Chapter 3

Linear algebra over $\mathbb{Z}$: basic algorithms for finite rings

We have seen in the previous chapter how to represent and carry out basic computations with finite abelian groups. This enables us to deal with the underlying additive group of a finite ring, which is crucial for all our algorithms. This chapter is a compendium of basic algorithms to do with finite rings.

We will represent finite rings using "basis representations" and describe algorithms that accomplish the following tasks:

1. compute homomorphism groups between two finite modules over a finite ring,
2. compute the ideal generated by a given set of elements,
3. compute sums, products and intersections of ideals,
4. compute the quotient of a finite ring by a two-sided ideal,
5. compute the characteristic, centre and prime subring of a finite ring.

We will also look at the problem of computing the Jacobson radical of a ring. In the case where the given ring is a finite-dimensional algebra over a field, this can be accomplished deterministically in polynomial time ([27, 75]). However, in general, such a result cannot be expected, in view of our inability to compute the largest square divisor of an integer.

Finally, we will briefly look at some other known algorithms for finite rings, and some open algorithmic questions.

3.1 Representing objects and basic constructions

3.1.1 Representing rings and modules

To represent finite rings inside algorithms, we will use *basis representations*. These are considered to be the "right" representations for complexity considerations, since they are neither too verbose (so as to make all problems quasipolynomial), nor too compact (so as to make all problems NP-hard). For more on the different representations and the complexity of problems on different representations, see [2, 4, 52, 53].

Definition 3.1.1. *Let R be a finite ring. A* basis representation *of R consists of a sequence of integers $d_1, \ldots, d_t \in \mathbb{Z}_{>1}$, for some $t \in \mathbb{Z}_{\geq 0}$ such that*

$$R^+ \cong \bigoplus_{i=1}^{t} (\mathbb{Z}/d_i\mathbb{Z}), \tag{3.1}$$

together with a bilinear map

$$\begin{aligned} \sigma : R^+ \times R^+ &\to R^+ & (3.2)\\ (e_i, e_j) &\mapsto e_i e_j, & (3.3)\end{aligned}$$

where e_i is a generator of the cyclic subgroup $\mathbb{Z}/d_i\mathbb{Z}$, for $1 \leq i \leq t$, and we express $e_i e_j$ linearly in terms of $\{e_i\}$, i.e. for each $1 \leq i, j \leq t$ we give a sequence $a_{ijk} \in \mathbb{Z}/d_k\mathbb{Z}$, for $1 \leq k \leq t$, such that $e_i e_j = \sum_{k=1}^{t} a_{ijk} e_k$.

Note 3.1.2. We have established in Definition 2.3.2 that the default representation of a finite abelian group is the exact-sequence representation. However, by the results of Section 2.6.5, we may assume that R^+ is in fact given by a direct-sum decomposition into cyclic groups.

Note 3.1.3. The map σ in (3.2) will be referred to as the *multiplication map* of R. Specifying σ amounts to giving t^3 integers a_{ijk}, which are called *structure constants*.

Note 3.1.4. Given $d_1, \ldots, d_t$ and a sequence of t^3 integers, a_{ijk}, we can check in polynomial time whether they define a ring. This amounts to checking a series of equalities and solving systems of linear equations over $\mathbb{Z}$.

Note 3.1.5. The size of a basis representation is equal to

$$\sum_{i=1}^{t} \log_2(d_i) \cdot t^2 = \log_2(|R|) \cdot t^2 \leq \log_2^3(|R|), \tag{3.4}$$

since $t \leq \log_2(|R|)$. Thus, when we say an algorithm with input R runs in polynomial time, we mean that the number of bit operations is bounded above by $(\log_2(2 + |R|))^C$, for some constant C. The 2 is added in order to accommodate the zero ring.

To input a finite module, we give a finite abelian group $(M, +)$ and a bilinear map

$$\alpha : R^+ \times M \to M, \tag{3.5}$$

which describes the action of R on M. For every additive generator of R and M, we express the image in terms of the additive generators of M.

