Polynomial-time algorithms in algebraic number theory

Written by

D.M.H. van Gent

Based on lectures of

H.W. Lenstra

Mathematisch Instituut
Universiteit Leiden
Netherlands
July 19, 2021

1 Introduction

In these notes we study methods for solving algebraic computational problems, mainly those involving number rings, which are fast in a mathematically precise sense. Our motivating problem is to decide for a number field K, elements $a_1, \ldots, a_t \in K^*$ and $n_1, \ldots, n_t \in \mathbb{Z}$ whether $\prod_i a_i^{n_i} = 1$, which turns out to be non-trivial. From basic operations such as addition and multiplication of integers we work up to computation in finitely generated abelian groups and algebras of finite type. Here we encounter lattices and Jacobi symbols. Some difficult problems in computational number theory are known, for example computing a prime factorization or a maximal order in a number field. We will however focus mainly on the positive results, namely the problems for which fast computational methods are known.

1.1 Algorithms

To be able to talk about polynomial-time algorithms, we should first define what an algorithm is. We equip the natural numbers $\mathbb{N} = \mathbb{Z}_{\geq 0}$ with a length function $l : \mathbb{N} \to \mathbb{N}$ that sends n to the number of digits of n in base 2, with $l(0) = 1$.

Definition 1.1. A *problem* is a function $f : I \to \mathbb{N}$ for some set of inputs $I \subseteq \mathbb{N}$ and we call f a *decision problem* if $f(I) \subseteq \{0, 1\}$. An *algorithm* for a problem $f : I \to \mathbb{N}$ is a 'method' to compute $f(x)$ for all $x \in I$. An algorithm for f is said to run in *polynomial time* if there exist $c_0, c_1, c_1 \in \mathbb{R}_{>0}$ such that for all $x \in I$ the time required to compute $f(x)$, called the *run-time*, is at most $(c_0 + c_1 l(x))^{c_2}$. We say a problem f is *computable* if there exists an algorithm for f.

This definition is rather empty: we have not specified what a 'method' is, nor have we explained how to measure run-time. We will briefly treat this more formally. The reader for which the above definition is sufficient can freely skip the following paragraph. The main conclusion is that we will not heavily rely on the formal definition of run-time in these notes.

In these notes, the word algorithm will be synonymous with the word *Turing machine.* For an extensive treatment of Turing machines, see [3]. A Turing machine is a model of computation described by Alan Turing in 1936 that defines an abstract machine which we these days think of as a computer. The main differences between a Turing machine and a modern day computer is that the memory of a Turing machine is a tape as opposed to random-access memory, and that a Turing machine has infinite memory. The run-time of a Turing machine is then measured as the number of elementary tape operations: reading a symbol on the tape, writing a symbol on the tape and moving the tape one symbol forward or backward. It is then immediately clear that it is expensive for a Turing machine to move the tape around much to look up data, as opposed to the random-access memory model where the cost of a memory lookup is constant, regardless of where the data is stored in memory. This also poses a problem for our formal treatment of run-time, as it may depend on our model of computation. However, both models of computation are able to emulate each other in such a way that it preserves the property of computability in polynomial time, even though the constants c_0, c_1 and c_2 as in Definition 1.1 may increase drastically. We use this as an excuse to be informal in these notes about determining the run-time of an algorithm.

1.2 Basic computations

In these notes we build up our algorithms from basic building blocks. First and foremost, we remark that the basic operations in $\mathbb{Z}$ and $\mathbb{Q}$ are fast. Addition, subtraction, multiplication and division (with remainder in the case of $\mathbb{Z}$) can be done in polynomial time, as well as checking the sign of a number and whether numbers are equal. We assume here that we represent a rational number by a pair of integers, a numerator and a denominator. We may even assume the numerator and denominator are coprime: Given $a, b \in \mathbb{Z}$ we can compute their greatest common divisor $\gcd(a, b)$ and solve the Bézout equation $ax + by = \gcd(a, b)$ for some $x, y \in \mathbb{Z}$ using the (extended) Euclidean algorithm in polynomial time. Applying these techniques in bulk we can also do addition, subtraction and multiplication of integer and rational matrices in polynomial time. Least trivially of our building blocks, using the theory of lattices we can compute bases for the kernel and the image of an integer matrix in polynomial time, which is the topic of Section 3.

1.3 Commutative algebra and number theory

Definition 1.2. Let R be a commutative ring. We write $\operatorname{spec} R$ for the set of prime ideals of R. We say R is *local* if R has a unique maximal ideal. For an ideal $I \subseteq R$ we define the *radical* $\sqrt{I}$ of I to be the ideal $\{x \in R \,|\, (\exists n \in \mathbb{Z}_{>0})\, x^n \in I\}$. We call $\operatorname{nil}(R) = \sqrt{0R}$ the *nilradical* of R and we call the elements of $\operatorname{nil}(R)$ the *nilpotents* of R. We say R is *reduced* when $\operatorname{nil}(R) = 0$. We say ideals $I, J \subseteq R$ are *coprime* when $I + J = R$.

Definition 1.3. A *number field* is a field K containing the field of rational numbers $\mathbb{Q}$ such that the dimension of K over $\mathbb{Q}$ as vector space is finite. A *number ring* is a subring of a number field.

For a number ring R, write $R_{\mathfrak{p}}$ for the localization at the prime ideal $\mathfrak{p}$ of R.

Definition 1.4. Let R be a number ring and let $K = R_{0R}$ be its field of fractions. A *fractional ideal* of R is a finitely generated non-zero R-submodule of K. A fractional ideal of R is *integral* if it is contained in R. A fractional ideal of R is *principal* if it is of the form xR for some $x \in K$. We write $I + J$ and $I \cdot J$ for the R-modules generated by $\{i + j \mid i \in I,\, j \in J\}$ respectively $\{i \cdot j \mid i \in I,\, j \in J\}$. A fractional ideal I of R is *invertible* if there exists some fractional ideal J of R such that $I \cdot J$ is principal. A non-invertible fractional ideal is called *singular*. For fractional ideals I and J of R write $I : J = \{x \in K \,|\, xJ \subseteq I\}$.

Definition 1.5. An *order* is a commutative ring whose additive group is isomorphic to $\mathbb{Z}^n$ for some $n \in \mathbb{Z}_{\geq 0}$. We say R is an order of a number field K if $R \subseteq K$ and $R_{0R} = K$.

Any reduced order is contained in some finite product of number fields. In our algorithms we encode an order by first specifying its rank n, and then writing down its $n \times n$ multiplication table for the standard basis vectors. By distributivity this completely and uniquely defines a multiplication on the order. We encode a number field K simply by an order R such that $R_{0R} = K$.

1.4 Exercises

Exercise 1.1. Let $\mathbb{Q}^{\text{alg}}$ be some algebraic closure of $\mathbb{Q}$.

- **a.** Show that there are precisely $\#\mathbb{N}$ number fields contained in $\mathbb{Q}^{\text{alg}}$ up to isomorphism.
- **b.** Show that there are precisely $\#\mathbb{R}$ subfields of $\mathbb{Q}^{\text{alg}}$ up to isomorphism.
- **c.** Do there exist $\#\mathbb{R}$ subfields of $\mathbb{Q}^{\text{alg}}$ that are pairwise isomorphic?
- **d.** Let $K \neq \mathbb{Q}$ be a number field. Show that there are precisely $\#\mathbb{N}$ orders and $\#\mathbb{R}$ subrings in K with field of fractions K.
- **e.** Argue why it is natural to restrict the input of our algorithms to orders and number fields as opposed to general number rings.

Exercise 1.2. Show that there exists a polynomial-time algorithm that, given a square integer matrix, determines whether it encodes an order.

Exercise 1.3. Let R be a commutative ring. Show that R is a local ring if and only if $R \setminus R^*$ is an additive subgroup of R.

Exercise 1.4. Let R be an order of a number field K. Prove that $R = \mathcal{O}_K$ if and only if the sum of every two invertible ideals is again invertible.

Exercise 1.5. Let R be a commutative ring. Show that $\operatorname{nil}(R)$ is equal to the intersection of all prime ideals of R. Moreover, show that if $\operatorname{nil}(R)$ is finitely generated, then $\operatorname{nil}(R)$ is nilpotent.

Exercise 1.6. Show that any finite commutative domain is a field. Conclude that in a general commutative ring prime ideals of finite index are maximal.

Exercise 1.7. Let R be a commutative ring and let $I_1, \ldots, I_m, J_1, \ldots, J_n \subseteq R$ be ideals such that for all i, j we have $I_i + J_j = R$. Show that $I_1 \cdots I_m + J_1 \cdots J_n = R$. Conclude that for any two distinct maximal ideals $\mathfrak{m}, \mathfrak{n} \subseteq R$ and any $m, n \in \mathbb{Z}_{\geq 0}$ we have $\mathfrak{m}^m + \mathfrak{n}^n = R$.

Exercise 1.8 (Chinese remainder theorem for ideals)**.** Let R be a commutative ring and let $I_1, \ldots, I_n \subseteq R$ be pair-wise coprime ideals. Show that $\bigcap_{i=1}^n I_i = \prod_{i=1}^n I_i$ and prove that the natural homomorphism

$$R/\Big(\bigcap_{i=1}^n I_i\Big) \to \prod_{i=1}^n (R/I_i)$$

is an isomorphism.

Exercise 1.9. Let I, J be fractional ideals in a number ring R.
- **a.** Show that if $IJ = R$, then $J = R : I$.
- **b.** Show that if I is invertible, then $I : I = R$.
- **c.** Show that IJ is invertible if and only if I and J are invertible.

Exercise 1.10. Let I be an ideal in a number ring R.
- **a.** Show that there exist prime ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_n \subseteq R$ that contain I and satisfy $\mathfrak{p}_1 \cdots \mathfrak{p}_n \subseteq I$.
- **b.** Suppose I is a product of prime ideals. Show that $I : I = R$ if and only if I is invertible.

Hint: First suppose I is prime. If I is singular and $a \in \mathfrak{p}$ is non-zero, then $\mathfrak{p}_1 \cdots \mathfrak{p}_t \subseteq aR \subseteq I$ and without loss of generality $\mathfrak{p}_1 = I$. Any $b \in \mathfrak{p}_2 \cdots \mathfrak{p}_n \setminus aR$ satisfies $bI \subseteq aR$.

Exercise 1.11. Let α be an algebraic integer of degree at least 3 and let p be a prime number. Show that $R = \mathbb{Z} + p\mathbb{Z}[\alpha]$ is a domain, and that $I = \alpha\mathbb{Z} + R$ is a fractional R-ideal. Moreover, prove that $I : I = R$ and that I is not invertible.
Note: This provides a counter-example to the converse of Exercise 1.9.b.

2 Coprime base factorization

In this section we treat the following problem, which will be the motivation for the *coprime base algorithm.*

Theorem 2.1. *There is a polynomial time algorithm that on input $t \in \mathbb{N}$, $q_1, \ldots, q_t \in \mathbb{Q}^*$ and $n_1, \ldots, n_t \in \mathbb{Z}$ decides whether*

$$\prod_{i=1}^{t} q_i^{n_i} = 1. \tag{2.1}$$

It is clear we can determine whether such a product has the correct sign: Simply take the sum of all n_i for which $q_i < 0$ and check whether the result is even. It is then sufficient to prove the following theorem instead.

Theorem 2.2. *There is a polynomial time algorithm that on input $t \in \mathbb{N}$, $a_1, \ldots, a_t, b_1, \ldots, b_t \in \mathbb{Z}_{>0}$ and $n_1, \ldots, n_t, m_1, \ldots, m_t \in \mathbb{Z}$ decides whether*

$$\prod_{i=1}^{t} a_i^{n_i} = \prod_{i=1}^{t} b_i^{m_i}. \tag{2.2}$$

In this form, the problem looks deceptively easy. Consider for example the most straightforward method to decide (2.2).

Method 2.3. Compute $\prod_{i=1}^{t} a_i^{n_i}$ and $\prod_{i=1}^{t} b_i^{m_i}$ explicitly and compare the results.

