Return $\mathcal{F}\big((a_0 b_0, \dots, a_{n-1} b_{n-1}), \omega^{-1}\big)$.
```

Corollary 1.25. *Provided K contains a primitive root ω of degree $2n = 2^k$, polynomials of degree up to n can be multiplied in $O(n \log n)$ operations in K.*

Proof. It is enough to prove that $\operatorname{char} K \neq 2$, hence $n \in K^*$ is invertible. But if $\operatorname{char} K = 2$, then $0 = \omega^n - 1 = (\omega - 1)^n$ and $\omega = 1$ which does not have order n in K^*. □

Compare with the naive approach which requires $O(n^2)$ operations in K! The basic idea is essentially due to Gauss, rediscovered in the 1960's: the Fourier transform is a K-algebra isomorphism from $K[X]/(X^n - 1)$ to K^n. It transforms the complicated algebra on the left to the trivial product algebra on the right. This is basically a homomorphic image scheme : we map to K^n, multiply there and come back; we can thus multiply quickly in $K[X]/(X^n - 1)$. Finally, Provided all polynomials considered have degree $< n/2$, multiplying in $K[X]/(X^n - 1)$ and $K[X]$ is the same.

For later reference, we note that the exact same multiplication algorithm performs equally well in a more general situation (cf. Corollary 1.23):

Algorithm 3. Fast multiplication in $D[Y]$, $D = R[X]/(X^{2m} + 1)$, $m = 2^k$

```
Input: $S, T \in D[Y]$, where $\deg S, \deg T < t$, where $t = m$ or $2m$.
Output: $t \times S \times T \pmod{Y^t - 1}$.
Let $\omega$ be the class of $X^2$ ($t = 2m$), resp. the class of $X^4$ ($t = m$); then $\omega$ is a
    primitive $t$-th root of 1 in $D$.
Compute $\omega^2, \dots, \omega^{t-1}$.
Compute $\mathcal{F}(S, \omega) = (a_0, \dots, a_{t-1})$.
Compute $\mathcal{F}(T, \omega) = (b_0, \dots, b_{t-1})$.
Return $\mathcal{F}\big((a_0 b_0, \dots, a_{t-1} b_{t-1}), \omega^{-1}\big)$.
```

1.5.2 The Schönhage-Strassen algorithm

A major problem in this presentation is the requirement that there exists a primitive n-th root of 1 with $n = 2^k$ in the base ring. A simple solution is to embed

K in $K[y]/(y^n - 1)$, but this is no longer a field and Lemma 1.21 needs not hold. Exercise 1.22 provides a solution: embed in $K[y]/(\Phi_n(y))$, for $n = 2^k$, or more generally $n = p^k$ using Exercise 1.24.

Unfortunately, an operation in this new quotient ring now requires about n operations in K (since $\deg \Phi_n = \phi(n) = n/2$), which does not look good. The details will be rather involved but the idea is remarkably simple and general: instead of multiplying polynomials of degree n using roots of order n, multiply polynomials of degree $\sqrt{n}$ using roots of order $\sqrt{n}$. Then a linear number of essentially linear-time operations in the larger ring still amounts to $\widetilde{O}(n)$ operations in the base ring R.

Let us make this more precise:

Definition 1.26. Let R be a ring, m be an integer and $T \in R[X]$.

- let $T^\sharp \in R[X, Y]$ be the unique bivariate polynomial such that $T^\sharp(X, X^m) = T(X)$, $\deg_X T^\sharp < m$. For instance, if $m = 3$ and
$$T(X) = 1 + 2X + 3X^2 + 4X^3 + 5X^4 + X^{10},$$
then
$$T^\sharp(X, Y) = (1 + 2X + 3X^2) + (4 + 5X)Y + XY^3.$$
Note that $\deg_Y T^\sharp = \lfloor \deg T/m \rfloor$.
- let $D = R[X]/(X^{2m} + 1)$, in which $\omega = X$ is a primitive $4m$-th root of 1. We let $T^*(Y) = T^\sharp(X, Y) \mod X^{2m} + 1 \in D[Y]$.

Definition 1.27. In all that follows, $f, g \in R[X]$ are such that $\deg fg < n = 2^k$. We let $m = 2^{\lfloor k/2 \rfloor}$ and $t = n/m = m$ or $2m$.

To recover fg, we need only compute $fg \mod (X^n + 1)$. For this it is enough to compute $f^\sharp g^\sharp \mod Y^t + 1$, more precisely to compute an $h^\sharp \in R[X, Y]$ such that
$$f^\sharp g^\sharp = q^\sharp \cdot (Y^t + 1) + h^\sharp, \quad \deg_Y h^\sharp < t. \tag{1.1}$$
Indeed, evaluation at $(X, Y) = (X, X^m)$, we obtain that
$$f(X)g(X) = q^\sharp(X, X^m) \cdot (X^n + 1) + h^\sharp(X, X^m) \equiv h^\sharp(X, X^m) \pmod{X^n + 1}.$$
Equation (1.1) implies that $\deg_X q^\sharp \leqslant \deg_X f^\sharp g^\sharp < 2m$, which in turn implies $\deg_X h^\sharp < 2m$. Reducing the equation mod $X^{2m} + 1$ this time we obtain
$$f^*(Y)g^*(Y) \equiv h^*(Y) \pmod{Y^t + 1}$$
in $D[Y]$. Note that since $\deg_X h^\sharp < 2m$, we can recover $h^\sharp(X, Y)$ from $h^*(Y) = h^\sharp(X, Y) \pmod{X^{2m} + 1}$. The key remark is that f^* and g^* have degree less than $t \leqslant 2m$, and D contains $\omega = \bar{X}$, a $4m$-th root of 1, hence a $2t$-th root of unity, $\eta = \omega$ $(t = 2m)$ or ω^2 $(t = m)$; we can thus compute $f^* g^*$ (up to powers of 2) using three $2t$-points FFTs and $2t$ multiplications in D. There are two further tricks, each allowing to save a further factor 2:

- since $\eta^t = -1$, the above congruence shows that
$$f^*(\eta Y)g^*(\eta Y) \equiv h^*(\eta Y) \pmod{Y^t - 1},$$
and we can use t-point FFTs instead of $2t$-points FFTs !

- We can gain another factor 2 during the multiplications in D : we only need the result modulo $X^{2m}+1$ this time, even though the inputs have representatives of degree $< 2m$. When *first* calling the algorithm, we must ensure that the inputs f, g satisfy $\deg fg < n$ in order to reconstruct fg from $fg \mod (X^n + 1)$. During the recursion, we no longer care.

This yields the following algorithm:

Algorithm 4. Fast multiplication in $R[X]$, $\operatorname{char} R \neq 2$ (Schönhage-Strassen)

```
Input: $f, g \in R[X]$ of degree $< n = 2^k$.
Output: $2^{e(n)} fg \mod X^n + 1$ for some $e(n) \in \mathbb{Z}_{\geqslant 0}$.
If $k \leqslant 2$, return $f \times g \mod X^n + 1$ using a naïve algorithm. In particular, $e(1) = e(2) = e(4) = 0$.
Let $m = 2^{\lfloor k/2 \rfloor}$ and $t = n/m = m$ or $2m$. Let $D = R[X]/(X^{2m} + 1)$, and $f^*, g^* \in D[Y]$ be as above, with degree $< t$. Let $\eta$ be a root of order $2t$ in $D$, namely $\eta = \omega$ $(t = 2m)$ or $\omega^2$ $(t = m)$.
Compute $\mathcal{F}(f^*(\eta Y), \eta^2) = (a_0, \ldots, a_{t-1}) \in D^t$.
Compute $\mathcal{F}(g^*(\eta Y), \eta^2) = (b_0, \ldots, b_{t-1}) \in D^t$.
Compute $\mathcal{F}\big((2^{e(2m)} a_0 b_0, \ldots, 2^{e(2m)} a_{t-1} b_{t-1}), \eta^{-2}\big) = 2^{e(2m)+t} h^*(\eta Y)$ in $D[Y]$, $\deg h^* < t$. We call ourselves recursively for the $t$ multiplications $a_i b_i$ in $D$, where we perform the multiplication on representatives of degree $< 2m$ in $R[X]$, yielding $2^{e(2m)} a_i b_i$ in $R[X]/(X^{2m} + 1)$.
Recover $h^*(Y)$ from $h^*(\eta Y)$, then $h^\sharp \in R[X, Y]$ from $h^*$ (lift all coefficients in $D$ to their representative of minimal $X$-degree). Finally use $h^\sharp(X, X^m) = (fg)(X)$ to recover $2^{e(n)} fg \in R[X]$, with $e(n) = e(2m) + t$.
```

Proposition 1.28. *The algorithm requires $O(n \log n \log\log n)$ operations in R.*

Proof. The three FFTs over D each require $O(t \log t)$ operations in D:

- additions : each addition in D corresponds to $O(m)$ additions in the base ring R;

- multiplications by the ω^i : this is a shift, followed by a reduction mod $X^{2m} + 1$, using $O(m)$ subtractions in R.

Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a non-decreasing function such that $f(n)$ is the algebraic complexity of the algorithm. We have

$$f(n) \leqslant t f(2m) + O(n \log n).$$

Replacing $k \to k+1$, hence $n \to 2n$, replaces $t \to 2m$ and $m \to t$, hence

$$f(2n) \leqslant 2mf(2t) + O(n \log n).$$

Let $g(x) = \frac{f(2x)}{x(\log x - 1)}$, where log denotes the base-2 logarithm; then the inequality simplifies to

$$g(n) \leqslant \frac{2(\log t - 1)}{\log n - 1} g(t) + O(1),$$

using $2mt = 2n$. The fraction is $2(\lceil k/2 \rceil - 1)/(k-1) \leqslant 1$ and we obtain

$$g(n) \leqslant g(t) + C,$$

for some universal constant C. Define by induction the two sequences

- $x_0 = k$, $x_{i+1} = \lceil x_i/2 \rceil$ for $i \geqslant 0$.
- $y_0 = k$, $y_{i+1} = (y_i + 1)/2$ for $i \geqslant 0$.

The motivation for this is that

$$g(2^{x_0}) \leqslant g(2^{x_1}) + C \leqslant \ldots \leqslant g(2^{x_i}) + C \times i.$$

By induction, we have $x_i \leqslant y_i$ for all i, and $y_i = 1 + (k-1)/2^i$. It follows that $x_i \leqslant 2$ for $i = O(\log k)$. Finally, $g(2^k) = O(\log k)$ and we are done ! □

Note that the algorithm can be trivially modified so as to return the precise value of $e(n)$, as well as the product $2^{e(n)}ST$.

All we have done is useless when $\operatorname{char} R = 2$ (we get 0) or more generally when 2 is not invertible in R (how to recover ST from $2^{e(n)}ST$?). In this case, in addition to using 2^k-th roots of unity and a 2-adic DFT, we can use 3^s-th roots and a 3-adic DFT as introduced in Exercise 1.24. We may also compute $3^s ST$ in essentially linear time, for some integer $s \geqslant 0$. From a Bezout relation $u2^k + v3^s = 1$ in $\mathbb{Z}$, we recover the product ST without requiring a division! The final result is

Theorem 1.29 (Cantor-Kaltofen). *Over any commutative ring R, polynomials of degree less than n can be multiplied in $O(n \log n \log\log n) = \widetilde{O}(n)$ operations in R.*

Theorem 1.30 (Schönhage-Strassen). *Two positive integers less than 2^n represented by bit-strings can be multiplied in time $O(n \log n \log\log n) = \widetilde{O}(n)$.*

Proof. (Rough sketch) This follows the same pattern as the polynomial multiplication. The idea is to represent integers as values of polynomials at powers of 2. We are reduced to multiplying $f, g \in \mathbb{Z}[X]$, $\deg fg < n = 2^k$, and $\|f\|_\infty$, $\|g\|_\infty \leqslant 2^\ell$. We can assume that $2\ell + 1 \leqslant n - k$ in which case $\|fg\|_\infty < 2^{n-1}$. This time we compute fg in $D[X]$ where $D = \mathbb{Z}/(2^n+1)\mathbb{Z}$ supports the FFT since 2 is a primitive $2n$-th root of 1. The multiplications in D are handled by a recursive call to the algorithm. □

1.6 Basic complexity results

1.6.1 In $\mathbb{Z}$

The size of an integer a is the number of bits required to store a, i.e $s(a) := \lfloor \log_2(a+1) \rfloor + 1$. Assume all operands have size less than n.

Operation	Naive	Fast
$a+b$	$O(n)$	$O(n)$
$a \times b$	$O(n^2)$	$\mathsf{M}_{\mathbb{Z}}(n) = \widetilde{O}(n)$
$a = bq + r$	$O(n^2)$	$O(\mathsf{M}_{\mathbb{Z}}(n))$
Extended gcd	$O(n^2)$	$O(\mathsf{M}_{\mathbb{Z}}(n) \log n)$
Chinese remainders	$O(n^2)$	$O(\mathsf{M}_{\mathbb{Z}}(n) \log n)$

- Multiplication : $\mathsf{M}_{\mathbb{Z}}(n)$ is the multiplication time in $\mathbb{Z}$ for two operands of size less than n. The fast algorithm is based on Schönhage-Strassen multiplication, in time $O(n \log n \log\log n)$.

  For inputs of different sizes, $s(a) \leqslant n$, $s(b) \leqslant m$, the naive (quadratic) algorithm runs in time $O((n+1)(m+1))$.

- Euclidean division : the input is (a,b), $b \neq 0$, and the output (q,r) with $0 \leqslant r < |b|$. The fast algorithm solves the equation $b - a/x = 0$ using the Newton Iteration
$$x_{n+1} = x_n - x_n(x_n b - a).$$
  (Let the precision increase with the iterations and use fast multiplication.) We then set $q = \lfloor x \rfloor$, then $r = a - bq$. The complexity stated assumes than $\mathsf{M}_{\mathbb{Z}}(n)$ satisfies properties like $M(n)/n \geqslant M(m)/m$ for all $n \geqslant m$, and $M(mn) \leqslant m^2 M(n)$, it is in particular applicable for the Schönhage-Strassen and the naive quadratic multiplication.

  If $s(a) \leqslant n$, $s(b) \leqslant m$, $n \geqslant m$, the naive (quadratic) algorithm runs in time $O((n-m+1)(m+1))$.

- Extended gcd: the input consists of two integers a, b and the output consists of the $\gcd(a,b)$ and two integers u, v such that $au + bv = \gcd(a,b)$. The fast gcd is again based on the divide an conquer paradigm.

- Chinese Remainder: the input consists of n congruences $x \equiv a_k \pmod{b_k}$ where the b_k are pairwise coprime with and the output expected is a solution for the above congruences. We assume $s(a_k) \leqslant s(b_k)$ and $\sum_k s(b_k) \leqslant n$. The fast algorithm uses three divide-and-conquer passes: first to compute a product tree, then all modular inverses simultaneously, then a standard recursion.

1.6.2 In $\mathbb{Z}/N\mathbb{Z}$

We choose a canonical representative in each congruence class. A natural choice are the integers in $[0, N-1]$; another is $]-N/2, N/2]$, which is often more efficient when we need small negative integers, but a little more complicated to describe. In both cases, the size of any input is less than $n := s(N)$.

An addition is implemented as an addition in $\mathbb{Z}$, possibly followed by a subtraction. Multiplication, is a multiplication in $\mathbb{Z}$, followed by a Euclidean division by N. Inversion is an extended gcd followed by a multiplication. So the costs are the same as in $\mathbb{Z}$, except for fast division which is more expensive by a factor $\log n$.

1.6.3 In $K[X]$ where K is a field :

Here the costs for operations in $K[X]$ count the number of operations in K; we may multiply by the cost of an elementary operation in K when this cost is bounded independently of the element considered, i.e. when K is finite. The operations taken into account are $+$, $-$, $\times$, $/$ in K. Let f, g be two polynomials in $K[X]$. If $h \in K[X]$, the size of h is $S(h) = \deg h + 1 \leqslant n$.

Operation	Naive	Fast
$f+g$	$O(n)$ $[+]$	$O(n)$ $[+]$
$f \times g$	$O(n^2)$ $[+, \times]$	$\mathsf{M}_{K[X]}(n) = \widetilde{O}(n)$
$f = gh + r$	$O(n^2)$	$O(\mathsf{M}_{K[X]}(n))$
Extended gcd	$O(n^2)$	$O(\mathsf{M}_{K[X]}(n)\log n)$
CRT	$O(n^2)$	$O(\mathsf{M}_{K[X]}(n)\log n)$

1.6.4 In $K[X]/(T)$:

Important special cases are finite field extensions $\mathbb{F}_q/\mathbb{F}_p$, and finite extensions of $\mathbb{Q}$. As in $\mathbb{Z}/N\mathbb{Z}$ we work with polynomials of size $\leqslant s(T)$, so the costs are as above. Again, fast modular division is slower by a factor $\log n$ than fast Euclidean division.

1.6.5 In $M_{n\times n}(K)$:

Again, we count the operations in K, for $A \in M_{n\times n}(K)$, $S(A) = n^2$.

Operations	Naive	Fast
$A+B$	$O(n^2)$	$O(n^2)$
$A \times B$	$O(n^3)$	$O(n^\omega), \omega = 2.376$
$A = LU$	$O(n^3)$	$O(n^\omega)$

The LU factorization is enough to solve most linear algebra problems over K: computing kernels, image, rank profile... In the above, ω is called a *feasible multiplication exponent*. The best value used for practical sizes is $\omega = \log_2 7 \approx 2.8$ (Strassen).

Black Box Linear Algebra: in this model, costs are calculated as the number of evaluations $x \mapsto Ax$ for a "black box matrix" A. (The name comes from the fact that we do not know anything about A except how it acts on vectors: it is an opaque operator, or a black box.) In general, matrix-vector multiplication for $A \in M_{n\times n}(K)$ is an $O(n^2)$ operation but most matrices encountered in practice have some structure which make evaluation cheaper, e.g. diagonal or band matrices, sparse matrix (as in the factorbase algorithms used to factor integers), Sylvester's matrix from the resultant definition, Berlekamp matrix (used to factor polynomials over finite fields), FFT matrix (= van der Monde on roots of unity), etc.

In this model, randomized algorithms are available that can compute the LU factorisation of A in $O(n)$ evaluations and $O(n^2)$ field operations on average. So we gain nothing on general matrices, where a matrix-vector multiplication requires $O(n^2)$ scalar operations; but quite a lot for special matrices.

Chapter 2

Lattices

2.1 $\mathbb{Z}$-modules

2.1.1 Definitions

Any abelian group $(G, +)$ can be made into a module over $\mathbb{Z}$ by the rules

- $0 \cdot g = 0_G$, the neutral element of G,
- $n \cdot g = g + \cdots + g$ (n summands), for $n > 0$,
- $(-n) \cdot g = -(n \cdot g)$, for $n < 0$, where $-x$ is the inverse of x.

We may thus identify abelian groups and $\mathbb{Z}$-modules, as well as submodules with subgroups.

- If $g = (g_i) \in G^n$ and $\lambda = (\lambda_i) \in \mathbb{Z}^n$, the linear combination $\sum_i \lambda_i g_i \in G$ will be denoted $g \cdot \lambda$. More generally, let m, n be two non-negative integers. Using freely the linear algebra formalism, we can associate to any matrix in $M_{m\times n}(\mathbb{Z})$ a $\mathbb{Z}$-linear operator $G^m \to G^n$ by "right multiplication" :

  $$(g_1, \ldots, g_m) \cdot \begin{pmatrix} \lambda_{1,1} & \cdots & \lambda_{1,n} \\ \vdots & \vdots & \vdots \\ \lambda_{m,1} & \cdots & \lambda_{m,n} \end{pmatrix} = \left(\sum_{i=1}^{n} \lambda_{i,1} g_i, \cdots, \sum_{i=1}^{n} \lambda_{i,n} g_i \right).$$

  The degenerate cases corresponding to $m = 0$ or $n = 0$ are handled gracefully by the convention $G^0 = (0)$: the trivial module, with a single element.

- If $A \subset G$, the submodule/subgroup generated by A is

  $$\langle A \rangle_{\mathbb{Z}} := \Big\{ \sum_{i \in I} \lambda_i a_i \colon (\lambda_i) \in \mathbb{Z}^I,\ (a_i) \in A^I,\ I \text{ finite} \Big\}$$

- G is of *finite type* if $G = \langle A \rangle_{\mathbb{Z}}$ for some finie A. In this case, we have $G = A \cdot \mathbb{Z}^{\# A}$; in other words, any element of G is of the form $A \cdot \lambda$. All our modules will be of this type.
- A family $g = (g_1, \dots, g_n)$ is *free* (linearly independent) if and only if
$$g \cdot \lambda, \ \lambda \in \mathbb{Z}^n \Rightarrow \lambda = 0.$$
- G (of finite type) is *free* if and only if there exist $(g_1, \dots, g_n) \in G^n$ which is free and generates G. In this case, $(g_1, \dots, g_n)$ is called a *basis*, and the integer n is well-defined: it is the *rank* of G.

Example 2.1. Not all modules have a basis: $G = \mathbb{Z}/2\mathbb{Z}$ is not free because $2x = 0$ for all $x \in G$, and $2 \neq 0$, so no non-empty subset of G can be free; but the empty set does not generate G. More generally, no module with non-trivial torsion elements can be free. However, $\mathbb{Z}^n$ is free, for all $n \geqslant 0$.

We state without proof two basic theorems about $\mathbb{Z}$-modules:

Theorem 2.2 (Adapted Basis). *Let G be a free $\mathbb{Z}$-module of rank n, and $H \leqslant G$ a submodule. There exists a basis $(g_1, \dots, g_n)$ of G and $d_n \mid \cdots \mid d_1$, $d_i \in \mathbb{Z}_{\geqslant 0}$, such that $\{d_i g_i \colon d_i > 0\}$ is a basis for H. In particular, H is free, with rank $\leqslant n$.*

The integers $d_1, \dots, d_n$ are well-defined: they do not depend on the basis (g_i).

Note that some of the d_i can be zero (the first ones).

Corollary 2.3 (Elementary Divisors). *Let G be a $\mathbb{Z}$-module of finite type. There exist $g_1, \dots, g_n$ in G such that*

$$\begin{aligned} G &= \bigoplus_{i=1}^{n} (\mathbb{Z}/d_i\mathbb{Z}) \cdot g_i, \quad \textit{where} \quad d_n \mid \cdots \mid d_1, \quad d_i \in \mathbb{Z}_{\geqslant 0}, \\ &= \underbrace{\bigoplus_{i=1}^{r} \mathbb{Z} \cdot g_i}_{\mathbb{Z}^r} \oplus \underbrace{\bigoplus_{r+1}^{n} (\mathbb{Z}/d_i\mathbb{Z}) \cdot g_i}_{G_{tor}} \quad \textit{if } d_1 = \cdots = d_r = 0, \textit{ and } d_{r+1} \neq 0, \end{aligned}$$

where G_{tor} is the torsion subgroup of G containing all the elements of finite order. If we further assume that $d_n \neq 1$, then the $(d_1, \dots, d_n)$ are unique and n is the minimal number of generators in any presentation of G.

The meaning of the direct sum in the Corollary is as follows:

$$\sum_{i=1}^{n} \lambda_i g_i = 0, \quad \lambda_i \in \mathbb{Z} \Leftrightarrow \lambda_i \in d_i\mathbb{Z}, \quad \forall i.$$

We call r the *rank* of G, generalizing the notion defined for free modules.

We now make these results explicit using linear algebra over $\mathbb{Z}$.

2.1.2 Hermite Normal Form (HNF)

Studying free modules is similar to studying vector spaces. If L is a free submodule of rank n in some $\mathbb{Z}^m$, it can be represented by an $m \times n$ matrix whose columns give the coordinates of a basis of L on the canonical basis of $\mathbb{Z}^m$. The representation is not unique, because it depends on a choice of basis for L. In vector spaces, any basis can be brought to column echelon form — even better, to canonical Gauss-Jordan form — using Gaussian elimination, but this algorithm uses division in an essential way, a forbidden operation over $\mathbb{Z}$.

The Hermite Normal Form generalizes the Gauss-Jordan form (over a field K) to modules (over a principal ring R). The simple underlying idea was introduced in Exercise 1.10. Here is a comparison of the two methods with 2×2 matrices. If $a \neq 0$, Gaussian elimination yields:

$$\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 1 & -b/a \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a & 0 \end{pmatrix}$$

Using a Bezout relation $au + bv = \delta := \gcd(a, b)$ instead, we can write:

$$\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} u & -b/\delta \\ v & a/\delta \end{pmatrix} = \begin{pmatrix} \delta & 0 \end{pmatrix}.$$

The multiplying matrix is no longer elementary, but nevertheless in $\mathrm{SL}_2(R)$. Before generalizing this idea (in order to zero chosen entries in a matrix while conserving some global structure), we first define our analog of the column echelon form, over the principal ring $R = \mathbb{Z}$. (Other normalizations could be chosen over an arbitrary principal ring, e.g. Howell form.)

Definition 2.4. The matrix $H = (H_{i,j}) \in M_{m\times n}(\mathbb{Z})$ is in *Hermite Normal Form* (HNF) if there exist $r \geqslant 0$ such that only the first r columns are non-zero, and a strictly decreasing function $f\colon \{1, \dots, r\} \to \{1, \dots, m\}$ satisfying

- [Echelon] $q_j := H_{f(j),j} > 0$, $H_{i,j} = 0$ if $i > f(j)$ and $H_{f(j),k} = 0$ if $k > j$,
- [Reduction] $0 \leqslant H_{f(j),k} < q_j$ if $k < j$.

In particular, the matrix rank is r and the last $n - r$ columns are zero. A matrix satisfying only the first condition is in *echelon form*. It is easier to tell what the definition is saying with a picture of the matrix:

$$\left(\begin{array}{ccccc|ccc}
\times & \times & \times & \times & q_5 & 0 & \cdots & 0 \\
\times & \times & \times & \times & & 0 & \cdots & 0 \\
\times & \times & \times & \times & & & & \\
\times & \times & \times & q_4 & & & & \\
\times & \times & q_3 & & & & & \\
\times & q_2 & & & & & & \\
\times & & & & & & & \\
q_1 & & & & & 0 & \cdots & 0
\end{array}\right)$$

Here $r = 5$, and the × entries can be zero, positive or negative. There are two conditions:

- $f(j)$ is the row where the pivot $q_j > 0$ lies. All coefficients to the right of a pivot (on the same row), or below it (in the same column), are 0.
- All the coefficients to the left of a pivot q_j (on the same row) are reduced mod q_j. So the × to the left of a q_j satisfy $0 \leqslant \times < q_j$.

The function f determines the rank profile: the matrix $\left(H_{f(i),j}\right)_{1\leqslant i,j\leqslant r}$ is non-singular.

Example 2.5. Assume $H \in M_{n\times n}(\mathbb{Z})$ has rank n. In this (simplest) case, $r = n$ and the rank profile f is the identity.

Theorem 2.6. *The $m \times n$ HNF matrices form a system of representatives of $M_{m\times n}(\mathbb{Z}) \,/\, \mathrm{GL}_n(\mathbb{Z})$.*

N.B. $\mathrm{GL}_n(\mathbb{Z})$ acts on $M_{m\times n}(\mathbb{Z})$ by right multiplication. The previous formulation is equivalent to the following one: if $A \in M_{m\times n}(\mathbb{Z})$, there exist a unique $H \in M_{m\times n}(\mathbb{Z})$ in HNF and a matrix $U \in GL_n(\mathbb{Z})$ (not necessarily unique) such that $H = AU$. We call H the *HNF reduction* of A.

Corollary 2.7. *If we fix a basis $(g_i)_{i\leqslant m}$ of a free module $G \simeq \mathbb{Z}^m$, any submodule H of G has a canonical basis, given by $(h_i)_{i\leqslant n} = (g_i) \cdot M$, where M is an HNF matrix. We call it the* HNF-basis *of H.*

Proof. It is sufficient to see that the submodule H can be represented by a matrix whose n columns are elements of one of its bases, described by their coordinates in the fixed basis of G. Since these bases are defined modulo $\mathrm{GL}_n(\mathbb{Z})$, the unicity of the HNF gives the result. □

Our specific definition of the HNF is unimportant: one could swap the columns or choose another system of representatives of $\mathbb{Z}/q_i\mathbb{Z}$ than $[0, q_i - 1]$ for instance. What matters is that, on the one hand, the HNF is a normal form (two modules are identical if and only if they have the same HNF basis), and on the other hand, the echelon form simplifies standard linear algebra problems. Here are direct applications of the normal form property: a submodule $G \subset \mathbb{Z}^m$ is defined by a matrix $A \in M_{m\times *}(\mathbb{Z})$, whose columns generate it.

- *Equality* of two submodules: let G_1 and G_2 be two submodules of $\mathbb{Z}^m$ defined by matrices A_1 and A_2. Then they are equal if and only if the HNF reductions of A_1 and A_2 are equal.
- *Sum* of two submodules: let G_1 and G_2 be two submodules of $\mathbb{Z}^m$ defined by matrices A_1 and A_2. Then the HNF of $(A_1 \mid A_2)$ gives an HNF-basis for $G_1 + G_2$.

- *Inclusion.* We have $G_1 \subseteq G_2 \iff G_1 + G_2 = G_2$, which can be checked using the previous two points. A generally more efficient test if G_2 is fixed and G_1 varies consists in computing the HNF H of the matrix associated to G_2, and then checking whether the triangular linear system $HX = A_1$ has a solution (see below).

Here are some classical linear algebra problems made simple by the echelon form. Let A be a matrix in $M_{m\times n}(\mathbb{Z})$ and $H = AU$ its HNF reduction.

- *Image*: the non-zero columns of H form a basis of the (free) $\mathbb{Z}$-module $\operatorname{Im}_{\mathbb{Z}} A := A\mathbb{Z}^n \subset \mathbb{Z}^m$ generated by the colums of A. In Example 2.11, $\operatorname{Im}_{\mathbb{Z}} A = \mathbb{Z}^2$.

- *Kernel*: to determine $\operatorname{Ker}_{\mathbb{Z}} A := \{x \in \mathbb{Z}^n : Ax = 0\}$, write $U = (U_1 \mid U_2)$ where $AU_2 = 0$ and $AU_1 =$ the non-zero columns of the HNF. Then the columns of U_2 form a basis of the module $\operatorname{Ker}_{\mathbb{Z}} A$.

  Proof. Obviously the columns of U_2 belong to $\operatorname{Ker}_{\mathbb{Z}} A$; as they were extracted from a non-singular matrix, they form a basis of the subspace they generate. Conversely, let $X \in \mathbb{Z}^n$ such that $AX = 0$ and let $Y = U^{-1}X \in \mathbb{Z}^n$. Since $HY = 0$, and H is in echelon form, the presence of the r pivots $H_{f(j),j} \neq 0$ for $1 \leqslant j \leqslant r$ leads to $Y_j = 0$ for $j \leqslant r$ and the result follows when writing $X = UY$. □

  In example 2.11, $\operatorname{Ker}_{\mathbb{Z}}(A)$ is generated by ${}^t(10, -7, 1)$.

- *Linear system*: solving $AX = Y$ amounts to solving $HX' = Y$, where $X' = U^{-1}X$; the latter system is "triangular" and easy to solve, then the solutions to the original system are of the form UX'. As usual, once a particular solution X_0 is found, the set of all solutions is $X_0 + \operatorname{Ker}_{\mathbb{Z}} A$.

We will see other applications later (2.1.5).

2.1.3 Smith Normal Form (SNF)

Definition 2.8. A matrix $\begin{pmatrix} 0 & D \end{pmatrix}$ or $\begin{pmatrix} 0 \\ D \end{pmatrix}$ is in *Smith Normal Form* (SNF) if D is diagonal, with diagonal $d_1, \dots, d_n$ such that $d_n \mid \cdots \mid d_1$ in $\mathbb{Z}_{\geqslant 0}$.

Theorem 2.9 (Restatement of elementary divisors theorem). *The $m \times n$ SNF matrices form a system of representatives of*

$$\operatorname{GL}_m(\mathbb{Z}) \backslash M_{m\times n}(\mathbb{Z}) / \operatorname{GL}_n(\mathbb{Z}).$$

N.B. $\mathrm{GL}_m(\mathbb{Z})$ and $\mathrm{GL}_n(\mathbb{Z})$ act by left and right multiplication respectively. The previous formulation is equivalent to the following one: if $A \in M_{m\times n}(\mathbb{Z})$, there exist a unique $S \in M_{m\times n}(\mathbb{Z})$ in SNF and matrices $U \in \mathrm{GL}_n(\mathbb{Z})$, $V \in \mathrm{GL}_m(\mathbb{Z})$ (not necessarily unique) such that $S = VAU$.

This relates quite explicitly to the elementary divisors theorem. In full generality, an abelian group G of finite type is given by a finite presentation (g, R), such that

- $g \in G^n$ is a finite set of n generators of G (i.e. $\langle g\rangle_{\mathbb{Z}} = G$), which we may see as a row vector. For $x \in \mathbb{Z}^n$, we note as in the introduction
$$g \cdot x = \sum_{i=1}^{n} x_i g_i.$$
- $R \in M_{n\times k}(\mathbb{Z})$ represents the module of all relations between the generators:
$$g \cdot x = 0 \text{ for } x \in \mathbb{Z}^n \Leftrightarrow x \in \mathrm{Im}_{\mathbb{Z}} R, \textit{i.e.}\ \exists y \in \mathbb{Z}^k,\ x = Ry.$$
In particular, $G \simeq \mathbb{Z}^n / \mathrm{Im}_{\mathbb{Z}} R$.

The content of the elementary divisors theorem is that we can choose g such that $R = \mathrm{diag}(d_1, \ldots, d_n)$ is an essentially unique diagonal matrix, satisfying the SNF property. Write $URV = S$ with S in SNF and U, V unimodular, then $g' := gU^{-1}$ is a set of suitable new generators.

Proof. Since U is unimodular, g' is a set of generators for G; as well $g'S = gU^{-1}URV = gR \cdot V = 0 \cdot V = 0$ so g' satisfies at least the right relations. Conversely, let $g'Y = 0$ be an arbitrary relation, $Y \in \mathbb{Z}^n$, then $gU^{-1}Y = 0$ hence $U^{-1}Y = RZ$ for some integral Z and $Y = SV^{-1}Z$. Since V is unimodular, $\mathrm{Im}_{\mathbb{Z}} SV^{-1} = \mathrm{Im}_{\mathbb{Z}} S$ and S indeed defines the full set of relations for g'. The matrix S need not be diagonal, but we can remove or add columns of zeros as needed (corresponding to a trivial relation $g' \cdot 0 = 0$). It has n rows; if we delete all columns of 0s, it will have less than n columns (as many as its rank $\leqslant n$). This means that we can complete S to an $n \times n$ diagonal matrix $\mathrm{diag}(d_1, \ldots, d_n)$, still in SNF. □

Finally,
$$G = \bigoplus_{i=1}^{n} (\mathbb{Z}/d_i\mathbb{Z}) \cdot g_i',$$
and we can also delete generators such that $d_i = 1$. We obtain in this case a presentation involving the minimal number of generators, with relations as predicted by the elementary divisors theorem; the d_i are then unique.

2.1.4 Algorithms and Complexity

A good reference for these algorithms is Arne Storjohann's PhD dissertation [22]. See also Cohen [6] which lacks details and complexity estimates but contains a wealth of applications. Here are the input and output of the algorithms:

Input: $A \in M_{m\times n}(\mathbb{Z})$, size $nm \log \|A\|_\infty$.

Output (HNF): $H \in M_{m\times n}(\mathbb{Z})$, $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $AU = H$ in HNF.

Output (SNF): $S \in M_{m\times n}(\mathbb{Z})$, $U \in \mathrm{GL}_n(\mathbb{Z})$, $V \in \mathrm{GL}_m(\mathbb{Z})$ such that $VAU = S$ in SNF.

Below is a simple, but inefficient algorithm for HNF. There exist efficient, more complex, algorithms (Theorem 2.13). Note: to simplify the presentation, we do not produce U at this point.

Algorithm 5. Naive Algorithm for HNF

Input: $A = (A_{i,j}) \in M_{m\times n}(\mathbb{Z})$.

Output: $H \in M_{m\times n}(\mathbb{Z})$ such that $AU = H$ in HNF, for some $U \in \mathrm{GL}_n(\mathbb{Z})$.