3.1.2 Representing ring and module homomorphisms

Let R_1 and R_2 be two finite rings. A ring homomorphism $\rho : R_1 \to R_2$ is a homomorphism of the underlying abelian groups that sends the unit element of R_1 to the unit element of R_2 and respects the multiplicative structure of the rings, i.e. for all $r, s \in R_1$, we have that

$$\rho(rs) = \rho(r)\rho(s). \tag{3.6}$$

Proposition 3.1.6. *There exists a deterministic polynomial-time algorithm that, given two finite rings R_1 and R_2, and a group homomorphism $\rho : R_1^+ \to R_2^+$, decides whether ρ is a ring homomorphism, and if it is, decides whether it is injective or surjective.*

Proof. As in Section 2.3.2, the map ϕ is given by a matrix which specifies the image of each additive generator of R_1 as a linear combination of additive generators of R_2. Given such a matrix, we can easily check if it induces a homomorphism of rings, by verifying that the induced map preserves multiplication and sends the unit element of one ring to the unit element of the other. This amounts to checking equalities over $\mathbb{Z}$. By Propositions 2.3.6 and 2.3.7, we can also check for injectivity or surjectivity. □

Let R be a ring and M, N two R-modules. A module homomorphism $\phi : M \to N$ is a homomorphism of abelian groups which is R-linear, i.e. for all $r \in R, m \in M$, we have that

$$\phi(rm) = r\phi(m). \tag{3.7}$$

Proposition 3.1.7. *There exists a deterministic polynomial-time algorithm that, given a finite ring R, two R-modules M and N, and a group homomorphism $\phi : M \to N$, decides whether ϕ is an R-module homomorphism, and if it is, decides whether it is injective or surjective.*

Proof. As in Chapter 2, Section 2.3.2, the map ϕ is given by a matrix which specifies the image of each additive generator of M as a linear combination of additive generators of N. Given such a matrix, we can easily check if it induces a module homomorphism by verifying that the induced map preserves scalar multiplication. This amounts to checking equalities over $\mathbb{Z}$. By Propositions 2.3.6 and 2.3.7, we can also check for injectivity or surjectivity. □

3.1.3 Homomorphism group

Proposition 3.1.8. *There exists a deterministic polynomial-time algorithm that, given a finite ring R and two finite R-modules M_1 and M_2, computes the homomorphism group* $\operatorname{Hom}_R(M_1, M_2)$.

Proof. We can certainly compute $\operatorname{Hom}_{\mathbb{Z}}(M_1, M_2)$. Then

$$\operatorname{Hom}_R(M_1, M_2) = \{f \in \operatorname{Hom}_{\mathbb{Z}}(M_1, M_2) \mid f(rx) = rf(x), \forall r \in R, \forall x \in M_1\}. \quad (3.8)$$

It is enough to ensure that the relation $f(rx) = rf(x)$ holds for the additive generators of R and M_1. Consider the map

$$\begin{aligned} \operatorname{Hom}_{\mathbb{Z}}(M_1, M_2) &\to \bigoplus_{\substack{r \text{ additive generator of } R \\ x \text{ additive generator of } M_1}} M_2 \\ f &\mapsto (f(rx) - rf(x))_{r,x}. \end{aligned}$$

Then $\operatorname{Hom}_R(M_1, M_2)$ is the kernel of this map, which we can compute. □

3.2 Computations with ideals

Let R be a ring. A left ideal I is, in particular, a left R-module, so it is given to the algorithm as an additive subgroup of R, together with a map $R \times I \to I$, as in (3.5). Right ideals and two-sided ideals are given in a similar way.

3.2.1 Computing the ideal generated by a given set of elements

Proposition 3.2.1. *There exists a deterministic polynomial-time algorithm that, given a finite ring R and a set $S \subset R$, computes the ideal generated by S in R.*

Proof. Suppose $S = \{s_1, \ldots, s_u\}$. The left ideal I, generated by S in R has underlying additive group generated by the set $\{e_i s_j \mid 1 \leq i \leq t, 1 \leq j \leq k\}$, which can be computed using Proposition 2.3.10. This will also produce an injective group homomorphism $I \hookrightarrow R$ specifying I^+ as a subgroup of R^+. To determine the R-action, we look at the map $\sigma : R^+ \times R^+ \to R^+$ as in (3.2), giving the multiplicative structure of R. Suppose $e_i s_j = \sum_{n=1}^{t} b_{ijn} e_n$. Then

$$e_k e_i s_j = \left(\sum_m a_{kim} e_m\right) s_j = \sum_{m,n} a_{kim} b_{mjn} e_n. \quad (3.9)$$

Now we express this sum as a linear combination of the $e_i s_j$. □

The right and two-sided ideals are dealt with in a similar manner.