This method is certainly correct in that it is able to decide (2.2). However, it fails to run in polynomial time even when $t = 1$. For $n \in \mathbb{Z}_{>0}$ we have that $l(2^n) = n + 1 \approx 2^{l(n)}$. Hence the length of 2^n is not bounded by any polynomial in $l(n)$. We wouldn't even have enough time to write down the number regardless of our proficiency in multiplication because the number is too long.

Another method uses the fundamental theorem of arithmetic, also known as unique prime factorization in $\mathbb{Z}$.

Method 2.4. Factor $a_1, \ldots, a_t, b_1, \ldots, b_t$ into primes and for each prime that occurs compute the number of times it occurs in the products $\prod_{i=1}^{t} a_i^{n_i}$ and $\prod_{i=1}^{t} b_i^{m_i}$ and compare the results.

It is true that once we have factored all integers into primes only a polynomial number of steps remains. If we write x_{ip} for the exponent of the prime p in a_i, then we may compute $\sum_{i=1}^{t} n_i x_{ip}$, the exponent of p in $\prod_{i=1}^{t} a_i^{n_i}$, in polynomial time. Moreover, the number of prime factors of $n \in \mathbb{Z}_{>0}$ is at most $l(n)$, so the number of primes occurring is at most $\sum_{i=0}^{t}(l(a_i) + l(b_i))$, which is less than the length of the combined input. The problem lies in the fact that we have not specified how to factor integers into primes. As of January 2020, nobody has been able to show that we can factor integers in polynomial time. Until this great open problem is solved, Method 2.4 is out the window.

An interesting observation is that the main obstruction in Method 2.3 lies in the exponents being large, while for Method 2.4 the obstruction is in the bases. Our proof for Theorem 2.2 will be to slightly tweak Method 2.4. Namely, observe that we do not need to factor into prime elements but that it suffices to factor into pairwise coprime elements. The following lemma follows readily from unique prime factorization.

Lemma 2.5 (Unique coprime factorization)**.** *Let $s \in \mathbb{N}$ and let $c_1, \ldots, c_s \in \mathbb{Z}_{>1}$ be pairwise coprime. If for $n_1, \ldots, n_s, m_1, \ldots, m_s \in \mathbb{Z}_{\geq 0}$ we have*

$$\prod_{i=1}^{s} c_i^{n_i} = \prod_{i=1}^{s} c_i^{m_i}, \tag{2.3}$$

then $n_i = m_i$ for all i. □

We now propose the following algorithm for deciding (2.2).

Method 2.6. Factor $a_1, \ldots, a_t, b_1, \ldots, b_t$ into pairwise coprime $c_1, \ldots, c_s \in \mathbb{Z}_{>0}$. For each c_i compute the number of times it occurs in $\prod_{i=1}^t a_i^{n_i}$ and $\prod_{i=1}^t b_i^{m_i}$ and compare the results.

Now to prove Theorem 2.2 and in turn Theorem 2.1 it suffices to prove the following.

Theorem 2.7 (Coprime base factorization). *There is a polynomial time algorithm that on input $t \in \mathbb{N}$ and $a_1, \ldots, a_t \in \mathbb{Z}_{>0}$ computes $s \in \mathbb{N}$, $c_1, \ldots, c_s \in \mathbb{Z}_{>1}$ and $(n_{ij}) \in \mathbb{Z}_{\geq 0}^{t \times s}$ such that $c_1, \ldots, c_s$ are pairwise coprime and $a_i = \prod_{j=1}^s c_j^{n_{ij}}$ for all i.*

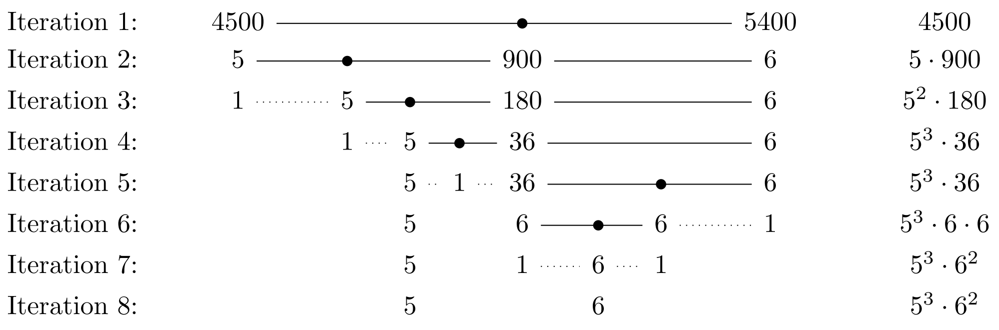
We state the algorithm first and prove the theorem later.

Method 2.8. Construct a complete simple graph G and label the vertices with $a_1, \ldots, a_s$. We call it a labeling because the map sending a vertex to its label need not be injective. While there are edges in G, repeat the following 5 steps:
1. Choose an edge $\{U, V\}$ of G and let u and v be the labels of U respectively V.
2. Compute $w = \gcd(u, v)$ using the Euclidean algorithm.
3. Add a vertex W labeled w to G and connect it to U, V and those vertices which are neighbours of both U and V.
4. Update the labels of U and V to u/w and v/w respectively.
5. For each $S \in \{U, V, W\}$, if the label of S is 1, then delete S and its incident edges from G.

Now $V = \{c_1, \ldots, c_s\}$ consists of the required pairwise coprime elements. The remaining output can now be computed in polynomial time.

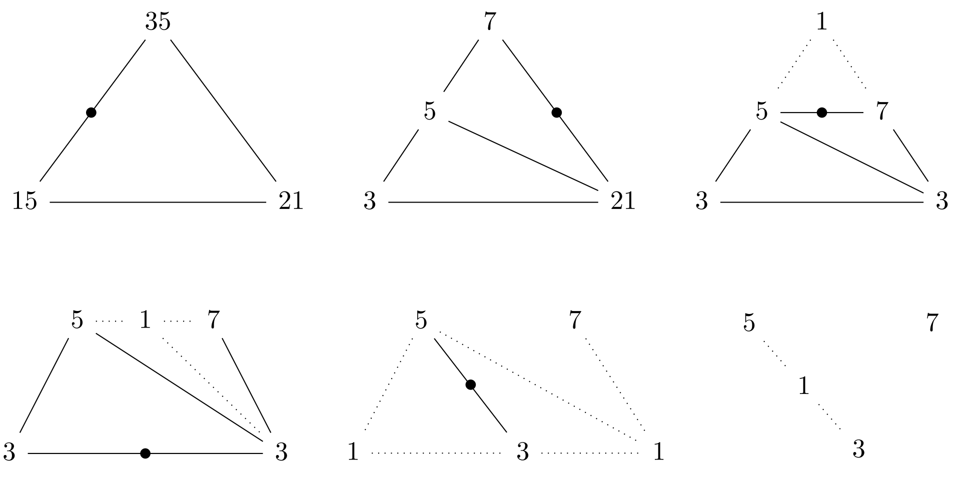
In the graph we construct and update the edges represent the pairs of numbers of which we do not yet know whether they are coprime, while a missing edge denotes that we know the pair to be coprime.

Example 2.9. We apply Method 2.8 to $(a_1, a_2) = (4500, 5400)$. Since there are only two vertices our graphs will fit on a single line. We denote the edge we choose in each iteration with a bullet and edges we have to erase are dotted. On the right we show how to keep track of the factorization of 4500 with minimal bookkeeping by writing it as a product of vertices in the graph.


Iteration 1: 4500 — 5400 — 4500
Iteration 2: 5 — 900 — 6 — $5 \cdot 900$
Iteration 3: 1 5 — 180 — 6 — $5^2 \cdot 180$
Iteration 4: 1 5 — 36 — 6 — $5^3 \cdot 36$
Iteration 5: 5 1 36 — 6 — $5^3 \cdot 36$
Iteration 6: 5 6 — 6 1 — $5^3 \cdot 6 \cdot 6$
Iteration 7: 5 1 6 1 — $5^3 \cdot 6^2$
Iteration 8: 5 6 — $5^3 \cdot 6^2$

We obtain $(c_1, c_2) = (5, 6)$ and $4500 = 5^3 \cdot 6^2$. By trial division we obtain $5400 = 5^2 \cdot 6^3$.

Example 2.10. We apply Method 2.8 to $(a_1, a_2, a_3) = (15, 21, 35)$.



The resulting coprime base is $(c_1, c_2, c_3) = (3, 5, 7)$. In the fifth graph something interesting happens. Vertex 7 suddenly becomes disconnected from the graph because we know it is coprime to one of the 3's.

Proof of Theorem 2.7. We claim Method 2.8 is correct and runs in polynomial time. One can show inductively that throughout the algorithm two vertices in the graph G are coprime when there is no edge between them. When the algorithm terminates because there are no edges in the graph, we may conclude that $c_1, \dots, c_s$ are coprime. Additionally, one shows inductively that the numbers $a_1, \dots, a_t$ can be written as some product of the vertices of G. Hence $c_1, \dots, c_s$ forms a coprime base for $a_1, \dots, a_t$, so Method 2.8 is correct. It remains to show that it is fast.

Write P_n for the product of all vertices in the graph at step n. Note that $P_0 = a_1 \cdots a_s$ and that $P_{n+1} \mid P_n$ for all $n \in \mathbb{N}$. Since P_0 has at most $B = l(P_0)$ prime factors counting multiplicities, there are at most B steps n for which $P_{n+1} < P_n$ and at most B vertices in G. The steps for which $P_n = P_{n+1}$ are those where the edge we chose is between coprime integers, meaning no vertices or edges are added to the graph and one edge is deleted. As the number of edges is at most B^2, then so is the number of consecutive steps for which $P_n = P_{n+1}$. Hence the total number of steps is at most B^3, which is polynomial in the length of the input. Lastly, note that each step takes only polynomial time because the values of the vertices are bounded from above by B and the Euclidean algorithm runs in polynomial time. Hence Method 2.8 runs in polynomial time. $\square$

The speed of this algorithm heavily depends on how fast the product of all vertices decreases. In each step we want to choose our edge $\{u, v\}$ such that $\gcd(u, v) \gg 1$. A heuristic for this could be to choose edges between large numbers. For those interested in the efficiency of coprime base factorization we refer to [1] for a provably faster algorithm.

Exercise 2.1. Show that there exists a polynomial-time algorithm that, given $a, b, c, d \in \mathbb{Z}_{>0}$ such that $ab = cd$, computes $w, x, y, z \in \mathbb{Z}_{>0}$ such that $(a, b, c, d) = (wx, yz, wz, xy)$.

Exercise 2.2 (Modified Euclidean algorithm)**.** For the sake of convention say $\gcd(0, 0) = 0$.

- **a.** Show that for all $a, b \in \mathbb{Z}$ with $b \neq 0$ there exist $r, q \in \mathbb{Z}$ with $a = qb + r$ and $|r| \leq |b|/2$.
- **b.** Show that for all $q, b, r \in \mathbb{Z}$ with $a = qb + r$ we have $\gcd(a, b) = \gcd(b, r)$ and $\gcd(a, 0) = |a|$.
- **c.** Prove that there exists a polynomial-time algorithm that, given $a, b \in \mathbb{Z}$, computes $\gcd(a, b)$ as well as $x, y \in \mathbb{Z}$ such that $ax + by = \gcd(a, b)$.
- **d.** Conclude that there exists a polynomial-time algorithm that, given $a, n \in \mathbb{Z}$ with $n > 1$, decides whether $a \in (\mathbb{Z}/n\mathbb{Z})^*$ and if so computes some $a' \in \mathbb{Z}$ such that $aa' \equiv 1 \bmod n$.
- **e.** Prove that there exists a polynomial-time algorithm that, given $a, b, m, n \in \mathbb{Z}$ with $n, m > 1$, decides whether there exists some $c \in \mathbb{Z}$ such that $c \equiv a \bmod m$ and $c \equiv b \bmod n$ and if so computes it.