```
 1: Set $R \leftarrow 1$
 2: for $i = m, m-1, \ldots, 1$ do    {row $i$}
 3:    for $j = R+1, \ldots, n$ do    {zero $A_{i,j}$ using $A_{i,R}$}
 4:       Let $\delta := \gcd(A_{i,R}, A_{i,j})$ and write
              $\begin{pmatrix} A_{i,R} & A_{i,j} \end{pmatrix} \begin{pmatrix} u & s \\ v & t \end{pmatrix} = \begin{pmatrix} \delta & 0 \end{pmatrix}$,
          using the extended Euclidean algorithm.
 5:       $\begin{pmatrix} A_{*,R} & A_{*,j} \end{pmatrix} \leftarrow \begin{pmatrix} A_{*,R} & A_{*,j} \end{pmatrix} \begin{pmatrix} u & s \\ v & t \end{pmatrix}$    {$A_{*,j}$ : $j^{th}$ column}
 6:    if $A_{i,R} \neq 0$ then
 7:       let $R \leftarrow R+1$
 8: Reset $R \leftarrow 1$    {will increase up to 1+rank of matrix}
 9: for $i = m, m-1, \ldots, 1$ do    {row $i$}
10:    if $A_{i,R} \neq 0$ then    {pivot; if no pivot, do nothing}
11:       Let $A_{*,R} \leftarrow A_{*,R} \times \mathrm{sign}(A_{i,R})$    {ensures that $A_{i,R} > 0$}
12:       for $j = 1, \ldots, R-1$ do
13:          Let $q \leftarrow \lfloor A_{i,j}/A_{i,R} \rfloor$    {$0 \leqslant A_{i,j} - qA_{i,R} < A_{i,R}$}
14:          $A_{*,j} \leftarrow A_{*,j} - qA_{*,R}$
15:       let $R \leftarrow R+1$
```

The algorithm is made of two main **for** loops, one on lines 2 to 7, another on lines 9 to 15. The first loop uses the extended Euclidean algorithm to bring the matrix into left-upper triangular form. The last entries of row i will be $(A_{i,R}\ 0\ \ldots\ 0)$. The second loop reduces the entries to the left of the pivots to bring the matrix into HNF.

Remark 2.10. $AU = H$, where $U = A^{-1}H$ is uniquely determined if A is invertible. To recover U in the general case, apply the algorithm to the matrix $\begin{pmatrix} \mathrm{Id} \\ A \end{pmatrix}$ instead.

Example 2.11. Let $A = \left(\begin{smallmatrix} 2 & 3 & 1 \\ 1 & 2 & 4 \end{smallmatrix}\right)$. The first loop transforms successively A in $\left(\begin{smallmatrix} 2 & 1 & 1 \\ 1 & 0 & 4 \end{smallmatrix}\right)$, then in $\left(\begin{smallmatrix} 2 & 1 & 1 \\ 1 & 0 & 0 \end{smallmatrix}\right)$ and finally in $\left(\begin{smallmatrix} 2 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$, for the successive choices $\left(\begin{smallmatrix} u & s \\ v & t \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 4 \\ 0 & -1 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 0 & 1 \\ 1 & -2 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1 & 7 \\ 0 & -1 \end{smallmatrix}\right)$. The second loop gives the desired HNF : $\left(\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$.

If we instead apply the algorithm to $B = \left(\begin{smallmatrix} \mathrm{Id} \\ A \end{smallmatrix}\right)$ we obtain

$$\begin{pmatrix} -3 & 2 & 10 \\ 2 & -1 & -7 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

from which we deduce

$$\begin{pmatrix} -3 & 2 & 10 \\ 2 & -1 & -7 \\ 0 & 0 & 1 \end{pmatrix}$$

as a possible value for U. The value obtained for U, contrary to what happens with the HNF (which is unique), depends on the matrices $\left(\begin{smallmatrix} u & s \\ v & t \end{smallmatrix}\right)$ chosen in the algorithm. For instance, if instead of $\left(\begin{smallmatrix} 1 & 7 \\ 0 & -1 \end{smallmatrix}\right)$ in the last step of the first loop we use $\left(\begin{smallmatrix} 8 & 7 \\ -1 & -1 \end{smallmatrix}\right)$ — which still ensures $\begin{pmatrix} 1 & 7 \end{pmatrix} \left(\begin{smallmatrix} u & s \\ v & t \end{smallmatrix}\right) = \begin{pmatrix} 1 & 0 \end{pmatrix}$ —, we now obtain

$$U = \begin{pmatrix} -3 & 12 & 10 \\ 2 & -8 & -7 \\ 0 & 1 & 1 \end{pmatrix},$$

another valid answer.

Measuring only algebraic complexity, the first loop requires $O(mnr)$ operations $(+, \times, \text{extended gcd})$ in $\mathbb{Z}$. The second loop requires $O(r^2m)$ operations $(+, \times, \div)$ in $\mathbb{Z}$. Therefore, the overall order is $O(rm(n+r))$. If in addition you would like to recover U, replace m by $(m+n)$ in the previous expression.

A major problem with this algorithm is that the size of entries increases quickly during iterations. Thus, it does *not* run in polynomial time, (input size)$^{O(1)}$. Kannan and Bachem (1979) found an algorithm that works in polynomial time. There are two main ideas:

- Reduce to the special case where 1) A is a square non-singular matrix of rank r; 2) the unimodular transformation matrix U is not needed. Assume that $A \in M_{r\times n}(\mathbb{Z})$ for simplicity (full row rank). Using fraction free Gaussian decomposition, find a permutation matrix P such that the first r columns of AP are independant, and complete this latter matrix with the last $n-r$ rows of the $n \times n$ identity matrix. Then the resulting matrix B is square of dimension n and non-singular. Let H be the HNF reduction of B, computed using the special case; then $U = PB^{-1}H$ is unimodular and AU is the HNF reduction of A.

- In the special case where A is square and non-singular, work modulo $N := |\det A| \neq 0$. This prevents the previous problem of the entries blowing up, because at every step, all the entries belong to a fixed system of representatives of $\mathbb{Z}/N\mathbb{Z}$. This works because the HNF of A and

$$\left(\begin{array}{c|ccc} & N & & 0 \\ A & & \ddots & \\ & 0 & & N \end{array}\right)$$

are identical. One can reconstruct the HNF from this modular computation, see the proof in Cohen [6]; it is a good exercise.

Example 2.12. Prove that SNF can be solved in polynomial time.

1. Prove that we can assume that the input matrix A is square of dimension n and non-singular.

2. Let $H = AU$ be the HNF reduction of A. Let $H' = {}^tHU'$ be the HNF reduction of the *transpose* of H. Then the entry at position $(1, n)$ of ${}^tU'AU$ is the gcd d of all entries in the last row and last column of A. Using obvious line and column operation, this reduces A to the shape $\left(\begin{smallmatrix} A' & 0 \\ 0 & d \end{smallmatrix}\right)$. Iterating, we can now assume that A is diagonal.

3. Let a, b be two integers and $au + bv = d$ a Bezout relation. Write $a = da'$ and $b = db'$; using the matrix identity

$$\begin{pmatrix} b' & -a' \\ u & v \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} 1 - ua' & 1 \\ -ua' & 1 \end{pmatrix} = \begin{pmatrix} ab/d & 0 \\ 0 & d \end{pmatrix},$$

explain how to reduce a diagonal matrix to its SNF reduction.

One can do better than the above naive algorithms:

Theorem 2.13 (Storjohann)**.** *Suppose $A \in M_{m\times n}(\mathbb{Z})$ has rank $r \leqslant \min(m, n)$ and let $B = \log \|A\|_\infty$. The HNF and SNF problems can be solved*

- *in time $\widetilde{O}(mnr^2B)$ and space $\widetilde{O}(\max(m, n)^3B)$.*

- *in time $\widetilde{O}(mnr^2B^3)$ using space $\widetilde{O}((m + n)rB)$.*

The first algorithm is fast, but requires a lot of memory. In the second, memory use is softly linear in the input size, essentially best possible. Actually, SNF *without U and V* can be solved faster than HNF, provided we allow Monte-Carlo method (randomized, with possibly wrong results):

Theorem 2.14 (Eberly, Giesbrecht, Villard)**.** *Let $A \in M_n(\mathbb{Z})$ and let $B = \log \|A\|_\infty$. There exist a probabilistic algorithm running in time $\widetilde{O}(n^{3.5}B^{2.5})$, which computes an SNF matrix that coincides with the SNF reduction of A with probability $\geqslant 1/2$.*

The running times above all use classical $O(n^3)$ matrix multiplication. Using asymptotically fast multiplication (a feasible multiplication exponent $2 \leqslant \omega < 3$) yields further improvements.

2.1.5 Applications

All these are straightforward applications of HNF and SNF.

Exercise 2.15. Let $A \in M_n(\mathbb{Z})$ and $(d_1, \ldots, d_n)$ be the diagonal of its SNF. Then

$$\mathbb{Z}^n / \operatorname{Im}_{\mathbb{Z}} A \simeq \bigoplus_{i=1}^{n} (\mathbb{Z}/d_i\mathbb{Z}).$$

Exercise 2.16. Solve $XA = Y$, where $Y \in M_{\ell\times n}(\mathbb{Z})$, $A \in M_{m\times n}(\mathbb{Z})$, unknown $X \in M_{\ell\times m}(\mathbb{Z})$. Hint: Write $AU = H$, H in HNF.

Exercise 2.17. Solve

$$AX = \begin{pmatrix} y_1 & \pmod{d_1} \\ & \vdots \\ y_n & \pmod{d_n} \end{pmatrix}$$

Hint: $AX = Y + DZ$, D diagonal $\Rightarrow (AI - D)\begin{pmatrix} X \\ Z \end{pmatrix} = Y$.

Exercise 2.18. Let

$$\begin{tikzcd} 0 \arrow[r] & \underset{(g_A,R_A)}{A} \arrow[r, "\phi"] & \underset{(g_B,R_B)}{B} \arrow[r, "\psi"] & \underset{(g_C,R_C)}{C} \arrow[r] & 0 \end{tikzcd}$$

be an exact sequence of abelian groups. Each abelian group G in the sequence is given by a finite presentation (g, R), which is a finite set of generators and relations as explained in §2.1.3 (we may assume that R is square, in SNF). We say that

- we know a group $G = (g, R)$ if we can express any element of G as $g \cdot x$, for some integral column vector x. (This is called the discrete logarithm problem. Do you see why?)

- we know a map $\phi : A \to B$ if for any $a \in A$ we can compute its image $\phi(a) \in B$, and for any $b \in \mathrm{Im}(\phi)$ we can compute an inverse image $a \in A$ such that $\phi(a) = b$.

Prove that if you know the two maps $\phi : A \to B$ and $\psi : B \to C$, as well as 2 out of the 3 groups in the sequence, then you also know the third one.

Proof. We treat just one case in this last exercise: suppose that we know A and B, how to compute C? Explicitly, assume that (g_A, R_A) and (g_B, R_B) are given. We first compute a suitable (g_C, R_C):

- $\psi(g_B) = (\psi(g_1), \dots, \psi(g_{n_B})) =: g_C$ is a set of generators for C.
- Let $\varphi(g_A) = g_B \cdot P$ where P is a known matrix. By definition, $g_C \cdot x = 0$ for some $x \in \mathbb{Z}^{n_B}$ if and only if $\psi(g_B) \cdot x = 0$. This is equivalent to

$$\begin{aligned}
&\psi(g_B \cdot x) = 0 && \text{(because } \psi \text{ is } \mathbb{Z}\text{-linear)}\\
\iff\ & g_B \cdot x \in \mathrm{Im}_{\mathbb{Z}}(\varphi) && \text{(exactness)}\\
\iff\ & g_B \cdot x = \varphi(g_A \cdot y), && \text{with } y \in \mathbb{Z}^{n_A}.\\
\iff\ & g_B \cdot (x - Py) = 0 &&\\
\iff\ & x - Py = R_B \cdot z, && \text{with } z \in \mathbb{Z}^{n_B}.
\end{aligned}$$

Finally, we want all $x \in \mathbb{Z}^{n_B}$ for which there exist y, z such that:

$$x = (P \mid R_B) \begin{pmatrix} y \\ z \end{pmatrix}.$$

Then $R_C = (P \mid R_B)$ is suitable. If needed, we can replace R_C by its Smith Normal Form, and obtain a minimal presentation for C (i.e. with minimal number of generators): let $D = UR_CV$ the SNF of R_C, then $G_C = g_C \cdot U$ are new generators, and (G_C, D) is such a presentation, once we delete the generators corresponding to elementary divisors equal to 1.

We now prove that the discrete logarithm problem can be solved in C. Let $c \in C$, that we must express on the the g_C. Then $c = \psi(b)$ for some in B. Since we can solve the discrete logarithm problem in B, we can write $b = g_B \cdot \beta$, for some $\beta \in \mathbb{Z}^{n_B}$; finally, $c = g_C \cdot \beta$, since $g_C = \psi(g_B)$. □

Exercise 2.19 (Subgroups)**.** Let $B = \mathbb{Z}^n / \mathrm{Im}_{\mathbb{Z}} R$ be a *finite* group, described by some relation matrix R. We can assume that R has rank n: replace it by its SNF and delete trivial generators corresponding to elementary divisors equal to 1 (thereby decreasing n).

1. Prove that the subgroups of B are in bijection with the subgroups $H \leqslant \mathbb{Z}^n$ containing $\mathrm{Im}_{\mathbb{Z}} R$.

2. A canonical basis for the submodule H is given by a square HNF matrix M of dimension n (and maximal rank). Prove that $\operatorname{Im}_{\mathbb{Z}} R \leqslant H$ if and only if $M^{-1}R$ has integer coefficients.

3. Conclude that the subgroups of B are in bijection with the square HNF matrices which are left-divisors of R. Write down explicitly the correspondance.

2.2 Lattices

2.2.1 Definitions and first results

Definition 2.20. A lattice (Λ, q) is a free $\mathbb{Z}$-module Λ of finite rank, together with a positive definite quadratic form q on $\Lambda \otimes_{\mathbb{Z}} \mathbb{R}$.

This definition is the most flexible one, but one can also define a lattice as already embedded in a fixed Euclidean space $E = (\mathbb{R}^n, q)$, where q is a positive definite quadratic form. Then (Λ, q) is a lattice if and only if $\Lambda \subset \mathbb{R}^n$ is a free $\mathbb{Z}$-module of maximal rank n.

Let $x \cdot y = \frac{1}{4}\big(q(x+y) - q(x-y)\big)$ be the scalar product associated to the quadratic form q. To any basis $(b_1, \dots, b_n)$ of E, we associate its Gram-Schmidt orthogonal basis $(b_1^*, \dots, b_n^*)$, defined by

$$b_i^* := b_i - \sum_{1 \leqslant j < i} \mu_{i,j} b_j^*, \quad 1 \leqslant i \leqslant n, \quad \text{where} \quad \mu_{i,j} := \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*}.$$

In particular, $b_1^* = b_1$. The recurrence formula follows from requiring that $b_i^* \cdot b_j^* = 0$ for $j < i$.

Note that if $\Lambda = \langle b_1, \dots, b_n \rangle_{\mathbb{Z}}$ is a lattice, the $(b_i^*)_{i \leqslant n}$ do not lie in Λ in general since the coefficients $\mu_{i,j}$ need not be integers. The $(b_i^*)_{i \leqslant n}$ do however form an orthogonal $\mathbb{R}$-basis for $\mathbb{R}^n$, a priori not orthonormal. More generally, $\langle b_1^*, \dots, b_r^* \rangle_{\mathbb{R}} = \langle b_1, \dots, b_r \rangle_{\mathbb{R}}$ for any $1 \leqslant r \leqslant n$ and b_k^* is orthogonal to this subspace for any $k > r$. This follows from

$$(b_1, \dots, b_r) = (b_1^*, \dots, b_r^*) \begin{pmatrix} 1 & \mu_{1,2} & \cdots & \mu_{1,r} \\ 0 & 1 & \cdots & \mu_{2,r} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}, \tag{2.1}$$

with a non-singular base-change matrix.

Remark 2.21. From the Gram-Schmidt process, we could assume that q is the standard Euclidean form. Namely, we can set

$$\delta_i = \frac{b_i^*}{\sqrt{b_i^* \cdot b_i^*}}$$

to get an *orthonormal* basis $(\delta_1, \dots, \delta_n)$. But for arithmetic applications, it is more flexible to retain the possibility of a general positive form. For instance, if $b_i \cdot b_j \in \mathbb{Z}$ for all i, j, then $\mu_{i,j} \in \mathbb{Q}$ for all i, j (proof by induction).

Let $E = (\mathbb{R}^n, q)$ be a Euclidean space, where $x \cdot x$ is the scalar product, and let Λ be a lattice with basis $(b_1, \dots, b_n)$.

Definition 2.22.

- Let $\mathrm{Gram}(b_1, \dots, b_n) := (b_i \cdot b_j)_{1 \leqslant i,j \leqslant n}$ be the Gram matrix of the b_i.
- The discriminant of Λ is
$$\mathrm{disc}(\Lambda) := \det(\mathrm{Gram}(b_i)).$$
- The determinant of Λ is
$$\mathrm{d}(\Lambda) := \sqrt{\mathrm{disc}(\Lambda)}.$$

Proposition 2.23. *The discriminant* $\mathrm{disc}(\Lambda)$ *is well-defined and is equal to* $\prod_{i=1}^n q(b_i^*)$. *In particular, the latter depends only on the lattice and not on the chosen basis.*

Proof. Consider $(b_1^*, \dots, b_n^*)$ the orthogonal basis of $\mathbb{R}^n$, such that $(b_i^*)A = (b_i)$, with $A \in \mathrm{GL}_n(\mathbb{R})$ an upper triangular matrix with determinant 1 as in (2.1). Then
$$\mathrm{Gram}(b_1, \dots, b_n) = {}^tA\,\mathrm{Gram}(b_1^*, \dots, b_n^*)A.$$
Since $\mathrm{Gram}(b_1^*, \dots, b_n^*)$ is diagonal, taking the determinant we obtain $\mathrm{disc}(\Lambda) = q(b_1^*) \dots q(b_n^*)$. Now any other basis of Λ is of the form $(b_i') = (b_i)U$ for some $U \in \mathrm{GL}_n(\mathbb{Z})$, replacing A by AU in the above. Since $\det U = \pm 1$, it follows that $\mathrm{disc}(\Lambda)$ is well-defined. □

Corollary 2.24 (Hadamard's inequality)**.** *Let* $B \in M_n(\mathbb{R})$ *be the matrix whose columns are some* $b_i \in \mathbb{R}^n$, *and let* $(\mathbb{R}^n, \|\cdot\|^2)$ *be the standard Euclidean space. Then*
$$|\det B| = \prod_{i=1}^n \|b_i^*\| \leqslant \prod_{i=1}^n \|b_i\|.$$

Proof. Let Λ be the lattice generated by the b_i equiped with $q = \|\cdot\|^2$. We have
$$\mathrm{disc}(\Lambda) = \det(\mathrm{Gram}(b_i)) = \det({}^tBB) = \det(B)^2 = \prod_{i=1}^n \|b_i^*\|^2,$$
by the previous Proposition. Since $b_i = b_i^* + \sum_{j<i} \mu_{i,j} b_j^*$ and since $(b_1^*, \dots, b_i^*)$ is orthogonal we have
$$\|b_i\|^2 = \|b_i^*\|^2 + \sum_{j<i} \mu_{i,j}^2 \left\|b_j^*\right\|^2 \geqslant \|b_i^*\|^2$$
and the result follows. □

We are interested in short vectors in lattices. We shall now see that short vectors do exist, where "short" only depends on the dimension and the discriminant of the lattice. But this theorem does not say how to find them.

2.2.2 Minkowski's Theorem

Theorem 2.25 (Minkowski)**.** *Let C be a subset of $\mathbb{R}^n$ such that:*

- *C is symmetric ($C = -C$),*
- *C is convex,*
- $\mathrm{Vol}(C) > 2^n d(\Lambda)$*;*

then there is a non-zero lattice point in C.

In the theorem, $\mathrm{Vol}(C)$ is the volume with respect to the Euclidean volume form, i.e. the Lebesgue measure if q is the standard form. More generally, if A is the base change matrix expressing an orthonormal basis of E in terms of the canonical basis, the volume form is the Lebesgue measure divided by $|\det(A)|$.

Lemma 2.26. *If $\mathrm{Vol}(C) > d(\Lambda)$, then there exists $x, y \in C$, $x \neq y$ such that $x \equiv y \bmod \Lambda$.*

Proof. (of Lemma) Let $(b_i)_{1\leqslant i\leqslant n}$ be a basis of Λ and $\mathcal{F}$ be the fundamental domain for Λ (a complete system of representatives for $\mathbb{R}^n/\Lambda$) given by:

$$\mathcal{F} = \left\{ \sum_{i=1}^{n} \lambda_i b_i \colon 0 \leqslant \lambda_i < 1 \right\}.$$

We have $\mathrm{Vol}(\mathcal{F}) = \mathrm{d}(\Lambda)$. Let us define

$$C_x := (C - x) \cap \mathcal{F}, \quad \text{where} \quad x \in \Lambda.$$

Since $C - x \cap \mathcal{F}$ is $C \cap (\mathcal{F} + x)$ translated, and translations conserve volumes, we have

$$\mathrm{Vol}(C_x) = \mathrm{Vol}(C \cap (\mathcal{F} + x)).$$

By construction the $\mathcal{F} + x$ are disjoint and cover $\mathbb{R}^n$: $\bigcup_{x\in\Lambda}(\mathcal{F} + x) = \mathbb{R}^n$. Now argue by contradiction: assume that the C_x are disjoint (if not, there exist distinct $x, x' \in \Lambda$ such that $c_1 - x = c_2 - x' \Leftrightarrow c_1 \equiv c_2 \pmod{\Lambda}$ which proves the Lemma). Since

$$\mathcal{F} \supset \bigcup_{x\in\Lambda} C_x,$$

we have

$$\begin{aligned}
\mathrm{d}(\Lambda) = \mathrm{Vol}(\mathcal{F}) &\geqslant \sum_{x\in\Lambda} \mathrm{Vol}(C_x), \quad \text{by disjointness,} \\
&= \sum_{x\in\Lambda} \mathrm{Vol}(C \cap (\mathcal{F}+x)) \\
&= \mathrm{Vol}\left(\bigcup_{x\in\Lambda} C \cap (\mathcal{F}+x)\right), \quad \text{by disjointness,} \\
&= \mathrm{Vol}(C \cap \mathbb{R}^n) = \mathrm{Vol}(C)
\end{aligned}$$

This is a contradiction. □

Proof. (Minkowski's theorem). Since

$$\mathrm{Vol}\left(\frac{C}{2}\right) = \frac{\mathrm{Vol}(C)}{2^n} > \mathrm{d}(\Lambda),$$

the Lemma yields distinct $c_1, c_2 \in C$ and $\lambda \in \Lambda \setminus \{0\}$, such that

$$\frac{c_1}{2} = \frac{c_2}{2} + \lambda,$$

hence $\lambda = \frac{1}{2}(c_1 - c_2)$. But C is symmetric so $-c_2 \in C$; by convexity, $\frac{1}{2}(c_1 - c_2) \in C$ and we are done. □

Corollary 2.27 (Minkowski). *Let C be a subset of $\mathbb{R}^n$ such that:*

- *C is compact,*
- *C is symmetric,*
- *C is convex,*
- $\mathrm{Vol}(C) \geqslant 2^n d(\Lambda)$*;*

then there is a non-zero lattice point in C.

Proof. Use Theorem 2.25 with $C_k = (1+1/k)C$, where $k \in \mathbb{Z}_{>0}$: C_k is symmetric, convex, and satisfies $\mathrm{Vol}(C) > 2^n\mathrm{d}(\Lambda)$. Then there exists $x_k \in \Lambda \setminus \{0\}$ such that $x_k \in C_k \subseteq 2C$. Since $2C$ is compact, we can extract from (x_k) a convergent subsequence. On the one hand, its limit x is in every C_k (because the C_k are closed, decreasing and $x_k \in C_k$), hence $x \in C$ (because C is closed). On the other hand, the lattice Λ being discrete, this subsequence is ultimately stationnary and $x \in \Lambda \setminus \{0\}$. □

Corollary 2.28. *Let (Λ, q) be a lattice of rank n. There exists $x \in \Lambda \setminus \{0\}$ such that*

$$q(x) \leqslant \gamma_n \operatorname{disc}(\Lambda)^{\frac{1}{n}},$$

where γ_n only depends on n.

Proof. We may assume that the Euclidean space (E, q) is $(\mathbb{R}^n, \|\cdot\|_2)$ (why?). Let $C = \{x \in \mathbb{R}, \|x\|_2 \leqslant R\}$, which is compact, convex and symmetric. In order to apply Corollary 2.27, we want $\operatorname{Vol} C = \delta_n R^n \geqslant 2^n \mathrm{d}(\Lambda)$, where δ_n is the volume of the unit ball in $\mathbb{R}^n$. We choose

$$R = 2\delta_n^{-1/n} \mathrm{d}(\Lambda)^{1/n}$$

and obtain the requested x. □

The proofs of these Minkowski variants is by contradiction, obviously ineffective.

2.2.3 From algebraic requirements to short vectors

In this section we study an example, to be developped in §2.4, to explain how knowing small vectors can help solving number-theoretic problems. Let $x \in \mathbb{R}$, given by a decimal approximation x_ε:

$$|x - x_\varepsilon| < \varepsilon.$$

We think of the $x_\varepsilon \in \mathbb{Q}$ as a sequence of approximations, which can be made arbitrarily precise. We want to answer the question: is x algebraic? Of course, there is no way to prove this only by knowing $x_\varepsilon \in \mathbb{Q}$ for a given ε, since this rational number is obviously algebraic! But there is a nice way to make good guesses provided ε is sufficiently small compared to the height and degree of x, which measure the "complexity" of an algebraic number.

We fix a positive integer n, and a big real number $C > 0$; think of n as an upper bound for the degree of a minimal polynomial of x over $\mathbb{Q}$. Consider the $(n+2) \times (n+1)$ matrix

$$A = \begin{pmatrix} 1 & \dots & \dots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & \dots & 1 \\ Cx^n & \dots & Cx & C \end{pmatrix}$$

and the $\mathbb{Z}$-module $\subset (\mathbb{R}^{n+2}, \|\cdot\|_2)$ generated by the columns of A, that is the set of

$$\begin{pmatrix} \lambda_n \\ \vdots \\ \lambda_0 \\ C\left(\sum_{i=0}^n \lambda_i x^i\right) \end{pmatrix}, \quad \lambda_i \in \mathbb{Z}.$$

We are interested in short vectors in this lattice, where “short” means small with respect to the Euclidean length. A short vector satisfies:

1. $\sum_{i=0}^{n} \lambda_i^2$ is small;

2. $C^2 \left(\sum_{i=0}^{n} \lambda_i x^i\right)^2$ is small.

If C is large, then we probably have $\sum_{i=0}^{n} \lambda_i x^i \approx 0$ for a short vector, so we let

$$P(X) = \sum_{i=0}^{n} \lambda_i X^i.$$

If x is algebraic of degree $\leqslant n$ then we can hope that $P(x) = 0$. We shall see how to *guarantee* this in §2.4. The point of the first condition is that if we allow λ_i to be arbitrarily large, the pigeonhole principle says there are many good approximations satisfying the second one, possibly bearing no relation to the minimal polynomial we look for. Assuming x is decent, λ_i should be relatively small.

Many problems can be thus translated into short vectors problem. Sometimes we want the shortest vector, sometimes we are happy a family of relatively short ones. Consider the set

$$S(c) = \{x \in \mathbb{R}^n \colon q(x) < c\}.$$

This is an ellipsoid, quite easy to describe geometrically (essentially it is a ball). Since Λ is discrete, $S(c) \cap \Lambda$ is finite for any c. This leads to an easy but inefficient algorithm to find short vectors, by enumerating all vectors with bounded length. In fact, if we call $N(c)$ the number of points of Λ contained in $S(c)$, then $N(c)$ is roughly proportional to c^n, so enumerating all the lattice points in $S(c)$ requires exponential time in the dimension n.

A second problem is that we do not know c in advance. So we use the following workaround:

1. start with an arbitrary c, e.g. $c = 1$;

2. if $S(c) \cap \Lambda$ is empty, replace c by $2c$, and iterate.

This requires $O(\log B)$ iterations if B is the final bound. Minkowski’s Theorem 2.25 tells us that the length B of the shortest (non-zero) vector is nicely bounded in terms of the size of the input, so this second problem is not very serious. But the first one is!

2.3 The LLL algorithm

Definition 2.29. We let $\lceil x \rfloor :=$ the nearest integer to $x = \lfloor x + \frac{1}{2} \rfloor$, then

$$a \operatorname{cmod} b := a - \left\lceil \frac{a}{b} \right\rfloor b.$$

(The letter c is for *center.*) We have $-\frac{|b|}{2} \leqslant a \operatorname{cmod} b < \frac{|b|}{2}$.

2.3.1 Introduction : towards an effective Minkowski ?

Our goal is to find a basis of short vectors for Λ, i.e. with $q(b_i)$ small. We will eventually achieve this with the LLL (or L^3) algorithm, invented by Arjen Lenstra, Hendrik Lenstra Jr. and László Lovász, in a landmark 1982 paper [16], with the intended application of factoring polynomials in $\mathbb{Q}[X]$.

The first idea is to generalize the Euclidean algorithm, which yields the optimal solution in dimension 1 (the gcd of the inputs). The latter iterates reductions ($a \leftarrow a \mod b$) and swaps ($a \leftrightarrow b$); we shall mimic this.

We want to define the meaning of "reduce b_i mod $\langle b_1 \dots b_{i-1} \rangle_{\mathbb{Z}}$". The idea is to reduce the size of b_i by removing a linear combination of the $(b_1, \dots, b_{i-1})$. Which one ?

Over $\mathbb{R}$. Let us first review the classical procedure over $\mathbb{R}$. The meaning of "reduce b_i mod $\langle b_1 \dots b_{i-1} \rangle_{\mathbb{R}}$" is clear: project on the orthogonal complement of $\langle b_1, \dots, b_{i-1} \rangle_{\mathbb{R}}$, i.e.

$$b_i \leftarrow b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*.$$

To generalize this, it is helpful to view this formula as a sequence of elementary operations: for all $j = i-1, \dots, 1$, let successively

$$b_i \leftarrow b_i - \mu_{i,j} b_j, \quad \text{where } \mu_{i,j} = \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*},$$

computed using the current value of b_i.

- In this algorithm, the $\mu_{i,j}$ change as b_i does; but the b_j^* do not, for $j < i$.
- We prove by induction that at the beginning of step j, the vector b_i is orthogonal to $b_{j+1}^*, \dots, b_{i-1}^*$; indeed the vector $b = b_i - \mu_{i,j} b_j$ satisfies
  - $b \cdot b_{j+1}^* = \cdots = b \cdot b_{i-1}^* = 0$ because all these vectors are orthogonal to b_i by the induction hypothesis, and to the subspace generated by $(b_1, \dots, b_j)$, to which b_j belongs.
  - $b \cdot b_j^* = 0$ by construction, since $b_j \cdot b_j^* = b_j^* \cdot b_j^*$.

- It follows that the final value of b_i lands in the orthogonal subspace to $H = \langle b_1 \dots b_{i-1} \rangle_{\mathbb{R}}$. Since it is equal to the original value of b_i plus an element of H, it is in fact the orthogonal projection of b_i on $H^{\perp}$. Hence, this final value of b_i is the same as before!

Over $\mathbb{Z}$. We approximate the above reduction process: for all $j = i-1, \dots, 1$, let successively

$$b_i \leftarrow b_i - \lceil \mu_{i,j} \rfloor \, b_j, \quad \text{where } \mu_{i,j} = \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*},$$

computed using the current value of b_i. Because of our rounding $\mu_{i,j}$, we obtain some approximation of the true orthogonal projection. Note that the above is a sequence of elementary operations over $\mathbb{Z}$, hence invertible.

Lemma 2.30. *Let b_i the resulting vector. It satisfies the following:*

$$|\mu_{i,j}| \leqslant \frac{1}{2}, \quad \textit{for all } j < i.$$

Proof. Decreasing induction. Let $|\mu_{i,j}| \leqslant \frac{1}{2}$ for all j such that $\ell < j < i$. The transformation $b_i \leftarrow b_i - \lceil \mu_{i\ell} \rfloor \, b_\ell$ has a nice behaviour: it does not affect the $\mu_{i,j}$ for $j > \ell$ and replaces $\mu_{i,\ell}$ by $\mu_{i,\ell} \operatorname{cmod} 1$. The result follows. □

A vector b_i satisfying the condition in the Lemma is called *size-reduced*, with respect to the family $(b_1, \dots, b_{i-1})$.

Corollary 2.31. *Given any $\mathbb{Z}$-basis of Λ we may change it into a size-reduced basis such that all the $|\mu_{i,j}| \leqslant \frac{1}{2}$ for every $j < i$. To size-reduce a basis requires $O(n^2)$ operations on vectors in Λ, hence $O(n^3)$ operations on scalar coordinates.*

Remark 2.32. Since the lattice does not change, $\prod_{i=1}^n q(b_i^*)$ remains fixed. Further

- $q(b_1^*) = q(b_1)$, and more generally $q(b_i^*) \leqslant q(b_i)$.
- $q(b_i) = q(b_i^*) + \sum_{j<i} \mu_{i,j}^2 q(b_j^*)$, since the b_j^* are an orthogonal family.
- $\mu_{i,j}^2 \leqslant \frac{1}{4}$ so the b_i are short provided the $q(b_j^*)$ are.

Problem

If all $q(b_j^*)$ are small, so is $q(b_i)$ by the above and we are happy. But what occurs if $q(b_j^*)$ is very small for some j ? Since the product $\prod_{i=1}^n q(b_i^*) = \operatorname{disc}(\Lambda)$ is constant, then some other Gram-Schmidt vector b_j^* is large, so at least one vector in the basis is large, since $q(b_j) \geqslant q(b_j^*)$.

So we want to avoid “tiny” Gram-Schmidt vectors. Ideally, we want the $q(b_i^*)$ to be roughly equal, of the order of $\operatorname{disc}(\Lambda)^{\frac{1}{n}}$ since their product is $\operatorname{disc}(\Lambda)$. We will swap vectors between reductions in order to ensure this.

2.3.2 Reduced bases

We are now ready to give definitions.

Definition 2.33. The basis $(b_1, \ldots, b_n)$ is *size reduced* if $|\mu_{i,j}| \leqslant \frac{1}{2}$ for every $j < i$.

Definition 2.34. $(b_i)_{i \leqslant n}$ is *Siegel reduced* if $q(b_i^*) \leqslant 2q(b_{i+1}^*)$ for $i < n$.

For such a basis, we have

$$q(b_1) = q(b_1^*) \leqslant 2q(b_2^*) \leqslant 2^2 q(b_3^*) \leqslant \ldots \leqslant 2^{n-1} q(b_n^*), \tag{2.2}$$

hence

$$q(b_1)^n \leqslant \prod q(b_i^*) \prod_{i<n} 2^i = \operatorname{disc}(\Lambda) \times 2^{n(n-1)/2}. \tag{2.3}$$

(Compare with Minkowski and Corollary 2.28.)

Definition 2.35. $(b_i)_{i \leqslant n}$ is called *reduced* if it is size reduced *and* Siegel reduced.

In a Siegel reduced basis, the values $q(b_i^*)$ cannot decrease too fast, hence a small $q(b_i^*)$ means that *all* $q(b_j^*)$ are small for $j \leqslant i$. If the basis is further size-reduced, all b_i are small in that same range (Remark 2.32) and we are happy. From the Siegel condition, if a $q(b_i^*)$ is large (so that $q(b_i)$ is large), then all following ones are large also. Since the product is controlled, no really large vector can occur.

The LLL algorithm produces a reduced basis from an arbitrary basis. Before giving the algorithm, we describe the nice properties of a reduced basis:

Theorem 2.36. *Let $x \in \Lambda$, $x \neq 0$, and let $(b_i)_{i \leqslant n}$ be a reduced basis for Λ. Then the following three properties hold:*

1. $q(x) \geqslant \min_{i \leqslant n} q(b_i^*)$ *(this one actually holds for any basis)*
2. $q(b_1) \leqslant 2^{n-1} q(x)$ *(in other words, b_1 is essentially as short as possible)*
3. *(Generalization of the previous case.) Let $(x_1, \ldots, x_t)$ be t independent vectors in Λ; in particular, $x_i \neq 0$ and $t \leqslant n$. Then*

$$q(b_t) \leqslant 2^{n-1} \max_{j \leqslant t} q(x_j).$$

Remark 2.37. The second property gives in general a much better bound than that given by Minkowski's theorem or (2.3), since here the bound for $q(b_1)$ depends on the shortest x for a specific lattice, while Minkowski runs through all possible lattices with a given discriminant. Recall that according to Corollary 2.28, there is a non-zero vector x in Λ such that $q(x) \leqslant \gamma_n \operatorname{disc}(\Lambda)^{\frac{1}{n}}$, for some γ_n only depending on n.

Proof. 1. Write $x = \sum_{i \leqslant n} \lambda_i b_i$, where $\lambda_i \in \mathbb{Z}$ and not all are 0 since $x \neq 0$. Let k be the maximal index such that $\lambda_k \neq 0$. Then

$$x = \sum_{i \leqslant k} \lambda_i (b_i^* + \sum_{j<i} \mu_{i,j} b_j^*) = \lambda_k b_k^* + \sum_{j<k} \nu_j b_j^* , \quad \nu_j \in \mathbb{R}.$$

Using the fact that $(b_1^*, \ldots, b_k^*)$ are orthogonal and $|\lambda_k| \geqslant 1$:

$$q(x) = \lambda_k^2 q(b_k^*) + \sum_{j<k} \nu_j^2 q(b_j^*) \geqslant q(b_k^*) \geqslant \min_{j \leqslant n} q(b_j^*).$$

2. $q(b_1) = q(b_1^*) \leqslant 2^{k-1} q(b_k^*) \leqslant 2^{k-1} q(x) \leqslant 2^{n-1} q(x)$.

3. (Thus far we have not made use of the assumption that the given basis is size reduced, only Siegel reduced. In practice however, decent algorithms giving a Siegel reduced basis yield a basis which is size reduced too.) We start by generalizing (2.2), noting that Siegel reducedness implies that for $j < i$:

$$q(b_j^*) \leqslant 2q(b_{j+1}^*) \leqslant \ldots \leqslant 2^{i-j} q(b_i^*).$$

Next we prove the following inequality between vectors in the reduced basis and corresponding vectors in the orthogonal basis:

Lemma 2.38. *It holds that*

$$1 \leqslant \frac{q(b_i)}{q(b_i^*)} \leqslant 2^{i-1}$$

Proof. From

$$b_i = b_i^* + \sum_{j<i} \mu_{i,j} b_j^*,$$

it follows that

$$q(b_i) = q(b_i^*) + \sum_{j<i} \mu_{i,j}^2 q(b_j^*),$$

hence

$$\frac{q(b_i)}{q(b_i^*)} = 1 + \sum_{j<i} \mu_{i,j}^2 \frac{q(b_j^*)}{q(b_i^*)}.$$

Note that the last term is non-negative. Now use $|\mu_{i,j}| \leqslant \frac{1}{2}$:

$$1 + \sum_{j<i} \mu_{i,j}^2 \frac{q(b_j^*)}{q(b_i^*)} \leqslant 1 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j} = 2^{i-2} + 2^{-1} \leqslant 2^{i-2} + 2^{i-2} = 2^{i-1}.$$

□

Now we prove our contention. Write the x_j in terms of the basis:

$$x_j = \sum_{i=1}^{n} r_{i,j} b_i \,, \quad r_{i,j} \in \mathbb{Z}.$$

For a fixed j, let $i(j)$ be the largest index i such that $r_{i,j} \neq 0$. Then by the proof of the first point (actually the second-last step) with $x = x_j$:

$$q(x_j) \geqslant q(b^*_{i(j)})$$

By renumbering the x_j, we may assume that

$$i(1) \leqslant i(2) \leqslant \ldots \leqslant i(t)$$

We proceed to prove by induction that $j \leqslant i(j)$. Firstly $1 \leqslant i(1)$ since $x_1 \neq 0$. Now suppose $j-1 \leqslant i(j-1)$. Since $i(j-1) \leqslant i(j)$ we have $j-1 \leqslant i(j)$. But $j-1 = i(j)$ would imply that $\{x_1, \ldots, x_j\}$ is contained in the subspace $\langle b_1, \ldots, b_{j-1} \rangle$ spanned by the first $j-1$ basis vectors. This contradicts the assumption that the x_i are linearly independent. Hence $j-1 < i(j)$, or $j \leqslant i(j)$. Now combine the various little results:

$$q(b_j) \leqslant 2^{j-1} q(b^*_j) \leqslant 2^{j-1} 2^{i(j)-j} q(b^*_{i(j)}) = 2^{i(j)-1} q(b^*_{i(j)}) \leqslant 2^{n-1} q(x_j),$$

which completes the proof.

□

2.3.3 The algorithm

Algorithm 6. LLL algorithm

Input: (b_i) a $\mathbb{Z}$-basis for $\Lambda \subset \mathbb{R}^n$. We assume that the b_i are in $\mathbb{Z}^n$, and $\mathrm{Gram}(b_i) \in M_n(\mathbb{Z})$. (Hence the $\mu_{i,j}$ are in $\mathbb{Q}$.)

Output: A reduced basis for Λ.