3.2.2 Sum, product and intersection of ideals

Proposition 3.2.2. *There exists a deterministic polynomial-time algorithm that, given a finite ring R and two ideals $I, J \subset R$, computes the ideals $I + J, I \cap J$ and IJ.*

Proof. Note that $(I+J)^+ = I^+ + J^+$ and $(I \cap J)^+ = I^+ \cap J^+$, and the action of R is induced by the ring multiplication map σ (as in (3.2)).

Suppose $x_1, \ldots, x_n$ is a set of additive generators of I and $y_1, \ldots, y_m$ is a set of additive generators of J. Recall that $IJ \subseteq J$. We have that

$$(IJ)^+ = \langle \{x_i \cdot y_j\}_{i,j} \rangle_{\mathbb{Z}}, \tag{3.10}$$

where the product $x_i \cdot y_j$ is computed by writing each x_i and y_j in terms of additive generators of R and then using the multiplication map σ to write each $x_i \cdot y_j$ as a linear combination of additive generators of R. The action of R on $(IJ)^+$ is again induced by σ. □

The right and two-sided ideals are dealt with in a similar manner.

3.2.3 Quotient of ring and two-sided ideal

Proposition 3.2.3. *There exists a deterministic polynomial-time algorithm that, given a finite ring R and a two-sided ideal I, computes the quotient ring R/I.*

Proof. Let I be a two-sided ideal of R. Then

$$(R/I)^+ = R^+/I^+, \tag{3.11}$$

which can be computed using Proposition 2.3.13. This will also produce a surjective group homomorphism $R^+ \twoheadrightarrow (R/I)^+$. The multiplication map for R/I is induced by the multiplication map for R. □

3.3 Computing the centre and the prime subring of a finite ring

Given a finite ring, we will often want to view it as an algebra over its centre or its prime subring. We show how to compute these.

Theorem 3.3.1. *There exists a deterministic polynomial-time algorithm that, given a finite ring R, computes its centre, prime subring and characteristic.*

Proof. To compute the centre of a finite ring R, we consider the map

$$\begin{aligned} \psi : R^+ &\longrightarrow \bigoplus_{r \text{ additive generator of } R} R^+ \\ s &\longmapsto (rs - sr)_r. \end{aligned}$$

The centre of R is the kernel of ψ, which we can compute. Its ring structure is induced by σ, the map defining multiplication in R.

To compute the prime subring of a finite ring R, we consider the map

$$\alpha : \mathbb{Z} \to R^+, \quad 1 \mapsto 1_R.$$

The prime subring of R is the image of α, which we can compute. The multiplication map is induced by σ.

The cardinality of the prime subring is the characteristic of R, which we can also compute by taking the lowest common multiple of the sizes of the cyclic direct summands of R^+. $\square$

3.4 Computing the Jacobson radical

When dealing with problems concerning rings, it is often convenient to reduce to the semisimple case. When the ring at hand is left-artinian, this reduces to computing the Jacobson radical (see Theorem 1.4.9, part (iv)). Thus, the natural question to ask is if Jacobson radicals can be efficiently computed. For our purposes, the appropriate version of this question is whether the Jacobson radical of a finite ring can be computed deterministically in polynomial time.