Exercise 2.3. Show that there exists a polynomial-time algorithm that, given $k, a_1, \dots, a_k, n \in \mathbb{Z}$ satisfying $n \geq 1$ and $a_i^2 \equiv 1 \mod n$ for all i, decides whether there exists some non-empty subset $I \subseteq \{1, \dots, k\}$ such that $\prod_{i \in I} a_i \equiv 1 \mod n$ and if so computes one such I. *Hint:* Factor n.

Exercise 2.4. We equip $\mathbb{Q}^2 \backslash \{(0, 0)\}$ with an equivalence relation $\sim$ where $(x_1, y_1) \sim (x_2, y_2)$ if and only if there exists some $\lambda \in \mathbb{Q}^*$ such that $(\lambda x_1, \lambda y_1) = (x_2, y_2)$. Write $\mathbb{P}^1(\mathbb{Q}) = (\mathbb{Q}^2 \setminus \{(0, 0)\})/\sim$ for the *projective line* and write $(x : y)$ for the image of (x, y) in $\mathbb{P}^1(\mathbb{Q})$. Let $a, b \in \mathbb{Z}_{>0}$ and let $c_1, \dots, c_n$ be the coprime base for a and b produced by Method 2.8.

- **a.** For $p \mid ab$ prime write $f(p) = (\mathrm{ord}_p(a) : \mathrm{ord}_p(b)) \in \mathbb{P}^1(\mathbb{Q})$. Show that every c_i naturaly corresponds to a fibre of f and give the prime factorization of c_i.
- **b.** Suppose $n = 7$. Show that $ab \geq 1485890406000$ and equality holds for 8 pairs (a, b).
- **c.** (difficult) Give an assymtotic formula for the minimum of ab in terms of n.

Exercise 2.5. Let $n \in \mathbb{Z}_{>0}$. We encode matrices $\overline{M} = (\overline{m}_{ij})_{i,j}$ over $\mathbb{Z}/n\mathbb{Z}$ as a matrix $M = (m_{ij})_{i,j}$ over $\mathbb{Z}$ such that $0 \leq m_{ij} < n$ and $m_{ij} \equiv \overline{m}_{ij} \mod n$ for all i, j. Show that there exist polynomial-time algorithms for the following problems:

- **a.** given $n \in \mathbb{Z}_{\geq 0}$ and matrices M and N over $\mathbb{Z}/n\mathbb{Z}$, compute $M + N$ and $M \cdot N$;
- **b.** given $n, k \in \mathbb{Z}_{>0}$ and a matrix M over $\mathbb{Z}/n\mathbb{Z}$, compute M^k;
  Note: An algorithm that takes k steps is not polynomial-time!

c. given $n \in \mathbb{Z}_{>0}$ and a matrix M over $\mathbb{Z}/n\mathbb{Z}$, compute a row-echelon form of M;

d. given $n \in \mathbb{Z}_{>0}$ and a square matrix M over $\mathbb{Z}/n\mathbb{Z}$, compute $\det(M)$ and $\mathrm{Tr}(M)$;

e. given $n \in \mathbb{Z}_{>0}$ and a matrix M over $\mathbb{Z}/n\mathbb{Z}$, decide whether M^{-1} exists and if so compute it.

You may use the following fact: For every $n, B \in \mathbb{Z}_{>0}$ and matrix $M = (m_{ij})_{i,j} \in \mathbb{Z}^{n\times n}$ with $|m_{ij}| \leq B$ for all i, j it holds that $|\det(M)| \leq B^n \cdot n^{n/2}$ (see Hadamard's inequality, Exercise 3.5).

f. Show that there exists a polynomial-time algorithm that, given an integer matrix M, computes $\det(M)$ and $\mathrm{Tr}(M)$.

Exercise 2.6. Show that there exists a polynomial-time algorithm that, given $a, b, k, n \in \mathbb{Z}$ with $k, n > 0$ and $a^k \equiv 1 \bmod n$ and $b^k \equiv -1 \bmod n$, computes some $c \in \mathbb{Z}$ such that $a \equiv c^2 \bmod n$. *Hint:* First consider n odd and k a power of 2.

Exercise 2.7. Show that there exist polynomial-time algorithms for the following problems:

a. given $a, p, q \in \mathbb{Z}$ with p and q prime and $\gcd(a, p) = 1$, compute $u, e \in \mathbb{Z}$ such that $\gcd(u, q) = 1$ and such that the order of a in $(\mathbb{Z}/p\mathbb{Z})^*$ equals uq^e;

b. given $a, p \in \mathbb{Z}$ with p prime, decide whether a is a square modulo p;

c. given $a, b, p \in \mathbb{Z}$ with p prime, a a square modulo p and b not a square modulo p, compute $c \in \mathbb{Z}$ such that $c^2 \equiv a \mod p$;

d. given $a, b, p \in \mathbb{Z}$ with p prime, compute $c \in \mathbb{Z}$ such that c^2 equals a, b or ab modulo p.

2.1 Coprime base factorization in number fields

We would like to generalize Theorem 2.1 to arbitrary number fields, by which we mean that there is an additional input K, a number field, and that we take $q_1, \ldots, q_t \in K^*$. The theorem we will prove in the final sections is the following.

Theorem 2.11. *There exists a polynomial-time algorithm that, given a number field K, an $n \in \mathbb{Z}_{\geq 0}$ and $a_1, \ldots, a_n \in K^*$, computes the kernel of the map $\mathbb{Z}^n \to \langle a_1, \ldots, a_n\rangle$ given by $(k_1, \ldots, k_n) \mapsto \prod_i a_i^{k_i}$.*

In Section 3 we will actualy define what it means to compute the kernel of a linear map. When we try to prove this theorem by generalizing the theorems from the previous section, we run into some classic problems in (computational) number theory.

Theorem 2.2, to which we reduce, is a statement about integers. Hence we replace $\mathbb{Z}$ by an order R in K. One problem is that R^* will generaly contain more than just $\{\pm 1\}$, and it is not obvious how to pick a set $R_{>0} \subseteq R \setminus \{0\}$ of representatives of $(R \setminus \{0\})/R^*$ like the positive integers for $\mathbb{Z}$. Another problem is that we would like to at least compute the set R^*, which is a finitely generated abelian group by Dirichlet's unit theorem, but it is not known how to do this in polynomial time. Even if we disregard run-time issues, we want a Lemma 2.5 for orders. However, generally R will not be a UFD like $\mathbb{Z}$, making Theorem 2.7 hard to generalize.

The 'correct' way to generalize the theory is to translate it into a theorem about ideals, since the maximal order $\mathcal{O}_K$ of K has unique ideal factorization. Moreover, as opposed to the elements of $\mathcal{O}_K \setminus \{0\}$ themselves, the ideals are invariant under multiplication by units. However, computing $\mathcal{O}_K$ is also difficult. Luckily this is something we can work around. First we generalize Lemma 2.5.

Lemma 2.12 (Unique coprime factorization for ideals)**.** *Let $s \in \mathbb{N}$, let R be an order and let $\mathfrak{c}_1, \ldots, \mathfrak{c}_s \subsetneq R$ be pairwise coprime invertible integral ideals. If for $n_1, \ldots, n_s, m_1, \ldots, m_s \in \mathbb{Z}_{\geq 0}$ we have*

$$\prod_{i=1}^{s} \mathfrak{c}_i^{n_i} = \prod_{i=1}^{s} \mathfrak{c}_i^{m_i}, \tag{2.4}$$

then $n_i = m_i$ for all i.

Proof. Since the ideals are invertible we may divide out $\mathfrak{c}_i^{\min\{n_i, m_i\}}$ and thus assume without loss of generality that $n_i = 0$ or $m_i = 0$ for all i. But then the product on the left hand side of (2.4) and the product on the right hand side of (2.4) are coprime, so the products equal R. By the Chinese remainder theorem for ideals we get $0 = R/(\prod_{i=1}^{s} \mathfrak{c}_i^{n_i}) = \prod_{i=1}^{s}(R/\mathfrak{c}_i^{n_i})$, so $\mathfrak{c}_i^{n_i} = R$ for all i. If $n_i > 0$ we have $\mathfrak{c}_i \supseteq \mathfrak{c}_i^{n_i} = R$, so $\mathfrak{c}_i = R$, a contradiction. Thus $n_i = m_i = 0$ for all i. □

We would now like to prove the following theorem.

Theorem 2.13. *There exists a polynomial-time algorithm that, given an order R, an $n \in \mathbb{Z}_{\geq 0}$ and non-zero ideals $\mathfrak{a}_1, \dots, \mathfrak{a}_n \subseteq R$, computes either an order $S \supsetneq R$ or a coprime base $\mathfrak{c}_1, \dots, \mathfrak{c}_m \subsetneq R$ of invertible ideals for $\mathfrak{a}_1, \dots, \mathfrak{a}_n$.*

To prove this theorem we we are required to do some more work. First of all we require some definitions on how to encode orders and ideals. Then, we should construct algorithms to do arithmetic on ideals in polynomial time. More generally, we will study algorithms for finitely generated abelian groups in the next section.

Exercise 2.8. Suppose we can compute a basis of the image and kernel of a morphism $\mathbb{Z}^n \to \mathbb{Z}^m$ encoded by an integer matrix in polynomial time (we can, but this is non-trivial). Show that for an order R and fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ of R, encoded as a $\mathbb{Z}$-basis, we may compute $\mathfrak{a} + \mathfrak{b}$, $\mathfrak{a} \cdot \mathfrak{b}$ and $\mathfrak{a} : \mathfrak{b}$ and decide whether $\mathfrak{a} = \mathfrak{b}$ in polynomial time.

Exercise 2.9. Suppose that we may compute for ideals $\mathfrak{a}$ and $\mathfrak{b}$ the ideals $\mathfrak{a} + \mathfrak{b}$, $\mathfrak{a} \cdot \mathfrak{b}$ and $\mathfrak{a} : \mathfrak{b}$ and decide whether $\mathfrak{a} = \mathfrak{b}$. Show that there exists an algorithm that, given an order R, $n \in \mathbb{Z}_{\geq 0}$ and non-zero ideals $\mathfrak{a}_1, \dots, \mathfrak{a}_n \subseteq R$, computes either an order $S \supsetneq R$ or a coprime base $\mathfrak{c}_1, \dots, \mathfrak{c}_m \subsetneq R$ of invertible ideals for $\mathfrak{a}_1, \dots, \mathfrak{a}_n$. Show that your algorithm runs in polynomial time when the input is restricted to ideals of the form aR with $a \in \mathbb{Z}_{>0}$. *Hint:* You can assume every ideal you encounter is invertible.

3 Finitely generated abelian groups

In this section we treat algorithms on finitely generated abelian groups. Many important objects in algebraic number theory are finitely generated abelian groups. For example, the additive group of orders R in number fields, as well as finitely generated modules over R, notably its ideals I and quotients R/I. In this section we will use additive notation for our abelian groups and we will use [2] as our reference. Other finitely generated abelian groups of interest are unit groups of finite commutative rings like $\mathbb{Z}/n\mathbb{Z}$ or $\mathbb{F}_q$, or an elliptic curve. However, as we will soon see, there is an obstruction in working these groups.

We begin by specifying a representation for our finitely generated groups. Recall that every finitely generated abelian group A fits in some exact sequence

$$\mathbb{Z}^m \xrightarrow{\alpha} \mathbb{Z}^n \xrightarrow{f} A \longrightarrow 0.$$

Namely, we obtain n and f by writing down some generators $a_1, \ldots, a_n \in A$ for A and let f map the i-th standard basis vector to a_i. For m and α we repeat the procedure with A replaced by $\ker(f)$. Note that α, being a morphism between free $\mathbb{Z}$-modules, has a natural representation as a matrix with integer coefficients. By the isomorphism theorem $A \cong \mathbb{Z}^n/\ker(f) = \mathbb{Z}^n/\operatorname{im}(\alpha) = \operatorname{coker}(\alpha)$, so A is completely defined by α. Thus we choose to encode A as the matrix corresponding to α. A morphism $f : A \to B$ of finitely generated abelian groups in terms of this representation gives a commutative diagram of exact sequences

$$\begin{tikzcd}
\mathbb{Z}^k \arrow[r, "\alpha"] & \mathbb{Z}^l \arrow[r] \arrow[d, dashed, "\varphi"] & A \arrow[r] \arrow[d, "f"] & 0 \\
\mathbb{Z}^m \arrow[r, "\beta"] & \mathbb{Z}^n \arrow[r] & B \arrow[r] & 0.
\end{tikzcd} \tag{3.1}$$

Here φ is any morphism that makes the diagram commute. We encode f by the matrix representing φ. Important to note is that not every φ defines a morphism $f : A \to B$. It defines a morphism precisely when $\operatorname{im}(\varphi \circ \alpha) \subseteq \operatorname{im}(\beta)$, however it is not immediately obvious how to test this. Computing the composition of morphisms and evaluating morphisms in this form is straightforward, as it is just matrix multiplication.