```
let $k := 2$, compute the $(b_i^*)_{i \leqslant n}$    {i.e. compute $(\mu_{i,j})_{j<i\leqslant n}$.}
while $(k \leqslant n)$ do    {$(b_1, \ldots, b_k)$ is reduced}
  Reduce $b_k$ mod $(b_1, \ldots, b_{k-1})$, update $(b_i^*)_{i \leqslant n}$    {Reduction step}
  if $(k > 1)$ and $q(b^*_{k-1}) > 2q(b^*_k)$ then    {Swap if Siegel condition fails}
    Exchange $b_{k-1}$ and $b_k$
    Update $(b_i^*)_{i \leqslant n}$
    Set $k := k - 1$
  else
    Set $k := k + 1$
return $(b_1, \ldots, b_n)$
```

Remark 2.39. It is not necessary to recompute the (b_i^*) from scratch each time a change is made: most do not change, and the others can be cheaply updated. See Cohen [6] for formulas. There are many variants on Siegel reducedness, simpler to check and a little harder to explain, but yielding essentially the same bounds.

Remark 2.40. Since $(b_1, \dots, b_n)$ is a basis, we can get the base change matrix going from the input basis to the output basis just from the input and output of the algorithm — this is easy linear algebra. What happens if the (b_i) are dependent vectors ? If the reduction of b_k mod $(b_1, \dots, b_{k-1})$ is 0 we discard b_k, and everything else works. But in that case, from input/output, we lose some information: the kernel. If that is a problem, one can add more steps updating an auxiliary matrix (initially Id_n) to keep track of elementary operations, as usual.

Remark 2.41. We assume the b_i and $\mathrm{Gram}(b_i)$ are integral because we want exact arithmetic throughout. Using floating point arithmetic complicates quite a bit the analysis — we need perturbation results —, although it certainly improves efficiency in practice. This is not a serious theoretical restriction since we may approximate all entries by rationals then clear denominators.

Theorem 2.42. *Let $A = \max_{i \leqslant n} q(b_i)$. The algorithm stops after $O(n^4 \log A)$ operations on integers of size $O(n \log A)$. Thus it takes $\widetilde{O}(n^5 (\log A)^2)$ time and requires $O(n^3 \log A)$ space.*

Remark 2.43. The final sentence is obvious since there are n^2 numbers of size $O(n \log A)$ each. In comparison, the advanced HNF algorithm uses $\widetilde{O}(n^4 (\log A)^3)$ time and $O(n^2 \log A)$ space. The LLL algorithm has the advantage however that it is easier to implement (coefficients do not blow up in a naive implantation), gives nicer base change matrices (smaller entries), and nicer output (bases whose vectors are provably small).

Proof. First step. Prove termination.

1. Some definitions: Let
$$\Lambda_i = \langle b_1, \dots, b_i \rangle_{\mathbb{Z}}$$
denote the lattice in $\Lambda_i \otimes_{\mathbb{Z}} \mathbb{R}$ generated by the first i basis vectors. So $\Lambda_n = \Lambda$. Then define
$$D_i = \mathrm{disc}(\Lambda_i) = \prod_{j \leqslant i} q(b_j^*)$$
$$D := \prod_{i=1}^{n-1} D_i = q(b_1^*)^{n-1} \dots q(b_{n-1}^*).$$
Then $D_i \in \mathbb{Z} \setminus \{0\}$, $i \leqslant n-1$, and hence $D \in \mathbb{Z}_{\geqslant 1}$. The idea is now to bound the maximum value that D can have at the beginning, and show

that D only changes during a swap, becoming less than $\frac{3}{4}D$ every time that it changes. This will bound the maximum number of swap steps, hence the number of iterations in the while loop. This proves termination, and will eventually give a bound on the running time of the algorithm, once we make sure coefficient explosion does not occur.

2. We now consider how D changes through the algorithm.

   Reduction of $b_k \mod (b_1, \ldots, b_{k-1})_{\mathbb{Z}}$ does not change the b_i^*, hence it leaves the D_i and D fixed.

   A swap step replaces (b_{k-1}^*, b_k^*) by (s, t), where
$$s = b_k^* + \mu_{k,k-1} b_{k-1}^*$$
   since it is the component of b_k orthogonal to $b_1, \ldots, b_{k-2}$. In particular it follows that $q(s) \geqslant q(b_k^*)$, which will be useful later. Since the product over all $q(b_i^*)$ stays constant, and only two of them change, we have
$$q(s)q(t) = q(b_{k-1}^*)q(b_k^*),$$
   from which we deduce $q(t) \leqslant q(b_{k-1}^*)$. (Here is another way to see this: t is the component of b_{k-1} orthogonal to $b_1, \ldots, b_{k-2}, b_k$, which spans a subspace containing $b_1, \ldots, b_{k-2}$. But b_{k-1}^* is defined as the component of b_{k-1} orthogonal to this latter subspace, hence $q(t) \leqslant q(b_{k-1}^*)$.)

   Summing up, the D_j are unaffected except D_{k-1}, which gets multiplied by $q(s)/q(b_{k-1}^*)$. From the expression for s we get
$$\frac{q(s)}{q(b_{k-1}^*)} = \frac{q(b_k^*)}{q(b_{k-1}^*)} + \mu_{k,k-1}^2 \leqslant \frac{1}{2} + \frac{1}{4} = \frac{3}{4},$$
   using the fact that we are swapping precisely because the Siegel condition was not satisfied at (b_{k-1}^*, b_k^*), and the size reduction consequence: $|\mu_{i,j}| \leqslant 1/2$.

   This proves that during each swap D gets replaced by an integer which is less than $\frac{3}{4}D$.

3. From the definitions of A and D it follows that at the beginning of the algorithm
$$D \leqslant q(b_1)^{n-1} \ldots q(b_{n-1}) \leqslant A^{\frac{n(n-1)}{2}}.$$
   By the above it follows that the number of swaps is bounded by
$$\log_{\frac{3}{4}} A^{\frac{n(n-1)}{2}} = O(n^2 \log A).$$
   Thus there are $O(n^2 \log A)$ loops, and since each loop involves $O(n^2)$ operations on the coordinates of the b_i and $\mu_{i,j}$, the total number of operations is $O(n^4 \log A)$, as claimed.

Second step. Bound the denominators.

Lemma 2.44. *At any point in the algorithm the following are true:*

1. $D_{k-1}b_k^* \in \mathbb{Z}^n$

2. $D_\ell \mu_{k,\ell} \in \mathbb{Z}$, *for all* $\ell < k \leqslant n$.

Proof. 1. We can express

$$b_k^* = b_k - \sum_{\ell<k} \lambda_{k,\ell} b_\ell$$

for certain $\lambda_{k,\ell} \in \mathbb{R}$. Recall that $b_k^* \cdot b_j = 0$ for $j < k$. Taking the inner product of every term in the above equation with b_j for j varying between 1 and $k-1$ gives the following $k-1$ linear equations in $k-1$ variables:

$$\sum_{\ell<k} \lambda_{k,\ell}(b_\ell \cdot b_j) = (b_k \cdot b_j)\ , \quad j < k.$$

The determinant of the linear system is D_{k-1}. Using Cramer's rule, we deduce that $D_{k-1}\lambda_{k,\ell} \in \mathbb{Z}$, as claimed.

2. Using the definitions and what was just proved gives the following:

$$D_\ell \mu_{k,\ell} = D_{\ell-1} q(b_\ell^*) \frac{b_k \cdot b_\ell^*}{q(b_\ell^*)} = b_k \cdot (D_{\ell-1} b_\ell^*) \in \mathbb{Z}$$

□

Thus bounding the D_i will bound all denominators. Recall that A was chosen such that $q(b_i)^* \leqslant q(b_i) \leqslant A$ for all i, at the beginning of the algorithm. Now we claim that $\max_i q(b_i^*)$ never increases, i.e. it is bounded throughout by A. This has essentially already been shown, since these values do not change except at the swap step, where both $q(s)$ and $q(t)$ were shown to be less than $q(b_{k-1}^*) \leqslant A$ by induction. Thus:

$$D_i = \prod_{j \leqslant i} q(b_i^*) \leqslant A^i.$$

Hence all the D_i are bounded by A^n. Then it follows from the previous step that the denominators of the b_k^* and $\mu_{k,l}$ are bounded by A^n, and since the b_i are anyway integers, this bounds all denominators of rational numbers in the algorithm by $O(n \log A)$.

Third step. Bound the absolute values of the entries. Using the previous step, this will bound the absolute values of the numerators.

1. We claim that

$$|\mu_{i,j}|^2 \leqslant D_{j-1} q(b_i).$$

This follows from the definitions and the Cauchy-Schwartz inequality:

$$|\mu_{i,j}|^2 = \left(\frac{b_i \cdot b_j^*}{q(b_j^*)}\right)^2 \leqslant \frac{q(b_i)}{q(b_j^*)}$$
$$q(b_j^*) = \frac{D_j}{D_{j-1}} \geqslant \frac{1}{D_{j-1}}$$

2. Next we need to bound the $q(b_i)$. The idea is to show that $q(b_i) \leqslant nA$ everywhere except possibly during the reduction step where b_i is being reduced, where the weaker bound $q(b_i) \leqslant n^2(4A)^{n+1}$ still holds. At the beginning $q(b_i) \leqslant A \leqslant nA$ by definition. The only place in the algorithm where $q(b_i)$ changes is at the reduction step when b_i is being reduced. To bound $q(b_i)$ during this step and at the end, we use the bound $q(b_i^*) \leqslant A$ and the expression

$$b_i = b_i^* + \mu_{i,i-1}b_{i-1}^* + \cdots + \mu_{i,1}b_1^*.$$

Letting $\mu_{i,i} = 1$ and using orthogonality gives:

$$q(b_i) = \sum_{j\leqslant i} |\mu_{i,j}|^2 q(b_j) \leqslant \sum_{j\leqslant i} m_i^2 A \leqslant nm_i^2 A,$$

where

$$m_i = \max\left\{|\mu_{i,j}| : 1 \leqslant j \leqslant i\right\}.$$

At the end of the reduction step $m_i = 1$ since $|\mu_{i,j}| \leqslant \frac{1}{2}$ for $j < i$ and 1 for $j = i$. Thus the bound $q(b_i) \leqslant nA$ always holds outside the reduction step.

To bound $q(b_i)$ during the reduction step,we need a bound for the m_i which holds throughout the reduction step (we fix some i from now on). We give a bound for m_i which holds at the beginning of the step, and then show that m_i does not grow too much during the reduction step. So at the beginning of the reduction step:

$$\begin{aligned} m_i^2 &= \max_i \left\{|\mu_{i,j}|^2 \colon 1 \leqslant j \leqslant i\right\} \\ &\leqslant \max_i \left\{D_{j-1}q(b_i) \colon 1 \leqslant j \leqslant i\right\} \\ &\leqslant A^{n-1}q(b_i) \leqslant A^{n-1}nA = nA^n \end{aligned}$$

Now consider how m_i changes during one step in the loop of the reduction step for some $1 \leqslant j < i$, i.e. when b_i is being replaced by $b_i - \lfloor \mu_{i,j} \rceil b_j$. As mentioned in the beginning of the lecture, after this step the new values of $\mu_{i,l}$ for $j \leqslant l < i$ are less than $\frac{1}{2}$ in absolute value. Thus they will not have an incremental effect on the new value of m_i. For $1 \leqslant l < j$ the following holds for the new value of $\mu_{i,l}$ (which is given by the first expression):

$$\begin{aligned}\frac{(b_i - \lfloor \mu_{i,j} \rceil b_j) \cdot b_l^*}{b_l^* \cdot b_l^*} &= |\mu_{i,l} - \lfloor \mu_{i,j} \rceil \mu_{j,l}| \\ &\leqslant |\mu_{i,l}| + |\lfloor \mu_{i,j} \rceil| \cdot |\mu_{j,l}| \\ &\leqslant m_i + (m_i + \frac{1}{2})\frac{1}{2} = \frac{3}{2} m_i + \frac{1}{4} \leqslant 2m_i\end{aligned}$$

Thus the new value of m_i cannot be more than twice the old value. So during the whole loop, m_i increases by at most a factor $2^{i-1} \leqslant 2^{n-1}$, so m_i^2 increases by at most a factor 2^{2n-2}. Thus it is always the case that $m_i^2 \leqslant nA^n 2^{2n-2} \leqslant n(4A)^n$. This gives

$$q(b_i) \leqslant n m_i^2 A \leqslant n^2 (4A)^n A \leqslant n^2 (4A)^{n+1}.$$

Fourth step. It remains to bound the absolute values of the numerators of the numbers occurring in the algorithm:

1. $||b_k|| = q(b_k)^{\frac{1}{2}} \leqslant n(4A)^{\frac{n+1}{2}}$
2. $||D_{k-1} b_k^*|| \leqslant A^n A^{\frac{1}{2}}$
3. $|D_l \mu_{k,l}| \leqslant A^n (D_{l-1})^{\frac{1}{2}} ||b_k|| \leqslant A^n A^{\frac{n}{2}} n(4A)^{\frac{n+1}{2}} \leqslant n(4A)^{2n+\frac{1}{2}}$

Thus the numerators all have length $O(n \log A)$ also, completing the proof. □

Here is a theorem on a variation of the LLL-algorithm, using floating point arithmetic:

Theorem 2.45 (NGuyen-Stehlé). *Let $\Lambda \subset \mathbb{Z}^d$ be a lattice given by a generating family of n vectors. Then a reduced basis can be obtained in time*

$$O(d^4 n (d + \log A) \log A).$$

When $n = d$ this becomes $O(n^6 \log^2 A)$.

2.4 Algebraicity test

We come back to our example in §2.2.3, as a nice application of the LLL algorithm, and as a motivation for our factorization efforts in $\mathbb{C}[X]$ in §3.3.

Theorem 2.46. *Let $z \in \mathbb{C}$. Assume*

1. *There exists $P \in \mathbb{Z}[X]$ irreducible, with $P(z) = 0$, $\deg P \leqslant n$, and $\|P\|_\infty \leqslant A$.*

2. *We can compute $\hat{z}(\varepsilon) \in \mathbb{Q}(i)$ such that $|z - \hat{z}(\varepsilon)| < \varepsilon$, for all $\varepsilon > 0$ with* $\log(1/\varepsilon) = O(n^2 \log A)$.

Then we can find such a P in deterministic polynomial time (actually, polynomial in $n \log A$).

Corollary 2.47. *Assume we can factor $P \in \mathbb{Z}[X]$ in $\mathbb{C}[X]$ in the above sense (approximate roots). Then we can factor it in $\mathbb{Q}[X]$ in polynomial time.*

Proof. Pick a complex root z of P, actually the approximation $\hat{z}$, and find its minimal polynomial by looking at the polynomials D with properly bounded size (i.e. $\|D\|_\infty \leqslant A$, A such that all divisors of P in $\mathbb{C}[X]$ satisfy $\|D\|_\infty \leqslant A$, see Landau's bound) and $\deg D < \deg P$. Either we get a factor, or we obtain a proof that P is irreducible. □

Proof. We now prove the Theorem: Let Λ be the free $\mathbb{Z}$-module generated by the columns of the following matrix

$$\begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ C\operatorname{Re}(\hat{z}^n) & \dots & C\operatorname{Re}(\hat{z}^0) \\ C\operatorname{Im}(\hat{z}^n) & \dots & C\operatorname{Im}(\hat{z}^0) \end{pmatrix},$$

where $C > 1$ is a large integer. In other words, the set of

$$\begin{pmatrix} \lambda_n \\ \vdots \\ \lambda_0 \\ C\operatorname{Re}(\sum \lambda_i \hat{z}^i) \\ C\operatorname{Im}(\sum \lambda_i \hat{z}^i) \end{pmatrix}, \quad \lambda_i \in \mathbb{Z}.$$

Let $v = (\lambda_n, \dots, \lambda_0, *, *) \in \Lambda$ be the first vector in a reduced basis, and let $Q(X) = \sum_{i=0}^n \lambda_i X^i$. We claim that $Q(z) = 0$, provided C was large enough! To prove this, we need three lemmas:

Lemma 2.48 (Cauchy's bound)**.** *Let $z \in \mathbb{C}$, $P = \sum_{i=0}^n \lambda_i X^i \in \mathbb{C}[X]$ such that $P(z) = 0$, $a_n \neq 0$. Then*

$$|z| \leqslant 2 \max_{0 \leqslant i < n} \left| \frac{a_i}{a_n} \right|^{1/(n-i)}.$$

Proof. Assume by contradiction that

$$|z|^{n-i} > 2^{n-i} \left| \frac{a_i}{a_n} \right|, \quad \text{for all } i < n.$$

Then $2^{i-n}\left|a_n\right|\left|z\right|^n > \left|a_i z^i\right|$ and

$$\left|a_n z^n\right| > \left|a_n z^n\right| \sum_{i<n} 2^{i-n} > \sum_{i<n} \left|a_i z^i\right| .$$

Hence $P(z) \neq 0$. □

Lemma 2.49. *Let $P \in \mathbb{Z}[X]$, irreducible over $\mathbb{Q}[X]$, $\deg P \leqslant n$, $\|P\|_\infty \leqslant A$, and $z \in \mathbb{C}$ a root of P. Then for all $Q \in \mathbb{Z}[X]$ such that $\deg Q \leqslant n$ and $\|Q\|_\infty \leqslant A$, we have either $Q(z) = 0$ or $|Q(z)| > \eta(A, n) > 0$, where $\log(1/\eta) = O(n^2 \log A)$.*

Proof. We may write $P = a_n \prod_{i=1}^n (X - z_i)$, $z_i \in \mathbb{C}$, with $z_1 = z$ say. From Lemma 2.48, $|z_i| < D = 2A$ and we may assume $D \geqslant 1$. Now consider

$$R := a_n^{\deg Q} \prod_{i=1}^n Q(z_i) = \operatorname{Res}(P, Q) \in \mathbb{Z}.$$

Recall that $\operatorname{Res}(P,Q) = 0$ if and only if P and Q have a common root in $\mathbb{C}$, which implies $P \mid Q$ since P is irreducible. Assume that $Q(z) \neq 0$, then P and Q have no common root and $|\operatorname{Res}(P,Q)| \geqslant 1$.

On the other hand, we bound $|Q(z_i)| < D^n A(n+1)$ for all $i > 1$, $|a_n|$ by A, and obtain

$$1 \leqslant |R| \leqslant Q(z) A^n \left(D^n A(n+1)\right)^{n-1}.$$

□

Lemma 2.50. *Let $z, \hat{z} \in \mathbb{C}$ such that $|z - \hat{z}| < \varepsilon < 1$, $|z| \leqslant A$, and let $P \in \mathbb{C}[X]$, $\deg P \leqslant n$, $\|P\|_\infty \leqslant A$. Then $|P(z) - P(\hat{z})| < \varepsilon B(A, n)$, where $\log B = O(n \log A)$.*

Proof. From the Mean Value Theorem, $P(z) - P(\hat{z}) = (z - \hat{z}) P'(\xi)$, $\xi = z + t(\hat{z} - z)$, for some $t \in [0,1]$. Hence $|\xi| \leqslant A + \varepsilon \leqslant A + 1$ and $|P'(\xi)| \leqslant n^2 A(A+1)^{n-1}$. □

We continue with the proof of theorem. Recall that C and ε are not fixed, we will choose them in order to get a contradiction.

- Assume by contradiction that $Q(z) \neq 0$. then $C\left|Q(z)\right| > C\eta(A,n)$ by Lemma 2.49. Hence, $C\left|Q(\hat{z}\right| > C(\eta - \varepsilon B)$ (Lemma 2.50). Note that the statement is empty if $\varepsilon B > \eta$.
- Since v is the first vector in a reduced basis, the LLL theorem says that

$$\begin{aligned} q(v) &\leqslant 2^n q(\text{smallest nonzero vector in } \Lambda) \\ &\leqslant 2^n \left(C^2 \left|P(\hat{z})\right|^2 + (n+1)A^2\right) \\ &\leqslant 2^n \left(C^2 \varepsilon^2 B^2 + (n+1)A^2\right) \quad \text{(Lemma 2.50, with } P(z) = 0\text{).} \end{aligned}$$

- On the other hand $q(v) \geqslant C^2 \left|Q(\hat{z})\right|^2 > C^2(\eta - \varepsilon B)^2$ if $\eta - \varepsilon B \geqslant 0$.

We get a contradiction if

$$\begin{aligned} C^2(\eta - \varepsilon B)^2 &\geqslant 2^n(C^2\varepsilon^2B^2 + (n+1)A^2), \\ \text{or} \quad \eta - \varepsilon B &\geqslant 2^{(n+1)/2}\varepsilon B, \end{aligned}$$

if we choose C such that $C\varepsilon B = \sqrt{n+1}A$. We now choose $\varepsilon \leqslant \eta/2B$ such that $2^{(n+1)/2}\varepsilon B \leqslant \eta/2$, i.e. $\varepsilon = \eta\big/B2^{(n+3)/2}$ is suitable. We find successively

$$\log(1/\varepsilon) = \log(1/\eta) + \log B + O(n) = O(n^2 \log A),$$

$$\log C = O(\log(nA) - \log(\varepsilon B)) = O(n^2 \log A),$$

hence the sizes of all inputs are polynomially bounded in terms of $n\log(A)$. □

Chapter 3

Polynomials

Factorization of univariate polynomials is our unifying theme, over finite fields, over p-adic fields and $\mathbb{C}$, and over $\mathbb{Q}$.

3.1 Factoring in $\mathbb{F}_q[X]$

3.1.1 Basic idea for factoring in a Euclidean ring R

To factor $N \in R$:

- Look for a zero divisor in $R/(N)$
- $d = \gcd(a, N)$ is a nontrivial factor of N, such that d and N/d are not units in R.
- Restart with d, N/d.

This algorithm works if we have a notion of size so that d and N/d are smaller than N, and we know the size cannot decrease indefinitely. This ensures the method terminates. The problem now becomes to find zero divisors.

Example 3.1. To factor $N \in \mathbb{Z}$, we want to find $x^2 \equiv y^2 \pmod{N}$, $x, y \in \mathbb{Z}$, so that$(x - y)(x + y) \equiv 0 \pmod{N}$. Hence $x \pm y$ are two potential zero divisors (provided none of them is 0).

Exercise 3.2. In this example, we must assume that N is not a prime power.

1. Why ?
2. How to reduce to this case? (How to detect N is of the form p^k, and compute p and k in that case?)

Exercise 3.3. If k is a field and $P \in k[X]$ we may assume that P has no repeated factors, i.e $\gcd(P, P') = 1$. Why?

3.1.2 A special case: roots over $\mathbb{F}_p$

Let $T \in \mathbb{F}_p[X]$, $\deg T = n$. We are not interested in all factors, only the linear ones. Eventually, we will reduce the whole factorization problem over $\mathbb{F}_q[X]$ to finding roots over the prime field $\mathbb{F}_p$.

Reduction

We may assume that T splits (has n distinct roots) over $\mathbb{F}_p$. Indeed, we may replace T by $\gcd(T, X^p - X)$: the new T is split and has the same roots as the old one. In fact

$$X^p - X = \prod_{\alpha \in \mathbb{F}_p} (X - \alpha).$$

Cost: assuming the coefficients of T are in $[0, p-1]$, the size of the input is $O(n \log p)$. We count the elementary operations in $\mathbb{F}_p$ in terms of n and $\log p$:

- Compute $y = X^p$ in $\mathbb{F}_p[X]/(T)$:
  - $O(\ln p)$ operations in $\mathbb{F}_p[X]/(T)$
  - in this algebra an operation costs $\widetilde{O}(n)$ operations in $\mathbb{F}_p$ (multiplication, addition).
- $\gcd(y, T)$: $\widetilde{O}(n)$ operations in $\mathbb{F}_p$.

So $\widetilde{O}(n \log p)$ operations.

Find a 0-divisor in $\mathbb{F}_q[X]/(T)$

Let $A := \mathbb{F}_p[X]/(T) \overset{\mathbb{F}_p-alg}{\simeq} (\mathbb{F}_p)^n$, from the Chinese Remainder Theorem. We can easily find 0-divisors in $(\mathbb{F}_p)^n$ (any non-zero vector with a zero coordinate), however the isomorphism $\mathbb{F}_p[X]/(T) \simeq (\mathbb{F}_p)^n$ is not explicit: for that we would need to know the decomposition of T! On the other hand, A is a quite concrete $\mathbb{F}_p$-vector space:

$$A = \left\langle 1, \overline{X}, ..., \overline{X}^{n-1} \right\rangle_{\mathbb{F}_p}.$$

Lemma 3.4. *For all $x \in A$, $x^p - x = \prod_{\alpha \in \mathbb{F}_p} (x - \alpha) = 0$.*

Any sub-product in the above is a potential 0-divisor, but it is not obvious how to find a non-trivial one, i.e. such that its cofactor is not 0 !

Let $x \in A \backslash \mathbb{F}_p$ and $y_j := x(x-1)(x-2)\dots(x-j)$. There exists a smallest $1 \leqslant j < p$ such that $y_j = 0$. Then $x - j$ is a 0-divisor, yielding the following algorithm; unfortunately, a priori, the worst case running time is $O(p)$, i.e. exponential in $\log p$.

Algorithm 7. 0-divisor in $\mathbb{F}_p[X]/(T)$, T split

```
Input: $x \in A \setminus \mathbb{F}_p$
Output: a 0-divisor
set $j := 0$, $y := x$;
while $y \neq 0$ do
    increase $j$ by 1 and set $y := y(x - j)$.
Return $x - j$    $\{\neq 0\}$
```

Exercise 3.5. Introduce the polynomial $P(X) = X(X-1)\dots(X-t)$, t close to $\sqrt{p}$ and improve the above into a $\widetilde{O}(\sqrt{p})$ algorithm.

Find a 0-divisor in $\mathbb{F}_q[X]/(T)$ with randomization assuming $p > 2$

All known deterministic algorithm for our problem are exponential time in the worst case. We now use randomization to avoid this exponential behavior.

Lemma 3.6. *Let $t := \frac{p-1}{2} \in \mathbb{Z}$. For all $x \in A$, $x(x^t - 1)(x^t + 1) = 0$.*

If $x \neq 0$ and $x^t \neq \pm 1$ then we have a 0-divisor, namely

$$\begin{cases} x(x^t - 1) & \text{if } \neq 0, \\ x & \text{otherwise.} \end{cases}$$

Pick x uniformly at random in A. The bad choices, not leading to a 0-divisor, are

$$\begin{cases} x = 0 & 1 \text{ choice,} \\ x^t = 1 & t^n \text{ choices,} \\ x^t = -1 & t^n \text{ choices.} \end{cases}$$

So, the probability $P(n)$ of picking a bad x is $(2t^n + 1)/p^n$. Notice that $P(n)$ is a decreasing function of n. As expected, $P(1) = 1$: there is no good choice if T is irreducible! Otherwise, for $n \geqslant 2$,

$$P(n) \leqslant P(2) = \frac{2t^2 + 1}{p^2} \leqslant \frac{1}{2}.$$

<u>Cost</u>: $\widetilde{O}(n \log p)$ operations in $\mathbb{F}_p$, no more than the initial reduction.

- Compute x^t in A: $\widetilde{O}(n \log p)$ operations in $\mathbb{F}_p$

- gcd : $\widetilde{O}(n)$

Algorithm 8. Split roots

```
Input: $T \in \mathbb{F}_p[X]$, totally split over $\mathbb{F}_p$, $p$ odd prime.
Output: 0-divisor in $A$.
1: Pick $\overline{x} \in A$ uniformly at random.
2: Compute $b$ a representative in $\mathbb{F}_p[X]$ of $x^t \in A$, $t = (p-1)/2$.
3: Compute $D = \gcd(b-1, T) \in \mathbb{F}_p[X]$.
4: If $\overline{D} \neq 1$ and $\overline{D} \neq 0$ in $A$, then return $D$.
5: Otherwise return FAIL.
```

The complete algorithm is now easy to write, a special case of Equal Degree Factorization (or EDF):

Algorithm 9. Equal Degree Factorization, degree 1

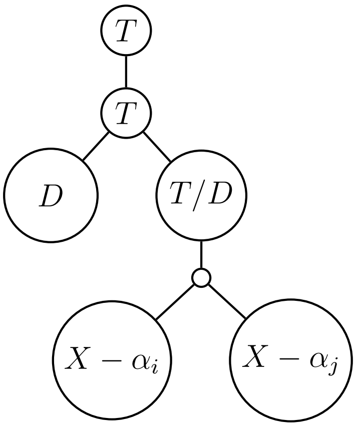
```
Input: $T \in \mathbb{F}_p[X]$, totally split over $\mathbb{F}_p$, $p$ odd prime.
Output: Factors of degree 1.
1: If $\deg T = 1$, return $T$.
2: Call Split until it returns a non-trivial factor $D$ of $T$.
3: Call oneself recursively on $D$ and $T/D$. Return the concatenation of the
   outputs.
```

Lemma 3.7. *Let $T = \lambda \prod_{i=1}^{s}(X - \alpha_i)$. Let $1 \leqslant i < j \leqslant n$ be two indices. Then* Split*(T) fails to separate α_i and α_j with probability $\leqslant 5/9 < 1$.*

Proof. Exercise. □

Theorem 3.8. *The algorithm finds all the roots of T in an expected number of operation in $\mathbb{F}_p$ which is $\widetilde{O}(n \log p)$.*

Proof. For a given sequence of random choices, define the execution tree as follows. Each node is a run of our Split algorithm, and is labelled with the polynomial T. Either it succeeds and the node has two offsprings (labelled with their respective factors), or it fails and we have a single son.



On a given level, the product of all labels is T (or a divisor of T if we remove the degree 1 factors as they are found). The cost of a node of degree m is $\widetilde{O}(m \log p)$, so the cost of a level is $\widetilde{O}(n \log p)$. (Since n is the sum of the m for the nodes on that level.) The proof follows from next Lemma. □

Lemma 3.9. *The expected depth (number of levels) of the tree is* $\widetilde{O}(\log n)$.

Proof. Fix $i < j$ and let $\delta = 5/9$. By Lemma 3.7:

$$\text{Prob}(\alpha_i \text{ and } \alpha_j \text{ are not separated in a given node}) \leqslant \delta.$$

Since at level k we went through k nodes already,

$$\text{Prob}(\alpha_i \text{ and } \alpha_j \text{ are not separated after level } k) \leqslant \delta^k.$$

We stop when all pairs of roots are separated. There are $n(n-1)/2 \leqslant n^2$ pairs of roots, so

$$p_k := \text{Prob}(\text{not done after level } k) \leqslant n^2 \delta^k.$$

Note that the probability that the tree stops at depth k is exactly $p_{k-1} - p_k$. Hence

$$\begin{aligned} \mathbb{E}(\text{depth}) &= \sum_{k \geqslant 1} (p_{k-1} - p_k) k \\ &= \sum_{k \geqslant 1} (k-1) p_{k-1} - k p_k + p_{k-1} = \sum_{k \geqslant 0} p_k \leqslant +\infty \end{aligned}$$

because $k p_k \leqslant n^2 k \delta^k$ tends to 0 as $k \to +\infty$. We now use either the trivial bound $p_k \leqslant 1$, or the one just proved:

$$\sum_{k \geqslant 0} p_k \leqslant \sum_{k \geqslant 0} \min(1, n^2 \delta^k).$$

Note that $n^2 \delta^k \leqslant 1$ if and only if $k \geqslant s := \lceil -\log_\delta n^2 \rceil$. So

$$\sum_{k \geqslant 0} p_k \leqslant s + \sum_{k \geqslant s} n^2 \delta^k \ll s + n^2 \delta^s \ll \log n.$$

□

3.1.3 Squarefree factorization

We now reduce the general problem of factorization over $k[X]$ to *squarefree* polynomials.

Definition 3.10. Let $T \in k[X]$ be monic non-constant, and $\prod T_i^{e_i}$ its decomposition into distinct irreducible monic factor. The squarefree part of T, or $\mathrm{core}(T)$, is $\prod T_i$.

The squarefree factorization of T is $T = f_1^1 f_2^2 \dots f_m^m$, where the $f_i \in k[X]$ are monic squarefree polynomials, pairwise coprime, and $f_m \neq 1$. The squarefree part of T is $f_1 \dots f_m$.

It is enough for our purpose to compute $\mathrm{core}(T)$, since once it is factored and the irreducible factors of T are known, we can compute the e_i as valuations (if they are at all needed).

Lemma 3.11. *Let $T \in k[X]$ of degree n, k a* perfect *field of characteristic $p \geqslant 0$. Assume $T = \prod_{i=1}^{s} T_i^{e_i}$, where the T_i are irreducible, pairwise non-associate. Let*

$$u := \gcd(T, T'), \quad v := T/u, \quad w := \frac{u}{\gcd(u, v^{\min(n-1, \deg u)})}$$

then

$$v = \prod_{p \nmid e_i} T_i, \quad \textit{and} \quad w = \prod_{p \mid e_i} T_i^{e_i}.$$

Proof. From the logarithmic derivative formula, we find

$$T' = \sum_{i=1}^{s} e_i \frac{T_i'}{T_i} T,$$

hence $T_i^{e_i - 1} \mid \gcd(T, T')$ and $T_i^{e_i}$ divides the GCD if and only if $T_i \mid e_i T_i'$, i.e $e_i = 0$ in k (since $T_i' = 0$ over the perfect field k would imply that T_i is a p-th power, hence not irreducible). Since the irreducible factors of v are among the T_i,

$$u = \prod_{p \nmid e_i} T_i^{e_i - 1} \prod_{p \mid e_i} T_i^{e_i},$$

and the first equality follows. For the second, $e_i - 1$ is less than $\deg u$ and $n - 1$ for all i, hence

$$\gcd(u, v^{\min(n-1, \deg u)}) = \prod_{p \nmid e_i} T_i^{e_i - 1}$$

and we are done. □

From the lemma, we can compute the squarefree part of T, in fact in a partially factored form:

Algorithm 10. Squarefree part over $\mathbb{F}_q$, core

```
Input: $T \in \mathbb{F}_q[X]$, $\deg T = n$.
Output: core($T$), the squarefree part of $T$.
Compute $u = \gcd(T, T')$, then $v = T/u$.
Compute a representative $a \in \mathbb{F}_q[X]$ of $v^{\min(n-1,\deg u)-1}$ in $\mathbb{F}_q[X]/(u)$, then
    $w = u/\gcd(u, a)$.   {w is of the form $\sum w_i X^{pi}$}
Let $W = \sum F^{-1}(w_i)X^i$, where $F^{-1} : x \mapsto x^{q/p}$.   {we have $W^p = w$}
Return $v \times$ core($W$), calling ourselves recursively.
```

Theorem 3.12. *Let $T \in k[X]$ be of degree n. If k has characteristic 0, or $k = \mathbb{F}_p$, p prime, the squarefree part of T is computed in $\widetilde{O}(n)$ operations in k. If $k = \mathbb{F}_q$ is finite but not a prime field, the squarefree part is computed in $\widetilde{O}(n \log q)$ operations.*

Proof. Computing u, v, v^{n-1} (mod u) and w all cost $\widetilde{O}(n)$ operations in k. This proves the case $\operatorname{char} k = 0$. Evaluating F^{-1} on $x \in \mathbb{F}_q$ costs $\widetilde{O}(\log(q/p))$ operations, i.e. nothing if $q = p$. If $f(n)$ is a the total cost of the computation for T of degree n, then $f(n) \leqslant \widetilde{O}(n \log(q/p)) + f(n/p)$, and the case $k = \mathbb{F}_q$ follows. □

Remark 3.13. If we are counting operations over $\mathbb{F}_p$, and write $\mathbb{F}_q = \mathbb{F}_p(y)$ then we can improve the above by precomputing $F^{-1}(y)$ then $F^{-1}(Q(y))$ as $Q(F^{-1}(y))$.

Yun's algorithm ([24, §14.6]) exploits the Lemma in a clever way to find the full squarefree factorization at the same cost as above.

3.1.4 Factorization over $\mathbb{F}_q[X]$, q odd

The beginning is actually also valid for q even. Let $T \in \mathbb{F}_q[X]$ be square-free, $T = \prod_{i=1}^{s} T_i$, T_i irreducible, pairwise coprime. Let $A := \mathbb{F}_q[X]/(T)$ and $F : x \mapsto x^q$ the Frobenius endomorphism (of $\mathbb{F}_q$-algebras) of A. By the Chinese Remainder Theorem,

$$\begin{array}{rcl} A & \xrightarrow{\sim} & \displaystyle\prod_{i=1}^{s} \mathbb{F}_q[X]/(T_i) \simeq \prod_{i=1}^{s} \mathbb{F}_{p^{\deg T_i}}. \\ a & \longmapsto & (\pi_1(a), ..., \pi_s(a)) \end{array}$$

Let $B := \ker(F - \mathrm{Id})$ the Berlekamp algebra, first introduced in this context in [4].

Lemma 3.14. $B \simeq (\mathbb{F}_q)^s$.

From now on q is odd. As before, for all $b \in B$, $b(b^t - 1)(b^t + 1) = 0$, where $t = \frac{q-1}{2}$. It is a little harder to split polynomials since we have no immediate way of detecting an irreducible factor (we can recompute another Berlekamp matrix, but this is expensive), so we cannot remove them from the tree as they are found. So we keep trying to split all polynomials obtained so far until the required s factors are found.