If the given ring R is a finite-dimensional algebra over a "nice" field $\mathbb{F}$, there do exist deterministic polynomial-time algorithms that compute the Jacobson radical of R (see [18, 27, 75]). For $\mathbb{F}$ a field of characteristic 0, this reduces to solving a system of linear equations over $\mathbb{F}$ by Dickson's theorem (see Theorem 1.4.10). If $\mathbb{F}$ is a finite field, then Friedl and Rónyai showed how to recursively construct a sequence of ideals of R, whose last element is equal to $\mathrm{J}(R)$. Using a very similar technique, Cohen, Ivanyos and Wales generalised these results, showing how to compute the Jacobson radical in the case that $\mathbb{F}$ is any field in which one can perform arithmetic and over which one can solve *semilinear equations* of the form $\sum_{i=1}^{k} a_i x_i^p = 0$, where $p = \mathrm{char}(\mathbb{F})$, $k \in \mathbb{Z}_{>0}$ and $a_i \in \mathbb{F}$ for all $1 \le i \le k$. Finite fields are examples of such fields.

Theorem 3.4.1 ([18])**.** *There exists a deterministic polynomial-time algorithm that, given a finite-dimensional algebra R over a field $\mathbb{F}$, where $\mathbb{F}$ is a field over which we can perform arithmetic and solve semilinear equations, computes the Jacobson radical of R.*

Note 3.4.2. We cannot in general expect to be able to compute the Jacobson radical for rings not containing a field. To see this, consider rings of the form $\mathbb{Z}/n\mathbb{Z}$, with $n \in \mathbb{Z}_{>0}$, for which the task ultimately reduces to finding square divisors of n. This is not something we know how to do deterministically in polynomial time.

3.5 Other known algorithms and open questions

Rings are ubiquitous. It is thus important to have a wide range of algorithms to deal with finite rings. This list of deterministic polynomial-time algorithms for finite rings however, is not as long as would be expected for such basic objects. One of the reasons for this is that many problems for finite rings reduce to rings of the type $\mathbb{Z}/n\mathbb{Z}$, and at this stage the fact that we cannot factor n efficiently becomes a serious issue.

The study of algorithmic problems involving automorphisms and isomorphisms of finite rings intensified after the first deterministic polynomial-time primality test was formulated by Agrawal, Saxena and Kayal in terms of automorphisms of a certain finite ring (see [1]). Subsequently, the same authors studied how some of the most important open algorithmic questions, like integer factorisation, polynomial factorisation over finite fields and graph isomorphism can be reduced to ring automorphism questions. The questions for which deterministic polynomial-time algorithms are sought are the following:

1. Ring Isomorphism Problem
   - (i) Decision version: Given two finite rings, decide if they are isomorphic.
   - (ii) Search version: Given two finite rings, find an isomorphism if one exists.
   - (iii) Counting version: Given two finite rings, compute the number of isomorphisms between them.
2. Ring Automorphism Problem
   - (i) Decision version: Given a finite ring, decide if it has a nontrivial ring automorphism.
   - (ii) Search version: Given a finite ring, find a nontrivial automorphism if one exists.
   - (iii) Counting version: Given a finite ring, compute the number of its automorphisms.

As far as deterministic polynomial-time algorithms are concerned, all of the above problems are open, with the exception of the decision version of the ring automorphism problem, which was shown to be in P in [53]. The algorithm given there relies on the classification of rigid rings (i.e. rings with no nontrivial automorphisms).

Theorem 3.5.1 ([53], Theorem 7.1)**.** *There exists a deterministic polynomial-time algorithm that, given a finite ring R, determines whether R has a nontrivial automorphism.*

It is shown in [53] that both integer factorisation and the graph isomorphism problem reduce to the counting version of the ring automorphism problem, which is unlikely to be NP-complete. The decision version of the ring isomorphism problem is shown to be at least as hard as the graph isomorphism problem. Moreover, integer factorisation reduces to the search version of the ring isomorphism problem.

For finite fields, the isomorphism problem can be handled deterministically in polynomial time.

Theorem 3.5.2 ([65], Theorem 1.2)**.** *There exists a deterministic polynomial-time algorithm that, given two finite fields of the same cardinality, exhibits an isomorphism between them.*

Apart from these problems, to which systematic study has been devoted, the list of algorithms for finite rings remains quite a short one. It is one of the goals of this thesis to expand on this list, thus supplementing the toolbox for algorithmically dealing with finite rings.