To work with abelian groups in our algorithms it takes more than just to specify an encoding. The following is a list in no particular order of operations we would like to be able to perform in polynomial time.

1. decide whether a matrix encodes a morphism of given groups;
2. compute kernels, images and cokernels of group homomorphisms;
3. test if a group homomorphism is injective/surjective and if bijective compute an inverse;
4. decide if two group homomorphisms are equal;
5. compute an element in the preimage of a given group element under a group homomorphism;
6. compute direct sums, tensor products and homomorphism groups of pairs of groups;
7. compute the order of a given group element;
8. compute the order/exponent of a finite group;
9. split exact sequences;
10. compute the torsion subgroup of a group;
11. write a group as a direct sum of cyclic groups.

We will spend this section working up to the last element of this list: An algorithmic version of the fundamental theorem of finitely generated abelian groups.

Finally, we addres an important subtlety that arises from our choice of encoding.

Lemma 3.1. *Assuming the above problems have polynomial time algorithms, we may solve the* discrete logarithm problem *in polynomial time. That is, given an abelian group A and elements $a, b \in A$, decide whether there exists some positive integer n such that $na = b$ and if so compute such n.*

It is well known that the discrete logarithm problem for $\mathbb{F}_q^*$ or elliptic curves over finite fields is difficult, i.e. not known to be solvable in polynomial time, even though both are finitely generated abelian groups. The difficulty is representing $\mathbb{F}_q^*$ and its elements in our encoding. For starters, we need to write down generators for $\mathbb{F}_q^*$ and subsequently write the input to our algorithms in terms of these generators. Doing so is almost equivalent to the discrete logarithm problem.

Exercise 3.1. Prove Lemma 3.1.

Exercise 3.2. Show that there exists a polynomial-time algorithm that, given finitely generated abelian groups A and B, computes the group $A \times B$ and the corresponding inclusions and projections.

3.1 Lattices and short bases

To understand general finitely generated abelian groups, we first need to understand the simplest instances, the free abelian groups. It will turn out to be fruitful to consider free abelian groups together with an inner product. This will allow us later to compute images and kernels of linear maps.

Definition 3.2. A *Euclidean (vector) space* is a finite-dimensional real inner product space. For an element x in an inner product space we will write $q(x) = \langle x, x\rangle$. A *lattice* is a discrete subgroup of a Euclidean space.

A Euclidean space we naturally encounter for any number field K is $K \otimes_{\mathbb{Z}} \mathbb{R}$, which we equip with the inner product

$$\langle x, y\rangle = \frac{1}{[K:\mathbb{Q}]} \sum_{\sigma : K\otimes_{\mathbb{Z}}\mathbb{R} \to \mathbb{C}} \sigma(x) \cdot \overline{\sigma(y)}$$

where the sum ranges over all $\mathbb{R}$-algebra homomorphisms. In this Euclidean space every order of K is a lattice.

Proposition 3.3. *A lattice Λ in a Euclidean space V is a free $\mathbb{Z}$-module with* $\operatorname{rk}\Lambda \leq \dim V$ *and the restriction of the inner product to Λ is $\mathbb{Z}$-bilinear, real-valued, symmetric and satisfies* $\inf\{\langle x, x\rangle \mid x \in \Lambda \setminus \{0\}\} > 0$. *Conversely, every free $\mathbb{Z}$-module Λ of finite rank equipped with a $\mathbb{Z}$-bilinear, real-valued, symmetric form φ for which* $\inf\{\varphi(x, x) \mid x \in \Lambda\setminus\{0\}\} > 0$ *can be embedded in a Euclidean vector space such that the inner product restricted to Λ equals φ.* □

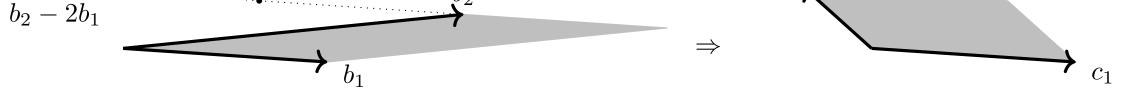
This proposition shows that we have an equivalent definition of a lattice that does not require an ambient vector space. The bilinear form φ in the proposition is again naturally given by a matrix $F = (\varphi(b_i, b_j))_{1\leq i,j\leq n}$ where $(b_1, \ldots, b_n)$ is the basis encoding Λ. For our computational purposes it is practical to restrict to matrices with rational entries. This will be our encoding for lattices.

An algorithmic problem we will encounter is computing a 'short basis' for a lattice $\Lambda \subseteq \mathbb{Z}^n$.

Definition 3.4. Let Λ be a lattice and let $(b_1, \ldots, b_n)$ be a basis of Λ. Consider the matrix $B = (\langle b_i, b_j\rangle)_{1\leq i,j\leq n}$. We define the *determinant* of Λ to be $\det(\Lambda) = |\det(B)|^{1/2}$.

Exercise 3.3. Show that the determinant of a lattice does not depend on the choice of basis.

The determinant $\det(\Lambda)$ also equals the volume of the parallelepiped span by a basis of Λ. Since the determinant is an invariant, finding a 'shorter' basis is equivalent to finding a 'more orthogonal' basis.



For a Euclidean space the Gram–Schmidt algorithm transforms a basis into an orthogonal one as follows.

Definition 3.5. Let V be a Euclidean space with basis $B = (b_1, \ldots, b_n)$. We iteratively define

$$\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \quad \text{for } 1 \leq j < i \text{ and } \quad b_i^* = b_i - \sum_{j<i} \mu_{ij} b_j^* \quad \text{for } 1 \leq i \leq n.$$

We call $B^* = (b_1^*, \ldots, b_n^*)$ and $(\mu_{ij})_{j<i}$ the *Gram–Schmidt basis* respectively *Gram–Schmidt coefficients* corresponding to B.

When interpreting $M = (\mu_{ij})_{j<i}$ as an upper-triangular matrix, we note that that $(\mathrm{id} + M)B^* = B$. In particular $\det(B) = \det(B^*)$. Since $b_1^*, \ldots, b_n^*$ are indeed pairwise orthogonal, they form an orthogonal basis of V. Sadly the Gramm–Schmidt coefficients will generally not be integers, meaning that if B is a basis for a lattice Λ, then generally B^* will not be. This is quite unsurprising, as not every lattice even has an orthogonal basis. A possible solution is to round the Gram-Schmidt coefficients to integers in every step, so that we are guaranteed to obtain a basis for Λ. However, this does not yield the necessary bounds on our basis.

Exercise 3.4. We say a basis $B = (b_1, \ldots, b_n)$ is *Gram–Schmidt reduced* if $|\mu_{ij}| \leq \frac{1}{2}$ holds for all Gram–Schmidt coefficients μ_{ij} of B. Show that the following algorithm is guaranteed to terminate, and hence computes a Gram–Schmidt reduced basis: Let $(b_1, \ldots, b_n)$ be a basis.

1. Compute the Gram–Schmidt coefficients $(\mu_{ij})_{j<i}$ of $(b_1, \ldots, b_n)$.
2. If $|\mu_{ij}| \leq \frac{1}{2}$ for all i, j, then return $(b_1, \ldots, b_n)$ and terminate.
3. Choose any i, j such that $|\mu_{ij}| > \frac{1}{2}$ and replace b_i by $b_i - \lceil \mu_{ij} \rfloor b_j$.
4. Go to step 1.

The algorithm of Exercise 3.4 will not run in polynomial time. In the next section we will state the existence of a better algorithm.

Exercise 3.5. Let $(b_1, \ldots, b_n)$ be a basis of a lattice Λ in a Euclidean space V. Write $\Lambda_k = \sum_{j \leq k} \mathbb{Z} b_i$ for all $0 \leq k \leq n$.

a. Show that

$$q(b_i) \geq q(b_i^*) = \Big(\frac{\det(\Lambda_i)}{\det(\Lambda_{i-1})} \Big)^2.$$

b. Conclude *Hadamard's inequality*: For B the matrix with columns $b_1, \ldots, b_n$ we have

$$|\det(B)| \leq \prod_{i=1}^n \|b_i\|$$

with equality if and only if the b_i are pairwise orthogonal.

c. Show that

$$\det(\Lambda)^2 \leq \prod_{i=1}^n q(b_i) \leq 2^{\binom{n}{2}} \det(\Lambda)^2.$$

Exercise 3.6. Let V be a Euclidean space. For a subspace $W \subseteq V$ we write $W^\perp = \{v \in V \mid \langle v, W \rangle = 0\}$.

a. Show that for all $W \subseteq V$ the natural map $W^\perp \to V/W$ is an isomorphism of vector spaces. We equip V/W with the natural Euclidean vector space structure induced by $W^\perp$. Suppose $\Lambda \subseteq V$ is a lattice with a sublattice Λ' such that Λ/Λ' is a torsion free group.

b. Show that the natural map $\Lambda/\Lambda' \to V/\mathbb{R}\Lambda'$ is injective and that its image is a lattice.

c. Show that $\det(\Lambda) = \det(\Lambda/\Lambda') \cdot \det(\Lambda')$.

Exercise 3.7. Let Λ be a lattice and define the $\Lambda^\dagger$ to be the group $\mathrm{Hom}(\Lambda, \mathbb{Z})$ together with the map $\langle \cdot, \cdot \rangle : \Lambda^\dagger \times \Lambda^\dagger \to \mathbb{R}$ given by

$$\langle f, g \rangle = \sup_{x \in \Lambda \setminus \{0\}} \frac{f(x)g(x)}{\langle x, x \rangle}.$$

a. Show that $\Lambda^\dagger$ is a lattice. We will call $\Lambda^\dagger$ the *dual lattice* of Λ.

b. Suppose $\Lambda \subseteq \mathbb{R}^n$ is full rank and let $\Lambda' = \{x \in \mathbb{R}^n \mid \langle x, \Lambda \rangle \subseteq \mathbb{Z}\}$. Show that Λ' is a lattice.

c. Show that $\Lambda^\dagger \cong \Lambda'$ and $(\Lambda^\dagger)^\dagger \cong \Lambda$.
d. Show that $\det(\Lambda^\dagger) = \det(\Lambda)^{-1}$.
e. For a homomorpism $\varphi : \Lambda_1 \to \Lambda_2$ of lattices write $\varphi^\dagger : \Lambda_2^\dagger \to \Lambda_1^\dagger$ for the map $f \mapsto f \circ \varphi$. Show that

$$\det(\ker(f)) \cdot \det(\operatorname{im}(f)) \cdot \det(\Lambda_1^\dagger) = \det(\ker(f^\dagger)) \cdot \det(\operatorname{im}(f^\dagger)) \cdot \det(\Lambda_2).$$

3.2 The LLL-algorithm

In this section we state the existence the LLL-algorithm, which produces a 'small' basis for a given lattice. We will not prove the correctness of the algorithm, nor will we actually describe the algorithm. What we will do is define what we mean by 'small' bases the context of the LLL-algorithm and derive some of their properties. In this section we will use [5] as our reference. A reference on the LLL-algorithm we do not draw upon which may be of interest to a reader is [6].