```
Algorithm 11. SPLIT over $\mathbb{F}_q[X]$, $q$ odd
Input: $(T, B, \mathcal{F})$, where $T \in \mathbb{F}_q[X]$, $B$ is given by an $\mathbb{F}_q$-basis, and $\mathcal{F} = \{f_1, \dots f_r\}$ is a collection of $f_i \in \mathbb{F}_q[X]$ such that $T = \prod_{i \leqslant r} f_i$ for some $r < s$.
Output: A larger collection of factors of $T$.
1: Pick uniformly a random $\overline{x} \in B$
2: Compute $\bar{y} := \overline{x}^t$ in $\mathbb{F}_q[X]/(T)$.    $\{O(\log t)$ multiplications$\}$
3: Compute $D := \gcd(y - 1, f_i)$ in $\mathbb{F}_q[X]$, for all $i \leqslant r$. Each time $D$ is non trivial, we split the corresponding $f_i$ in two.
4: Return the new collection of factors.
```

The global algorithm is then as follows:

```
Algorithm 12. Berlekamp's algorithm over $\mathbb{F}_q[X]$, $q$ odd
Input: $T \in \mathbb{F}_q[X]$, squarefree, $q$ odd.
Output: The irreducible factors of $T$.
1: Compute an $\mathbb{F}_q$-basis of $B = \ker(F - \mathrm{Id})$ acting on $\mathbb{F}_q[X]/(T)$.
2: Let $s = \dim_{\mathbb{F}_q} B$. If $s = 1$, we are done: $T$ is irreducible.
3: Otherwise, set $\mathcal{F} = \{T\}$ then $\mathcal{F} = \text{SPLIT}(T, B, \mathcal{F})$ until $\#\mathcal{F} = s$.
4: Return $\mathcal{F}$.
```

Lemma 3.15. *The expected cost of "Splitting B", i.e. of splitting T using B is $\widetilde{O}(n \log q)$ operations in $\mathbb{F}_q$.*

Proof. It is essentially the same as for roots. Left as an exercise. □

What is the cost of computing a basis of B? We count operations over $\mathbb{F}_q$:

1. Compute $F(X) = X^q$ in $A = \mathbb{F}_q[X]/(T)$: $\widetilde{O}(n \log q)$ operations.

2. $F(X^2) = X^{2q}, \dots, X^{(n-1)q}$: $n - 1$ multiplications in A, for a cost of $\widetilde{O}(n^2)$ operations ($\widetilde{O}(n)$ for one multiplication).

3. Compute the kernel : $\widetilde{O}(n^\omega)$.

Theorem 3.16. *The total cost of Berlekamp's algorithm is $\widetilde{O}(n^\omega + n \log q)$ expected operations in $\mathbb{F}_q$. The computation of the Berlekamp algebra B is deterministic, and yields the number of irreducible factors; only the splitting is randomized.*

3.1.5 Factorization over $\mathbb{F}_{2^f}[X]$

As before, let $B := \ker(F - \mathrm{Id})$ be the Berlekamp algebra. For q odd, we used $x(x^t - 1)(x^t + 1) = 0$. In characteristic 2, we consider instead

$$\mathrm{Tr}(x) = \sum_{k=1}^{f} x^{2^{k-1}}.$$

Lemma 3.17. *1.* $\mathrm{Tr} : \mathbb{F}_{2^f} \to \mathbb{F}_2$ *is a surjective* $\mathbb{F}_2$*-linear map.*

2. For a random $x \in \mathbb{F}_{2^f}$, $\mathrm{Tr}(x) = 0$ *or* 1 *with probability* $1/2$.

3. For all $b \in B$, $\pi_i(\mathrm{Tr}(b)) \in \mathbb{F}_2$.

Proof. Exercise. □

Corollary 3.18. *For a random* $b \in B$, $\mathrm{Tr}(b) \in \mathbb{F}_2$ *with probability* $2^{1-s} \leqslant 1/2$ *if* $s \geqslant 2$

We thus obtain our last variant of the Split algorithm:

Algorithm 13. Split over $\mathbb{F}_q[X]$, $q = 2^f$ even

```
Input: $(T, B, \mathcal{F})$, where $T \in \mathbb{F}_q[X]$, $B$ is given by an $\mathbb{F}_q$-basis, and $\mathcal{F} = \{f_1, \dots f_r\}$ is a collection of $f_i \in \mathbb{F}_q[X]$ such that $T = \prod_{i \leqslant r} f_i$ for some $r < s$.
Output: A larger collection of factors of $T$.
1: Pick uniformly a random $\overline{x} \in B$
2: Compute $D := \gcd(\mathrm{Tr}(b), T)$ in $\mathbb{F}_q[X]$, for all $i \leqslant r$. Each time $D$ is non trivial, we split the corresponding $f_i$ in two.
3: Return the new collection of factors.
```

3.1.6 Conclusion

Theorem 3.19. *The expected cost of the Berlekamp Algorithm using any of the* Split *subroutines is the same:* $\widetilde{O}(n^\omega + n \log q)$ *operations in* $\mathbb{F}_q$.

Proof. Exercise. □

Using black-box algebra and randomizing also the computation of the Berlekamp algebra B, the above improves to $\widetilde{O}(n^2 + n \log q)$ (Kaltofen & Lobo).

A quite different approach (iterated Frobenius), was introduced by von zur Gathen & Shoup, improving on a classical algorithm by Cantor & Zassenhaus: it avoids completely linear algebra, and computes simultaneously the X^{q^d} for $d \leqslant n$ in $\widetilde{O}(n^2 + n \log q)$ operations in $\mathbb{F}_q$ (this is non-trivial within that time bound, contrary to the X^{qd} required by Berlekamp!).

After taking the gcds $(X^{q^d} - X, T)$, it remains to split products of irreducible polynomials having the same degree d, using a generalization of Algorithm 3.1.2. This algorithm achieves the same complexity: $\widetilde{O}(n^2 + n \log q)$ expected operations in $\mathbb{F}_q$. Both algorithms are quite practical.

3.2 Factoring in $\mathbb{Q}[X]$

3.2.1 $\mathbb{Q}_p$ and $\mathbb{Z}_p$

$\mathbb{R}$ is a convenient "limit field" associated to convergent sequences of rational approximations: for all $x \in \mathbb{R}$, there exists $\widehat{x}(\epsilon) \in \mathbb{Q}$ such that $|x - \widehat{x}| < \epsilon$. From a computational point of view we only know (some of) the approximations $\widehat{x}$.

We need an analogous ring associated to a different metric: now we want closer and closer approximations modulo ever larger powers of a given prime p: for all $x \in \mathbb{Z}_p$ there exists $\widehat{x}(n) \in \mathbb{Z}$ such that $|x - \widehat{x}|_p < p^{-n}$.

Definition 3.20. Let $|\cdot|_p : \mathbb{Q} \to \mathbb{R}^+$ defined by $x \mapsto p^{-v_p(x)}$ for $x \neq 0$ and $|0|_p = 0$. It is a non-Archimedean absolute value, satisfying $|x+y|_p \leqslant \max(|x|_p, |y|_p)$.

Definition 3.21. Equip $\mathbb{Q}$ with the topology afforded by the $|\cdot|_p$ metric. Let $\mathbb{Q}_p$ be the ring of Cauchy sequences in $\mathbb{Q}^{\mathbb{N}}$, modulo the ideal of sequences converging to 0. (Both "Cauchy sequence" and "converging to 0" are understood with repect to the p-adic metric!) The absolute value $|\cdot|_p$ extends to $\mathbb{Q}_p$ (with values in $\mathbb{Q}$), making it a topological *field*.

Definition 3.22. Let $\mathbb{Z}_p = \Big\{x \in \mathbb{Q}_p\colon\ |x|_p \leqslant 1\Big\}$ be the unit ball in $\mathbb{Q}_p$. This is a compact ring, whose field of fraction is $\mathbb{Q}_p$, in fact $\mathbb{Q}_p = \mathbb{Z}_p[1/p]$. $\mathbb{Z}_p$ is a principal local ring with unique maximal ideal $p\mathbb{Z}_p = \Big\{x \in \mathbb{Q}_p\colon\ |x|_p < 1\Big\}$ the open unit ball. One proves that the natural map $\mathbb{Z}_p/p^k\mathbb{Z}_p \to \mathbb{Z}/p^k\mathbb{Z}$ is an isomorphism.

An alternative construction for $\mathbb{Z}_p$ and $\mathbb{Q}_p$ is

$$\mathbb{Z}_p = \varprojlim_{n \geqslant 1} \mathbb{Z}/p^n\mathbb{Z},$$

that is the subring of $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p^2\mathbb{Z} \times \cdots$, containing the $(x_1, x_2, \dots)$ such that $x_i \equiv x_j \pmod{p^i}$, for all $i \leqslant j$. Then define $\mathbb{Q}_p = \operatorname{Frac} \mathbb{Z}_p = \mathbb{Z}_p[1/p]$.

The important property for us is that $\mathbb{Z}$ is dense in $\mathbb{Z}_p$ (and $\mathbb{Q}$ in $\mathbb{Q}_p$). We represent an $x \in \mathbb{Z}_p$ by an approximation $\widehat{x} \in \mathbb{Z}$ such that $x \equiv \widehat{x} \pmod{p^\ell}$ and say that x is known modulo p^ℓ. The next theorem is the crux of all modern factorization algorithms over $\mathbb{Q}[X]$: one can refine an approximate factorization in *coprime* factors to an arbitrary accuracy.

Theorem 3.23 (Hensel's lemma). *Let $T \in \mathbb{Z}_p[X]$ be a monic polynomial, and a collection of polynomials $T_i \in \mathbb{Z}_p[X]$, $i \leqslant r$, which are monic and pairwise coprime, such that*

$$T \equiv \prod_{i=1}^{r} T_i \pmod{p} \quad (= \bmod p\mathbb{Z}_p[X]).$$

(This is an equality in $\mathbb{Z}_p[X]/p\mathbb{Z}_p[X] \simeq \mathbb{F}_p[X]$.) There exist unique $T_i^* \in \mathbb{Z}_p[X]$ such that $T = \prod_{i=1}^{r} T_i^*$ and $T_i^* \equiv T_i \pmod{p}$ for $i \leqslant r$.

Proof. Let $k > 0$. Starting from a factorization $T = \prod_{i=1}^{r} T_i \pmod{p^k}$, we construct explicitly $t_i \in \mathbb{Z}_p[X]$, $\deg t_i < \deg T_i$ such that

$$T = \prod_{i=1}^{r} (T_i + t_i p^k) \pmod{p^{2k}}.$$

In fact, from a Bezout relation in the principal ring $\mathbb{Q}_p[X]$, we obtain $t_i \in \mathbb{Q}_p[X]$, $\deg t_i < \deg T_i$ such that

$$\sum_{i=1}^{s} t_i \prod_{j \neq i} T_j = \frac{T - \prod_{i=1}^{r} T_i}{p^k} \in \mathbb{Z}_p[X].$$

We claim the t_i belong to $\mathbb{Z}_p[X]$. If not, let t_i have the largest denominator p^d; clear denominators by multiplying by p^d and reduce modulo p: we see that T_i divides $p^d t_i$ modulo p, hence $t_i = 0$, a contradiction.

By induction this yields $T_i^{(k)} \in \mathbb{Z}_p[X]$, $\deg T_i^{(k)} \leqslant \deg T$ such that $T_i^{(k)} \equiv T_i^{(\ell)} \pmod{\ell}$ for all $\ell \leqslant k$, i.e. $k \mapsto T_i^{(k)}$ is a Cauchy sequence in $\mathbb{Z}_p[X]_{\leqslant \deg T}$, which is complete. Therefore, it converges to $T_i^* \in \mathbb{Z}_p[X]$. To prove unicity, notice that $\mathbb{Z}_p$ is a UFD (it is principal!), hence $\mathbb{Z}_p[X]$ is also a UFD. □

Remark 3.24. It is a little easier to lift a Bezout relation in $\mathbb{F}_p[X] = \mathbb{Z}_p[X]/(p)$ to ensure a factorization modulo p^{k+1}, instead of p^{2k}. This would be sufficient for the above proof, but would be less efficient in the computational version to follow.

To put this proof in a computationally convenient form, we must deal with approximations in $\mathbb{Z}/p^k\mathbb{Z}$ at each stage. E.g. knowing the t_i modulo p^k is enough to know $T_i^{(2k)} := T_i^{(k)} + p^k t_i$ modulo p^{2k}. The problem is the Bezout relation, since $(\mathbb{Z}/p^k\mathbb{Z})[X]$ is a terrible ring, certainly not principal. To avoid this, we lift simultaneously the polynomials and the Bezout relation:

Algorithm 14. Hensel step

```
Input: $R$ (commutative, unital), $f, g, h \in R[X]$ monic, such that $\deg s < \deg h$,
    $\deg t < \deg g$, $f \equiv gh \pmod{m}$, $sg + th \equiv 1 \pmod{m}$.
Output: $g^*, h^* \in R[X]$ monic, $s^*, t^* \in R[X]$ such that $\deg s^* < \deg h^* = \deg h$,
    $\deg t^* < \deg g^* = \deg g$, $f \equiv g^*h^* \pmod{m^2}$, $s^*g^* + t^*h^* \equiv 1 \pmod{m^2}$.
1: Let $e = f - gh$,
    $G = et \pmod{m^2, g}$,    $g^* = g + G$
    $H = es \pmod{m^2, h}$,    $h^* = g + H$
2: Let $e = 1 - (sg^* + th^*)$
    $S = et \pmod{m^2, h^*}$,    $s^* = s + S$
    $T = es \pmod{m^2, g^*}$,    $t^* = t + T$
```

Proof. Letting $g^* = g + G$, $h^* = h + H$, $s^* = s + S$, $t^* = t + T$, we look for $G, H, S, T \equiv 0 \pmod{m}$ with $\deg G < \deg g$, $\deg H < \deg h$, $\deg S < \deg S$, $\deg T < \deg t$.

First lift g and h. We want

$$g^*h^* \equiv gh + Hg + Gh \pmod{m^2} \equiv f.$$

Letting $e = f - gh \equiv 0 \pmod{m}$, we try to solve $Hg + Gh \equiv e \pmod{m^2}$.

The quotient ring R/m^2 is not a field so we cannot solve this in the usual way (extended Euclidean algorithm). Fortunately, a Bezout relation is provided with the input, and $G = et$, $H = es$ is a solution. Unfortunately, we want $\deg G < \deg g$ and $\deg H < \deg h$.

But we can modify H by a multiple of h and G by the corresponding multiple of g, so write: $G = qg + r$, $\deg r < \deg g$ (division possible because g is monic), and replace $G \leftarrow G - qg$, $H \leftarrow H + qh$.

1. $Hg + Gh$ does not change.

2. $G = r$ satisfies $\deg G < \deg g$.

3. $\deg e < \deg f = \deg g + \deg h$. (Since f, g, h are monic, leading terms cancel). Now use $Gh+Hg = e$; since $\deg Gh = \deg G + \deg h < \deg g + \deg h$, we finally obtain $\deg(Hg) < \deg g + \deg h$. In other words, $\deg H < \deg h$ as required.

4. $f \equiv g^*h^* \pmod{m^2}$.

Now lift s, t: we have

$$s^*g^* + t^*h^* = sg^* + th^* + Sg^* + Th^*, \quad sg^* + th^* \equiv 1 \pmod{m},$$

and we solve $Sg^* + Th^* \equiv 1 - (sg^* + th^*) \pmod{m^2}$ exactly as before. □

The most important applications are $R = \mathbb{Z}$, $m = p^\ell$, p prime and $R = k[X]$, $m = X^\ell$.

Lemma 3.25. *If $R = \mathbb{Z}$, $\deg f = n$, and all inputs satisfy $\|\cdot\|_\infty < m^2$, the cost of a Hensel step is $\widetilde{O}(n \log m)$, i.e. essentially linear in the input size.*

Exercise 3.26. Over $\mathbb{Z}$, if $f \equiv gh \pmod{p^\ell}$, and we do not wish to lift s, t, we can still solve $Hg + Gh \equiv f - gh$ but modulo $p^{\ell+1}$, no longer modulo $p^{2\ell}$. Develop the corresponding algorithm (linear lift).

Algorithm 15. Hensel multi-lift

Input: R commutative ring, $p \in R$ such that R/p is a field, $\ell \in \mathbb{Z}_{>0}$. $f \in R[X]$ monic. $f_1, f_2, \ldots, f_r$ monic in $R[X]$, pairwise coprime in $(R/p)[X]$ such that

$$f = \prod_{i=1}^{r} f_i \pmod{p}.$$

Output: Monic $f_1^*, \ldots, f_r^* \in R[X]$ such that

$$f \equiv \prod_{i=1}^{r} f_i^* \pmod{p^\ell}.$$

```
1: If $r = 1$, return $f$.
2: Let $k = \lfloor r/2 \rfloor$, $g = f_1 \ldots f_k$, $h = f_{k+1} \ldots f_r$, and find $s, t$ such that $sg + th \equiv 1 \pmod{p}$ (Extended Euclidean Algorithm).
3: Compute $g^*, h^*$ such that $f \equiv g^* h^* \pmod{p^\ell}$, using $O(\log \ell)$ Hensel Steps.
4: Call ourselves recursively with $g^*, (f_1 \ldots f_k)$, then $h^*, (f_{k+1} \ldots f_r)$ and return the concatenation of the factors.
```

Theorem 3.27. *Assume $R = \mathbb{Z}$, $\deg f = n$, and all inputs are reduced modulo p^ℓ. Then the lifting cost is $\widetilde{O}(n \log p^\ell)$, essentially linear.*

Proof. Exercise. See [24, §15]. □

3.2.2 Bounds

Let

$$T = a_n \prod_{i=1}^{n} (X - \alpha_i), \quad \alpha_i \in \mathbb{C}, \quad a_n \neq 0.$$

Define the Mahler measure

$$M(T) := |a_n| \prod_{i=1}^{n} \max(|\alpha_i|, 1).$$

Theorem 3.28 (Landau). *$M(T) \leqslant \|T\|_2 \leqslant 2^{\deg T} M(T)$.*

Proof. We first prove the left-hand side: let

$$U(X) = a_n \prod_{|\alpha_i| \leqslant 1} (\bar{\alpha}_i X - 1) \prod_{|\alpha_i| > 1} (X - \alpha_i).$$

We have

$$M(T) = |U(0)| \leqslant \|U\|_2 = \|T\|_2 ,$$

where the last equality uses the following lemma, whose proof is an easy exercise:

Lemma 3.29. *If $z \in \mathbb{C}$, $\|(X - z)T\|_2 = \|(\bar{z}X - 1)T\|_2$.*

As for the right hand side, writing the relations between roots and coefficients for $T = \sum_{i=0}^{n} a_i X^i$, we have

$$|a_i| \leqslant \binom{n}{i} |a_n| \max_{j_1 < \cdots < j_i} |\alpha_{j_1} \cdots \alpha_{j_i}| ,$$

and the max is less than $M(T)/|a_n|$. □

Corollary 3.30. *If $S \mid T$ in $\mathbb{C}[X]$ then $\|S\|_2 \leqslant 2^{\deg S} \|T\|_2 \leqslant 2^{\deg T} \|T\|_2$.*

Proof. $\|S\|_2 \leqslant 2^{\deg S} M(S) \leqslant 2^{\deg S} M(T) \leqslant 2^{\deg T} \|T\|_2$. □

In particular, if $S, T \in \mathbb{Z}[X]$, $\deg T = n$, $\|T\|_\infty = A$, we have

$$\|S\|_\infty \leqslant \|S\|_2 \leqslant 2^n (n+1)^{1/2} A,$$

yielding a finite number of possible divisors, hence a naive algorithm to factor T. We shall now improve drastically on such an exhaustive search.

3.2.3 Zassenhaus's algorithm

Our goal is to factor the polynomial T in $\mathbb{Q}[X]$. By clearing denominators and rescaling ($T \leftarrow a^{n-1} T(X/a)$ if T has degree n and leading coefficient a), we may assume that T is *monic* in $\mathbb{Z}[X]$. Gauss lemma (multiplicativity of contents) then says that the monic irreducible factors of T in $\mathbb{Q}[X]$ are in fact in $\mathbb{Z}[X]$.

From squarefree factorization, we can assume further that T is squarefree.

Definition 3.31. Le p prime, $k \in \mathbb{Z}_{>0}$ and R one of the rings $\mathbb{Z}$, $\mathbb{Z}_p$ or $\mathbb{Z}/p^i\mathbb{Z}$ for some $i \geqslant k$. If $a \in R$, we write $\mathrm{MOD}\left(a, p^k\right)$ for the unique representative in $\mathbb{Z}$ of $a \pmod{p^k}$, which lies in $]-p^k/2, p^k/2]$. We extend this definition coefficient-wise to polynomials in $R[X]$.

Algorithm 16. Zassenhaus's algorithm

```
Input: $T \in \mathbb{Z}[X]$, monic, squarefree
Output: The monic irreducible factors of $T$ in $\mathbb{Q}[X]$.
Pick $p$ prime such that $T \in \mathbb{F}_p[X]$ remains squarefree and compute
    $$T \equiv \prod_{i=1}^{r} T_i \pmod{p}$$
    where the $T_i$ are distinct, monic and irreducible in $\mathbb{F}_p[X]$.
Let $B = 2^{\det T} \|T\|_2$   {= bound for the sup-norm of a factor in $\mathbb{Z}[X]$}
Compute $\ell$ minimal such that $p^\ell > 2B$ and lift
    $$T \equiv \prod_{i=1}^{r} T_i^* \pmod{p^\ell}.$$
Let $S_0 = \{1, \dots, r\}$.
for $S \subset S_0$, by increasing size do
    Compute
        $$A_S := \mathrm{MOD}\left(\prod_{i \in S} T_i^*, p^\ell\right) \in \mathbb{Z}[X].$$
    if $A_S \mid T$ then
        Output $A_S$.  {$A_S$ is an irreducible factor}
        Replace $S_0 \leftarrow S_0 \setminus S$ and $T \leftarrow T/A_S$.
```

Proof. We first prove that all factors appear as A_S for some S: by Corollary 3.30 a factor Q satisfies $\|Q\|_\infty \leqslant B$, and we certainly have $Q \equiv A_S \pmod{p^\ell}$ for a unique S (since the T_i are distinct: apply Hensel Lemma and reduce mod p^ℓ the factorization in $\mathbb{Z}_p[X]$). Hence

$$\|Q - A_S\|_\infty \leqslant B + \frac{p^\ell}{2} < p^\ell,$$

by the choice of ℓ. Since this polynomial is divisible by p^ℓ, is is identically 0 and $Q = A_S$.

It remains to prove that the A_S we output are irreducible. This is guaranteed by the order in which we consider the set S: if A_S is not irreducible, then $A_S = BC$ in $\mathbb{Q}[X]$, for non-constant B and C. By what we have just proved B is of the form $A_{S'}$ for a proper subset S' of S, which cannot occur because smaller sets are considered first and the indices in S' would already have been removed from S_0. □

Remark 3.32. In the loop over the subsets S, we may stop as soon as $\#S > \#S_0/2$, since A_S is a factor if and only if $A_{S'}$ is a factor, with S' the complement of S in S_0. In particular, we consider at most 2^{r-1} sets S.

Remark 3.33. There is a "big prime" variant of this algorithm where we directly chose $p > 2B$ and skip the Hensel lift (take $\ell = 1$). But this is a bad idea since Hensel lifting is essentially linear time ($\widetilde{O}(n \log p^{\ell})$) whereas factorization in $\mathbb{F}_p[X]$ for big p definitely is not: $\widetilde{O}(n^2 \log p + \log^2 p)$ using the fastest randomized algorithms from Section 3.1.6. Also it is not obvious a priori how to construct a "big prime", whereas small primes are easy to find using a sieve. (See later.)

We now examine the various costs involved, letting $n = \deg T$, $A = \|T\|_{\infty}$. We make use of the following classical result in analytic number theory (see Tenenbaum [23])

Theorem 3.34 (Prime Number Theorem (PNT))**.** *As $x \to +\infty$, we have*

$$\Theta(x) := \sum_{p \leqslant x} \log p \sim x.$$

Effectively, one has $\Theta(x) > 0.98x$ for $x > 7481$.

The usual, more transparent, formulation is in terms of the function $\pi(x) := \# \{p \leqslant x\}$, satisfying $\pi(x) \sim x/\log x$. The above function is the one which turns up in our case.

Exercise 3.35. Using integration by part (Abel's summation), show that the two forms of the PNT are equivalent.

Theorem 3.36. *There exists a prime $p = O(n \log(nA))$ such that T is squarefree in $\mathbb{F}_p[X]$. In particular $\log p = O(\log n + \log\log A)$.*

Proof. Let $D = \operatorname{Res}(T, T')$. This is an integer (since $T, T' \in \mathbb{Z}[X]$), which is non-zero since T is squarefree in $\mathbb{Q}[X]$, hence $\gcd(T, T') = 1$. Since D is a polynomial in the coefficients of T and T' and reduction mod p is a ring homomorphism, the resultant in $\mathbb{F}_p[X]$ of $\bar{T}$ and $\bar{T}'$ is $\bar{D}$, and this is non-zero if and only if $\bar{T}$ is squarefree in $\mathbb{F}_p[X]$.

By Hadamard's bound and $\deg T' \leqslant n-1$, $|D| \leqslant \|T\|_2^{n-1} \|T'\|_2^{n}$, from which we obtain $\log |D| = O(n \log(nA))$. Assume now that all primes $p \leqslant x$ divide D, for some $x > 7481$. Then

$$\log |D| \geqslant \sum_{p \leqslant x} \log p > 0.98x,$$

which is a contradiction for $x \gg n \log(nA)$. □

Corollary 3.37. *The cost of finding a suitable p is $\widetilde{O}(n^2 \log A)$*

Proof. Reducing the $n+1$ coefficients of T modulo p costs $O(n \log A \log p)$. Once the inputs have sup norm less than p, computing $\gcd(\bar{T}, \bar{T}')$ costs $\widetilde{O}(n \log p)$, negligible before the reduction cost. Summing this for all $p = \widetilde{O}(n \log A)$, and using the PNT, we obtain $\widetilde{O}(n \log A)^2$. □

In practice, one chooses a few random primes less than *twice* an explicit upper bound for $|D|$ (T and T' are known !). This guarantees that less than half the primes up to the bound are unsuitable, so we expect $\leqslant 2$ trials before hitting one, replacing the above estimate by $\widetilde{O}(n \log A)$ (no longer deterministic). Even after hitting a suitable prime, we still compute a few more and choose the one such that the number of modular factors r is minimal.

Note that p is so small that we can factor deterministically in $\mathbb{F}_p[X]$ (in time exponential in $\log p$) and still remain polynomial-time with respect to n and $\log A$. This is important for our later theoretical result that factoring in $\mathbb{Q}[X]$ can be done in deterministic polynomial time. In practice, one uses a fast randomized algorithm, for a cost of $\widetilde{O}(n^2 \log p + n \log^2 p)$:

Corollary 3.38. *The (randomized) cost of factoring over* $\mathbb{F}_p$ *is* $\widetilde{O}(n^2 \log\log A + n(\log\log A)^2)$.

Corollary 3.39. *The cost of the Hensel lift is* $\widetilde{O}(n\ell \log p) = \widetilde{O}(n^2 + n \log A)$.

Proof. We have $B \leqslant 2^n\sqrt{n+1}A$ and want $p^\ell > 2B$, i.e. $\ell = \log(2B)/\log p + O(1)$. Hensel lifting costs

$$\widetilde{O}(n\ell \log p) = \widetilde{O}(n \log(2B) + n \log p).$$

Using the upper bound $\log B = O(n + \log(A))$ and $\log p = O(\log n + \log\log A)$, we obtain $\widetilde{O}(n^2 + n \log A)$. □

So far, so good. Unfortunately, the number of sets S to consider is a priori exponential in n. In fact, if T is irreducible, we have to consider 2^{r-1} sets. And for *bad* polynomials, r can be as large as $n/2$, independently of p. Take T the minimal polynomial of

$$\alpha_k = \sqrt{2} + \sqrt{3} + \cdots + \sqrt{p^k},$$

where p_k is the k-th prime number. The polynomial T is irreducible of degree $n = 2^k$, and $\mathbb{Q}(\alpha_k)/\mathbb{Q}$ is Galois with group $G = (\mathbb{Z}/2\mathbb{Z})^k$. A theorem of Frobenius relates the cycle structure of elements of G (viewed as conjugacy classes in S_n) with the factorization of T modulo p. In this example, it asserts that T has either n or $n/2$ factors in $\mathbb{F}_p[X]$, whenever it is squarefree.

More generally, if T is irreducible, generating a Galois extension of $\mathbb{Q}$ with Galois group of exponent e, we have $r \geqslant n/e$. Hence Zassenhaus algorithm runs in deterministic polynomial time up to the last loop, which unfortunately is exponential time in n. (But not in $\log A$: for small degrees, it is fine.)

On the other hand, *if* there exist primes modulo which r (number of modular factors) is not much bigger than s (number of true factors), then effective forms of the theorem of Chebotarëv tell us that a relatively small such p exist.

3.2.4 The LLL algorithm

This provides a replacement for the last step in Zassenhaus algorithm, avoiding the exhaustive enumeration which make it exponential time in the worst case. This results in a deterministic polynomial time algorithm, though not a very practical one.

Lemma 3.40. *Let $f, g \in \mathbb{Z}[X]$ have positive degrees. Suppose $u \in \mathbb{Z}[X]$ is non-constant, monic, and divides f and g in $(\mathbb{Z}/m\mathbb{Z})[X]$, for some $m > |\mathrm{Res}(f,g)|$. Then $\gcd(f,g) \in \mathbb{Q}[X]$ is non-constant.*

Proof. There exist $s, t \in \mathbb{Z}[X]$ such that $sf + tg = \mathrm{Res}(f,g) =: R \in \mathbb{Z}$. Hence $\bar{u}$ divides $\bar{R}$ in $(\mathbb{Z}/m\mathbb{Z})[X]$; since u is monic, non-constant, $\bar{u}$ has degree $\geqslant 1$ which implies $\bar{R} = 0$. Since $m > |R|$, $R = 0$ and f, g have a common factor in $\mathbb{Q}[X]$. □

The idea is to use the lemma together with the LLL algorithm as follows: let $m = p^\ell$, $f = T \in \mathbb{Z}[X]$ of degree n and $u \in \mathbb{Z}[X]$ which is an irreducible factor of T modulo m, e.g. $u = T_1^*$ from Zassenhaus algorithm. Then look for a $g \in \mathbb{Z}[X]$ of degree $k < n$ such that

- $u \mid g$ modulo m,
- $\|g\|_2^n < m \|T\|_2^{-k}$, which implies $|\mathrm{Res}(T,g)| < m$ by Hadamard.

If T is reducible, then there exists such a “small” g, provided m is large enough. How large ? We know $k \leqslant n-1$ and $\|g\|_2 \leqslant B$, hence want $m > \|T\|_2^{n-1} B^n$. We only need to express the divisibility condition with a lattice and we have a short vector problem! From any short vector, the lemma finds a factor. On the other hand if no short vector is found, or the short vector does not yield a factor, it proves that T was irreducible.

So let $j \geqslant \deg u$ and $\Lambda_j \subset \mathbb{Z}[X]$ be the lattice (of rank j) generated by the

$$\left\{uX^i, i < j - \deg u\right\} \cup \left\{mX^i, i < \deg u\right\}$$

Lemma 3.41. *$g \in \Lambda_j$ if and only if $\deg g < j$ and u divides g modulo m.*

Lemma 3.42. *Let $m \geqslant 2^{n^2/2} B^n$, and g the first vector in an LLL-reduced basis of Λ_{n-1}. Then either $\gcd(g,T)$ is a non trivial factor of T or T is irreducible.*

Proof. Assume T reducible. Then there exists h in Λ_j which divides T in $\mathbb{Z}[X]$ (take the irreducible rational factor containing T_1^*). From Landau's bound, we know $\|h\|_2 \leqslant B$. Let $j = n-1$; by the properties of LLL-reduced bases,

$$\|g\|_2 \leqslant 2^{(j-1)/2} \|h\|_2 \leqslant 2^{n/2} B.$$

Then Lemma 3.40 ensures we get a factor provided

$$\|g\|_2^n \|T\|_2^{n-1} < m.$$

Now use the trivial bound $\|T\|_2 \leqslant B$. □

Corollary 3.43. *In order to apply the LLL method we must lift modulo p^ℓ, where $\log p^\ell = O(n^2 + n\log A)$, for a cost $\widetilde{O}(n(n^2 + n\log A))$.*

Note that this lifting bound is n times larger than Landau's bound used in Zassenhaus algorithm.

Theorem 3.44. *Let $T \in \mathbb{Z}[X]$ be monic. We can factor T in $\mathbb{Q}[X]$ in deterministic polynomial time $\widetilde{O}(n^{10} + n^8\log^2 A)$.*

Proof. An LLL reduction costs $\widetilde{O}(n^5\log^2 m) = \widetilde{O}(n^9 + n^7\log^2 A)$, the final modular gcd has negligible cost $\widetilde{O}(n^2 + n\log B)$ by Theorem 1.17. We must perform the above at most $2n$ times since T has at most n factors (so at most n splittings and at most n failures denoting an irreducible polynomial). □

A practical improvement is to take for u some T_i^*, then consider successively $\Lambda_{\deg u}$, $\Lambda_{2\deg u}$, …, $\Lambda_{2^k\deg u}$, trying to guess the degree of a factor g containing u by dichotomy. The cost now becomes $\widetilde{O}(d^9 + d^7\log^2 A)$ to find a factor of degree d. Such a factor may be reducible, but any factor containing u has degree $> d/2$. By dichotomy (knowing a factor of degree d_2 and that no factor of degree d_1 exist, we try for degree $(d_1 + d_2)/2$) we arrive at the irreducible factor of degree d containing u in $O(\log d)$ steps, and the cost remains $\widetilde{O}(d^9 + d^7\log^2 A)$. Since $\sum d = n$, we have $\sum d^t \leqslant (\sum d)^t = n^t$ for any $t > 0$ and the total cost goes down to $\widetilde{O}(n^9 + n^7\log^2 A)$.

Another improvement is to try smaller lattices or lift to smaller accuracy than p^ℓ: if the polynomial is reducible, we may get lucky and find a factor. Unfortunately, when the polynomial is irreducible, there is no way to avoid computing up to the worst case bounds. This is still horrendously expensive for large degrees, e.g. $n = 1000$: 10^{27} operations are beyond the capabilities of any current computer.

3.2.5 Van Hoeij's algorithm

Van Hoeij's algorithm (2002), in the version we shall describe, has the same *proven* complexity as LLL (in particular polynomial time) but is orders of magnitude faster in practice. Moreover, it gives all the factors at once, whereas LLL gives just one. Variants of van Hoeij's method can be proven to perform better than LLL, but they are more technical to prove and describe so we will be content with heuristics at this point. The setup is the same as for Zassenhaus and LLL:

Input: $f \in \mathbb{Z}[X]$, monic, square-free, of degree n, $f = \prod_{i=1}^r g_i \in \mathbb{Z}_p[X]$. The $g_i \in \mathbb{Z}_p[X]$ are monic, irreducible, pairwise distinct, and $\mathrm{MOD}\left(g_i, p^\ell\right) \in \mathbb{Z}[X]$ is known for all i.

Output: $f = \prod_{i=1}^s f_j \in \mathbb{Q}[X]$, $s \leqslant r$. The $f_j \in \mathbb{Z}[X]$, are monic and irreducible.

Notations

Let $G_p = \langle g_1, \cdots, g_r\rangle$ be the multiplicative subgroup of $\mathbb{Q}_p(X)^*$ generated by the g_i and $G_\mathbb{Q} = \langle f_1, \cdots, f_s\rangle$ the subgroup generated by the f_j. Note that $G_\mathbb{Q}$ is a subgroup of G_p and they both are free $\mathbb{Z}$-modules of rank r and s respectively. Our goal is to compute $G_\mathbb{Q}$; this is sufficient because the HNF algorithm enables us to deduce the f_j from any $\mathbb{Z}$-basis of $G_\mathbb{Q}$:

Lemma 3.45. *Using (g_i) as a fixed basis for G_p, a suitable permutation of the (f_j) form an HNF basis for $G_\mathbb{Q}$. More precisely, if $f_j = \prod_i g_i^{h_{i,j}}$ and $H = (h_{i,j})$ is a matrix with $\{0,1\}$-coefficients then, up to a reordering of its columns, H is in HNF.*

Proof. This is true for any matrix with $\{0,1\}$ coefficients containing a most a single 1 on each line. □

Corollary 3.46. *If the $b_j = \prod_i g_i^{a_{i,j}}$, $j \leqslant s$, form a basis of $G_\mathbb{Q}$, then in the notations of the Lemma, H is the HNF of $(a_{i,j})$.*

The idea is to find a basis of $G_\mathbb{Q}$ as a set of short vectors. First we linearize the problem. Let φ be the following map:

$$\begin{array}{rcl}\varphi\colon (G_p, \times) & \to & (\mathbb{Q}_p(X), +)\\ g & \mapsto & f\frac{g'}{g}\end{array}$$

From the derivative laws $(fg)' = f'g + g'f$ so $\frac{(fg)'}{fg} = \frac{f'}{f} + \frac{g'}{g}$ and φ is a morphism. In fact,

$$\varphi\Big(\prod_{i=1}^{r} g_i^{e_i}\Big) = \sum_{i=1}^{r} e_i g_i' \frac{f}{g_i} \in \mathbb{Q}_p[X],$$

with degree $< n$. Analogously, we see that $\varphi\left(G_\mathbb{Q}\right) \subset \mathbb{Z}[X]_{<n}$.

More generally, define

$$\Phi(g_i) = \varphi(g_i) + X^{n+r-i},$$

and extend it additively to G_p. In effect, we concatenate an identity matrix above the matrix giving the $\varphi(g_i)$. This will keep track of column operations; in a short vector algorithm, it further ensures that perturbations of the input remain small.

It is difficult to work with $\varphi\left(G_p\right)$ because it is not a $\mathbb{Z}$-module of finite type, and the g_i are not known exactly. So we introduce the lattice Λ, which is the $\mathbb{Z}$-module generated by $\Phi(g_i)$, $i \leqslant r$ and by $p^\ell X^i$, $i \leqslant n$. Expressed on the natural basis $1, X, \ldots, X^{n+r-1}$, Λ is generated by the columns of the following matrix:

$$\begin{pmatrix}\mathrm{Id}_r & 0\\ A & p^\ell\,\mathrm{Id}_n\end{pmatrix},$$

where A is the $n \times r$ matrix of the MOD $\left(\varphi(g_i), p^\ell\right)$, written by decreasing degree. (Recall that $\varphi(g_i)$ has degree $< n$.) A generic vector in Λ has the form

$$\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_r \\ \sum_i e_i \text{MOD}\left(\varphi(g_i), p^\ell\right) + p^\ell Q \end{pmatrix}$$

with $Q \in \mathbb{Z}[X]_{<n}$. The above vector is $\Phi(\prod g_i^{e_i})$, where the part of degree $< n$ is taken modulo p^ℓ, and

$$\left(\Phi(G_p) + p^\ell \mathbb{Z}_p[X]_{<n}\right) \cap \mathbb{Z}[X] = \Lambda.$$

The $\Phi(f_j)$ have $(e_1, \ldots, e_r)$ in $\{0, 1\}$. Compared to LLL, the base change matrix is very simple: in LLL it is given by the coefficients of a factor, which may be large. We shall prove that, if the accuracy p^ℓ is large enough, the first s vectors in an LLL-reduced basis of Λ will form a basis of $\Phi(G_{\mathbb{Q}})$, enabling us to find $G_{\mathbb{Q}}$ and the rational factors. The recipe to find s is simple: eliminate the big basis vectors. But first we need some lemmas:

Lemma 3.47 (Mahler, 1961). *Let $A \in \mathbb{C}[X]$, then $M(A') \leqslant \deg(A) M(A)$.*

Proof. The proof is surprisingly difficult and we shall only sketch it, see [10, Appendix D] for details. Letting $\alpha_1, \ldots, \alpha_d$ and $\beta_1, \ldots, \beta_{d-1}$ be the complex roots of A and A' respectively, we must prove

$$\prod_{j=1}^{d-1} \max(1, |\beta_j|) \leqslant \prod_{j=1}^{d} \max(1, |\alpha_j|).$$

For given $\alpha_1, \ldots, \alpha_{d-1} \in \mathbb{C}$, and $t \in \mathbb{R}$, let

$$f_t(\alpha_d) = \sum_{j \leqslant d} \frac{1}{e^{2i\pi t} - \alpha_j}.$$

Using Jensen's formula, Mahler's measure admits the alternative expression

$$\log M(A) = \int_0^1 \log \left|A(e^{2i\pi t})\right| \, dt.$$

(The integral is well-defined, since any singularity is of the form log(polynomial).) Using

$$\frac{A'}{A}(z) = \sum_{j \leqslant d} \frac{1}{z - \alpha_j},$$

a simple modification of the proof shows that

$$\int_0^1 \log|f_t(\alpha_d)|\,dt = \log\frac{M(A')}{M(A)} = \log d + \log\prod_{j=1}^{d-1}\max(1,|\beta_j|) - \log\prod_{j=1}^{d}\max(1,|\alpha_j|).$$

Hence the inequality to prove becomes equivalent to

$$\int_0^1 \log|f_t(\alpha_d)|\,dz \leqslant \log d. \tag{3.1}$$

The function $\alpha_d \mapsto \log|f_t(\alpha_d)|$ is subharmonic on $\mathbb{C}\setminus\{z\colon\ |z|=1\}$, hence so is

$$\alpha_d \mapsto \int_0^1 \log|f_t(\alpha_d)|\,dt.$$

From the maximum modulus principle for subharmonic functions applied to $\{z\colon\ |z|<1\}$ and $\{z\colon\ |z|>1\}$, it follows that the integral in (3.1) above is maximal for some α_d with $|\alpha_d|=1$. By symmetry, it is enough to prove the inequality when $|\alpha_1|=\cdots=|\alpha_d|=1$.

In that case, it follows immediately from Lucas's theorem: the zeroes of A' are in the convex hull of the zeroes of A, which implies that $|\beta_j|\leqslant 1$ for all j. To prove Lucas's result, let β be a root of A', then

$$0 = \frac{A'}{A}(\beta) = \sum_{j=1}^{d}\frac{1}{\beta-\alpha_j} = \sum_{j=1}^{d}\frac{\overline{\beta-\alpha_j}}{|\beta-\alpha_j|^2}.$$

Taking the conjugate, it follows that β is a linear combination of the α_j with *positive* coefficients. □

Lemma 3.48. *If $g\in\mathbb{Z}[X]$ divides f, then $\|\varphi(g)\|_2 \leqslant n2^n\cdot\|f\|_2$*

Proof. As in Landau's theorem, let $M(A)$ be the Mahler measure of A. For two polynomials A, B, we have $M(AB)=M(A)M(B)$. Remember that $M(A)\leqslant \|A\|_2 \leqslant 2^{\deg A}M(A)$. Since $\deg\varphi(g)<n$ and $\frac{f}{g}$ is a polynomial, we have

$$\|\varphi(g)\|_2 \leqslant 2^n\cdot M(fg'/g) = 2^n M(g')M(f/g).$$

Using Mahler's lemma and Landau's estimate,

$$M(g')M(f/g)\leqslant n2^n\cdot M(g)M(f/g) = n2^nM(f)\leqslant n2^n\cdot\|f\|_2\,.$$

□

Corollary 3.49. *For all $j\leqslant s$, $\|\Phi(f_j)\|_2 \leqslant n2^n\,\|f\|_2+\sqrt{r}$.*

Proof. Exercise. □

For $g = \prod_i g_i^{e_i} \in G_p$, we define the *support* of g by

$$\operatorname{Supp}(g) = \{i \colon e_i \neq 0\}.$$

Lemma 3.50. *$g_i \mid \varphi(g) \Leftrightarrow i \notin \operatorname{Supp}(g)$.*

Proof. Since φ is a morphism, we have $\varphi(g) = \sum_i e_i \varphi(g_i)$. Now, $g_i \mid \varphi(g_j)$ if and only if

$$g_i \mid \frac{g_j'}{g_j} \cdot f = \prod_{k \neq j} g_k \cdot g_j'$$

Since g_j is irreducible, and $\mathbb{Q}_p$ has characteristic 0, g_j' and g_j are coprime. Hence $g_i \mid \varphi(g_j) \Leftrightarrow i \neq j$. Since g_i divides all the summands in $\varphi(g)$ except the i-th one, $g_i \mid \varphi(g)$ if and only if the latter is 0, i.e. $e_i = 0$. □

Lemma 3.51. *Let G be a subgroup of G_p containing strictly $G_\mathbb{Q}$ and $u \in G \setminus G_\mathbb{Q}$. Then there exists $g \in G \setminus G_\mathbb{Q}$ such that:*

1. *$g_i \mid \varphi(g)$ for some $i \leqslant r$*

2. *$f_j \nmid \varphi(g)$ for all $j \leqslant s$*

3. *$\operatorname{MOD}\left(\Phi(g), p^\ell\right)$ is not much larger than $\operatorname{MOD}\left(\Phi(u), p^\ell\right)$.*

Proof. We start with $g = u$ and modify it until the conditions hold. Because of Lemma 3.50, the first two conditions mean that $i \notin \operatorname{Supp} g$ and $\operatorname{Supp} f_j \cap \operatorname{Supp} g \neq \emptyset$ for all j.

So for all j such that $\operatorname{Supp}(f_j) \cap \operatorname{Supp}(g) = \emptyset$, we we replace g by gf_j. After this step, the second condition is satisfied. Since we have multiplied $u \notin G_\mathbb{Q}$ by elements in the subgroup $G_\mathbb{Q}$, the resulting g is not in $G_\mathbb{Q}$.

Write $g = \prod_i g_i^{e_i}$. We want to make sure that one of the e_i is zero. For $j \leqslant s$, let $S_j = \{e_i \colon i \in \operatorname{Supp} f_j\}$. There exists j such that $\#S_j > 1$, otherwise $S_j = \{s_j\}$ for all j and $g = \prod_j f_j^{s_j} \in G_\mathbb{Q}$, a contradiction.

Pick one $i \in S_j$, since g_i divides f_j to the first power, we replace g by $g/f_j^{e_i}$, ensuring that g_i does not divide g, without affecting the other conditions.

As for the last condition, the $\Phi(f_j)$ are bounded by Corollary 3.49, and so is e_i. □

Lemma 3.52. *Let (Λ, q) be a lattice, $(b_1, \ldots, b_n)$ a basis of Λ and $(b_1^*, \ldots, b_n^*)$ the associated Gram-Schmidt orthogonalized basis. Assume $q(b_j^*) > B$, for all $j > j_0$. Then if $v \in \Lambda$ satisfies $q(v) \leqslant B$, we have $v \in \langle b_1, \ldots, b_{j_0} \rangle_\mathbb{Z}$.*

Proof. Exercise (look at the proof of Theorem 2.36). □

This is true for any basis, but of course an LLL-basis is most suitable: since $2q(b_{i+1}^*) \geqslant q(b_i^*)$, the $q(b_i^*)$ do not decrease too fast. In fact, we expect them to increase quickly if the basis contains both small and large vectors. The idea is then that we expect non-rational inputs taken mod p^ℓ to yield big integers. (Just like we expect the average irrational number not to have a long sequence of 0 in its decimal development.)

Algorithm 17. van Hoeij's algorithm

```
1: Compute an LLL-reduced basis $(b_1, \ldots, b_m)$ of the lattice $\Lambda$ generated by
   $$\left\{\mathrm{MOD}\left(\Phi(g_i), p^\ell\right), i \leqslant r\right\} \cup \left\{p^\ell X^i, i < n\right\}.$$
2: Let $B = 2^n(n\|f\|_2 + \sqrt{r})$.
3: Let $s = j_0$ be minimal such that $\left\|b_j^*\right\|_2 > B$ for all $j > j_0$.   {we have $\Lambda_{\mathbb{Q}} \subset \langle b_1, \ldots, b_{j_0}\rangle_{\mathbb{Z}}$.}
4: Return $(b_1, \ldots, b_s)$
```

Theorem 3.53. *There exists an effective $C = O(n^2 + n\log\|f\|_2)$ such that if $\log(p^\ell) > C$, the above algorithm returns a basis of $\Lambda_{\mathbb{Q}}$.*

Proof. Assume there exists $i \leqslant j_0$ such that $b_i \notin \Lambda_{\mathbb{Q}}$. Then there exists a vector $\mathrm{MOD}\left(\Phi(g), p^\ell\right)$ in $\langle b_1, \ldots, b_{j_0}\rangle_{\mathbb{Z}} \setminus \Lambda_{\mathbb{Q}}$ not much larger than b_i satisfying the conditions of Lemma 3.51. Let us be more precise: since $\left\|b_{j_0}^*\right\|_2 \leqslant B$, by definition of j_0, we have $\|b_i^*\|_2 \leqslant 2^{(j_0-i)/2}B$ from Siegel condition, and

$$\|b_i\|_2 \leqslant 2^{(i-1)/2}\|b_i^*\|_2 \leqslant 2^{(j_0-1)/2}B \leqslant 2^{(n-1)/2}B,$$

where we have used Lemma 2.38. To go from b_i to $\Phi(g)$, we follow the proof of Lemma 3.51,

- we first add $\Phi(\prod f_j)$ for some factor $\prod f_j$ of f, whose length is less than B, so the resulting vector has length $\leqslant (2^{(n-1)/2}+1)B$, which is an upper bound for $|e_i|$,
- we then subtract $e_i\Phi(f_i)$ whose length is less than $|e_i|\,B$.

Let $G = \mathrm{MOD}\left(\varphi(g), p^\ell\right) \in \mathbb{Z}[X]$ be the part with degree $< n$. Because of the above, $\|G\|_2 \leqslant (2^{(n-1)/2}+1)B(1+B) =: C$. Because of Lemma 3.51, we have

1. $\mathrm{Res}(G, f) \equiv 0 \pmod{p^\ell}$,
2. $\mathrm{Res}(G, f) \neq 0$,
3. $|\mathrm{Res}(G, f)| < C^n\|f\|_2^{n-1} =: R$ is effectively bounded.

The last point is Hadamard's inequality, with $\deg G < n$. If $p^\ell \geqslant R$, we have a contradiction. Finally, one easily checks that $\log R = O(n^2 + n \log \|f\|_2)$. □

This version of Van Hoeij's Algorithm turns out to have the same proven complexity as LLL. But there are a number of practical improvements, which make it superior in practice:

First we do not work with Λ but with the $\mathbb{Z}$-module generated by the columns of the matrix

$$\begin{pmatrix} I_r & 0 \\ A_1 & B_1 \end{pmatrix},$$

where A_1 is the first line of the $n \times r$ matrix A, and B_1 is the first line of $p^\ell I_n$. We apply LLL to this lattice and eliminate large vectors as above. This gives us a new lattice generated by the columns of a matrix $\begin{pmatrix} M \\ N \end{pmatrix}$. We work now with the lattice generated by by the columns of

$$\begin{pmatrix} M & 0 \\ A_2 & B_2 \end{pmatrix},$$

where A_2 is the second line of the matrix $A \times M$. And so on introducing successively the lines of A (suitably updated from the successive base changes). We expect to eliminate quickly many basis vectors, as if we had worked with all lines of A simultaneously.

A second similar improvement is to take into account only the leading p-adic digits of A (we need to modify the bounds for the discarded vectors of course) then iterate, feeding more and more digits.

Intuitively, we expect to quickly detect true small vectors in the original lattice Λ from these approximate (simpler) lattices: since the small vectors in Λ are small perturbations of the input vectors, they are short vectors in all these other lattices and we hope to detect them early, *before* all the lines and all the digits are input. Theorems exist to support this intuition but are a little harder to formulate and prove!

From these heuristics, the lattices we reduce in practice have much smaller size than the ones in LLL. As a result, van Hoeij's Algorithm is the current best practical method to factor polynomials in $\mathbb{Q}[X]$.

3.3 Factoring in $\mathbb{C}[X]$

We fix $\varepsilon > 0$ and $P \in \mathbb{Q}(i)[X] \subset \mathbb{C}[X]$, P is monic and of degree n. We want to find $u_1, \cdots, u_n, \in \mathbb{Q}(i)$ such that

$$\left\| P - \prod_{i=1}^{n} (X - u_i) \right\|_1 \leqslant \varepsilon \|P\|_1 . \tag{3.2}$$

Theorem 3.54 (Schönhage)**.** *If the coefficients of P are given by floating point numbers, we can compute n floating point numbers $(u_1, \dots, u_n)$ satisfying* (3.2) *in time*

$$\widetilde{O}(n^3 \ln(s) + n^2 s), \quad \textit{with} \quad \varepsilon = 2^{-s}.$$

Remark 3.55. It is an impressive result, because it does not depend on the *size* of the input polynomial but only on the degree and requested accuracy. We shall not prove the theorem, which is rather technical, but introduce the main ideas.

Remark 3.56. The problem of root finding is ill-conditioned, i.e. sensitive to perturbations (hence difficult), even though the roots are a continuous function of the coefficients (of monic polynomials of given degree). Indeed, let $P = X^n$ and $\widehat{P} = X^n - \varepsilon^n$ for a small $0 < \varepsilon < 1$. The polynomials P and $\widehat{P}$ are very close but ε is a root of $\widehat{P}$ whose distance to the unique root 0 of P is ε, much larger than the distance from P to $\widehat{P}$ ($= \varepsilon^n$).

Remark 3.57. If $P = \prod_i (X - z_i)$ then $|u_i - z_i|$ will be small (see Ostrowski's Theorem 3.64). The problem in this form is well suited for approximate inputs: instead of estimating the error between the result and the true roots, we estimate the distance from the input to a virtual input which would produce the approximate roots as an exact result. (Backward error analysis.)

3.3.1 Idea of this algorithm

The naive root finding idea is to use random Newton iterations : pick a random $x_0 \in \mathbb{C}$ and for $n \geqslant 0$, define

$$x_{n+1} = x_n - \frac{P(x_n)}{P'(x_n)}.$$

This sequence will probably converge to a root of P. It works nicely for small degrees but it is quite unstable, especially if the degree of P is big (say, $\deg P \geqslant 10$) or if $\|P\|_\infty$ is big. Already in degree 3, consider all polynomials in a given ball B, whose roots belong to some other ball Ω. There exists a set of polynomials of positive measure in B, such that a positive measure of initial points in Ω do not yield converging sequences.

Schönhage's idea is deterministic and recursive.

1. Look for a separating circle Γ containing exactly k roots of P: $u_1, u_2, \cdots, u_k$. Ideally, $k \approx n/2$ and the roots of P are well away from Γ (Lemma 3.58).

2. Use Cauchy formula: for all $m \leqslant k$, approximate the Newton sums

$$s_m := u_1^m + u_2^m + \cdots + u_k^m = \frac{1}{2i\pi} \oint_\Gamma \frac{P'(z)}{P(z)} \cdot z^m \, dz.$$

3. Thanks to Newton formulas, from approximations of the s_m, $m \leqslant k$, we obtain an approximation P_0 of $\prod_{i=1}^{i=k}(X - u_i)$.

4. Reapply to P_0 and P/P_0.

In order to obtain an efficient algorithm, one must prove perturbation results so as to use floating point arithmetic (incurring rounding errors) with smallest possible accuracy and still realistic error bounds.

In practice, numerical integration is very costly and we alternate these with Newton-like iterations (seeded by the values from the integrals), falling back to integration when the iteration diverges. We will neglect these aspects, see [11] for details.

Then the main problem is to find the separating circle Γ, which ultimately depends on our ability to approximate the modulus of the roots.

3.3.2 Numerical integration

In this section we assume Γ is known. By translation and scaling, we may further assume Γ is the unit circle. We discretize Γ at H-th roots of 1 and approximate the integral by Riemann sums:

Lemma 3.58. *Let*

$$W_m = \frac{1}{H}\sum_{j=0}^{H-1}\frac{P'}{P}(\omega^j)\omega^{(m+1)j}, \quad \omega = e^{2i\pi/H}.$$

Assume P of degree n has no root in the annulus $e^{-\delta} < |z| < e^{\delta}$, for some $\delta > 0$. Then for all $m \leqslant n < H$, we have

$$|W_m - s_m| \leqslant \frac{ne^{-\delta(H-m)}}{1 - e^{-\delta H}} = O_{n,\delta}(e^{-\delta H}).$$

Proof. Let $u_1, \dots, u_k$ be the roots of P within Γ, and $u_{k+1}, \dots, u_n$ the others. By assumption,

$$|u_1|, \dots, |u_k|, |u_{k+1}|^{-1}, \dots, |u_n|^{-1}$$

are all less than $e^{-\delta} < 1$. Let

$$s_m = \sum_{i \leqslant k} u_i^m, \quad S_m = \sum_{i > k} u_i^{-m}$$

and develop $(P'/P)(\omega^t)$ in Fourier series,

$$\frac{P'}{P}(\omega^j) = \sum_{\ell \in \mathbb{Z}} c_\ell \omega^{\ell j}.$$

From the logarithmic derivative formula, one obtains

$$\begin{aligned}\frac{P'}{P}(\omega^j) &= \sum_{i\leqslant k}\frac{1}{\omega^j-u_i}+\sum_{k<i}\frac{1}{\omega^j-u_i}\\ &= \sum_{i\leqslant k}\omega^{-j}\sum_{\ell\geqslant 0}(u_i\omega^{-j})^\ell-\sum_{k<i}u_i^{-1}\sum_{\ell\geqslant 0}(\omega^j u_i^{-1})^\ell\\ &= \sum_{\ell\geqslant 0}s_\ell\omega^{-j(\ell+1)}-\sum_{\ell\geqslant 0}\omega^{j\ell}S_{\ell+1}.\end{aligned}$$

So that

$$c_\ell=\begin{cases}-S_{\ell+1} & \text{if } \ell\geqslant 0,\\ s_{-\ell-1} & \text{if } \ell<0.\end{cases}$$

From this we gather $W_m=\sum c_\ell$, where $m+1+\ell\equiv 0 \pmod H$, and $s_m=c_{-(m+1)}$. Finally, $W_m-s_m=\sum c_\ell$, where ℓ is of the form $-(m+1)+\lambda H$, $\lambda\in\mathbb{Z}\setminus\{0\}$.

From $|s_m|\leqslant ke^{-m\delta}$ and $|S_m|\leqslant(n-k)e^{-m\delta}$, we get

$$\begin{aligned}|W_m-s_m| &\leqslant \sum_{\lambda>0}(n-k)e^{-\delta(\lambda H-m)}+\sum_{\lambda<0}ke^{-\delta(-\lambda H+m)}\\ &\leqslant e^{\delta m}(k+(n-k))\sum_{\lambda>0}e^{-\delta\lambda H}.\end{aligned}$$

□

Remark 3.59. If n is large, to compute the Riemann sums W_m, we choose H a power of 2 and use the FFT to compute the $P(\omega^j)$ and $P'(\omega^j)$ (saving essentially a factor n).

3.3.3 Choosing Γ

Let $(u_1,\dots,u_n)$ be the roots of $P=\sum_{i\leqslant n}a_iX^i$, ordered by increasing modulus, and $\rho_1(P)\leqslant\ldots\leqslant\rho_n(P)$ their absolute values. We assume $n\geqslant 2$. In this section we show that if we can approximate the ρ_i, then we can find a suitable Γ and compute k and δ as above. We always assume that $\rho_1(P)>0$ (since we can first remove the roots equal to 0).

1. Translate: replacing P by $P(X-a_{n-1}/a_n)$, we can assume that the barycenter of the u_i is 0, i.e. that $\sum u_i=0$.

2. Rescale: replacing P by $P(X/\rho_n)$, we can assume that the largest root is close to the unit circle.

3. Test for center: let $S=\{\pm 2,\pm 2i\}$. From Lemma 3.60, there exists a point $\Omega\in S$ such that ρ_n/ρ_1 for $P(X-\Omega)$ is bigger than $2/\left|2-e^{i\pi/4}\right|\approx 1.35$. Fix the center of Γ at Ω.

4. Choose radius: by translation again, we may now assume that Γ is centered at 0. If $\rho_i < R < \rho i + 1$ is the chosen radius, the optimal δ such that no z in the annulus $Re^{-\delta} < |z| < Re^{\delta}$ is a root of P satisfies
$$e^{\delta} = \min(\rho_{i+1}/R, R/\rho_i).$$
This is maximal for $R = \sqrt{\rho_i \rho_{i+1}}$, with value $e^{\delta} = \sqrt{\rho_{i+1}/\rho_i}$. So we choose k such that ρ_{k+1}/ρ_k is maximal and set the radius R of Γ as above.

   Since the product of the ρ_{i+1}/ρ_i is bigger than 1.35, one of them is $> 1.35^{1/n}$. Finally, $e^{\delta} > 1.35^{1/2n}$ is bounded away from 1.

Lemma 3.60. *With the notations of 3), let $\Omega \in S$ a closest point to a largest root of P. Then ρ_n/ρ_1 for $P(X - \Omega)$ is bigger than $2/\left|2 - e^{i\pi/4}\right|$.*

Proof. There exists a root u of P such that $|\Omega - u| \geqslant 2$ (since 0 is their center of gravity). There exists a root v of P such that $|\Omega - v| \leqslant \left|2 - e^{i\pi/4}\right|$ (a closest root). The result follows. □

3.3.4 Estimate $\rho_k(P)$ (Graeffe's method)

We start with two lemmas, giving very rough bounds, see [12].

Lemma 3.61. *Let $P = \sum_{i \leqslant n} a_i X^i \in \mathbb{C}[X]$ with $a_0 a_n \neq 0$ and $n \geqslant 2$. Let $k \geqslant 0$ such that $|a_k| = \max_i |a_i|$. Then $\rho_k(P) \leqslant 2n$ and $\rho_{k+1}(P) \geqslant 1/2n$.*

Proof. The second bound follows from the first applied to the reciprocal polynomial $X^n P(1/X)$ of P. The case $k = 0$ is Cauchy's bound 2.48. The case $k > 0$ follows from Cauchy's bound applied to P/Q, $Q = (X - u_n) \dots (X - u_{k+1})$. Details left as an exercise (use the series development of $1/Q$). □

Lemma 3.62. *Let $P = \sum_{i \leqslant n} a_i X^i$, $n \geqslant 1$. One can choose $r > 0$ such that $Q := P(rX) = \sum_{i \leqslant n} b_i X^i$ is such that there exists ℓ, h in $[0, n]$, with*
$$\ell < k \leqslant h, \quad |b_\ell| = |b_h| \geqslant |b_j|, \quad \forall j \leqslant n).$$
In particular
$$\frac{1}{2n} \leqslant \rho_{\ell+1}(Q) \leqslant \rho_k(Q) \leqslant \rho_h(Q) \leqslant 2n.$$

Proof. Let C be the upper convex hull of the $M_j = (j, \log |a_j|) \in \mathbb{R}^2$ (omit the points with $a_j = 0$). Let ℓ be the largest $j < k$ such that $M_j \in C$, and h the smallest $j \geqslant k$ such that $M_j \in C$. Then set $r = |a_\ell / a_h|^{1/(h-\ell)}$. □

We now use Graeffe's construction to "amplify" the behavior we want to detect: let $G(P)$ be the polynomial G such that $G(X^2) = P(X)P(-X)$ (the letter G stands for Gräffe, the method is also independently due to Dandelin and Lobachevsky). Obviously, G has the same degree as P and its roots are the squares of the roots of P.

Algorithm 18. Graeffe's method for $\rho_k(P)$