Definition 3.6. Let Λ be a lattice with basis $B = (b_1, \ldots, b_n)$ and let $(b_1^*, \ldots, b_n^*)$ and $(\mu_{ij})_{j<i}$ be its corresponding Gram–Schmidt basis and coefficients as defined in Definition 3.5. Let $\frac{4}{3} \geq c$. We say B is c-reduced if
(1) For all $1 \leq j < i \leq n$ we have $|\mu_{ij}| \leq \frac{1}{2}$.
(2) For all $1 \leq k < n$ we have $cq(b_{k+1}^*) \geq q(b_k^*)$.

Note that the first condition states that B is Gram–Schmidt reduced in the sense of Exercise 3.4. We may compute a c-reduced basis, and in particular it always exists, by Exercise 3.10. That we may in fact compute it in polynomial time is non-trivial.

Theorem 3.7 (LLL-algorithm). *Let $c > \frac{4}{3}$. There exists a polynomial-time algorithm that, given a lattice Λ, produces a c-reduced basis of Λ.* □

Although the algorithm does not fundamentally differ when we modify c, it cannot be part of the input because then the algorithm would no longer run in polynomial time. We should warn the reader that the literature contains various definitions of a 'reduced basis', and even in the context of the LLL-algorithm there are at least two.

Definition 3.8. Let Λ be a lattice of rank n. For $0 < i \leq n$ we define the *i-th successive minimum* to be the value

$$\lambda_i(\Lambda) = \min\{r \in \mathbb{R}_{\geq 0} \mid \operatorname{rk}\langle x \in \Lambda \mid q(x) \leq r \rangle \geq i\}.$$

A basis of vectors attaining the successive minima is the gold standard of 'small' bases, although it does not always exist. The following proposition gives bounds on how far away a reduced basis can be from these minima.

Proposition 3.9. *Let $c \geq \frac{4}{3}$ and suppose $(b_1, \ldots, b_n)$ is a c-reduced basis for a lattice Λ. Then for all $0 < i \leq n$ we have $c^{1-n} \cdot q(b_i) \leq \lambda_i(\Lambda) \leq c^{i-1} \cdot q(b_i)$.*

Proof. See Exercise 3.9. □

Exercise 3.8. Show that there exists a lattice Λ for which no basis $b_1, \ldots, b_n$ attains the successive minima, i.e. satisfies $q(b_i) = \lambda_i(\Lambda)$ for all i. *Hint:* Consider $2\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$.

Exercise 3.9. Let $c \geq \frac{4}{3}$ and $0 < i \leq n$ and suppose $(b_1, \ldots, b_n)$ is a c-reduced basis of Λ.
a. Show that $q(b_j^*) \leq c^{i-j} q(b_i^*)$ for all $j \leq i$.
b. Recall that $b_i = b_i^* + \sum_{j<i} \mu_{ij} b_j^*$. Show that $q(b_i) \leq c^{i-1} q(b_i^*)$.
c. Show that $q(b_j) \leq c^{i-1} q(b_i^*) \leq c^{i-1} q(b_i)$ for all $j \leq k$.
d. Conclude that $\lambda_i(\Lambda) \leq \max\{q(b_j) \mid j \leq i\} \leq c^{i-1} q(b_i)$.
Write $\Lambda_k = \sum_{j \leq k} \mathbb{Z} b_j$ for all $0 \leq k \leq n$.
e. Prove that for all $0 < k \leq n$ and $x \in \Lambda_k \setminus \Lambda_{k-1}$ we have $q(x) \geq q(b_k^*)$.
Write $S = \{x \in \Lambda \mid q(x) \leq \lambda_i(\Lambda)\}$ and let k be minimal such that $S \subseteq \Lambda_k$.
f. Show that $k \geq \operatorname{rk}\langle S \rangle \geq i$.
g. Conclude that $\lambda_i(\Lambda) \geq q(b_k^*) \geq c^{1-n} q(b_i)$.

Exercise 3.10. Let $c \geq \frac{4}{3}$. Show that the following algorithm is guaranteed to terminate, and hence computes a c-reduced basis: Let $(b_1, \ldots, b_n)$ be a basis.

1. Compute the Gram–Schmidt basis $(b_1^*, \ldots, b_n^*)$ and coefficients $(\mu_{ij})_{j<i}$ of $(b_1, \ldots, b_n)$.
2. If there exists some $1 \leq k < n$ such that $cq(b_{k+1}^*) < q(b_k^*)$, choose any such k, swap b_{k+1} and b_k and go to step 1.
3. If there exist some $1 \leq j < i \leq n$ such that $|\mu_{ij}| > \frac{1}{2}$, choose any such i, j, replace b_i by $b_i - \lceil \mu_{ij} \rfloor b_j$ and go to step 1.
4. Return $(b_1, \ldots, b_n)$ and terminate.

Hint: Let $\Lambda_k = \sum_{i=1}^{k} \mathbb{Z} b_i$. What can you say about $\prod_{k=1}^{n} \det(\Lambda_k)$ in step 2?

3.3 The kernel-image algorithm

The LLL-algorithm allows us to prove the kernel-image algorithm, from which most of the algorithms for finitely generated abelian groups from the beginning of this section follow without much effort. We first need a theorems from linear algebra.

Theorem 3.10 (Cramer's rule). *Suppose $A \in \mathbb{R}^{n \times n}$ is an invertible matrix and let $b \in \mathbb{R}^n$. Write A_i for the matrix obtained from A by replacing the i-th column with b. Then there exists a unique $x \in \mathbb{R}^n$ such that $Ax = b$, and it is given by $x = \det(A)^{-1} \cdot (\det(A_1), \ldots, \det(A_n))$.* □

Exercise 3.11. Let $n \geq 0$. Suppose $N \subseteq M \subseteq \mathbb{Z}^n$ are subgroups such that $N \oplus P = \mathbb{Z}^n$ for some $P \subseteq \mathbb{Z}^n$. Show that $\mathrm{rk}(M) = \mathrm{rk}(N)$ if and only if $M = N$.

Theorem 3.11 (Kernel-image algorithm). *There exists a polynomial-time algorithm that, given a linear map $\varphi : \mathbb{Z}^n \to \mathbb{Z}^m$, computes the rank r of φ and injective linear maps $\iota : \mathbb{Z}^r \to \mathbb{Z}^n$ and $\kappa : \mathbb{Z}^{n-r} \to \mathbb{Z}^n$ such that* $\mathrm{im}(\varphi \circ \iota) = \mathrm{im}(\varphi)$ *and* $\mathrm{im}(\kappa) = \ker(\varphi)$.

Proof. Write B for the largest absolute value of a coefficient of the matrix defining φ. Compute

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

and note that its length is polynomially bounded by the size of the input. Hence we can consider the lattice $L = \mathbb{Z}^n$ together with the bilinear form given by $q(x) = \|x\|^2 + \omega \|\varphi(x)\|^2$. Using the LLL-algorithm, compute a 2-reduced basis $(b_1, \ldots, b_n)$ of L. We will show that this basis has the following properties:

(a) $q(b_i) < \omega$ for $0 < i \leq n - r$;
(b) $(b_1, \ldots, b_{n-r})$ is a basis for $\ker(\varphi)$;
(c) $q(b_i) \geq \omega$ for $n - r < i \leq n$;
(d) $(\varphi(b_{n-r+1}), \ldots, \varphi(b_n))$ is a basis for $\mathrm{im}(\varphi)$.

Once we have shown this, it is clear how to compute r and the maps ι and κ in polynomial time.

Claim: For all $0 < i < n - r$ we have $\lambda_i(\Lambda) \leq n^{n+1} B^{2n}$.

Proof. By Cramer's rule, we may find linearly independent vectors $a_1, \ldots, a_{n-r} \in \ker(\varphi)$ for which the coefficients are determinants of $r \times r$ submatrices of F. Then by Hadamard's inequality (Exercise 3.5), each coefficient is bounded in absolute value by $r^{r/2} B^r \leq n^{n/2} B^n$. Hence $q(a_i) = \|a_i\|^2 \leq n^{n+1} B^{2n}$ for all $0 < i \leq n - r$. The claim now follows from the independence of the a_i. ■

From Proposition 3.9 and the claim it follows that

$$q(b_i) \leq 2^{n-1} \cdot \lambda_i(\Lambda) \leq 2^{n-1} \cdot n^{n+1} \cdot B^{2n} < \omega$$

for all $0 < i \leq n - r$, proving (a). Clearly for all $x \in \Lambda$ such that $\omega > q(x) = \|x\|^2 + \omega \|\varphi(x)\|^2$ we have $\|\varphi(x)\|^2 = 0$ and thus $x \in \ker(\varphi)$. In particular, we have linearly independent $b_1, \ldots, b_{n-r} \in \ker(\varphi)$. From Exercise 3.11 we may conclude it is in fact a basis for $\ker(\varphi)$, proving (b). It follows from (b) that $b_i \notin \ker(\varphi)$ and thus $q(b_i) \geq \omega$ for all $n - r < i \leq n$, proving (c). Lastly, (d) follows from (b) and the homomorphism theorem. □

Note that in the proof of Theorem 3.11 we could have constructed a c-reduced basis for values of c other than 2. Moreover, the exact value of ω is not important, as long as it is sufficiently large. Exercise 3.14 will prove a version of the kernel image algorithm for general finitely generated abelian groups.

3.4 Applications of the kernel-image algorithm

In this subsection we provide a polynomial-time algorithm for most problems in the beginning of this section. An immediate consequence of the kernel-image algorithm is the following.

Corollary 3.12. *There exists a polynomial-time algorithm that, given a linear map $\varphi : A \to B$ of finitely generated free abelian groups, decides whether φ is injective/surjective.* □

Exercise 3.12. Show that for a matrix $\varphi : \mathbb{Z}^n \to \mathbb{Z}^n$ we have $\#\operatorname{coker}(\varphi) = |\det(\varphi)|$. Conclude that there exist a polynomial time algorithm that, given an abelian group A, decides whether A is finite and if so computes $\#A$. *Note:* The matrix representing A need not be injective.

Proposition 3.13. *There exists a polynomial-time algorithm that, given linear maps $\varphi : A \to C$ and $\psi : B \to C$ of free abelian groups, decides whether* $\operatorname{im}(\psi) \subseteq \operatorname{im}(\varphi)$.

Proof. Using Theorem 3.11 compute the kernel $\kappa : K \to A \times B$ of $A \times B \to C$ as in the following diagram.

$$\begin{tikzcd}
& & A \arrow[dr, "\varphi"] & \\
K \arrow[r, "\kappa"] \arrow[drr, dashed, bend right] & A \times B \arrow[ur, "\pi_A"] \arrow[rr] \arrow[dr, "\pi_B"'] & & C \\
& & B \arrow[ur, "\psi"'] &
\end{tikzcd}$$

The image of $\pi_B \circ \kappa$ is precisely the set of elements $b \in B$ for which there exists an $a \in A$ such that $\varphi(a) = \psi(b)$. Hence it suffices to decide using Corollary 3.12 whether $\pi_B \circ \kappa$ is surjective. □

Corollary 3.14. *There exists a polynomial time algorithm that, given finitely generated abelian groups A and B, represented by linear maps $\alpha : A_0 \to A_1$ respectively $\beta : B_0 \to B_1$, and a linear map $\varphi : A_1 \to B_1$, decides whether φ represents a morphism $f : A \to B$.*

Proof. Recall φ represents a morphism precisely when $\operatorname{im}(\varphi \circ \alpha) \subseteq \operatorname{im}(\beta)$. □

Corollary 3.15. *There exists a polynomial-time algorithm that, given finitely generated abelian groups A and B and morphisms $f, g : A \to B$, decides whether $f = g$.*

Proof. Considering $h = f - g$ it suffices to be able to decide whether a morphism is zero. With η the matrix representing h and β the matrix representing B, we have $h = 0$ precisely when $\operatorname{im}(\eta) \subseteq \operatorname{im}(\beta)$. □

Proposition 3.16. *There exists a polynomial-time algorithm that, given a morphism $f : A \to B$ of finitely generated abelian groups, decides whether f is injective/surjective.*

Proof. Write $\alpha : A_0 \to A_1$ and $\beta : B_0 \to B_1$ for the representatives of A respectively B and φ for the representative of f. Note that f is surjective if and only if $B = \operatorname{im}(f) = \operatorname{im}(\varphi)/\operatorname{im}(\beta)$ if and only if $B_1 = \operatorname{im}(\varphi) + \operatorname{im}(\beta)$. It suffices to decide whether the map $A_1 \times B_0 \to B_1$ induced by φ and β is surjective, for which we have Corollary 3.12. Note that f is injective if and only if $\ker(\varphi) \subseteq \operatorname{im}(\alpha)$. Using Theorem 3.11 we compute a linear map $\kappa : K \to A_1$ with $\operatorname{im}(\kappa) = \ker(\varphi)$, and apply Proposition 3.13 to decide $\operatorname{im}(\kappa) \subseteq \operatorname{im}(\alpha)$. □

Proposition 3.17. *There exists a polynomial-time algorithm that, given a linear map $\varphi : A \to B$ of free abelian groups A and B and $b \in B$, decides whether an element $a \in A$ exists such that $\varphi(a) = b$, and if so computes one.*

Proof. Consider the linear map $\psi : A \times \mathbb{Z} \to B$ that sends (a, x) to $\varphi(a) + xb$.

$$\begin{tikzcd}
I \arrow[dr, "\iota"] & & & A \arrow[dr, "\varphi"] & \\
& K \arrow[r, "\kappa"] \arrow[drr, dashed, bend right, "\chi"'] & A \times \mathbb{Z} \arrow[ur] \arrow[rr, "\psi"] \arrow[dr, "\pi_{\mathbb{Z}}"'] & & B \\
& & & \mathbb{Z} \arrow[ur, "b"'] &
\end{tikzcd}$$

Compute using Theorem 3.11 the kernel $\kappa : K \to A \times \mathbb{Z}$ of ψ and in turn $\iota : I \to K$ a preimage of $\chi = \pi_{\mathbb{Z}} \circ \kappa$. Note that $\varphi(a) = b$ has a solution precisely when -1 is in the image of χ. Moreover, if we find $k \in K$ such that $\chi(k) = -1$, then its image under $K \to A \times \mathbb{Z} \to A$ gives an element $a \in A$ such that $\varphi(a) = b$. Note that $I \subseteq \mathbb{Z}$. If $I = 0$ no solution to $\chi(k) = -1$ exists, and otherwise $k \in \iota(\{\pm 1\})$ gives a solution if it exists. □

Corollary 3.18. *There exists a polynomial-time algorithm that, given a homomorphism $f : A \to B$ of finitely generated abelian groups A and B and $b \in B$, decides whether an element $a \in A$ exists such that $f(a) = b$, and if so computes one.*

Proof. Let $\varphi : A_1 \to B_1$ be the representative of f. Simply apply Proposition 3.17 to φ and some representative of b in B_1. □

Exercise 3.13. Give a direct proof of Proposition 3.13 or Proposition 3.17 by giving an algorithm that applies the LLL-algorithm only once, similar to Theorem 3.11.

Exercise 3.14. Show that there exists a polynomial-time algorithm that, given a morphism $f : A \to B$ of finitely generated abelian groups, computes morphisms $k : K \to A$ and $i : I \to A$ such that $\mathrm{im}(k) = \ker(f)$, $\mathrm{im}(f \circ i) = \mathrm{im}(f)$ and k and $f \circ i$ are injective.

3.5 Homomorphism groups of finitely generated abelian groups

Although we can algorithmically work with individual morphisms $f : A \to B$ of finitely generated abelian groups, we have yet to treat the group of homomorphisms $\mathrm{Hom}(A, B)$ as a whole. Certainly, we would like to compute $\mathrm{Hom}(A, B)$, but we have to decide what that means. Firstly, we have to give an abelian group H represented by $H_0 \to H_1$ such that $H \cong \mathrm{Hom}(A, B)$. Secondly, we want to evaluate elements of $\mathrm{Hom}(A, B)$ at elements of A. For this we give two possibilities:

- Note that morphisms are already encoded as matrices in $\mathbb{Z}^{n\times m}$, so we simply give a map $H_1 \to \mathbb{Z}^{n\times m}$ that maps H_1 to matrices representing morphisms $A \to B$ and H_0 to the zero morphisms.
- We give a bilinear map $A \times H \to B$, i.e. a linear map $A \otimes H \to B$, which corresponds to the evaluation map $A \times \mathrm{Hom}(A, B) \to B$ under the isomorphism $H \cong \mathrm{Hom}(A, B)$.

Regardless of which representation we choose, when we talk about computing $\mathrm{Hom}(A, B)$ we mean computing both a group isomorphic to $\mathrm{Hom}(A, B)$ as well as a way to evaluate its elements in A.

Exercise 3.15. Show that the above representations for $\mathrm{Hom}(A, B)$ are 'polynomially equivalent', i.e. there exist polynomial-time algorithm that transforms one representation of $\mathrm{Hom}(A, B)$ into the other.

The moral of Exercise 3.15 is that as long as it is easy to describe how elements from H correspond to homomorphisms, it does not matter how we encode this.

Theorem 3.19. *There exists a polynomial-time algorithm that, given finitely generated abelian groups A and B, computes* $\mathrm{Hom}(A, B)$.

Proof. Consider the case where A is a free abelian group and B is a finitely generated abelian group represented by $\beta : B_0 \to B_1$. We may compute the matrix $\beta_* : \mathrm{Hom}(A, B_0) \to \mathrm{Hom}(A, B_1)$ given by $f \mapsto \beta \circ f$. Since A is free we get an exact functor $\mathrm{Hom}(A, \_)$ such that

$$\left[\ B_0 \xrightarrow{\beta} B_1 \to B \to 0\ \right] \xRightarrow{\mathrm{Hom}(A,\_)} \left[\ \mathrm{Hom}(A, B_0) \xrightarrow{\beta_*} \mathrm{Hom}(A, B_1) \to \mathrm{Hom}(A, B) \to 0\ \right],$$

and thus $\mathrm{Hom}(A, B) \cong \mathrm{coker}(\beta_*)$. Evaluating an element of $\mathrm{Hom}(A, B)$ in A reduces to evaluating an element of $\mathrm{Hom}(A, B_1)$ in A, which is just matrix multiplication.

Now consider the general case where A and B are general finitely generated abelian groups represented by $\alpha : A_0 \to A_1$ respectively β. We note that the functor $\mathrm{Hom}(\_, B)$ is left-exact and contravariant. Applied to the exact sequence of A we get

$$\left[\ A_0 \xrightarrow{\alpha} A_1 \to A \to 0\ \right] \xRightarrow{\mathrm{Hom}(\_,B)} \left[\ 0 \to \mathrm{Hom}(A, B) \to \mathrm{Hom}(A_1, B) \xrightarrow{\alpha^*} \mathrm{Hom}(A_0, B)\ \right].$$

Hence $\mathrm{Hom}(A, B) \cong \ker(\alpha^*)$. By the previous case we may compute $\mathrm{Hom}(A_1, B)$ and $\mathrm{Hom}(A_0, B)$ since A_1 and A_0 are free. It is not difficult to show we may then compute α^* and in turn its kernel using Exercise 3.14. Evaluation in A of elements in $\mathrm{Hom}(A, B)$ reduces to evaluation in A_1 of elements in $\mathrm{Hom}(A_1, B)$, which we may also do by the previous case. □

Exercise 3.16 (Group exponent)**.** Show that there exists a polynomial-time algorithm that, given a finitely generated abelian group A

(a) and an element $a \in A$, decides whether a is torsion and if so computes $\mathrm{ord}(a)$;
(b) decides whether A is finite and if so computes its exponent and an $a \in A$ with that order.

Exercise 3.17 (Splitting exact sequences)**.** Show that there exists a polynomial-time algorithm that, given morphisms $f : A \to B$ and $g : B \to C$ of finitely generated abelian groups,

(a) decides whether the sequence $0 \to A \xrightarrow{f} B \xrightarrow{g} C \to 0$ is exact;
(b) if so, decides whether the sequence is split;
(c) if so, produces a left-inverse of f and a right-inverse of g.

Hint: Consider the map $g_* : \mathrm{Hom}(C, B) \to \mathrm{Hom}(C, C)$.

Note that by taking $C = 0$ in Exercise 3.17 we may conclude that there exists a polynomial-time algorithm that, given a morphism $f : A \to B$ of finitely generated abelian groups, decides whether f is an isomorphism and if so computes its inverse.

3.6 Structure theorem for finitely generated abelian groups

The structure theorem for finitely generated abelian groups is the following.

Theorem 3.20. *Suppose A is a finitely generated abelian group. Then there exists a unique sequence $(r, m, n_1, n_2, \ldots, n_m)$ of integers with $r, m \geq 0$ and $n_1, \ldots, n_m > 1$ such that $n_m \mid \cdots \mid n_2 \mid n_1$, for which*

$$A \cong \mathbb{Z}^r \times \prod_{k=1}^{m} (\mathbb{Z}/n_m\mathbb{Z}).$$

In this section we will prove its algorithmic counterpart.

Exercise 3.18. Let $n > 0$ and $M \subseteq \mathbb{Q}^n$. For subgroups $H \subseteq M$ write $H^\perp = \{x \in M \mid \langle x, H\rangle = 0\}$. Show that for all subgroups $N \subseteq M$ we have $(N^\perp)^\perp = (\mathbb{Q}N) \cap M$.
Hint: First consider the case where M and N are $\mathbb{Q}$-vector spaces.

Lemma 3.21. *There exists a polynomial-time algorithm that, given a finitely generated abelian group A, computes its torsion subgroup.*

Proof. For a subgroup H of $\mathbb{Z}^n$ write $H^\perp = \{x \in \mathbb{Z}^n \mid \langle x, H\rangle = 0\}$, or equivalently for a map $h : H \to \mathbb{Z}^n$ write $h^\perp : H^\perp \to \mathbb{Z}^n$ for the kernel of the map $\mathbb{Z}^n \to \mathrm{Hom}(H, \mathbb{Z})$ given by $x \mapsto (y \mapsto \langle x, h(y)\rangle)$. Note that using Theorem 3.11 we may compute $h^\perp$ in polynomial time. In particular, we may compute $(\alpha^\perp)^\perp : T \to A_1$ for the representative $\alpha : A_0 \to A_1$ of A. It follows from Exercise 3.18 that $(\alpha^\perp)^\perp$ is the torsion subgroup of A: Its image is precisely the set of those elements of A_1 for which a positive integer multiple is in $\alpha(A_0)$. □

Theorem 3.22. *There exists a polynomial-time algorithm that, given a finitely generated abelian group A, computes integers $(r, m, n_1, \ldots, n_m)$ with $r, m \geq 0$ and $n_1, \ldots, n_m > 1$ such that $n_m \mid \cdots \mid n_1$ and computes for A an isomorphism*

$$A \cong \mathbb{Z}^r \times \prod_{k=1}^{m} (\mathbb{Z}/n_m\mathbb{Z}),$$

i.e. projections to and inclusions from the individual factors on the right hand side.