```
Input: $P \in \mathbb{C}[X]$, $k \leqslant \deg P$, $\delta > 0$
Output: $R$ such that $Re^{-\delta} \leqslant \rho_k(P) \leqslant Re^{\delta}$.
let $m$ be minimal such that $(2n)^{1/2^m} \leqslant e^{\delta}$.
Define $P_i$, $Q_i$, $r_i$ by
    $P_0 = P, \quad Q_j = P_j(r_j X), \quad P_{j+1} = G(Q_j),$
    where $(Q_j, r_j)$ come from Lemma 3.62.  {The roots of $Q_j$ are the $(u_i/R_j)^{2^j}$,
    $1 \leqslant i \leqslant n$, where $R_j = r_0 r_1^{1/2} \dots r_j^{1/2^j}$.}
Return $R_m$.
```

Proof. By Lemma 3.62, we have $1/2n \leqslant \rho_k(Q_m) \leqslant 2n$, hence

$$R_m(2n)^{-1/2^m} \leqslant \rho_k(P) \leqslant R_m(2n)^{1/2^m}.$$

□

3.3.5 Continuity of the roots

This is the following result:

Theorem 3.63. *Let $\mathcal{C}$ be the set of monic polynomials of degree n in $\mathbb{C}[X]$, equipped with the natural product topology (we identify $\mathcal{C}$ with $\mathbb{C}^n$). Then $T : \mathbb{C}^n/S_n \to \mathcal{C}$ given by $(z_1, \dots, z_n) \mapsto \prod_{i \leqslant n}(X - z_i)$ is a homeomorphism.*

Proof. Since $\mathbb{C}$ is algebraically closed (fundamental theorem of algebra), T is a bijection, obviously continuous (polynomial map). Let $S = T^{-1}$ and $P_k \in \mathcal{C}$ be such that $P_k \to P$, we want to prove that $S(P_k) \to S(P)$. But $S(P_k)$ is bounded (Cauchy's bound). By compacity, passing to a subsequence, we may assume that $S(P_k) \to Q$. By continuity of T, we have $P_k = T(S(P_k)) \to T(Q)$, hence $P = T(Q)$ and $Q = S(P)$. □

The following theorem is an effective version, giving concrete approximations to the roots of P from Schönhage's algorithm.

Theorem 3.64 (Ostrowski)**.** *Let $\rho = \max(1, |z_1|, \dots, |z_n|)$. Let $P \in \mathbb{C}[X]$ be monic such that*

$$\|P - (X - z_1)\dots(X - z_n)\|_1 < \varepsilon^n.$$

There exists a permutation $(u_1, \dots, u_n)$ of the complex roots of P such that

$$(1 - 4\varepsilon)|u_i - z_i| < 4\rho\varepsilon.$$

Proof. Let z be a root of $\widehat{P} = (X-z_1)\dots(X-z_n)$. Consider the continuous family of polynomials $H_t = tP + (1-t)\widehat{P} = \widehat{P} - t(\widehat{P} - P)$, $t \in [0,1]$. In particular, $H_1 = P$, $H_0 = \widehat{P}$. By continuity of the roots, there exists a continuous path $t \mapsto d_t(z) \in \mathbb{C}$ such that $d_0 = 0$ and $z + d_t$ is a root of H_t for all t. In particular, $u(z) := z + d_1$ is a root of P. (Choose the $d_t(z)$ such that $z \mapsto u(z)$ is 1-to-1.) Let $D = |u - z| = |d_1|$; by continuity of d_t, d_t takes at least all values in $[0, D]$.

On the other hand, we have

$$\left|\widehat{P}(z+d_t)\right| = t\left|(\widehat{P} - P)(z+d_t)\right| < \varepsilon^n(\rho + |d_t|)^n,$$

$$\left|\widehat{P}(z+d_t)\right| \geqslant \left|\prod_{i=1}^{n} |d_t| - |z - z_i|\right|.$$

Hence, for all $d \in [0, D]$,

$$\varepsilon^n(\rho + D)^n \geqslant \varepsilon^n(\rho + d)^n > \left|\prod_{i=1}^{n} d - |z - z_i|\right|.$$

By the minimax property of Chebyshev polynomials (Lemma 3.65), the right hand side is larger than $2(D/4)^n$ for some $d \in [0, D]$, whence $\varepsilon(\rho + D) > D/4$ and the result follows. □

For the following minimax result used above, see [5].

Lemma 3.65. *If $P \in \mathbb{R}[X]$ runs through the monic polynomials of degree n, we have*

$$\min_{P} \max_{x \in [a,b]} |P(x)| \geqslant \left(\frac{b-a}{2}\right)^n 2^{1-n} = 2\left(\frac{b-a}{4}\right)^n,$$

Remark 3.66. The minimal value is realized by a suitable Chebyshëv polynomial, e.g. if $[a,b] = [-1,1]$, $2^{1-n}T_n$ (where $T_n(X) = \cos\arccos(nX)$ for $X \in [-1,1]$).

Chapter 4

Integers

The main reference for this chapter is Cohen [6].

4.1 “Elementary” algorithms

4.1.1 Introduction

If N is a positive integer, given by a string of binary digits, we consider three basic different problems:

1. (Primality) let N be prime, prove it is so.
2. (Compositeness) let N be composite, prove it is so.
3. (Split) let N be composite, find a non trivial factor $d \neq 1, N$.

In fact, the problem we are actually most interested in would be

4. (Factor) factor N completely, i.e find all prime divisors p of N, the $v_p(N)$, and prove that each p is a prime number.

But an algorithm solving 4. also solves the three others. Conversely, solving 1., 2. and 3. implies we can solve 4.: apply 3. $O(\log N)$ times and run simultaneously 1. and 2. on each factor (stop whenever one succeeds). In theory, 2. is not needed, since 3. is obviously stronger (exhibiting a factor proves compositeness). In practice, ordering the problems by increasing order of difficulty: $2 \ll 1 \ll 3$, and we will consider them in this order. Some landmarks:

Theorem 4.1 (Rabin)**.** *Problem 2. can be solved in randomized time* $\widetilde{O}(\log N)^2$.

Theorem 4.2 (Agrawal-Kayal-Saxena, 2002)**.** *1. and 2. can be solved in polynomial time* $O(\log N)^{10.5}$.

This timing was later improved to $O(\log N)^{6+\varepsilon}$ by Lenstra and Pomerance.

Theorem 4.3 (Miller, 1976). *Assuming the Generalized Riemann Hypothesis holds, 1. and 2. can be solved in time* $\widetilde{O}(\log N)^4$.

Conjecture 4.4 (Goldwasser-Killian, Atkin, Shallit). *Fast variants of the ECPP algorithm solve 1. in randomized time* $\widetilde{O}(\log N)^4$.

If successful, the Elliptic Curve Primality Proving algorithm (ECPP) produces a primality proof, i.e. a tailor-made algorithm which proves the primality of N in time $\widetilde{O}(\log N)^3$. Something which Miller's algorithm is not capable of.

Definition 4.5. Let $0 \leqslant \alpha \leqslant 1$, we define

$$L_\alpha(N) = \exp\big((\log N)^\alpha (\log\log N)^{1-\alpha}\big).$$

Note that $L_0(N) = \log N$ and $L_1(N) = N$.

Theorem 4.6 (Lenstra-Pomerance). *The integer N can be factored in randomized time $L_{1/2}(N)$.*

Conjecture 4.7 (Pollard). *Using the Number Field Sieve algorithm (NFS), the integer N is factored in randomized time $L_{1/3}(N)$.*

4.1.2 Characters

We introduce here an important notion, which we need to explain the Solovay-Strassen test, and will be the key to Miller's Theorem 4.3.

Definition 4.8. A character χ modulo N is a group homomorphism from the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^*$ to $\mathbb{C}^*$. We lift χ to a function $\chi : \mathbb{Z} \to \mathbb{C}$ by setting $\chi(a) = 0$ if $(a, N) > 1$ and $\chi(a) := \chi(a \mod N)$ otherwise. We write χ_0 for the trivial character: $\chi_0(a) = 1$ whenever $(a, N) = 1$.

An important example is the Jacobi symbol, generalizing the Legendre symbol. For odd $N = \prod p$, where the primes p are repeated according to their multiplicity, let

$$\left(\frac{a}{N}\right) := \prod \left(\frac{a}{p}\right),$$

where $\left(\frac{a}{p}\right)$ is Legendre's symbol (-1, 0, or 1 if a is a non-square, 0 or a non-zero square in $\mathbb{F}_p$).

The function $a \mapsto \left(\frac{a}{N}\right)$ is a character modulo N, whose values can be computed efficiently using the quadratic reciprocity law, and a variant of the Euclidean algorithm, in essentially linear time $\widetilde{O}(N)$ if we use fast arithmetic. If N is prime, then $\left(\frac{a}{N}\right) \equiv a^{(N-1)/2} \pmod{N}$.

4.1.3 Compositeness

The following congruences are the basis of three historically important compositeness tests:

Theorem 4.9. *If $N > 2$ is prime and $0 < a < N$, then the following equalities hold in $\mathbb{Z}/N\mathbb{Z}$:*

1. $a^{N-1} = 1$,
2. $a^{(N-1)/2} = \left(\frac{a}{N}\right) \neq 0$,
3. *Write $N - 1 = 2^e q$, q odd, $e \geqslant 1$ and set $b := a^q$. Then $b = 1$ or there exists a unique $0 \leqslant i < e$ such that $b^{2^i} = -1$.*

Proof. (1) is Fermat's little theorem and (2) is one of the elementary properties of the Legendre symbol. The condition $\left(\frac{a}{N}\right) \neq 0$ is empty if N is prime, but we shall need it for general N later.

The last congruence follows from the polynomial equality

$$X^{2^e} - 1 = (X - 1)(X + 1)(X^2 + 1) \dots (X^{2^{e-1}} + 1)$$

(which you can prove directly or as a property of cyclotomic polynomials), evaluated at $X = b$: the left hand side is 0 by Fermat so one of the factors on the right hand side is 0. The index i is obviously unique since from then on, the b^{2^j}, $j > i$ are obtained by successive squarings, hence all are equal to 1.

This yields a more enlightening proof: by Fermat, $a^{N-1} = 1 = b^{2^e}$. In a (commutative) field of characteristic different from 2, the equation $X^2 = 1$ has just two solutions -1 and 1. Since we obtain 1 by a sequence of squarings from b, either all b^{2^i} are 1 or we hit -1 somewhere. □

These congruences yield three probabilistic compositeness tests: pick a random $0 < a < N$, and check one of (1), (2) or (3) above, yielding respectively Fermat's test (1640), Solovay-Strassen test (1977), and Rabin-Miller test. As for the last test, Miller (1976) devised a conditional deterministic test (see Theorem 4.3), which Rabin modified to the above probabilistic compositeness test in 1980. For completeness:

Algorithm 19. Solovay-Strassen compositeness test

Input: N an odd integer, $a \in \mathbb{Z}/N\mathbb{Z}$, $a \neq 0$.

Output: $SS(a) =$ `Composite` or `Fail`

```
If $\left(\frac{a}{N}\right) \neq a^{(N-1)/2}$ return Fail.
Return Composite.
```

Algorithm 20. Rabin-Miller compositeness test

```
Input: $N$ an odd integer, $a \in \mathbb{Z}/N\mathbb{Z}$, $a \neq 0$.
Output: $RM(a) =$ Composite or Fail
1: Write $N - 1 = 2^e q$, $q$ odd.
2: Compute $b = a^q$.
3: If $b = 1$ or $b^{2^i} = -1$ for some $i = 0, \dots, e-1$ return Fail.
4: Return Composite.
```

Theorem 4.10. *All three tests (Fermat, Solovay-Strassen, Rabin-Miller) run in time* $\widetilde{O}(\log N)^2$.

Proof. This is obvious for Fermat, easy for Rabin-Miller (note that we need $O(\log q + e) = O(\log N)$ multiplications in $\mathbb{Z}/N\mathbb{Z}$), and a little more involved for Solovay-Strassen: the powering part is easy again, but one needs to adapt the complexity proof of Euclid's algorithm to the computation of Jacobi's symbol $\left(\frac{a}{N}\right)$ using quadratic reciprocity. This requires specifying precisely the latter, and you can check out the result in Cohen [6]. □

Definition 4.11. An a such that $SS(a)$ or $RM(a)$ returns `Composite` is called a *witness* (of compositeness). If N is composite, an a such that they return `Fail` is called a *liar*.

Note that a witness testifies to the compositeness of N, but provides only circonstancial evidence: no explicit factor is exhibited.

Theorem 4.12. *Let* $N > 2$ *be an odd integer (no longer a prime),* $0 < a < N$, *and consider again the congruences from Theorem 4.9. Then (3) implies (2) implies (1).*

Proof. (1) follows from (2) by squaring: since we impose $\left(\frac{a}{N}\right) \neq 0$ in (2), the Jacobi symbol is ± 1. We now prove (3) $\Rightarrow$ (2), which is surprisingly intricate. Note that (3) $\Rightarrow$ (1) is obvious again by squarings, which already implies $a \in (\mathbb{Z}/N\mathbb{Z})^*$ and $\left(\frac{a}{N}\right) = \pm 1$. Since q is odd, we have $\left(\frac{a}{N}\right) = \left(\frac{a}{N}\right)^q = \left(\frac{b}{N}\right)$.

1. If (3) holds because $b = 1$, then $a^{(N-1)/2} = b^{2^{e-1}} = 1$ and $\left(\frac{b}{N}\right) = 1 = \left(\frac{a}{N}\right)$, hence (2) holds.

2. We now assume that $b^{2^i} = -1$ for a unique $0 \leqslant i < e$. We write $p - 1 = 2^{e_p} q_p$, q_p odd, for all $p \mid N$. The congruence $b^{2^i} \equiv -1 \pmod{p}$ holds for all $p \mid N$, meaning that the order of $b \in \mathbb{F}_p^*$ is 2^{i+1}. This must divide $\#\mathbb{F}_p^* = p - 1$, hence $i + 1 \leqslant e_p$.

3. From $\left(\frac{b}{p}\right) \equiv b^{2^{e_p - 1} q_p} \pmod{p}$, $b^{2^i} \equiv -1 \pmod{p}$ and $i + 1 \leqslant e_p$, it follows that $(-1)^{2^{e_p - i - 1}} = \left(\frac{b}{p}\right)$ (we also use that q_p odd). Hence $\left(\frac{b}{p}\right) = -1$ if and only if $e_p = i + 1$.

4. Consider

$$\begin{aligned} N = \prod_p (1 + 2^{e_p} q_p) &\equiv 1 + 2^{i+1} \sum_{p \colon \left(\frac{b}{p}\right) = -1} q_p \pmod{2^{i+2}} \\ &\equiv 1 + 2^{i+1} \# \left\{ p \colon \left(\frac{b}{p}\right) = -1 \right\} \pmod{2^{i+2}}, \end{aligned}$$

where each p is repeated according to its multiplicity. It follows that $i+1 = e$ if and only if $\# \left\{ p \colon \left(\frac{b}{p}\right) = -1 \right\}$ is odd, that is if and only if $\left(\frac{b}{N}\right) = -1$.

5. Finally $a^{(N-1)/2} = b^{2^{e-1}} = -1$ if and only if $i = e - 1$.

In other words $a^{(N-1)/2}$ and $\left(\frac{a}{N}\right)$, which are both equal to ± 1, are equal to -1 under the exact same conditions. Thus they are equal. □

In other words, Rabin-Miller's test is stronger than Solovay-Strassen, itself stronger than Fermat. It is easy to find specific counter examples showing that the reverse implications do not hold in general. To check whether our tests are any good we must now prove that for any odd composite N, there exist sufficiently many $0 < a < N$ that violate (1), (2) or (3).

Theorem 4.13. *If $N > 1$ is composite, then*

$$\left\{ 0 < a < N \colon a^{N-1} \equiv 1 \pmod{N} \right\}$$

is a subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$. If is equal to the full group $(\mathbb{Z}/N\mathbb{Z})^*$ if and only if N is squarefree and $p \mid N \Rightarrow p - 1 \mid N$ for all prime divisors p of N.

Proof. The subgroup assertion is clear. Now assume $a^{N-1} \equiv 1 \pmod{N}$ for all a coprime to N. Then it holds mod p for $p \mid N$, and a a primitive root (of order $p-1$) in $\mathbb{F}_p^*$. This implies $p - 1 \mid N - 1$. If further $p^2 \mid N$, then taking a of order $p(p-1)$ in the cyclic group $(\mathbb{Z}/p^2\mathbb{Z})^*$ we obtain $p(p-1) \mid N-1$. This contradicts $p \mid N$ since $\gcd(N, N-1) = 1$. □

A bad composite N as above, which completely wrecks Fermat's test, is called a Carmichael number. They were actually previously defined and studied by Korselt (1899), who could not find an actual example. In 1910, Carmichael found the smallest such number 561. These have no equivalent for the Solovay-Strassen test, nor for the stronger Rabin-Miller test:

Theorem 4.14. *If $N > 1$ is composite, then*

$$\left\{ 0 < a < N \colon a^{(N-1)/2} \equiv \left(\frac{a}{N}\right) \neq 0 \pmod{N} \right\}$$

is a proper subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$.

Proof. The subgroup assertion is clear again, and we must prove it is not the whole group. Since it is a subgroup of the one considered in the previous theorem, we may assume that N is a Carmichael number, and we must find an a not belonging to the given set.

If $p \mid N$ is an odd prime, let a be a non-square mod p and $a \equiv 1 \pmod{N/p}$, which exists by the Chinese Remainder Theorem. In fact, since N is squarefree, we have $\gcd(p, N/p) = 1$. This implies $\left(\frac{a}{N}\right) = -1$, but $a^{(N-1)/2} \not\equiv -1 \pmod N$ since this is not true modulo p. □

Corollary 4.15. *If N is an odd composite number, an $0 < a < N$ chosen uniformly at random is a witness with probability $\geqslant 1/2$.*

Proof. A proper subgroup has index greater than 2. □

This is a quick and beautiful argument, telling us witnesses are relatively abundant. In fact for most N almost all a are witnesses. It is not hard to give a much more precise result, or to improve on the Corollary:

Theorem 4.16. *1. If $N = \prod p^{f_p}$, let $\omega = \#\{p \mid N\}$,*

$$e_p = v_2(p-1), \quad E = \min_{p \mid N} e_p.$$

Then the number of a such that $RM(a)$ returns `Fail` is

$$\prod_{p \mid N} \gcd(q, p-1) \left(1 + \frac{2^{E\omega} - 1}{2^\omega - 1}\right).$$

2. If N is composite, an $0 < a < N$ chosen uniformly at random is a witness with probability $\geqslant 3/4$.

Proof. An a such that $RM(a)$ fails belongs to $(\mathbb{Z}/N\mathbb{Z})^*$. By the Chinese Remainder Theorem, $(\mathbb{Z}/N\mathbb{Z})^*$ is ring-isomorphic to $\prod_{p \mid N} (\mathbb{Z}/p^{f_p}\mathbb{Z})^*$. Each factor is cyclic, which linearizes the equations. We can count the (a_p) in the product $\prod_{p \mid N} \mathbb{Z}/(p-1)p^{f_p - 1}\mathbb{Z}$ such that $RM(a)$ returns `Fail`:

$$\#\{(a_p) : qa_p = 0\} + \sum_{i=0}^{e-1} \#\left\{(a_p) : q2^i a_p = (p-1)p^{f_p-1}/2\right\}$$

(these sets are disjoints). Using $(q, (p-1)p^{f-1}) = (q, p-1)$ and $E \leqslant e$ (a product of integers $\equiv 1 \pmod{2^E}$ is $\equiv 1 \pmod{2^E}$), we obtain

$$\left(1 + \sum_{i=0}^{E-1} 2^{i\omega}\right) \prod_p (q, p-1) = \left(1 + \frac{2^{\omega E} - 1}{2^\omega - 1}\right) \prod_p (q, p-1)$$

and the first result follows. Note that the special case $\omega = 1$, $p = N$, $E = e$ reproves case (3) in Theorem 4.9. We have already seen two rather more enlightening proofs.

To prove the second point, note that the proportion of witnesses is

$$\frac{\prod_{p|N} \gcd(q, p-1)}{N-1} \left(1 + \frac{2^{\omega E} - 1}{2^{\omega} - 1}\right).$$

We bound $\gcd(q, p-1) \leqslant (p-1)/2^{e_p} \leqslant (p-1)/2^E$. We treat first the case $\omega = 1$: $N = p^f$, $f > 1$. Then $(p-1)/(p^f - 1) \leqslant 1/(p+1) \leqslant 1/4$, and the result follows.

If $\omega > 1$, we bound $\prod(p-1) \leqslant N-1$ and the proportion is less than

$$2^{-\omega E} \left(1 + \frac{2^{\omega E} - 1}{2^{\omega} - 1}\right).$$

For a fixed ω, this is a decreasing function of E, which equals $2^{1-\omega}$ for $E = 1$. Hence the result unless $\omega = 2$, in which case we obtain only $1/2$... Coming back to the original $\gcd(q, p-1) \leqslant (p-1)/2^{e_p} \leqslant (p-1)/2^E$, we see that our final upper bound is divided by at least 2 unless we have equality throughout. The only remaining bad case is then $N = pq$, $p - 1 \mid N - 1$, $q - 1 \mid N - 1$, where $p < q$ are distinct primes. But $N - 1 \equiv p - 1 \pmod{q-1}$, hence $q - 1 \mid p - 1$, a contradiction. So this case cannot happen. This proves in particular that a Carmichael number has at least three distinct prime factors. □

Definition 4.17. Let $w(N)$ be the least witness for the compositeness of N, i.e. the smallest a such that $RM(a)$ returns `Composite`. For N prime, we let $w(N) = 0$.

If N is a Carmichael number whose prime divisors are $\equiv 3 \pmod 4$, there are exactly $\varphi(N)2^{1-\omega}$ liars and $3/4$ is not that far off. But, in general, the lower bound $3/4$ is very pessimistic: for large random N, we expect that $(q, p-1)$ is small for $p \mid N$; then the proportion of liars is essentially bounded by $2^{(\omega-1)e}/q$ which is small. Precisely, we have the following very strong *average* result:

Theorem 4.18 (Burthe)**.** *The average value of $w(N)$ over odd integers is 2:*

$$\sum_{N<X,\ 2\nmid N} w(N) \sim 2 \sum_{N<X,\ 2\nmid N} 1.$$

Which means that 2 is almost always a reliable witness. Note that from the prime number theorem, the primes contribute a negligible amount $O(X/\log X)$ to the right hand side $\sim X$. On the other hand, it is known by work of Alford-Granville-Pomerance that there are infinitely many Carmichael numbers and that $\limsup_{N\to\infty} w(N) = +\infty$.

4.1.4 Primality

Miller's theorem

First we explain how the GRH can turn either Solovay-Strassen or the Miller-Rabin test into a good primality prover. How does the Riemann Hypothesis come into play ? For χ a character modulo N, we introduce the Dirichlet L-function

$$L(\chi, s) = \sum_{n \geqslant 1} \chi(n) n^{-s}.$$

These are instrumental in proving that there exist infinitely many primes such that $p \equiv a \pmod{N}$ whenever $\gcd(a, N) = 1$. Just as the Riemann zeta function (which we essentially recover when $\chi = \chi_0$) is required to prove the prime number theorem.

For details on the basic theory of L-functions, see Serre [20]. For more advanced material, see Iwaniec-Kowalski [13]. In particular, it is a standard fact that $L(s, \chi)$ extends to a meromorphic function on $\mathbb{C}$ with at most a simple pole in $s = 1$ (if and only if $\chi = \chi_0$). For $\mathrm{Re}(s) > 1$, it satisfies an Euler product

$$L(s, \chi) = \prod_{p} (1 - \chi(p) p^{-s})^{-1}.$$

The theorem of Miller, in an effective version due to Bach [3], is as follows:

Theorem 4.19. *Assume the Dirichlet L-functions $L(s, \chi)$ have no 0 in the half-plane $\mathrm{Re}(s) > 1/2$, for all characters modulo N (GRH). If N is composite, there exists a witness $a \leqslant 2(\log N)^2$.*

Since $RM(a)$ runs in time $\widetilde{O}(\log N)^2$, this theorem yields a conditional primality test in essentially quartic time $\widetilde{O}(\log N)^4$. Unconditionally, we only know that $w(N) \leqslant N^{1/(6\sqrt{e})+\varepsilon}$ for all $\varepsilon > 0$, and N large. Not sufficient for a polynomial-time test.

Proof. (rough idea). Let $G = (\mathbb{Z}/N\mathbb{Z})^*$. Since N is composite, the subgroup $H = \left\{a \in G \colon a^{(N-1)/2} = \left(\frac{a}{N}\right)\right\}$ is proper. In particular, there exists a non trivial character χ modulo N such that χ is trivial on H (lift a non-trivial character of G/H). Assume by contradiction that $w(N) > x$, then $\sum_{a \leqslant x} \chi(a) = \lfloor x \rfloor$, and we have a non-trivial character masquerading χ_0. Assuming GRH, we have good bound on character sums, which yields a contradiction.

More precisely, taking the logarithmic derivative of the Euler product, we obtain

$$-\frac{L'}{L}(s, \chi) = \sum_{n \geqslant 1} \chi(n) \Lambda(n) n^{-s},$$

where Λ is the Riemann-von Mangoldt function: $\Lambda(n) = \log p$ if $n = p^k$ is a prime power, and $\Lambda(n) = 0$ otherwise.

Perron's formula says that

$$\sum_{n\leqslant x}\chi(n)\Lambda(n)=\frac{1}{2i\pi}\int_{\mathrm{Re}(s)=c}-\frac{L'}{L}(s,\chi)x^s\,\frac{ds}{s},$$

for $x>0$ not in $\mathbb{Z}$ and any $c>1$. Subtracting the formula for χ_0, we obtain

$$\int_{\mathrm{Re}(s)=c}\frac{L'}{L}(s,\chi)x^s\,\frac{ds}{s}=\int_{\mathrm{Re}(s)=c}\frac{L'}{L}(s,\chi_0)x^s\,\frac{ds}{s}.$$

We move the line of integration to the left up to the left of the line $\mathrm{Re}(s)=1/2$: the GRH enables us to keep a tight control on the singularities of L'/L: at the zeroes of L for $\mathrm{Re}(s)=1/2$, and a simple pole at $s=1$ for $(L'/L)(s,\chi_0)$. Keeping track of the residues, we find an equality of the form $O(x^{1/2})=x+O(x^{1/2})$, a contradiction if x is large.

The true proof is technically more demanding because the error terms depend on N: we must use an "explicit formula", and integrate with respect to more involved kernel functions. □

Primality certificates (Pratt)

Besides being conditional, Miller's algorithm is not entirely satisfactory: if the answer is no, I have a witness, i.e. a proof of compositeness; but if the answer is yes, I am left with no evidence, hence no convincing argument besides "I did program the test correctly". We are no better off than when trial dividing up to the square root of the number. In this section, we explain the idea of *primality certificate*, or succinct proof of primality.

We use the following idea: $N>1$ is prime if and only if $(\mathbb{Z}/N\mathbb{Z})^*$ is cyclic. If I can exhibit a generator and prove that it has order $N-1$, I am done. Unfortunately, this will require factoring $N-1$, which is hard. But the person I am handing the certificate to will not care: creating a proof may be hard, but checking it is easy.

Theorem 4.20 (Pocklington). *Let $N>1$ be an integer and $p\mid N-1$ a prime such that $v_p(N-1)=e$. Assume $a\in\mathbb{Z}$ satisfies*

- $a^{N-1}\equiv 1 \pmod N$,
- $\gcd(a^{(N-1)/p}-1,N)=1$.

Then all divisors $d\mid N$ satisfy $d\equiv 1 \pmod{p^e}$.

Proof. We may assume that d is prime, since a product of integers $\equiv 1 \pmod M$ is congruent to $1\equiv \pmod M$. Since $d\mid N$, $a^{N-1}\equiv 1 \pmod d$, which implies that $\gcd(a,d)=1$ hence $a^{d-1}\equiv 1 \pmod d$ by Fermat's little theorem.

Since $a^{(N-1)/p}\not\equiv 1 \pmod d$, the order r of a in $(\mathbb{Z}/d\mathbb{Z})^*$ satisfies $r\mid N-1$, but $r\nmid (N-1)/p$, hence $p^e\mid r$. On the other hand, $r\mid d-1$ and we are done. □

Corollary 4.21. *Let $N > 1$ be an integer. Assume $N - 1 = FU$ with $F \geqslant \sqrt{N}$, where the prime divisors of F are known, and for all such p, and a_p as above is given. Then N is prime.*

Proof. If $d \mid N$, then $d \equiv 1 \pmod{F}$ by the Chinese remainder Theorem and Theorem 4.20. Hence $d = 1$ or $d \geqslant F + 1 > \sqrt{N}$. The latter implied $d = N$, since $N/d < \sqrt{N}$ must be 1. Hence N is prime. □

Of course, the catch is that one needs to factor $N - 1$. But when this is done, the rest is easy: if N is prime, all $a \in (\mathbb{Z}/N\mathbb{Z})^*$ satisfy the first condition, and exactly $(N-1)/p$ elements satisfy $a^{(N-1)/p} = 1$, hence a random $a \in (\mathbb{Z}/N\mathbb{Z})$ is a suitable a_p with probability $1 - 1/p \geqslant 1/2$.

We can now define recursively a primality certificate $C(N)$: it is a factorization $N = \prod p^{e_p}$ together with a set of triples $(p, a_p, C(p))$, where $C(p)$ recursively certifies p. In order to avoid infinite recursion, we allow the empty certificate for $p = 2$.

Exercise 4.22. Bound the size of $C(N)$. How fast can you check it ?

4.1.5 Producing primes

We recall the Prime Number Theorem in its standard form:

Theorem 4.23 (Prime Number Theorem (PNT))**.** *As $x \to +\infty$, we have*

$$\pi(x) := \#\{p \leqslant x\colon p \text{ prime}\} \sim \frac{x}{\log x}.$$

Effectively, we have

$$\frac{x}{\log x}\Big(1 + \frac{1}{2\log x}\Big) \leqslant \pi(x) \leqslant \frac{x}{\log x}\Big(1 + \frac{3}{2\log x}\Big),$$

where the left-hand side is valid for $x \geqslant 59$, and the right-hand side for $x > 1$.

From the prime number theorem, the number of primes in $]N, 2N]$ is

$$\pi(2N) - \pi(N) \sim \frac{N}{\log N}.$$

Picking an integer at random in the interval, the expected number of trials before hitting a prime is $\log N$. We can test the numbers produced for compositeness, then for primality once we have a good candidate (an integer which fails many compositeness tests is often declared a “probable prime”). This is a suitable algorithm to find a *big* prime. We now examine a dual, nicer construction: a sieve constructs simultaneously many *small* primes, for an essentially constant unit cost. According to the divide-and-conquer principle, “small primes” algorithms followed by chinese remaindering (and to a lesser degree p-adic algorithms using Hensel lifting) perform better than “large prime” algorithms, although the latter are conceptually simpler.

Algorithm 21. Eratosthenes’s sieve

Input: An integer B.

Output: The set of primes $p \leqslant B$.

```
Initialize an array $A[2] = \cdots = A[B] = 1$.
for $n = 2, \ldots, \sqrt{B}$ do
    if $A[n] = 1$ then    {$n$ is prime}
        for $k = 2, \ldots, B/n$ do    {cross out multiples of $n$}
            Set $A[kn] = 0$,    {$kn$ not a prime}
Return the $n$ such that $A[n] = 1$.
```

The number of array operations involved is

$$2B + \sum_{p \leqslant \sqrt{B}} \lfloor B/p \rfloor = 2B + O(\sqrt{B}) + B \sum_{p \leqslant \sqrt{B}} \frac{1}{p} \sim B \log\log B,$$

using $\sum_{p \leqslant x} p^{-1} \sim \log\log x$, which follows for instance from the Prime Number Theorem.

4.1.6 Split

This section introduces the important notion of smooth integers, and serves as a warm up for our later factoring with elliptic curves.

Definition 4.24. Let $B > 0$. A positive integer $N > 0$ is

- B-smooth (or B-friable) if $p \mid N$ implies $p \leqslant B$ for prime p.
- B-powersmooth if $p^k \mid N$ implies $p^k \leqslant B$ for prime p and positive k.

Theorem 4.25 (de Bruijn). *Let $\psi(x, y) = \#\{n \leqslant x \colon n \text{ is } y\text{-smooth}\}$. Provided $(\log x)^{\varepsilon} \leqslant u \leqslant (\log x)^{1-\varepsilon}$ for some $0 < \varepsilon < 1$, we have*

$$\frac{\psi(x, x^{1/u})}{x} = u^{-u+o(u)}$$

as x tends to infinity.

We will abuse this theorem to estimate the probability that integers in certain sequences are $x^{1/u}$ smooth. Our reasoning will not be rigorous: a random integer in our sequence is considered to be $x^{1/u}$-smooth with the same probability u^{-u} as if it were taken form a uniform distribution. It is possible to rigorously prove some estimates, using much more technical arguments, beyond the scope of our lectures. Here is a simple application of smooth numbers:

Algorithm 22. Pollard's $p-1$ method

Input: N an integer, B a smoothness bound.

Output: A non-trivial factor of N or `Fail`.

```
Using Eratosthenes's sieve, compute all primes $p \leqslant B$.
Pick a random $a \in \mathbb{Z}/N\mathbb{Z}$. Let $b = a$.
for $p \leqslant B$ do    {compute $b = a^{\mathrm{lcm}(2,\dots,B)}$}
    Let $k$ be maximal such that $p^k \leqslant B$.
    Set $b := b^{p^k}$.
if $d = \gcd(b-1, N)$ is a non-trivial divisor of $N$ then
    Return $d$.
else
    Return Fail.
```

The algorithm succeeds when the order of a modulo some divisor of N is B-powersmooth. Equivalently, the order of a modulo a *prime* divisor p of N is B-powersmooth. Unless we are lucky this means that $p-1$ is B-powersmooth. Indeed, if $\ell > B$ is a large prime dividing $p-1$, then a random $a \in (\mathbb{Z}/N\mathbb{Z})^*$ has order divisible by ℓ in $\mathbb{F}_p^*$ with probability $1 - 1/\ell \approx 1$. ($p-1$ also fails to be B-powersmooth if $\ell^k > B$ divides $p-1$ for a small ℓ and huge k, but this is implausible.)

At the end of the **for** loop, $b = a^{\mathrm{lcm}(2,\dots,B)}$, which is congruent to 1 modulo p by our assumption that $p-1$ is B-powersmooth. In other words, $p \mid b-1$. If we are lucky, $N \nmid b-1$ and d is a non-trivial factor. The algorithm can fail for two reasons:

- $d = 1$: this proves that there is no prime divisor p of N such that $p-1$ is B-powersmooth. We must increase B and retry.

- $d = N$: the order of a in $(\mathbb{Z}/N\mathbb{Z})^*$ is B-powersmooth. As above, unless we were lucky, this means that $\varphi(N)$ is B-powersmooth. We must decrease B and retry. (If B is not very large, we can try a different a, just in case.)

The cost of the method is dominated by the powering, in time

$$\sum_{p^k \leqslant B} (\log p^k) \widetilde{O}(\log N) = \widetilde{O}(B \log N).$$

Assuming that $p-1$ is B-powersmooth is a little extreme. We increase our chances by assuming it is B_1-powersmooth, up to a single prime less than $B_2 \gg B_1$. There is a nice way of implementing this idea, based on the fact that primes are relatively plentiful, and the differences $p_{i+1} - p_i$ between consecutive primes are rather small. (To be taken with a grain of salt: it is an easy exercise to show that $\limsup_i p_{i+1} - p_i = +\infty$; the "converse" statement $\liminf_i p_{i+1} - p_i = 2$ is the famous *Twin primes conjecture*.)

Algorithm 23. Pollard's $p-1$ method, with B_2 phase

Input: N an integer, (B_1, B_2) smoothness bounds.

Output: A non-trivial factor of N or `Fail`.

```
1: Pick a random $a \in \mathbb{Z}/N\mathbb{Z}$ and compute $b = a^{\mathrm{lcm}(2,\dots,B_1)}$ as in the standard method.
2: Let $S = \{b^{p_{i+1}-p_i}\}$, where the $p_i$, $1 \leqslant i \leqslant K$ are the consecutive primes $B_1 < p \leqslant B_2$. This is a (small) set indexed by the $p_{i+1} - p_i$.
3: Set $b \leftarrow b_1^p$;
4: for $i = 1, \dots, K-1$ do
5:   If $d = \gcd(b-1, N)$ is a non-trivial divisor of $N$, return $d$.
6:   Replace $b \leftarrow b \times S[p_{i+1} - p_i]$.   $\{b = a^{\mathrm{lcm}(2,\dots B_1) p_{i+1}}\}$
7: Return Fail.
```

In effect, instead of a powering cost of $\sum_{p<B_2} \log p \sim B_2$ multiplications, we have $\pi(B_2) \sim B_2/\log B_2$ multiplications. If B_2 is so large that a full fledged sieving of $[0, B_2]$ becomes impractical, we can precompute the primes up to $\sqrt{B_2}$ and sieve out smaller slices $[kM, (k+1)M]$ with $k = \lfloor B_2/M \rfloor$.

The $p-1$ method has the very nice feature of being sensitive to the size of the smallest prime divisor of N: smaller numbers are smoother. On the other hand the first point above is a big problem: either there exists a prime divisor p with $p-1$ smooth, or there does not and we are lost. The elliptic curve method nowadays completely supersedes $p-1$: it will try *many* orders $\#E(\mathbb{F}_p)$ instead of the single $\#\mathbb{F}_p^* = p-1$, by varying the curve E. All these orders have roughly the same size as $p-1$, hence supposedly the same chance of being smooth, using the heuristic principle discussed above.

4.2 Primality proving & factoring with elliptic curves

4.2.1 Elliptic curves over $\mathbb{Z}/N\mathbb{Z}$

Let $N > 0$ be an integer coprime to 6. An "elliptic curve" over $\mathbb{Z}/N\mathbb{Z}$ is a Weierstrass equation

$$E\colon Y^2Z = X^3 + aXZ^2 + bZ^3, \quad a, b \in \mathbb{Z}/N\mathbb{Z}, \quad 4a^3 + 27b^2 \in (\mathbb{Z}/N\mathbb{Z})^*.$$

This defines a "nonsingular curve" in the projective space $\mathbb{P}^2(\mathbb{Z}/N\mathbb{Z})$:

$$\left\{(x, y, z) \in (\mathbb{Z}/N\mathbb{Z})^3, \gcd(x, y, z, N) = 1\right\} /\text{multiplication by } \lambda \in (\mathbb{Z}/N\mathbb{Z})^*.$$

The class of (x, y, z) in $\mathbb{P}^2(\mathbb{Z}/N\mathbb{Z})$ is denoted $(x : y : z)$ as usual. If $p \mid N$, there is a natural projection $\pi : \mathbb{P}^2(\mathbb{Z}/N\mathbb{Z}) \to \mathbb{P}^2(\mathbb{Z}/p\mathbb{Z})$ inducing a natural map from E to E_p, the curve over $\mathbb{Z}/p\mathbb{Z}$ defined by reducing the equation of E modulo p, i.e. we can reduce $P \in E(\mathbb{Z}/N\mathbb{Z})$ to $\pi(P) \in E_p(\mathbb{Z}/p\mathbb{Z})$. Basic facts about elliptic curves can be found in Silverman [21].

Theorem 4.26. *If N is prime*

1. *$E(\mathbb{Z}/N\mathbb{Z})$ has a natural structure of commutative group. The group law is denoted $(P, Q) \mapsto P+Q$ with neutral element $O_E = (0 : 1 : 0)$. It is given by polynomial mappings (depending on the coefficients of E) in the coordinates of P and Q, formally the same ones as in the "chord and tangent process" over $\mathbb{R}$ or $\mathbb{Q}$.*

2. *$E(\mathbb{Z}/N\mathbb{Z})$ has at most two cyclic components.*

3. *$(\sqrt{N} - 1)^2 < \#E(\mathbb{Z}/N\mathbb{Z}) < (\sqrt{N} + 1)^2$ (Hasse's bound).*

Now a non-theorem: if N is not prime, $E(\mathbb{Z}/N\mathbb{Z})$ is definitely not a group. But we may still try to add points applying the same defining formulae as if N were prime. The worse obstruction we may encounter is a non-invertible $d \in (\mathbb{Z}/N\mathbb{Z}) \setminus \{0\}$. In this case, $\gcd(d, N)$ is a non-trivial factor of N. In the text, we now assume that all computation depicted in curves over $\mathbb{Z}/N\mathbb{Z}$ (e.g. compute $P + Q$) do succeed. Whenever they do not we obtain a non-trivial factor of N, which is usually our main motivation.

More generally, we define $[m]P$ for $m \in \mathbb{Z}$ and $P \in E(\mathbb{Z}/N\mathbb{Z})$ by

$$[0]P := O_E, \quad [m]P := [m-1]P + P, \text{ for } m > 0$$

and $[m]P := [-m]P$ for $m < 0$. If N is prime, we have genuine associative group law and $[m]P = P + \cdots + P$ with m summands, for $m > 0$; otherwise, the result is undefined if one of the addition fails. The only result we will need about this pseudo group law is that if $p \mid N$ is a prime, $\pi : \mathbb{P}^2(\mathbb{Z}/N\mathbb{Z}) \to \mathbb{P}^2(\mathbb{Z}/p\mathbb{Z})$ is the natural projection, and $P, Q \in E(\mathbb{Z}/N\mathbb{Z})$ such that $P + Q$ is well-defined, then

$$\pi(P + Q) = \pi(P) + \pi(Q), \quad \text{and} \quad \pi([m]P) = [m]\pi(P),$$

where the right-hand side additions use the ordinary group law on $E_p(\mathbb{Z}/p\mathbb{Z})$.

We will subsequently use freely operations in $E(\mathbb{Z}/N\mathbb{Z})$, with the convention that any such operation failing provides a non trivial factor of N and aborts all computations.

4.2.2 The basic idea (Goldwasser-Killian)

We now adapt Pocklington's theorem 4.20, in the simplest setting, not meant to be practical.

Theorem 4.27. *Let N be an integer coprime to 6, let E be an elliptic curve over $\mathbb{Z}/N\mathbb{Z}$, $P \in E(\mathbb{Z}/N\mathbb{Z})$, and $m > 0$ an integer such that*

- *There exists a prime divisor q of m with $q \geqslant (N^{1/4}+1)^2$,*
- *$[m]P = O_E$, but $[m/q]P = (x : y : z)$, with $(z, N) = 1$.*

Then N is prime.

Proof. Assume N is composite, and let $p \leqslant N^{1/2}$ be the smallest prime divisor of N. The order r of $\pi(P) \in E_p(\mathbb{Z}/p\mathbb{Z})$ divides m but not m/q, hence $q \mid r$. On the other hand, by Hasse's bound,

$$r \leqslant \#E_p(\mathbb{Z}/p\mathbb{Z}) < (p^{1/2}+1)^2 \leqslant (N^{1/4}+1)^2,$$

hence $q < (N^{1/4}+1)^2$, a contradiction. □

If N is prime the above just uses a curve E and $P \in E(\mathbb{Z}/N\mathbb{Z})$ of provably large order (divisible by q). It is natural to choose $m = \#E(\mathbb{Z}/N\mathbb{Z})$ and try random points on the curve:

Proposition 4.28. *Let $N > 3$ be prime, E an elliptic curve over $\mathbb{Z}/N\mathbb{Z}$, $m = \#E(\mathbb{Z}/N\mathbb{Z})$, and q a prime factor of m. Then a random point $P \in E(\mathbb{Z}/N\mathbb{Z})$ satisfies $[m/q]P = O_E$ with probability $1/q$.*

Proof. Since $m < (\sqrt{N}+1)^2$, we have $v_q(m) = 1$. The requested property is true for any abelian group G satisfying $v_q(\#G) = 1$. Indeed, we may write $G = \mathbb{Z}/q\mathbb{Z} \oplus H$ for some abelian group H or order h (coprime to q); the condition $h(a \oplus b) = 0$ is equivalent to $a = 0$, which is true for a fraction $h/\#G = 1/q$ of all elements of G. □

Algorithm 24. Goldwasser-Killian primality test

Input: $N > 1$, coprime to 6.

Output: A primality proof for N.

```
1: Pick $a, b \in \mathbb{Z}/N\mathbb{Z}$ at random and let $E : Y^2 Z = X^3 + aXZ^2 + bZ^3$.
2: Compute $m = \#E(\mathbb{Z}/N\mathbb{Z})$, as if $N$ were prime.
3: Try to factor $m$: if it factors completely leaving a factor $q \geqslant (N^{1/4}+1)^2$
   which looks prime (fails a few compositeness tests).
4: Recursively prove the primality of $q$. If it fails, start over at (1).
5: Find $P \in E(\mathbb{Z}/N\mathbb{Z})$ such that $[m/q]P = (x : y : z)$, with $\gcd(z, N) = 1$ and
   $[m]P = O_E$.    {pick a random $x \in \mathbb{Z}/N\mathbb{Z}$ until $x^3 + ax + b$ is a square, then
   let $y$ be a square root and set $P = (x : y : 1)$.}
```

In the above we factor the quadratic polynomial $Y^2 - (x^3 + ax + b)$ as if N were prime: if it fails, N is composite. We can simplify the last step by picking simultaneously E and P: pick randomly $x, y, a \in \mathbb{Z}/N\mathbb{Z}$ and set $b = y^2 - x^3 - ax$.

The big problem with the algorithm is that m is difficult to compute. Goldwasser and Killian use Schoof's algorithm which runs in time $O(\log N)^8$, assuming N is prime. Even with a lot of improvements since (in particular by Elkies and Atkin), this is still impractical to prove the primality of large integers, say 10000 digits. Atkin's idea is to consider curves with complex multiplication, so that m is known in advance from a simple formula.

4.2.3 Introduction to complex multiplication

An elliptic curve over $\mathbb{C}$ is a torus $\mathbb{C}/\Lambda$, where Λ is a lattice, i.e. a rank 2 submodule. In particular, Λ is not contained in $\mathbb{R}$.

Definition 4.29. An *elliptic function* with period lattice Λ is a meromorphic function f, such that $f(z+w) = f(z)$ for all $w \in \Lambda$. In other words it induces a well-defined function on $\mathbb{C}/\Lambda$ with finitely many poles deleted.

One of the simplest examples is Weierstrass $\wp$-function:

$$\wp(z;\Lambda) := \wp(z) = \frac{1}{z^2} + \sum_{w\in\Lambda\setminus\{0\}} \left(\frac{1}{(z-w)^2} - \frac{1}{w^2}\right).$$

It satisfies the following differential equation:

$$\wp'(z)^2 = 4\wp(z)^3 - g_2\wp(z) - g_3, \quad \text{where } g_2 = 60 \sum_{w\in\Lambda\setminus\{0\}} \frac{1}{w^4}, \quad g_3 = 140 \sum_{w\in\Lambda\setminus\{0\}} \frac{1}{w^6}.$$

(The proof is easy: an elliptic function without poles is constant, because it is entire and bounded. Apply this to the difference $4\wp(z)^3 - g_2\wp(z) - g_3 - \wp'(z)^2$.) Note that g_2 and g_3 are well-defined, in fact

Lemma 4.30. *For any lattice $\Lambda \subset \mathbb{C}$, the series*

$$\sum_{w\in\Lambda\setminus\{0\}} \frac{1}{|w|^s},$$

converges if $s > 2$.

Proof. Prove that $C_k = \#\{w \in \Lambda, |w| \leqslant k\} = C(\Lambda)k^2 + O(k)$, for some constant $C(\Lambda)$. It follows that $\#\{w \in \Lambda, k \leqslant |w| < k+1\} = O(k)$. Summing by parts, the sum converges for $s > 2$. □

Proposition 4.31. *We have $g_2^3 - 27g_3^2 \neq 0$, i.e the projective curve $y^2z = 4x^3 - g_2xz^2 - g_3z^3$ is non-singular.*

Remark 4.32. The mapping

$$\begin{array}{rcl} \mathbb{C}/\Lambda & \to & \mathbb{P}^2(\mathbb{C}) \\ z & \mapsto & \begin{cases} (\wp(z) : \wp'(z) : 1) & \text{if } z \notin \Lambda, \\ (0:1:0) & \text{if } z \in \Lambda, \end{cases} \end{array}$$

is a complex isomorphism between the torus $\mathbb{C}/\Lambda$ and the non-singular projective curve $E : y^2z = 4x^3 - g_2xz^2 - g_3z^3$. This endows E with a natural group structure, which coincides with the chord-and-tangent law.

We now consider morphisms between complex elliptic curve, that is holomorphic, $\mathbb{Z}$-linear maps:

Theorem 4.33. *1. The curves $\mathbb{C}/\Lambda$ and $\mathbb{C}/\Lambda'$ are isomorphic if and only if there exists $\alpha \in \mathbb{C}^*$, $\Lambda' = \alpha\Lambda$.*

2. $\operatorname{End}(\mathbb{C}/\Lambda) = \{\alpha \in \mathbb{C} \colon \alpha\Lambda = \Lambda\}$.

3. $\operatorname{End}(\mathbb{C}/\Lambda) = \mathbb{Z}$ *or an order $\mathcal{O}$ in an imaginary quadratic field (i.e. $\mathcal{O} = \mathbb{Z} + w\mathbb{Z}$ with* $\operatorname{Im}(w) > 0$ *and $w^2 - sw + d = 0$ for some $s, d \in \mathbb{Z}$).*

4. Let $j(E) = j(\Lambda) = 1728g_2^3/(g_2^3 - 27g_3^2)$. This j-invariant characterizes the isomorphism class of $E = \mathbb{C}/\Lambda$: $j(E) = j(E')$ if and only if $E \cong E'$.

5. $j(\alpha\Lambda) = j(\Lambda)$ for any $\alpha \in \mathbb{C}^*$.

Corollary 4.34. *It is easy to write a Weierstrass equation for a complex curve with given j-invariant:*

1. If $j = 0$, take $y^2 = x^3 - 1$.

2. If $j = 1728$, take $y^2 = x^3 - x$.

3. If $j \neq 0, 1728$, let $c = j/(1728 - j)$ and take $y^2 = x^3 + 3cx + 2c$. The right hand side has discriminant $-2^2 3^3 c^2(c + 1) \neq 0$.

We denote $E(j)$ the equation given above. The formula is in fact valid over any base whose characteristic is not 2 or 3.

Definition 4.35. E has complex multiplication by $\mathcal{O}$ if $\operatorname{End}(E) = \mathcal{O}$ is strictly larger than $\mathbb{Z}$. We say E has CM by $\mathcal{O}$.

Example 4.36. The curve $\mathbb{C}/\mathbb{Z}[i]$ has CM by $\mathbb{Z}[i]$. It is isomorphic to $E : y^2 = x^3 - x$, which has extra endomorphism $(x, y) \mapsto (-x, iy)$.

From now on, we insist that a basis (ω_1, ω_2) for Λ be positively oriented: $\operatorname{Im}(\omega_1/\omega_2) > 0$, which is easily achieved by swapping ω_1 and ω_2. Since $j(\alpha\Lambda) = j(\Lambda)$, we may assume that $\Lambda = \langle \tau, 1 \rangle_{\mathbb{Z}}$, with $\operatorname{Im} \tau > 0$. Given a basis (ω_1, ω_2) such that $\tau = \omega_1/\omega_2$ has positive imaginary part, we set $j(\tau) := j(\langle \tau, 1 \rangle_{\mathbb{Z}})$.

Remark 4.37. The reason for the weird normalizing constant 1728 for j is to ensure the following identity

$$j(\tau) = \frac{1}{z} + 744 + \sum_{n\geqslant 1} c_n q^n, \quad \text{with } q = \exp(2i\pi\tau),$$

where the c_n are positive *integers*.

The fact that $j(\Lambda)$ is a function on lattices, which we compute using any positively oriented basis is equivalent to saying that $j(\tau)$ is invariant under the natural action of $\mathrm{SL}_2(\mathbb{Z})$:

$$\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \cdot \tau = \frac{a\tau + b}{c\tau + d}.$$

Namely $(\omega_1, \omega_2) \to (a\omega_1 + b, c\omega_2 + d)$, $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{GL}_2(\mathbb{Z})$, gives the general base change and

$$\operatorname{Im}\left(\frac{a\tau + b}{c\tau + d}\right) = \operatorname{Im}\left(\frac{(a\tau + b)\overline{(c\tau + d)}}{|c\tau + d|^2}\right) = \frac{(ad - bc)\operatorname{Im}\tau}{|c\tau + d|^2} > 0$$

restricts us to $\mathrm{SL}_2(\mathbb{Z})$. Hence, we may further impose that τ belongs to the standard fundamental domain for the action of $\mathrm{SL}_2(\mathbb{Z})$ on Poincaré's half-plane $\operatorname{Im}(z) > 0$:

$$|\operatorname{Re}(z)| \leqslant \frac{1}{2}, \quad |z| \geqslant 1.$$

In particular $\operatorname{Im}\tau \geqslant \sqrt{3}/2$ and $|q| \leqslant \exp(-\pi\sqrt{3}) \approx 0.0043$. From this, $j(\tau)$ is easy to approximate numerically. In fact, we find

$$g_2 = \frac{1}{12}\left(\frac{2\pi}{\omega_2}\right)^4 \left(1 + 240 \sum_{n\geqslant 1} \frac{n^3 q^n}{1 - q^n}\right),$$

$$g_3 = \frac{1}{216}\left(\frac{2\pi}{\omega_2}\right)^6 \left(1 + 504 \sum_{n\geqslant 1} \frac{n^5 q^n}{1 - q^n}\right).$$

This is not the fastest way to compute j, but already quite efficient: $n^5 q^n$ tends very quickly to 0.

4.2.4 Some algebraic number theory

A $z \in \mathbb{C}$ is *algebraic* if it is a root of a non-zero polynomial in $\mathbb{Z}[X]$. It is an *algebraic integer* if that polynomial can be chosen to be *monic*.

A number field $K \subset \mathbb{C}$ of degree n is a finite extension of $\mathbb{Q}$ of degree n, i.e. a field which is a $\mathbb{Q}$-vector space of finite dimension n. The algebraic integers in K form a ring $\mathbb{Z}_K$. An *integral ideal* is a non-zero ideal in $\mathbb{Z}_K$, a *fractional ideal* is a subset $\mathfrak{A}$ of K such that $d\mathfrak{A}$ is integral for some $d \in K$. A fractional ideal, in particular $\mathbb{Z}_K$, is a free $\mathbb{Z}$-modules of rank n.

Remark 4.38. We define number fields as embedded in $\mathbb{C}$. This is not a necessity, and is in fact the wrong point of view. We could view K as an abstract field embedding $\mathbb{Q} \subset K$ or, more concretely but less adequately, as a quotient ring $K = \mathbb{Q}[X]/(T)$, for some irreducible $T \in \mathbb{Q}[X]$ of degree n (this follows from the primitive element theorem, and is not suitable to define extensions over more general bases than $\mathbb{Q}$, e.g. $\mathbb{F}_p(t)$). To each complex root α_i of T corresponds an embedding $K \to \mathbb{C}$, sending the class of X to α_i, and each complex embedding is of this form. In this way, an abstract number field of degree n comes equipped with n canonical embeddings into the complex numbers, or any algebraically closed field. There is no reason to favor any of them.

For instance, the field $\mathbb{Q}[X]/(X^3-2)$ has three different embeddings, $\mathbb{Q}(2^{1/3})$, $\mathbb{Q}(2^{1/3}j)$ and $\mathbb{Q}(2^{1/3}j^2)$, which are isomorphic as fields, but definitely not identical. For one thing, the first one is a subset of $\mathbb{R}$.

There is a natural multiplication on the set of integral ideals: $\mathfrak{A}\mathfrak{B}$ is the smallest integral ideal containing all ab, $a \in \mathfrak{A}$, $b \in \mathfrak{B}$. This extends in a natural way to fractional ideals.

Theorem 4.39. *1. $\mathbb{Z}_K$ is a Dedekind ring: the fractional ideals of K form an abelian group under multiplication, with neutral element $\mathbb{Z}_K$. Any fractional ideal can be written uniquely as a product of maximal ideals: $\mathfrak{A} = \prod \mathfrak{p}^{e_\mathfrak{p}}$, where all $e_\mathfrak{p}$ but finitely many are 0. $\mathfrak{A}$ is integral, if and only if $e_\mathfrak{p} \geqslant 0$ for all $\mathfrak{p}$. If $\mathfrak{p} \subset \mathbb{Z}_K$ is a maximal ideal, then $\mathbb{Z}_K/\mathfrak{p}$ is a finite field.*

2. If $\mathfrak{A}$ is an integral ideal, then the norm of $\mathfrak{A}$, $N\mathfrak{A} := \mathbb{Z}_K/\mathfrak{A}$ is finite, this is a multiplicative map: $N(\mathfrak{A}\mathfrak{B}) = N\mathfrak{A} \cdot N\mathfrak{B}$, which extends to fractional ideals by multiplicativity.

3. Let I_K be the group of fractional ideals, and P_K be the subgroup of principal *fractional ideals. The quotient group I_K/P_K is a finite abelian group* $\mathrm{Cl}(K)$, *the class group of K.*

4. There exists a finite Galois extension H_K/K, the Hilbert class field of K, whose Galois group is $\mathrm{Cl}(K)$. *A maximal ideal $\mathfrak{p} \subset \mathbb{Z}_K$ splits completely in H_K if and only if it is principal.*

(The ideal $\mathfrak{p}$ splits completely in L/K if $\mathfrak{p}\mathbb{Z}_L$ is a product of $[L : K]$ distinct maximal ideals in $\mathbb{Z}_L$; they all satisfy $\mathbb{Z}_L/\mathfrak{P} = \mathbb{Z}_K/\mathfrak{p}$.)

We shall see in the next lecture how to compute class groups of imaginary quadratic fields. From this, the j-invariant will enable us to compute Hilbert class fields. There are beautiful and important algorithms for the general case of arbitrary number fields, and generalizations of the Hilbert class field, just beyond the scope of our lectures. Class fields provide a complete solution to the inverse Galois problem for abelian groups. In short, they describe all abelian extensions of a given number field K, in terms of arithmetic data depending on K only. For an introduction to this so-called Class Field Theory in a context very close to ours, see Cox [8], then Cohen [6, 7] for a computational approach.

Let us consider a few examples :

1. $K = \mathbb{Q}$, one finds $\mathbb{Z}_K = \mathbb{Z}$, $\mathrm{Cl}(K) = \{1\}$ ($\mathbb{Z}$ is principal, in fact Euclidean), hence $H_K = K$.

2. $K = \mathbb{Q}(i)$, one finds $\mathbb{Z}_K = \mathbb{Z}[i]$, $\mathrm{Cl}(K) = \{1\}$ ($\mathbb{Z}[i]$ is again Euclidean), hence $H_K = K$.

3. $K = \mathbb{Q}(\sqrt{-6})$, one finds $\mathbb{Z}_K = \mathbb{Z}[\sqrt{-6}]$, $\mathrm{Cl}(K) \simeq \mathbb{Z}/2\mathbb{Z}$, generated by the maximal ideal $\mathfrak{p}$ generated by 2 and $\sqrt{-6}$, which is *not* principal; in this case $\mathbb{Z}_K/\mathfrak{p} = \mathbb{F}_2$. The Hilbert class field H_K is
$$K(\sqrt{-3}) = K(\sqrt{2}) = \mathbb{Q}(\sqrt{2} + \sqrt{-3}).$$

Remark 4.40. The notation $\sqrt{d}$ for $d < 0$ denotes the complex number $i\sqrt{d} \in \mathbb{C}$. Had we taken the better point of view that $K = \mathbb{Q}(\sqrt{d})$ is really $\mathbb{Q}[X]/(X^2 - d)$, for some non-square d, we could just say it is the class of X in K.

Exercise 4.41. Let d be a squarefree integer, $K = \mathbb{Q}(\sqrt{d})$. Prove that $\mathbb{Z}_K = \mathbb{Z}[\omega]$, where
$$\omega = \begin{cases} \sqrt{d} & \text{if } d \equiv 2, 3 \pmod 4, \\ \dfrac{1+\sqrt{d}}{2} & \text{if } d \equiv 1 \pmod 4. \end{cases}$$
(When is $(u + \sqrt{d})/2$ an algebraic integer, if $u, v \in \mathbb{Q}$?)

The *discriminant* of a quadratic field $\mathbb{Q}(\sqrt{d})$, d squarefree, is $D = 4d$ if $d \equiv 2, 3 \pmod 4$, and d otherwise. Then $\mathbb{Z}_K = \mathbb{Z}[\frac{D+\sqrt{D}}{2}]$ in all cases.

4.2.5 Class groups of imaginary quadratic fields

Let $K = \mathbb{Q}(\sqrt{D})$ be the imaginary quadratic field of discriminant $D < 0$. The most efficient representation for handle ideal classes in $\mathbb{Z}_K$ uses an isomorphism with classes of integral binary quadratic forms modulo the action by $\mathrm{SL}_2(\mathbb{Z})$ given by change of variables. We will stick to ideals, which entails minor inefficiencies. See [6, Chap.5] for details.

The proofs of the following lemmas are not difficult and left as exercises:

Lemma 4.42. *If $\alpha \in K \subset \mathbb{C}$, then $N(\alpha) = |\alpha|^2$.*

Lemma 4.43. *Let K be a quadratic field of discriminant D. All integral ideals in $\mathbb{Z}_K$ are of the form $\mathfrak{A} = \delta\big(a\mathbb{Z} + \frac{-b+\sqrt{D}}{2}\mathbb{Z}\big)$ for some $a, \delta \in \mathbb{Z}_{>0}$, $b \in \mathbb{Z}$, such that $-a < b \leqslant a$ and $b^2 \equiv D \pmod{4a}$. Conversely, these $\mathbb{Z}$-modules are* distinct *ideals (= $\mathbb{Z}_K$-modules). Further, $\mathfrak{A} \cap \mathbb{Z} = \delta a\mathbb{Z}$ and $N\mathfrak{A} = a\delta^2$.*

Proof. Tedious but simple computations. (Write $\mathbb{Z}_K = \left\langle 1, \frac{D+\sqrt{D}}{2} \right\rangle_{\mathbb{Z}}$ and $\mathfrak{A}$ as a submodule given by an HNF basis.) Note that the lemma holds for real and imaginary fields. □

Definition 4.44. An integral ideal is *primitive* if it is not of the form $\delta\mathfrak{A}$ with $\mathfrak{A}$ integral and $\delta > 1$. We represent the primitive ideal $\left(a\mathbb{Z} + \frac{-b+\sqrt{D}}{2}\mathbb{Z}\right)$ by the triple $(a, b, c) \in \mathbb{Z}^3$, with $a > 0$, $-a < b \leqslant a$ and $c := (b^2 - D)/(4a) > 0$. Since c can be deduced from a, b and D, we may omit it in the notation, as in $(a, b, *)$.

The condition $-a < b \leqslant a$ only says that $b = \operatorname{MOD}(b, 2a)$, and corresponds to the obvious fact that $\langle a, \tau\rangle_{\mathbb{Z}} = \langle a, \tau - qa\rangle_{\mathbb{Z}}$ for any $q \in \mathbb{Z}$. Since we are mostly interested in ideal classes, it does not hurt to assume that ideals are primitive. Note also that if $\tau = aX - \frac{-b+\sqrt{D}}{2}Y$ lies in the ideal (a, b, c), i.e. if $X, Y \in \mathbb{Z}$, then

$$N(\tau) = |\tau|^2 = a(aX^2 + bXY + cY^2) \tag{4.1}$$

(Here come the binary quadratic forms.)

Lemma 4.45. *With the previous notations, let*

$$\mathfrak{A} = (a, b, c), \quad \mathfrak{B} = (c, \operatorname{MOD}(-b, 2c), *), \quad \tau = \frac{b + \sqrt{D}}{2a} \in K.$$

Then $\mathfrak{A}$ and $\mathfrak{B}$ are in the same ideal class. In fact, $\mathfrak{B} = (\tau)\mathfrak{A}$.

Corollary 4.46. *Each ideal class in $\mathbb{Z}_K$ contains a unique ideal of the form $\mathfrak{A} = (a, b, c)$ such that*

$$|b| \leqslant a \leqslant c,$$

and $b \geqslant 0$ if one inequality is an equality. Such an ideal is called reduced.

Proof. Existence is easy: pick a representative in the ideal class such that $a \in \mathbb{Z}_{>0}$ is minimal. Then $a \leqslant c$ by minimality and the previous lemma. The other conditions are easy.

We now prove unicity. Using (4.1), we claim that

$$a^2 = \min_{x \in \mathfrak{A}\setminus\{0\}} |x|^2, \quad ac = \min_{x \in \mathfrak{A}\setminus\mathbb{Z}} |x|^2.$$

Let us prove the first one: a^2 is obviously attained ($X = 1$, $Y = 0$); using the defining equalities $|b| \leqslant a \leqslant c$, we have

$$aX^2 + bXY + cY^2 \geqslant a(X^2 - |XY| + Y^2) \geqslant a,$$

since $X, Y \in \mathbb{Z}$ implies

$$X^2 - |XY| + Y^2 = (X - |Y|/2)^2 + \frac{3}{4}Y^2 \geqslant 1, \quad (X, Y) \in \mathbb{Z}^2 \setminus \{(0, 0)\}.$$

The second equality follows analogously, by investigating the consequences of $Y \neq 0$.

Assume now that $\mathfrak{A} = (a, b, c)$ and $\mathfrak{A}' = (a', b', c')$ are reduced and in the same ideal class: $\mathfrak{A} = \alpha\mathfrak{A}'$ for some $\alpha \in K^*$. This implies $N\mathfrak{A} = N(\alpha)N\mathfrak{A}'$ hence $a = |\alpha|^2 a'$ by the multiplicativity of norms and Lemma 4.42. Then

$$\left\{|x|^2 : x \in \mathfrak{A}' \setminus \{0\}\right\} = |\alpha|^2 \left\{|x'|^2 : x' \in \mathfrak{A}' \setminus \{0\}\right\},$$

hence they have the same minimum and $a^2 = |\alpha|^2 a'^2$. It follows that $a = a'$ and $|\alpha| = 1$. Then $\mathfrak{A} \cap \mathbb{Z} = \mathfrak{A}' \cap \mathbb{Z} = a\mathbb{Z}$ and

$$\left\{|x|^2 : x \in \mathfrak{A}' \setminus \mathbb{Z}\right\} = \left\{|x'|^2 : x' \in \mathfrak{A}' \setminus \mathbb{Z}\right\},$$

from which we obtain $ac = a'c'$, hence $c = c'$. Since $b^2 - 4ac = b'^2 - 4a'c'$, it follows that $b = \pm b'$. By the convention on the sign of b, we can assume $0 < |b| < a < c$, otherwise $b, b' \geqslant 0$ and we are done.

In these conditions, the only solutions of $aX^2 + bXY + cY^2 = a$ are $(\pm 1, 0)$ (refine the computations above). Since $a \in \mathfrak{A}$, there exists $u = Xa' - Y\frac{-b'+\sqrt{D}}{2} \in \mathfrak{A}'$, $X, Y \in \mathbb{Z}$, such that $a = \alpha u$, hence $|u|^2 = a^2$, which now implies that $(X, Y) = (\pm 1, 0)$. Finally $\alpha = \pm 1$, and $\mathfrak{A} = \mathfrak{A}'$. □

Note that $\mathbb{Z}_K$ is always reduced and is represented either by $(1, 0, -D/4)$ or by $(1, 1, (1 - D)/4)$ depending on $D \bmod 4$. More importantly, the triples representing reduced ideals are easily bounded: in fact, let $\Delta = |D| = -D$, the definitions $|b| \leqslant a \leqslant c$ and $D = b^2 - 4ac$ imply

$$-D = 4ac - b^2 \geqslant 4a^2 - a^2 = 3a^2,$$

hence $a \leqslant \sqrt{\Delta/3}$. This yields a simple algorithm to enumerate $\mathrm{Cl}(K)$:

```
Algorithm 25. Class group of imaginary quadratic fields
Input: $D < 0$, discriminant of a quadratic fields $K$; we let $\Delta = -D$.
Output: A set of reduced ideal representatives of $\mathrm{Cl}(K)$.
for $a = 1, \dots, \sqrt{\Delta/3}$ do
    for $b = 0, \dots, a$ such that $b \equiv \Delta \pmod 2$ do
        Let $c = (b^2 + \Delta)/4a$.
        if $c \in \mathbb{Z}$ and $a \leqslant c$ then
            Print $(a, b, c)$.
            If $b \neq a \neq c$ and $b \neq 0$, print $(a, -b, c)$.
```

Theorem 4.47. *This algorithm prints all reduced ideals in $\mathbb{Z}_K$ in time $\widetilde{O}(D)$.*

Proof. The two outer loops have length $O(\sqrt{\Delta})$. □

One proves that $\#\,\mathrm{Cl}(D) \ll \sqrt{\Delta}\log\Delta$, hence the algorithm above is certainly not optimal. In fact the Brauer-Siegel theorem states that $\log\#\,\mathrm{Cl}(D) \sim \log(\sqrt{\Delta})$ as D tends to infinity, so the size of the output is indeed roughly $\sqrt{\Delta}$. It is possible to enumerate $\mathrm{Cl}(D)$ in time essentially linear in the size of the output by describing first $\mathrm{Cl}(D)$ in terms of generators and relations, see §5.6. Note that the error terms in Brauer-Sieger are *ineffective* unless we assume a Riemann Hypothesis.

Algorithm 26. Reduction of ideals in imaginary quadratic fields

Input: A primitive integral ideal $\mathfrak{A} = (a, b, c)$ in $\mathbb{Z}_K$.

Output: The reduced representative $\mathfrak{B}$ of the ideal class of $\mathfrak{A}$, and a $\tau \in K^*$ such that $\mathfrak{B} = \tau\mathfrak{A}$.

```
Set $\tau \leftarrow 1$.
while $a > c$ or $(b < 0$ and $a = c)$ do
    Let $a' = c$, $b' = \mathrm{MOD}\,(b, 2a')$, $c' = (b'^2 - D)/(4a')$.
    Set $(a, b, c) \leftarrow (a', b', c')$.
    Set $\tau \leftarrow \tau \times \frac{b+\sqrt{D}}{2a}$.
```

Proof. Since we assume $-a < b \leqslant a$, it is impossible that $b = -a$ and the output is correct by repeated application of Lemma 4.45. The only thing to prove is that the algorithm stops. If $a = c$ and $b < 0$ at the beginning of the loop then (a, b, a) is replaced by $(a, -b, a)$ and we exit the loop in the next iteration. Otherwise, $a \in \mathbb{Z}_{>0}$ decreases strictly from one loop to the next, and we are done. With a little care [6, Prop 5.4.3], one proves $a' \leqslant a/2$ except in the very last iteration, so there are $O(\log a)$ iterations. □

Algorithm 27. Solving $U^2 - DV^2 = 4N$

Input: $D < 0$ discriminant of a quadratic field; N an odd prime coprime to D.

Output: Return a solution (U, V) of $U^2 - DV^2 = 4N$, or `False` if none exist.

```
Factor the quadratic polynomial $X^2 - D$ in $\mathbb{Z}/N\mathbb{Z}$, as if $N$ were prime.
If the polynomial is irreducible, return False.
Let $0 < b < N$ be a square root of $D$ modulo $N$. If $b$ and $D$ have different
  parities, set $b \leftarrow N - b$.
    {Now, $b^2 \equiv D \pmod{4N}$.}
Let $c = \frac{b^2-D}{4N} \in \mathbb{Z}$. Using the previous reduction algorithm with input $\mathfrak{A} =$
  $(N, b, c)$, find a reduced representative $\mathfrak{B} = (a, *, *)$ of $\mathfrak{A}$, and $\tau \in K$ such
  that $\mathfrak{B} = \tau\mathfrak{A}$.
If $a \neq 1$, return False.
Write $1/\tau = \frac{U+V\sqrt{D}}{2}$ and return $(U, V)$.
```

Proof. The equation means $N = \alpha\overline{\alpha}$, where $\alpha = (U + V\sqrt{D})/2$ is an algebraic integer, since it is a root of the monic $X^2 - UX + N \in \mathbb{Z}[X]$. The integral ideal (α) has norm N, which is prime, hence it must be primitive and can be represented as $(N, b, *)$ where $b^2 \equiv D \pmod{4N}$. Provided this is possible, the equation has a solution if and only if this ideal is principal, i.e. $\mathfrak{B} = \mathbb{Z}_K$, which is the same as $a = 1$. □

The proof shows that, if $\alpha = (U + V\sqrt{D})/2$ is the solution returned for the equation $\mathrm{Norm}\alpha = N$, the others are of the form $\zeta\alpha$ or $\zeta\bar{\alpha}$, where $\zeta \in \mathbb{Z}_K^*$. Hence there are either 0 or exactly $2w(D)$ solutions, where $w(D) = \#\mathbb{Z}_K^*$ is the number of units in $\mathbb{Z}_K$, i.e. $w(D) = 2$ for $D < -4$, $w(-4) = 4$, $w(-3) = 6$.

4.2.6 ECPP

Let $K = \mathbb{Q}(\sqrt{D})$ be an *imaginary* quadratic field of discriminant $D < 0$. If $\mathfrak{A} \subset \mathbb{Z}_K \subset \mathbb{C}$ is an integral ideal, $\mathbb{C}/\mathfrak{A}$ is an elliptic curve with CM by $\mathbb{Z}_K$.

Theorem 4.48. *1. The Weierstrass equation E of $\mathbb{C}/\mathfrak{A}$ is defined over H_K, in particular $j(E) = j(\mathfrak{A}) \in H_K$. Note that $j(\mathfrak{A})$ only depends on the class of $\mathfrak{A}$ in $\mathrm{Cl}(K)$. In fact, if $\mathfrak{A}$ runs through the classes of $\mathrm{Cl}(K)$, the $j(\mathfrak{A})$ define the $\mathbb{C}$-isomorphism classes of elliptic curves with CM by $\mathbb{Z}_K$.*

2. Let

$$\Phi(X) = \prod_{\mathfrak{A} \in \mathrm{Cl}(K)} \big(X - j(\mathfrak{A})\big).$$

Then $\Phi(X) \in \mathbb{Z}[X]$ is irreducible and any root of Φ generates H_K/K.

3. A prime N splits completely in $H_K/\mathbb{Q}$ if and only if the equation $U^2 - DV^2 = 4N$ has a solution in integers U, V. In this case, maximal ideals $\mathfrak{p}$ above N in H_K satisfy $\mathbb{Z}_K/\mathfrak{p} = \mathbb{Z}/N\mathbb{Z}$ and are principal.

Exercise 4.49. Using the algorithm in the previous lecture, show that $K = \mathbb{Q}(\sqrt{-163})$ has trivial class group. Then explain why $\exp(\pi\sqrt{163})$ is very close to an integer.

For a root $j \in H_K$ of the modular polynomial $\Phi(X)$, let $E(j)$ the complex curve of Corollary 4.34, which is defined over H_K. In the situation of (3), N splits completely and we can intuitively reduce the equation E modulo $\mathfrak{p}$, to obtain $\bar{E}$ over $\mathbb{Z}/N\mathbb{Z}$. Unfortunately, even though j is integral, the equation $E(j)$ does not have integral coefficients, and denominators are a nuisance. The proper way to proceed is as follows: $\Phi(X)$ splits into distinct linear factors modulo N and its roots $\bar{j}$ are the reductions of the j invariants of the $\mathbb{C}$-isomorphism classes of curves with CM by $\mathbb{Z}_K$. The whole point of the construction, besides surprisingly producing explicit curves $\bar{E} = E(\bar{j})$ over $\mathbb{Z}/N\mathbb{Z}$ from complex analytic data, is that the $\#\bar{E}(\mathbb{Z}/N\mathbb{Z})$ are known:

Theorem 4.50. 1. *In the above situation, $\#\bar{E}(\mathbb{Z}/N\mathbb{Z}) = N+1-U$ for some solution (U,V) of $U^2 - DV^2 = 4N$. Note that Hasse's bound $|U| < 2\sqrt{N}$ is obvious in this case.*

2. *Conversely, there are $w(D)$ such solutions U, where $w(D) = \#\mathbb{Z}_K^*$, and each give rise to the cardinality of a curve*

We can now formulate roughly the main ideas in Atkin's algorithm. Its complexity is not rigorously analyzed, but optimized variants are the fastest known methods in practice.

Algorithm 28. ECPP primality test

Input: N an integer

Output: `True` if N is prime, `False` otherwise.

```
for $D = -3, -4, -7, \dots$ do   {Loop over imaginary quadratic fields $K = \mathbb{Q}(\sqrt{D})$}
    if $D$ is a discriminant and $4N = U^2 - DV^2$ has integer solutions then
        Compute $m_U = N + 1 - U$ for all solutions $(U, V)$. If one of these is
        completely factored up to a large probable prime $q \geqslant (N^{1/4} + 1)^2$.
Compute representatives $\mathfrak{A}$ for the elements of Cl($K$).
Compute floating point approximations of the $j(\mathfrak{A})$, then of the polynomial
$\Phi(X)$ and round its coefficients to the nearest integer.
    {If $N$ is prime, then $\Phi$ splits in $\mathbb{Z}/N\mathbb{Z}$}
Compute a root $\bar{j}$ of $\Phi$ in $\mathbb{Z}/N\mathbb{Z}$, and write a Weierstrass equation for $\bar{E}$ with
$j$-invariant $\bar{j}$.
Test a few random points $P$ until we believe that $\#\bar{E}(\mathbb{Z}/N\mathbb{Z}) = m_U$, i.e. we
always have $[m_U]P = O_E$. If for even a single $P$, the test fails, replace $E$ by
its quadratic twist. (If $D = -3, -4$, consider all quadratic twists until we
find one whose cardinality is probably $m_U$.)
Use the Goldwasser-Killian test to prove that $N$ is prime, assuming that $q$
is: pick a random point $P$ and check that $[m_U/q]P = (x : y : z)$, with
$\gcd(z, N) = 1$.
Then recursively prove the primality of $q$.
```

The algorithm is not completely formalized and it is clear there are many places where it can be improved. For instance,

- One should consider the D by increasing class numbers. Also, better invariants than j are known (smaller polynomials than Φ) to compute H_K/K.
- With some more theoretical effort, one can avoid almost all the guesswork when matching m_U to a specific curve (not up to quadratic twist as shown above).

- A heuristic analysis shows that the most time consuming part is the solving of $b^2 \equiv D \pmod{N}$, so we can restrict the D to be products of small primes p_i and precompute their square roots. Then the square root of D can be recovered by a multiplication. This is the main idea in the FastECPP algorithm, which is conjectured to run in randomized time $\widetilde{O}(\log N)^4$, where the original algorithm was conjectured to run in randomized times $\widetilde{O}(\log N)^5$. Current records for primality proofs of general integers use FastECPP and lie around 20000 decimal digits (june 2006).

As in §4.1.4, ECPP produces a primality certificate for N, of the form $C(N)$, which consists in

- the integer N,
- a curve E over $\mathbb{Z}/N\mathbb{Z}$ and a point $P \in E(\mathbb{Z}/N\mathbb{Z})$,
- an integer m divisible by $q \geqslant (N^{1/4} + 1)^2$, such that $[m]P = O_E$ and $[m/q]P = (x : y : z)$ with $\gcd(z, N) = 1$,
- a certificate $C(q)$ for q.

4.2.7 Factoring with elliptic curves

This is straightforward generalization of the $p - 1$ method from §4.1.6.

Algorithm 29. Lenstra's ECM factorization algorithm

Input: N an integer, B a smoothness bound.

Output: A non-trivial factor of N or `Fail`.