Proof. We may compute using Lemma 3.21 the torsion subgroup T of A. Using the image algorithm of Exercise 3.14 we may compute an isomorphism $\mathbb{Z}^r \to A/T$. We have an exact sequence $0 \to T \to A \to A/T \to 0$ which splits, hence by Exercise 3.17 we may compute maps $A \to T$ and $A/T \to A$ such that $A \cong T \times (A/T)$. Replacing A by T we may now assume A is torsion. If $A = 0$ we are done. Using Exercise 3.16 we may compute an element $a \in A$ with order equal to the exponent e of A. Again we have an exact sequence $0 \to \mathbb{Z}a \to A \to A/(\mathbb{Z}a) \to 0$ which is split to which we apply Exercise 3.17. We proceed recursively with A replaced by $A/(\mathbb{Z}a)$. Note that the exponent of $A/(\mathbb{Z}a)$ is a divisor of the exponent of A, so indeed we will get $n_m \mid \cdots \mid n_1$. □

Corollary 3.23. *There exists a polynomial-time algorithm that, given a finitely generated abelian group A and a set S of integers, computes $r, m \in \mathbb{Z}_{\geq 0}$ and $c_1, \ldots, c_m \in \mathbb{Z}_{>1}$ and $n_1, \ldots, n_m \in \mathbb{Z}_{>0}$ such that any two c_i are either coprime or a power of the same integer, and every c_i either divides some power of an element of S or is coprime to all elements of S, and computes for A an isomorphism*

$$A \cong \mathbb{Z}^r \times \prod_{k=1}^{m} (\mathbb{Z}/c_k\mathbb{Z})^{n_k}.$$

Proof. Apply Theorem 3.22 and compute a coprime basis from $S \cup \{n_1, \ldots, n_m\}$ using Theorem 2.7. Write every n_i in terms of this basis and proceed as in Theorem 3.22. □

4 Computing symbols

In algebraic number theory we find lots of 'symbols'. It is not well-defined what a symbol is, but generally they are maps which encode algebraic properties of its parameters which also satisfies some reciprocity law. In this section we will use [4] as reference. In this section we will define and give algorithms for computing some of these symbols. The father of all symbols is the Legendre symbol.

Definition 4.1. Let p be an odd prime and let a be an integer coprime to p. We define the *Legendre symbol*

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & a \text{ is a square in } \mathbb{Z}/p\mathbb{Z} \\ -1 & \text{otherwise} \end{cases}$$

or equivalently $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p$.

It is easy to see that we can compute the Legendre symbol directly from the (equivalent) definition in polynomial time using a square-and-multiply algorithm modulo p.

Theorem 4.2 (Quadratic reciprocity). *Suppose p and q are distinct odd primes. Then*

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

Definition 4.3. Let b be a positive odd integer and a an integer coprime to b. In terms of the prime factorization $b = p_1^{k_1} \cdots p_n^{k_n}$ of b we define the *Jacobi symbol*

$$\left(\frac{a}{b}\right) = \left(\frac{a}{p_1}\right)^{k_1} \cdots \left(\frac{a}{p_n}\right)^{k_n},$$

where $\left(\frac{a}{p_i}\right)$ is the Legendre symbol defined previously.

Note that the Jacobi symbol extends the Legendre symbol, which justifies using the same notation for both. To compute the Jacobi symbol directly from the definition we need to be able to factor b, which is unfeasible. Quadratic reciprocity for the Jacobi symbol gives us a better method.

Proposition 4.4 (Quadratic reciprocity). *Suppose a and b are coprime positive odd integers. Then*

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

Proof. Note that for fixed b the maps $a \mapsto \left(\frac{a}{b}\right)\left(\frac{b}{a}\right)$ and $a \mapsto (-1)^{\frac{a-1}{2}\cdot\frac{b-1}{2}}$ are multiplicative. Hence it suffices to prove the proposition for a prime. Applying the same reasoning to b we have reduced to Theorem 4.2. $\square$

Exercise 4.1. Let p be an odd prime. Show that 2 is a square in $\mathbb{F}_p$ if and only if $p \equiv \pm 1 \mod 8$. Conclude that $\left(\frac{2}{b}\right) = (-1)^{(b^2-1)/8}$ for all odd $b > 0$. *Hint:* Write $\sqrt{2}$ in terms of 8-th roots of unity.

Exercise 4.2. Let $x_0, x_1, x_2, x_3 \in \mathbb{Z}$ such that x_1 is odd and $x_{n+2} \equiv x_n \mod x_{n+1}$ for all n. Show that when the expressions are well-defined we have the following equalities.

a. $\left(\frac{x_0}{x_1}\right) = (-1)^{(x_1-1)(x_2-1)/4} \cdot \left(\frac{x_1}{x_2}\right)$ if $x_2 \equiv 1 \bmod 2$, **b.** $\left(\frac{x_0}{x_1}\right) = \left(\frac{x_2}{x_3}\right)$ if $x_2 \equiv 0 \bmod 4$

and **c.** $\left(\frac{x_0}{x_1}\right) = (-1)^{(x_1x_3-1)(x_1x_3+x_2-1)/8} \cdot \left(\frac{x_2}{x_3}\right)$ if $x_2 \equiv 2 \mod 4$

Theorem 4.5. *There exists a polynomial time algorithm that, given coprime positive odd integers a and b, computes the Jacobi symbol $\left(\frac{a}{b}\right)$.*

Proof. We define the *gcd sequence* of positive integers x_0 and x_1 to be the sequence $(x_0, \dots, x_N)$ where x_{n+2} is the unique integer such that $0 \leq x_{n+2} < x_{n+1}$ and $x_{n+2} \equiv x_n \mod x_{n+1}$ and with $x_N = 0$. The proof of the Euclidean algorithm shows that this sequence contains only linearly many elements and can be computed in polynomial time. Moreover, $x_{N-1} = \gcd(x_0, x_1)$.

Compute the gcd sequence of a and b. As $\gcd(x_n, x_{n+1}) = \gcd(a, b) = 1$ for all $n < N$, the symbols $\left(\frac{x_n}{x_{n+1}}\right)$ are defined. Using Exerise 4.2 we may express $\left(\frac{a}{b}\right) = s_n \cdot \left(\frac{x_n}{x_{n+1}}\right)$ for some $s_n \in \{\pm 1\}$ iteratively for n with x_{n+1} odd. Clearly we may compute these s_n in polynomial time. As $\left(\frac{x_{N-2}}{x_{N-1}}\right) = \left(\frac{x_{N-2}}{1}\right) = 1$, we simply return s_{N-1}. □

Exercise 4.3. The Euclidean algorithm implied by Exercise 2.2 produces a different type of 'gcd sequence' than those used in Theorem 4.5, namely those where $|x_{n+2}| \leq |x_{n+1}|/2$ and $x_{n+2} \equiv x_n \mod x_{n+1}$ for all n. Give a proof of Theorem 4.5 using such gcd sequences.

The Jacobi symbol is defined on a subset of the integers. As is the theme in this document, we will 'extend' the Jacobi symbol to number rings.

Remark 4.6. One is sometimes interested in a more general Jacobi symbol $\left(\frac{a}{b}\right)$ where b is not necessarily odd or positive, by defining

$$\left(\frac{a}{2}\right) = (-1)^{\frac{a^2-1}{8}} \quad \text{and} \quad \left(\frac{a}{-1}\right) = \frac{a}{|a|} = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{if } a < 0 \end{cases}.$$

This is called the *Kronecker symbol.* The Kronecker symbol can be computed in polynomial time by writing $b = uc2^k$ where $u = \pm 1$ and c is odd and positive, and applying Theorem 4.5 to the factor $\left(\frac{a}{c}\right)$ in $\left(\frac{a}{b}\right) = \left(\frac{a}{u}\right)\left(\frac{a}{c}\right)\left(\frac{a}{2}\right)^k$.

4.1 Jacobi symbols in number rings

First we define the Legendre symbol for a general number ring.

Definition 4.7. Let $\mathfrak{p}$ be a prime ideal in a number ring R of odd index $n_\mathfrak{p} = (R : \mathfrak{p})$ and let $a \in R$ such that $aR + \mathfrak{p} = R$. We define the *Legendre symbol*

$$\left(\frac{a}{\mathfrak{p}}\right) = \begin{cases} +1 & \text{if } a \text{ is a square in } R/\mathfrak{p} \\ -1 & \text{otherwise} \end{cases},$$

or equivalently $\left(\frac{a}{\mathfrak{p}}\right) = a^{\frac{n_\mathfrak{p}-1}{2}} \mod \mathfrak{p}$.

Extending the definition to general ideals as for the Jacobi symbol cannot be done similarly, unless R is a Dedekind domain, because it would require prime factorization of ideals. Instead, we consider the following.

Definition 4.8. Suppose $\mathfrak{b}$ is an ideal of a number ring R. For a prime $\mathfrak{p}$ we define $l_\mathfrak{p}(\mathfrak{b})$ such that $(R_\mathfrak{p} : \mathfrak{b}_\mathfrak{p}) = (R : \mathfrak{p})^{l_\mathfrak{p}(\mathfrak{b})}$. For $(R : \mathfrak{b})$ odd and $a \in R$ with $aR + \mathfrak{b} = R$ we define the *Jacobi symbol* by

$$\left(\frac{a}{\mathfrak{b}}\right) = \prod_{\mathfrak{p} \in \operatorname{spec} R} \left(\frac{a}{\mathfrak{p}}\right)^{l_\mathfrak{p}(\mathfrak{b})}.$$

Theorem 4.9. *There exists a polynomial time algorithm that, given an order R, an ideal $\mathfrak{b}$ of R such that $(R : \mathfrak{b})$ is odd and $a \in R$ such that $aR + \mathfrak{b} = R$, computes the Jacobi symbol $\left(\frac{a}{\mathfrak{b}}\right)$.*

We will prove this theorem by expressing the Jacobi symbol in terms of yet another symbol.

Exercise 4.4. Let $\mathfrak{p}$ and $\mathfrak{b}$ be ideals in a number ring R with $\mathfrak{p}$ prime and take any composition series $0 = M_0 \subsetneq M_1 \subsetneq \cdots \subsetneq M_n = R/\mathfrak{b}$ of $R/\mathfrak{b}$ as R-module. Show that $l_\mathfrak{p}(\mathfrak{b})$ equals the number of quotients M_{i+1}/M_i that is isomorphic to $R/\mathfrak{p}$ as an R-module.

4.2 Signs of permutations

In this section we will consider the following symbol.

Definition 4.10. Let B be a finite abelian group and let $\sigma \in \mathrm{Aut}(B)$. We define (σ, B) to be the sign of σ as element of the permutation group on B.

Lemma 4.11. *Suppose A and C are sets and $\alpha \in \mathrm{Aut}(A)$ and $\gamma \in \mathrm{Aut}(C)$ are permutations. Write $\alpha \sqcup \gamma$ respectively $\alpha \times \gamma$ for the induced permutation on the disjoint union $A \sqcup C$ and product $A \times C$. Then $\mathrm{sgn}(\alpha \sqcup \gamma) = \mathrm{sgn}(\alpha) \cdot \mathrm{sgn}(\gamma)$ and $\mathrm{sgn}(\alpha \times \gamma) = \mathrm{sgn}(\alpha)^{\#C} \cdot \mathrm{sgn}(\gamma)^{\#B}$.*

Proof. That $\mathrm{sgn}(\alpha \sqcup \gamma) = \mathrm{sgn}(\alpha) \cdot \mathrm{sgn}(\gamma)$ follows from the fact that $\alpha \sqcup \gamma = \alpha\gamma$ when $\mathrm{Aut}(A)$ and $\mathrm{Aut}(C)$ are naturally mapped to $\mathrm{Aut}(A \sqcup C)$.

For the second part write $\alpha' = \alpha \times \mathrm{id}_C$ and $\gamma' = \mathrm{id}_A \times \gamma$ and note that $\alpha \times \gamma = \alpha' \cdot \gamma'$. Now α' acts as α on $\#C$ disjoint copies of A, hence by the previous $\mathrm{sgn}(\alpha') = \mathrm{sgn}(\alpha)^{\#C}$. Mutatis mutandis we obtain the same for γ', and the lemma follows from multiplicativity of the sign. $\square$

Proposition 4.12. *Suppose B is a finite abelian group and $\beta \in \mathrm{Aut}(B)$. Suppose we have an exact sequence $0 \to A \to B \to C \to 0$ such that β restricts to an automorphism α of A. Then β induces an automorphism γ of C such that the following diagram commutes*

$$\begin{tikzcd}
0 \arrow[r] & A \arrow[r, "f"] \arrow[d, "\alpha"] & B \arrow[r, "g"] \arrow[d, "\beta"] & C \arrow[r] \arrow[d, "\gamma"] & 0 \\
0 \arrow[r] & A \arrow[r, "f"] & B \arrow[r, "g"] & C \arrow[r] & 0
\end{tikzcd}$$

and if $\#C$ is odd we have $(\beta, B) = (\alpha, A) \cdot (\gamma, C)^{\#A}$.

Proof. The map γ exists by a diagram chasing argument. Since $\#C$ is odd we may write $C = \{0\} \sqcup D \sqcup (-D)$ for some subset $D \subseteq C$. Choosing any section $D \to B$ of g (which is not a group homomorphism!), we may extend it to a section $h : C \to B$ in such a way that $h(-c) = -h(c)$. Now the maps f and h together give a bijection of sets $A \times C \to B$ and let β' be the induced action of β on $A \times C$. By Lemma 4.11 we have that $(\alpha \times \gamma, A \times C) = (\alpha, A)^{\#C} \cdot (\gamma, C)^{\#A}$, hence to prove the proposition it suffices to show that $\sigma = (\alpha \times \gamma)^{-1} \cdot \beta'$ is an even permutation. For all $d \in D$ the action of σ restricts to $A \times \{d\}$. Note that σ commutes with -1, hence the action of σ on $A \times \{-d\}$ is isomorphic to the action on $A \times \{d\}$. Hence the restriction of σ to $A \times (C \setminus \{0\})$ is even. Finally, σ is the identity on $A \times \{0\}$, so we conclude σ is even, as was to be shown. $\square$

The exact sequence $0 \to 2\mathbb{Z}/4\mathbb{Z} \to \mathbb{Z}/4\mathbb{Z} \to \mathbb{Z}/2\mathbb{Z} \to 0$ resists application of Proposition 4.12. If we choose the non-trivial automorphism σ given by $x \mapsto -x$ on $\mathbb{Z}/4\mathbb{Z}$ we see that its sign is -1, while the induced maps on the other terms are trivial. Hence $(\sigma, \mathbb{Z}/4\mathbb{Z})$ is not an $\mathbb{F}_2$-linear combination of $(\sigma, 2\mathbb{Z}/4\mathbb{Z})$ and $(\sigma, \mathbb{Z}/2\mathbb{Z})$.

Exercise 4.5. Show that for $k \in \mathbb{Z}_{\geq 2}$ and $a \in (\mathbb{Z}/2^k\mathbb{Z})^*$ we have $(x \mapsto ax, \mathbb{Z}/2^k\mathbb{Z}) = (-1)^{\frac{a-1}{2}}$. *Hint:* Write $\mathbb{Z}/2^k\mathbb{Z} = (\mathbb{Z}/2^k\mathbb{Z})^* \sqcup (2\mathbb{Z}/2^k\mathbb{Z})$ and show $(x \mapsto ax, (\mathbb{Z}/2^k\mathbb{Z})^*) = -1$ if and only if a generates $(\mathbb{Z}/2^k\mathbb{Z})^*$.

Exercise 4.6. Suppose B is an abelian group. For $b \in B$ write $\lambda_b : B \to B$ for the map $x \mapsto x + b$.

a. Show that $\mathrm{sgn}(\lambda_b) = -1$ if and only if $(B : \langle b \rangle)$ is odd and $2 \mid \#B$.

Suppose B has order 2^k for some $k \geq 1$ and let $\beta \in \mathrm{Aut}(B)$.

b. Show that there exists a subgroup $A \subseteq B$ such that $\beta(A) = A$ and $(B : A) = 2$.

c. Let $b \in B$ such that $B = A \cup (b + A)$. Show that $(\beta, B) = -1$ if and only if $A = \langle \beta(b) - b \rangle$.

d. Suppose $(\beta, B) = -1$. Show that $B = \mathbb{Z}/2^k\mathbb{Z}$ or $B = (\mathbb{Z}/2\mathbb{Z})^2$.

e. Show that there exists a polynomial time algorithm that, given a finite abelian group B and $\beta \in \mathrm{Aut}(B)$ such that $2 \mid \#B$, computes (β, B).

4.3 Computing signs of group automorphisms

In this section we will prove we can compute the sign of group automorphisms in polynomial time. We will need an elementary lemma about determinants that mirrors Proposition 4.12.

Lemma 4.13. *Let $\mathbb{F}$ be a field and let $0 \to A \to B \to C \to 0$ be an exact sequence of finite dimensional $\mathbb{F}$-vector spaces together with automorphism α, β and γ such that the diagram*

$$\begin{tikzcd}
0 \arrow[r] & A \arrow[r] \arrow[d, "\alpha"] & B \arrow[r] \arrow[d, "\beta"] & C \arrow[r] \arrow[d, "\gamma"] & 0 \\
0 \arrow[r] & A \arrow[r] & B \arrow[r] & C \arrow[r] & 0
\end{tikzcd}$$

commutes. Then $\det(\beta) = \det(\alpha) \cdot \det(\gamma)$. □

In terms of matrices, the above lemma simply states that for square matrices A and C and a matrix P that fits, the block matrix $B = \left(\begin{smallmatrix} A & P \\ 0 & C \end{smallmatrix}\right)$ satisfies $\det(B) = \det(A) \cdot \det(C)$.

Theorem 4.14. *Suppose $b \in \mathbb{Z}_{>0}$ is odd and B is a free $(\mathbb{Z}/b\mathbb{Z})$-module of finite rank. Then for all $\sigma \in \operatorname{Aut}(B)$ we have $(\sigma, B) = \left(\frac{\det(\sigma)}{b}\right)$.*

Proof. For $B = 0$ the theorem clearly holds, so assume $b \neq 1$ and that B has rank at least 1.

First suppose b is prime and B has rank 1. Then σ is given by multiplication with $a \in (\mathbb{Z}/b\mathbb{Z})^*$. If a generates $(\mathbb{Z}/b\mathbb{Z})^*$, then the corresponding permutation fixes 0 and acts transitively on the $b-1$ remaining elements of B, so that $(\sigma, B) = -1 = \left(\frac{a}{b}\right) = \left(\frac{\det(\sigma)}{b}\right)$. By multiplicativity of both symbols in σ, this also proves the case where a is not a generator.

Now we prove using induction the case for general rank of B. Suppose σ is given by an upper or lower triangular matrix. Then there exists a subspace $0 \subsetneq A \subseteq B$ such that σ restricts to A. Hence we have a split exact sequence $0 \to A \to B \to C \to 0$ with $C = B/A$ and let α and γ be the induced maps on A respectively C. Then by Proposition 4.12, the induction hypothesis and Lemma 4.13 we get

$$(\sigma, B) = (\alpha, A) \cdot (\gamma, C) = \left(\frac{\det(\alpha)}{b}\right) \cdot \left(\frac{\det(\gamma)}{b}\right) = \left(\frac{\det(\alpha)\det(\gamma)}{b}\right) = \left(\frac{\det(\sigma)}{b}\right).$$

Since every matrix can be written as a product of upper and lower triangular matrices, the case for general α follows.

Now we prove the theorem for general b with induction to the number of divisors of b. We have just proven the induction base with b prime. For b not prime we may take a divisor $1 < d < b$ of b. Let $A = dB$ and $C = B/A$ and note that they are free modules over $\mathbb{Z}/\frac{b}{d}\mathbb{Z}$ respectively $\mathbb{Z}/d\mathbb{Z}$. Moreover, σ induces maps α and γ on A respectively C that make the usual diagram commute. It follows from the definition of the determinant that $\det(\alpha) \equiv \det(\sigma) \mod \frac{b}{d}$ and $\det(\gamma) \equiv \det(\sigma) \mod d$. Then

$$(\sigma, B) = (\alpha, A)(\gamma, C) = \left(\frac{\det(\alpha)}{b/d}\right)\left(\frac{\det(\gamma)}{d}\right) = \left(\frac{\det(\sigma)}{b/d}\right)\left(\frac{\det(\sigma)}{d}\right) = \left(\frac{\det(\sigma)}{b}\right).$$

The theorem now follows by induction. □

Theorem 4.15. *There exists a polynomial-time algorithm that, given an finite abelian group B and an automorphism σ of B, computes the symbol (σ, B).*

Proof. If $2 \mid \#B$ we have Exercise 4.6, so suppose B has odd order. Using Theorem 3.22, write B as a product $\prod_{k=1}^{m} (\mathbb{Z}/n_k\mathbb{Z})$ of non-trivial cyclic groups such that $n_j \mid n_k$ for all $j > k$. Note that B fits in an exact sequence $0 \to A \to B \to C \to 0$ with $A = n_m B$ and $C = B/A$, such that σ restricts to A and C. Then

$$A = \prod_{k=1}^{m} (n_m\mathbb{Z}/n_k\mathbb{Z}) \cong \prod_{k=1}^{m-1} \left(\mathbb{Z}\Big/\frac{n_k}{n_m}\mathbb{Z}\right) \quad \text{and} \quad C \cong (\mathbb{Z}/n_m\mathbb{Z})^m.$$

Since C is a free $(\mathbb{Z}/n_m\mathbb{Z})$-module, we may compute (σ, C) using Theorem 4.14 in polynomial time. Note that A is a product of strictly less cyclic groups than B, as well as having smaller order. While $A \neq 0$ we compute (σ, A) recursively and apply Proposition 4.12 to compute (σ, B). Since m is polynomially bounded in the length of the input, there is only polynomially many recursive steps and the algorithm runs in polynomial time. □

4.4 Computing Jacobi symbols in number rings

To compute Jacobi symbols in polynomial time it now suffices to reduce to Theorem 4.15.

Lemma 4.16. *Suppose $\mathfrak{b}$ is an ideal in a number ring R odd index $(R:\mathfrak{b})$, and suppose $a \in R$ satisfies $aR + \mathfrak{b} = R$. Then $\left(\frac{a}{\mathfrak{b}}\right) = (\alpha, R/\mathfrak{b})$ where the map α is multiplication by a.*

Proof. First suppose $\mathfrak{b}$ is a prime ideal and suppose a generates $(R/\mathfrak{b})^*$. Then $\left(\frac{a}{\mathfrak{b}}\right) = -1$. As a acts transitively on an even number of elements $(R/\mathfrak{b}) \setminus \{0\}$, we conclude that $(x \mapsto ax, R/\mathfrak{b}) = -1 = \left(\frac{a}{\mathfrak{b}}\right)$. The case for general a follows from multiplicativity of both symbols.

Now consider the case of general $\mathfrak{b}$. Choose some composition series $0 = M_0 \subsetneq M_1 \subsetneq \cdots \subsetneq M_n = R/\mathfrak{b}$ of $R/\mathfrak{b}$ as R-module. Consider the exact sequence $0 \to M_i \to M_{i+1} \to M_{i+1}/M_i \to 0$ and note that $M_{i+1}/M_i \cong R/\mathfrak{p}$ as R-modules for some prime ideal $\mathfrak{p}_i$ of R. By applying Proposition 4.12 inductively we obtain $(\alpha, B) = \prod_{i=1}^n (\alpha, M_{i+1}/M_i) = \prod_{i=1}^n \left(\frac{\alpha}{\mathfrak{p}_i}\right)$. It follows from Exercise 4.4 that the latter equals $\left(\frac{\alpha}{\mathfrak{b}}\right)$. ☐

Proof of Theorem 4.9. Compute $R/\mathfrak{b}$ and the map $\alpha : R/\mathfrak{b} \to R/\mathfrak{b}$ given by $x \mapsto ax$. Using Theorem 4.15 compute $(\alpha, R/\mathfrak{b})$, which equals $\left(\frac{a}{\mathfrak{b}}\right)$ by Lemma 4.16. ☐

References

[1] Daniel J. Bernstein. Factoring into coprimes in essentially linear time. *Journal of Algorithms*, 54(1):1–30, 2005.

[2] I. Ciocanea Teodorescu. *Algorithms for finite rings.* PhD thesis, Leiden Univ., 2016.

[3] J. E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages and Computations.* Pearson, 3 edition, 2006.

[4] H. W. Lenstra, Jr. Computing jacobi symbols in algebraic number fields. *Nieuw Archief voor Wiskunde*, 14(3):421–426, 1995.

[5] H. W. Lenstra, Jr. Lattices. *Algorithmic Number Theory*, 44:127–181, 2008.

[6] P. Q. Nguyen and B. Vallée, editors. *The LLL Algorithm, Survey and Applications.* Springer, 2010.