```
Using Eratosthenes's sieve, compute all primes $p \leqslant B$.
Pick a random curve $E$ over $\mathbb{Z}/N\mathbb{Z}$ and $P \in E(\mathbb{Z}/N\mathbb{Z})$. Let $Q = P$.
for $p \leqslant B$ do    {compute $Q = [\mathrm{lcm}(2, \dots, B)]P$}
    Let $k$ be maximal such that $p^k \leqslant B$.
    Set $Q := [p^k]Q$. If during the computation we hit a point $R = (x : y : z)$
    with $z \neq 0$ but $d = \gcd(z, N) > 1$, the computation is impossible, but we
    are happy and return the factor $d$!
Return Fail.
```

Just as for the $p - 1$ method, we can introduce a B_2-phase to cater for $\#E(\mathbb{F}_p)$ which are B_1-powersmooth up to a single larger prime $\leqslant B_2$.

Conjecture 4.51. *With suitable parameters, ECM runs in randomized time $L_{1/2}(p)^{1/\sqrt{2}+o(1)}$, where p is the smallest prime divisor of N. Since $p \leqslant \sqrt{N}$, this is $L_{1/2}(N)^{1+o(1)}$.*

4.3 Sieving algorithms

4.3.1 The basic idea

We want to find $x, y \in \mathbb{Z}$ such that

$$x^2 \equiv y^2 \pmod{N}, \quad x \not\equiv \pm y \pmod{N}.$$

Since this is impossible if $(\mathbb{Z}/N\mathbb{Z})^*$ is cyclic (unless we stumble directly on x, y not belonging to $(\mathbb{Z}/N\mathbb{Z})^*$, which is highly implausible but would indeed yield factors), we must check that N is an not a prime power before embarking on this course of action. We may as well require that it is not a pure power, of the form $N = q^k$. Note that $k = O(\log N)$ and an approximation to $N^{1/k}$ is easily computed for each given k using Newton's method (or approximating $\exp(\frac{1}{k}\log N)$), so this is not a costly pre-condition :

Algorithm 30. Generic sieving factorization algorithm

Input: An odd integer N, not a pure power.

Output: A non-trivial factor of N or `Fail`.

```
1: Choose a factorbase $\mathcal{B}$, e.g. $\{-1\} \cup \{p \leqslant B\}$
2: Produce many congruences of the form
       $$x_j^2 \equiv \prod_{i \in \mathcal{B}} i^{e_{i,j}} \pmod{N}, \quad j \leqslant J,$$
   where $e_{i,j} \in \mathbb{Z}$, but may as well be taken in $\{0, 1\}$.
3: If $v = (v_j)$ is a non-zero vector in the kernel of $(e_{i,j})$, viewed as a matrix
   over $\mathbb{F}_2$, then $\sum_j e_{i,j} v_j \equiv 0 \pmod{2}$ for all $i \in \mathcal{B}$ and
       $$\left(\prod_j x_j^{v_j}\right)^2 \equiv \left(\prod_{i \in \mathcal{B}} i^{\frac{1}{2}\sum_j e_{i,j} v_j}\right)^2 \pmod{N},$$
   which is of the required form $x^2 \equiv y^2 \pmod{N}$.
4: If $x \equiv \pm y \pmod{N}$, return Fail.
```

Of course, in practice, one never returns `Fail`, but computes further kernel vectors. The number of square roots of 1 in $\mathbb{Z}/N\mathbb{Z}$ is 2^ω, where ω is the number of distinct prime divisors of N. So two random (x, y) such that $x^2 \equiv y^2 \pmod{N}$ yield a trivial factor with probability $2^{1-\omega} \leqslant 1/2$ provided $\omega \geqslant 2$.

The whole difficulty is now to find relations, and then to choose suitably the factorbase depending of our relation-finding algorithm. Let

$$B := \left\lfloor \sqrt{N} \right\rfloor.$$

The simplest idea (Dixon's random squares) is to pick $t > B$ at random. If $\mathrm{MOD}\,(t^2, N)$ factors on $\mathcal{B}$, we have found a relation. Of course, we want $t \approx B$ to that $\mathrm{MOD}\,(t^2, N)$ be of the order of $\sqrt{N}$ and no larger. This method provably runs in expected time $L_{1/2}(N)^{O(1)}$.

For instance, we may try $t = B + a$, for $a = 1, 2, \ldots$. Note that if a is not too large, then

$$\mathrm{MOD}\left(t^2, N\right) = (B + a)^2 - N,$$

which is the basic idea behind the quadratic sieve.

4.3.2 The quadratic sieve

It is quite costly to check directly that a given integer is B-smooth: about $B/\log B$ divisions. Sieves are slower, but they can check a whole range of numbers simultaneously, at an essentially constant cost per number. Just like Eratosthenes's sieve produces many more primes than a single primality proof. The main problem with the trial division approach is that almost all numbers tested are not smooth, so we have a huge amount of wasted work.

Let $Q(X) = (X + B)^2 - N$, with $B = \left\lfloor \sqrt{N} \right\rfloor$. Since this is a polynomial in $\mathbb{Z}[X]$, if $m \mid Q(a)$, then $m \mid Q(a + \lambda m)$ for all $\lambda \in \mathbb{Z}$.

Algorithm 31. Quadratic sieve

Input: An integer N, a factorbase $\mathcal{B}$, a sieving bound $M > 1$. We assume that all $p \in \mathcal{B}$ satisfy $\left(\frac{N}{p}\right) = 1$.

Output: A set of $a \leqslant M$ such that $Q(a)$ is $\mathcal{B}$-smooth: $p \mid Q(a)$ implies $p \in \mathcal{B}$.

```
Build an array $A[a] = Q(a)$, for $1 \leqslant a \leqslant M$.
If $2 \in \mathcal{B}$, replace $A[a]$ by its largest odd factor for all $a \leqslant M$.
for $p \in \mathcal{B}$ odd prime do
    Find the largest $k$ such that $p^k \leqslant Q(M)$.
    Find $a_k, b_k$ such that $Q(a_k), Q(b_k) \equiv 0 \pmod{p^k}$.    {Two solutions mod $p$ then Hensel lift.}
    for $i = k, k-1, \ldots, 1$ do
        If $i < k$, set $(a_i, b_i) \leftarrow (a_{i+1}, b_{i+1}) \bmod p^i$.
        for $\lambda \leqslant M/p^i$, $p \nmid \lambda$ do
            Divide $A[a_i + \lambda p^i]$ by $p^i$ in place.    {exact division}
            Divide $A[b_i + \lambda p^i]$ by $p^i$ in place.    {exact division}
Return all the $a$ such that $A[a] = 1$.
```

Proof. The reason for the condition $\left(\frac{N}{p}\right)$ is that p cannot divide $Q(a)$ unless N is a square modulo p. Of course, if $p \mid N$, we have factored N. So the assumption is harmless. Since we ensure $p \neq 2$, it follows that the equation $x^2 \equiv N \pmod{p^k}$ has exactly 2 solutions for all k. The result is correct since we divide the $A[a]$ by primes belonging to $\mathcal{B}$. Note that we may miss some smooth numbers this way, but this agrees with the specifications. □

Provided M is large enough, the sieve proper dominates the running time. In an actual implementation, two basic improvements are useful:

- Initialize $A[a]$ by a rough approximation to $\log Q(a)$, then subtract approximations to $i \log p$ instead of dividing by p^i: subtractions are cheaper than divisions. Of course, the $\log p$ are precomputed.
- Do not sieve by small primes, which cost a lot (there are lots of them) and do not decrease much the size of $A[a]$. Then we may as well not sieve by prime powers, since most integers will not be divisible by too many squares of a not-so-small primes.

In the end, we check directly by trial division the a such that $A[a]$ is not too large. Due to the above two approximations, we may miss quite a few smooth numbers, which seems wasteful. But we still expect a large number of the corresponding $Q(a)$ to be smooth. Since the sieve is now much faster, proper tuning results in a net gain: finding twice fewer relations in each sieving range is not a major problem if we find them three times faster!

4.3.3 The Multiple Polynomials Quadratic Sieve (MPQS)

Our polynomial Q is nice but it stands all alone, and the $Q(a)$ increase relatively fast with a. We now replace the polynomial $(X+B)^2 - N$ with a more general polynomial $Q(X) = AX^2 + 2BX + C$, with reduced discriminant $B^2 - AC = N$, hence $AQ(X) = (AX+B)^2 - N$.

What are suitable parameters ? $Q(X)$ is minimal at $-B/A$, with value $-N/A$, which is fine compared to our old $(X+B)^2 - N$ if A is not much smaller than $\sqrt{N}$. We can get a symmetric range of small values: for all $x \in [-B/A-M, -B/A+M]$, we have

$$-N = AQ(-B/A) \leqslant AQ(x) \leqslant AQ(-B/A+M) = (AM)^2 - N,$$

so if $A \leqslant \sqrt{2N}/M$, we have $|Q(x)| \leqslant N/A \approx M\sqrt{N/2}$.

Algorithm 32. Recipe to find Q for the quadratic sieve

Input: An odd integer N, a sieving bound $M > 1$.

Output:

```
1: Find $A$ odd prime such that $A \approx \sqrt{2N}/M$ and $\left(\frac{N}{A}\right) = 1$.
2: Find $B$, $B^2 \equiv N \pmod{A}$ and let $C := (B^2 - N)/A$
3: Return $Q = AX^2 + BX + C$.
```

We take A prime and $\left(\frac{N}{A}\right) = 1$ so that the congruence be solvable (factor a quadratic polynomial over a finite field). Such A are found by trial and error, trying consecutive integers larger than $\sqrt{2N}/M$ until they fail a few compositeness tests and the Legendre symbol has the right value. Now when the $Q(a)$ become large, we can find a new polynomial Q with different arithmetic properties, and still relatively small values!

We must also compute the roots of Q modulo primes in the factorbase. Provided $p \nmid A$, the roots of $Q(a) \equiv 0 \pmod{p}$ are the $(-B + a_1)A^{-1} \pmod{p}$, $(-B + b_1)A^{-1} \pmod{p}$, with a_1, b_1 the square roots of $N \pmod{p}$ as before. If $p \mid A$ there is a single root $-BC^{-1}$ modulo p. Note that it is more difficult to sieve modulo prime powers when $p \mid A$: another reason to avoid it.

4.3.4 The Self Initializing MPQS, Large Prime variations

It is unfortunately rather costly to change Q, so we cannot change it as often as we please. A very simple idea takes care of the problem, reminiscent of FastECPP: we do not need A to be prime, only that $B^2 \equiv N \pmod{A}$ be easy to solve. We consider $A = \prod p_i \approx \sqrt{2N}/M$ for some distinct small primes p_i such that $\left(\frac{N}{p_i}\right) = 1$, with precomputed squares root $B_i^2 \equiv N \pmod{p_i}$. Then we recover B by Chinese remaindering.

A final very important practical improvement are the Large Prime variations (Single, Double, etc.), reminiscent of the B_2 phase in $p - 1$ and ECM. We now maintain a database of relations which are smooth, but for a single Large Prime (or up to *a few* large primes). If we hit another relation which is almost smooth but for the *same* large primes, we can combine them and get new relations; or at least get relations involving fewer large primes. Due to the birthday paradox, we expect to find quite a few new relations this way. This idea can be used in all the sieving factorization algorithms.

In practice, the Single and Double large prime variations are easy to implement, but the combinatorics and the costs of handling the associated graphs become quickly horrendous as we allow more of them.

4.3.5 The Number Field Sieve

Chapter 5

Algebraic Number Theory

5.1 Introduction and definitions

See [19] for a quick introduction to the notions developped in this chapter and [15] for more in-depth discussions and developments. [17] contains a wealth of detailed computations and explicit results.

A *number field* K is a finite extension of $\mathbb{Q}$; we may write $K = \mathbb{Q}(x)$ for some x in K (primitive element theorem). All the elements of K are algebraic over $\mathbb{Q}$. An *algebraic integer* in K is an element $x \in K$ satisfying one of the following equivalent properties:

1. the minimal polynomial of x belongs to $\mathbb{Z}[X]$,

2. there exists $Q \in \mathbb{Z}[X]$, Q monic, such that $Q(x) = 0$,

3. $\mathbb{Z}[x]$ is a $\mathbb{Z}$-module of finite type,

4. there exists $M \subset K$ a $\mathbb{Z}$-module of finite type containing $\mathbb{Z}[x]$.

The set of algebraic integers $\mathbb{Z}_K \subset K$ is a ring, which is the proper analog of $\mathbb{Z}_{\mathbb{Q}} = \mathbb{Z} \subset \mathbb{Q}$ for the arithmetic of K.

A number field K has $\dim_{\mathbb{Q}} K = r_1 + 2r_2$ field embeddings $\sigma : K \hookrightarrow \mathbb{C}$. Among them, r_1 embeddings have image contained in $\mathbb{R}$, and $2r_2$ further pairwise conjugate embeddings. The *norm*, *trace*, *characteristic polynomial* of x in K is the determinant, trace, characteristic polynomial of the multiplication by x seen as a $\mathbb{Q}$-linear endormorphism of K. In particular, $N : K^* \to \mathbb{Q}^*$ and $\mathrm{Tr} : K \to \mathbb{Q}$ are group morphisms (for the multiplicative and additive structure, respectively). Concretely,

$$N(x) = \prod_{\sigma : K \hookrightarrow \mathbb{C}} \sigma(x), \quad \mathrm{Tr}(x) = \sum_{\sigma : K \hookrightarrow \mathbb{C}} \sigma(x), \quad \mathrm{Char}_x(T) = \prod_{\sigma : K \hookrightarrow \mathbb{C}} \bigl(T - \sigma(x)\bigr).$$

The norm, trace and characteristic polynomial of an algebraic integer are integral. In particular, the units $\mathbb{Z}_K^*$ in $\mathbb{Z}_K$ have norm ± 1.

We consider

$$K \otimes \mathbb{R} \simeq_{\mathbb{R}\text{-algebra}} \mathbb{R}^{r_1} \times \mathbb{C}^{r_2}$$

as a Euclidean space, endowed with the canonical Euclidean form $T_2(x) := \sum_{\sigma: K \hookrightarrow \mathbb{C}} |\sigma(x)|^2$. The *discriminant* Δ_K of K is the discriminant of the lattice $(\mathbb{Z}_K, T_2)$. Algebraically, it is the absolute value of the determinant of the matrix $(\mathrm{Tr}(w_i w_j))$, where $\mathbb{Z}_K = \langle w_1, \dots, w_n \rangle_{\mathbb{Z}}$.

An *integral ideal* is a non-zero ideal of $\mathbb{Z}_K$, a *fractional ideal* is a sub $\mathbb{Z}_K$-module of K of rank 1 (equivalent definition: $\mathfrak{A}$ is a fractional ideal if and only if $d\mathfrak{A}$ is integral for some $d \in \mathbb{Z}_{>0}$). Since $\mathbb{Z}_K$ is a Dedekind domain, the fractional ideals form a group and every fractional ideal can be written uniquely as a product of maximal ideals:

$$\mathfrak{A} = \prod_{\mathfrak{p}} \mathfrak{p}^{v_{\mathfrak{p}}(\mathfrak{A})},$$

where $v_{\mathfrak{p}}(\mathfrak{A}) = 0$ for all but finitely many $\mathfrak{p}$. (Note that we exclude the 0 ideal.) A maximal ideal $\mathfrak{p}$ contains a unique prime number p (the generator of $\mathfrak{p} \cap \mathbb{Z}$), and the quotient $\mathbb{Z}_K/\mathfrak{p}$ is a finite field of characteristic p.

A fractional ideal is *principal* if it is of the form $(\alpha) := \alpha\mathbb{Z}_K$ for some $\alpha \in K^*$. The norm $N : K \to \mathbb{Q}$ generalizes to the group of principal ideals by $N(x\mathbb{Z}_K) := |N(x)|$. (It wouldn't be well-defined without the absolute value: x is defined up to units, and units may have norm -1.) This extends to a multiplicative function on the whole group of fractional ideals. For an integral ideal $\mathfrak{A}$, we have $N\mathfrak{A} = \#(\mathbb{Z}_K/\mathfrak{A})$.

The class group of $\mathrm{Cl}(\mathbb{Z}_K)$ is the quotient of the group of fractional ideals by the subgroup of principal ideals. It is a *finite* abelian group. In fact, a simple application of Minkowski's theorem proves that each ideal class contains an integral ideal of norm $O(\sqrt{\Delta_K})$.

A *place* of K is an equivalence class of non-trivial absolute values on K. Concretely, canonical representatives for these classes are given as follows:

- r_1 real places : $|x|_\sigma := |\sigma(x)|$ where $\sigma : K \hookrightarrow \mathbb{R}$ is a real embedding.
- r_2 complex places : $|x|_\sigma := |\sigma(x)|$ where $\sigma : K \hookrightarrow \mathbb{C}$ runs through a system of non-congugate complex embeddings.
- one finite place for each maximal ideal $\mathfrak{p}$: $|x|_{\mathfrak{p}} := N\mathfrak{p}^{-v_{\mathfrak{p}}(x)}$.

The $r_1 + r_2$ real and complex places are called *infinite* places. Given a finite set of places S containing all infinite places, we define the S-integers as

$$\mathbb{Z}_{K,S} = \{x \in K \colon v_{\mathfrak{p}}(x) \geqslant 0 \text{ for all } x \notin S\}.$$

The subgroup of invertible elements is

$$\mathbb{Z}_{K,S}^* = \{x \in K \colon v_{\mathfrak{p}}(x) = 0 \text{ for all } x \notin S\}.$$

This is a $\mathbb{Z}$-module of finite type, a direct product of a (finite) cyclic subgroup $\mu(K)$ containing all roots of unity in K and a free module of rank $\#S - 1$. If S contains only the places at infinity, we obtain the ordinary integers and units in K. Note that $\mathbb{Z}_{K,S}$ is still a Dedekind ring (not of finite type over $\mathbb{Z}$), and we can also define a notion of S-class group, by quotienting the $\mathbb{Z}_{K,S}$-fractional ideals by principal ones. It turns out that $\mathrm{Cl}(\mathbb{Z}_{K,S}) \simeq \mathrm{Cl}(\mathbb{Z}_K)/\left\langle \mathfrak{p}\colon \mathfrak{p} \in S\right\rangle$.

Computational algebraic number theory concerns itself with computing and handling all these objects effectively.

5.2 Concrete representations

We assume that K is given abstractly by the minimal polynomial of a generating element: $K = \mathbb{Q}[X]/(T)$, where T is irreducible in $\mathbb{Q}[X]$. It is no loss of generality to assume that T is integral of degree $n = r_1 + 2r_2$. We shall furthermore assume that T is monic. (We can reduce to this case by a change of variable, although most algorithms can be adapted to work directly with non-monic inputs, more efficiently than on the transformed monic polynomial.) We will not explicitly evaluate the complexity of most of our algorithms computing invariants of K, but their input size would be $n \log ||T||_\infty$, and we hope to obtain runtimes bounded by a polynomial in this input size. (We shall not be successful.)

We may then work in K either as in any (univariate) polynomial quotient ring, or as a $\mathbb{Q}$-vector space with canonical basis $1, X, \ldots, X^{n-1}$. In particular, this yields a direct way to compute the norm, trace and characteristic polynomial of $x \in K$ using $\mathbb{Q}$-linear algebra.

The $r_1 + 2r_2$ embeddings are given by $\sigma : X \mapsto \alpha_\sigma$, where the α_σ are the complex roots of T. Since we know how to approximate the complex roots of T within a guaranteed fixed accuracy, we may approximate any $\sigma(x) \in \mathbb{C}$ as floating point complex numbers to an arbitray fixed precision. This gives a different, analytic, method to compute norms, traces and characteristic polynomials (bound denominators, approximate, then round).

$\mathbb{Z}_K$ and all fractional ideals are free $\mathbb{Z}$-module of rank n and can be represented by a $\mathbb{Z}$-basis. Once we fix a $\mathbb{Z}$-basis $(w_1, \ldots, w_n)$ for $\mathbb{Z}_K$, any integral ideal has a canonical basis, given by the Hermite Normal Form, i.e. of the form $(w_i)H$ where H is a square matrix in HNF. Note that $\det H$ is the index of the submodule, hence the norm of the ideal. The product $\mathfrak{A} \times \mathfrak{B}$ is the $\mathbb{Z}$-module generated by all n^2 products $a_i b_i$ of the generators of $\mathfrak{A}$ and $\mathfrak{B}$. The quotient $\mathfrak{A}\mathfrak{B}^{-1}$ is $(\mathfrak{A} : \mathfrak{B})$, where

$$(A : B) := \{\alpha \in K, \alpha B \subset A\},$$

for any $\mathbb{Z}$-modules A and B. Given $\mathbb{Z}$-bases, this quotient can also be computed using $\mathbb{Z}$-linear algebra (i.e. the HNF algorithm).

Units and ideal classes are represented in the obvious way: as elements of K and by any (integral) ideal representative respectively. It is not yet clear how to compute with ideal classes, nor how to enumerate a complete system of representatives. Analogousy, if $\mu(K)$ is relatively straightforward (amounts to factoring cyclotomic polynomials over K, which can be done using generalizations of the techniques we saw over $\mathbb{Q}$), how to find non-torsion units ?

5.3 The maximal order $\mathbb{Z}_K$

An *order* of K is a subring $\mathcal{O}$ (in particular, containing 1) such that $\operatorname{rank}_{\mathbb{Z}} \mathcal{O} = \dim_{\mathbb{Q}} K$. Note that $\mathbb{Z}_K$ is an order and that all orders are contained in $\mathbb{Z}_K$ (property 4. from §5.1), with finite index. We define the discriminant $\Delta_{\mathcal{O}}$ of $\mathcal{O}$, from any $\mathbb{Z}$-basis of $\mathcal{O}$, by mimicking the definition of $\Delta_K = \Delta_{\mathbb{Z}_K}$. A direct computation involving a van der Monde matrix shows that if x is an algebraic integer generating K, with minimal polynomial T, then $\Delta_{\mathbb{Z}[x]} = \operatorname{disc} T := \operatorname{Res}(T, T')$.

The main point of orders is that

- they are easy to construct: $\mathbb{Z}[x]$ is an order for any $x \in \mathbb{Z}_K$, in particular. Given $T \in \mathbb{Z}[X]$ monic such that $K = \mathbb{Q}[X]/(T)$, we obtain a canonical *equation order* (generated by the class of X).

- they approximate $\mathbb{Z}_K$. In fact they have finite index in $\mathbb{Z}_K$, which is easy to bound since we have $\Delta_{\mathcal{O}} = [\mathbb{Z}_K : \mathcal{O}]^2 \Delta_K$ and we know $\Delta_{\mathcal{O}}$.

- they are somewhat imperfect compared to the maximal order, but still have interesting arithmetic properties: they are noetherian integral domains of dimension 1. But a non-maximal order is not integrally closed and the norm $\mathcal{O}/\mathfrak{A}$ is no longer multiplicative, finitely many maximal ideals are not invertible, etc.

Exercise 5.1. More generally, orders appear as rings of stabilizers: if $A \subset K$ is a $\mathbb{Z}$-module of rank n, prove that $\mathcal{O} = (A : A)$ is an order.

In many applications (e.g. factoring polynomials in $K[X]$, splitting primes in K) it is enough to know any order, the computation being more efficient if its index is reasonably small. For instance, given a $\mathbb{Z}$-basis (w_i) for an order $\mathcal{O}$, any integer of K is a $\mathbb{Q}$-linear combination of the w_i, with denominator bounded by the exponent of the additive group $\mathbb{Z}_K/\mathcal{O}$ (itself obviously bounded by the index). On the other hand, to compute more subtle invariants like the class groups and units we will need $\mathbb{Z}_K$ itself.

Definition 5.2. Let m be an integer. On order $\mathcal{O} \subset \mathbb{Z}_K$ is *m-maximal* if m and the index $[\mathbb{Z}_K : \mathcal{O}]$ are coprime.

Theorem 5.3 (Zassenhaus)**.** *Given an order $\mathcal{O}$ and a prime p, we can find in polynomial time a basis for an order $\mathcal{O}_p \supset \mathcal{O}$ such that $\mathcal{O}_p$ is p-maximal.*

Proof. In fact, we will compute $\mathcal{O}_p$ minimal with respect to the requested property, i.e. $[\mathcal{O}_p : \mathcal{O}]$ will be a power of p. Let $I_p = \operatorname{Rad}(p\mathcal{O})$ the radical of $p\mathcal{O}$: by definition, $I_p/p\mathcal{O}$ is the ideal of nilpotents in the finite ring $\mathcal{O}/p\mathcal{O}$. Since a nilpotent in the n-dimensional $\mathbb{F}_p$ vector space $\mathcal{O}/p\mathcal{O}$ has index $\leqslant n$, $I_p/p\mathcal{O}$ is the kernel of the $\mathbb{F}_p$-linear endomorphism of $\mathcal{O}/p\mathcal{O}$ given by $F^t : x \mapsto x^{p^t}$, where t is minimal such that $p^t \geqslant n$.

I_p is the intersection of the minimal prime ideals containing p and we also have

$$I_p := \bigcap_{\mathfrak{p}\,:\, p \in \mathfrak{p}} \mathfrak{p} = \prod_{\mathfrak{p}\,:\, p \in \mathfrak{p}} \mathfrak{p}.$$

There are finitely many such ideals $\mathfrak{p}$ and they are maximal in $\mathcal{O}$. The idea of the method is that $\mathcal{O}$ is p-maximal if and only if all the $\mathfrak{p}$ are invertible (as $\mathcal{O}$-ideals), if and only if I_p is invertible.

Lemma 5.4. *Let $\mathcal{O}' := (I_p : I_p)$. The order $\mathcal{O}$ is p-maximal if and only if $\mathcal{O} = \mathcal{O}'$. Otherwise, $p \mid [\mathcal{O}' : \mathcal{O}] \mid p^n$.*

Proof. $\mathcal{O}'$ is an order containing $\mathcal{O}$ (I_p is a $\mathbb{Z}$-module of rank n, apply Exercice 5.1); since $p \in I_p$, we have $p\mathcal{O}' \subset I_p \subset \mathcal{O}$ hence $[\mathcal{O}' : \mathcal{O}] \mid p^n$. It follows that $\mathcal{O} = \mathcal{O}'$ if $\mathcal{O}$ is p-maximal.

Conversely, if $\mathcal{O} = \mathcal{O}'$, let $R \subset \mathbb{Z}_K$ the smallest p-maximal order containing $\mathcal{O}$. Since I_p and R have finite rank, we have $I_p^m \subset p\mathcal{O}$ and $p^m R \subset \mathcal{O}$ for $m \gg 1$. Finally $RI_p^m \subset \mathcal{O}$ for m large enough. We now show that in fact $m = 0$, hence $R \subset \mathcal{O}$ and $\mathcal{O}$ is p-maximal.

By contradiction, suppose there exists $m \geqslant 0$ such that $RI_p^m \not\subset \mathcal{O}$ and pick it maximal. Choose $\alpha \in RI_p^m \setminus \mathcal{O}$; then $\alpha I_p \subset \mathcal{O}$, refined to $\alpha I_p \subset I_p$: for $k \gg 1$, $I_p^k \subset p\mathcal{O}$, hence $(\alpha I_k)^{k\ell} \subset p^\ell R \subset p\mathcal{O}$ for $\ell \gg 1$. Hence $\alpha \in (I_p : I_p) = \mathcal{O}' = \mathcal{O}$; contradiction. □

The algorithm is now obvious: compute $I_p/p\mathcal{O}$, lift to I_p, then compute $\mathcal{O}'$. Either $\mathcal{O} = \mathcal{O}'$ is p-maximal or we replace $\mathcal{O}$ by $\mathcal{O}'$ and restart, dividing the index at least by p. □

Corollary 5.5. *Given the primes p such that $p^2 \mid \operatorname{disc} T$, we can compute a $\mathbb{Z}$-basis for $\mathbb{Z}_K$ in polyomial time.*

Proof. If $p^2 \nmid \operatorname{disc} T$, the equation order is p-maximal. Otherwise, compute $\mathcal{O}_p$ for all those given p and return $\sum \mathcal{O}_p$, using the HNF algorithm. □

Interestingly, Zassenhaus's algorithm still works if p is only assumed to be squarefree. Either the algorithm exhibit a zero divisor in $\mathbb{Z}/p\mathbb{Z}$ (from which we can factor p and restart), or it produces a p-maximal order. Using these ideas one can prove a stronger result:

Theorem 5.6 (Buchmann-Lenstra). *There are polynomial time algorithm that given a number field K and one of 1), 2) below determines the other:*

1. *the ring of integers of K,*

2. *the largest squarefree divisor of Δ_K.*

Finding the largest squarefree divisor of a given integer is currently essentially as hard as full integer factorization, hence computing $\mathbb{Z}_K$ is difficult; but it becomes easy if an explicit factorization is given. To see why 2) is at least as hard as 1), consider the simplest case of a quadratic field $K = \mathbb{Q}(\sqrt{D})$ for some integer D. How would you compute $\mathbb{Z}_K$ without assuming that D is squarefree ?

5.4 Dedekind's criterion

A very important byproduct of Zassenhaus's algorithm is that it is trivial to check whether a given order $\mathcal{O}$ is p-maximal for p prime (or squarefree, using Buchmann and Lenstra's trick). The recipe simplifies if $\mathcal{O}$ is the equation order:

Theorem 5.7 (Dedekind). *Let p be a prime number. Let $K = \mathbb{Q}(X)/(T)$, $T \in \mathbb{Z}[X]$ monic, such that*

$$T \equiv \prod_i P_i^{e_i} \pmod{p\mathbb{Z}[X]},$$

where the $P_i \in \mathbb{Z}[X]$ are monic, irreducible and distinct modulo p. Let

$$f := \prod P_i, \quad g := \prod P_i^{e_i - 1}, \quad h := (T - fg)/p \in \mathbb{Z}[X].$$

Then the equation order is p-maximal if and only if $\gcd(\overline{f}, \overline{g}, \overline{h}) = 1$ in $\mathbb{F}_p[X]$.

Proof. It follows from the Chinese Remainders that $I_p = p\mathbb{Z}[\theta] + f(\theta)\mathbb{Z}[\theta]$, where θ is the class of X modulo T. From this, one computes $(I_p : I_p)$ explicitly. □

Corollary 5.8. *An Eisenstein polynomial at p yields a p-maximal equation order.*

Proof. $f = X$, $g = X^{n-1}$, $h = (T - X^n)/p$. The gcd is 1. □

5.5 Splitting of primes

Theorem 5.9 (Kummer). *Let $K = \mathbb{Q}[X]/(T)$, $T \in \mathbb{Z}[X]$ monic and $\theta = X \pmod T$. If the equation order is p-maximal, "the factorization of T mod p mirrors the factorization of $p\mathbb{Z}_K$". More precisely, if*

$$T \equiv \prod_i P_i^{e_i} \pmod{p\mathbb{Z}[X]},$$

where the P_i are monic, irreducible and distinct modulo p. Then

$$p\mathbb{Z}_K = \prod_i \mathfrak{p}_i^{e_i},$$

where the $\mathfrak{p}_i := p\mathbb{Z}_K + P_i(\theta)\mathbb{Z}_K$ are distinct maximal ideals, with residual degree $\deg P_i$.

If Dedekind's criterion tells us that the equation order is *not* p-maximal, we can still compute a p-maximal order $\mathcal{O}$ using Zassenhaus's method (Theorem 5.3). In fact, for simplicity, assume we know $\mathbb{Z}_K$. Then we can compute $I_p = \prod \mathfrak{p}_i$ as in Zassenhaus's method, and finding the $\mathfrak{p}_i$ is equivalent to splitting the separable algebra $\mathbb{Z}_K/I_p$, which can be done using an adaptation of Berlekamp's algorithm.

5.6 Ideal class group and units

5.7 Smaller generating sets for the class group

5.8 Class field theory

Bibliography

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, PRIMES *is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793. MR MR2123939

[2] W. R. Alford, Andrew Granville, and Carl Pomerance, *There are infinitely many Carmichael numbers*, Ann. of Math. (2) **139** (1994), no. 3, 703–722. MR MR1283874 (95k:11114)

[3] E. Bach, *Explicit bounds for primality testing and related problems*, Math. Comp. **55** (1990), no. 191, 355–380. MR 91m:11096

[4] E. R. Berlekamp, *Factoring polynomials over large finite fields*, Math. Comp. **24** (1970), 713–735. MR 43 #1948

[5] Peter Borwein and Tamás Erdélyi, *Polynomials and polynomial inequalities*, Graduate Texts in Mathematics, vol. 161, Springer-Verlag, New York, 1995. MR MR1367960 (97e:41001)

[6] Henri Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138, Springer-Verlag, Berlin, 1993. MR MR1228206 (94i:11105)

[7] ______, *Advanced topics in computational number theory*, Graduate Texts in Mathematics, vol. 193, Springer-Verlag, New York, 2000. MR MR1728313 (2000k:11144)

[8] David A. Cox, *Primes of the form x^2+ny^2*, A Wiley-Interscience Publication, John Wiley & Sons Inc., New York, 1989, Fermat, class field theory and complex multiplication. MR MR1028322 (90m:11016)

[9] Richard Crandall and Carl Pomerance, *Prime numbers*, second ed., Springer, New York, 2005, A computational perspective. MR MR2156291 (2006a:11005)

[10] Graham Everest and Thomas Ward, *Heights of polynomials and entropy in algebraic dynamics*, Universitext, Springer-Verlag London Ltd., London, 1999. MR MR1700272 (2000e:11087)

[11] Xavier Gourdon, *Algorithmique du théorème fondamental de l'algèbre*, Rapport de recherche 1852, INRIA, 1993.

[12] Peter Henrici, *Applied and computational complex analysis*, Wiley-Interscience [John Wiley & Sons], New York, 1974, Volume 1: Power series—integration—conformal mapping—location of zeros, Pure and Applied Mathematics. MR MR0372162 (51 #8378)

[13] Henryk Iwaniec and Emmanuel Kowalski, *Analytic number theory*, American Mathematical Society Colloquium Publications, vol. 53, American Mathematical Society, Providence, RI, 2004. MR MR2061214 (2005h:11005)

[14] Serge Lang, *Algebra*, second ed., Addison-Wesley Publishing Company Advanced Book Program, Reading, MA, 1984. MR MR783636 (86j:00003)

[15] ______, *Algebraic number theory*, second ed., Graduate Texts in Mathematics, vol. 110, Springer-Verlag, New York, 1994. MR MR1282723 (95f:11085)

[16] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), no. 4, 515–534. MR 84a:12002

[17] Władysław Narkiewicz, *Elementary and analytic theory of algebraic numbers*, second ed., Springer-Verlag, Berlin, 1990. MR 91h:11107

[18] Christos H. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994. MR 95f:68082

[19] Pierre Samuel, *Théorie algébrique des nombres*, Hermann, Paris, 1967. MR MR0215808 (35 #6643)

[20] J.-P. Serre, *A course in arithmetic*, Springer-Verlag, New York, 1973, Translated from the French, Graduate Texts in Mathematics, No. 7. MR MR0344216 (49 #8956)

[21] J. H. Silverman, *The arithmetic of elliptic curves*, Springer-Verlag, New York, 1986. MR 87g:11070

[22] A. Storjohann, *Algorithms for matrix canonical forms*, Ph.D. thesis, ETH Zurich, 2000, `http://www.cs.uwaterloo.ca/~astorjoh/dissA4.ps`.

[23] G. Tenenbaum, *Introduction à la théorie analytique et probabiliste des nombres*, Pub. Inst. Elie Cartan, 1990.

[24] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, Cambridge University Press, New York, 1999. MR 2000j:68205