MODEL – Euclide's paradigm

In this course, we mainly study Euclide's algorithm, some of its variants and its immediate applications (essentially to what is called rational reconstruction).

A little bit more on algebraic structures. Here we may need to introduce algebraic closures and quotient rings.

Lemma 1. *Let $\mathbb{K}$ be a field and $f \in \mathbb{K}[X]$ of degree d. Then f has at most d roots in $\mathbb{K}$.*

The proof of this lemma is immediate and any student should know how to prove it.

Lemma 2. *Let $f \in \mathbb{K}[X]$ and $a \in \mathbb{K}$ such that $f(a) = 0$. Then, $X - a$ divides f.*

Again, the proof of this lemma is immediate and any student should know how to prove it.

Definition 3. *Let $\mathbb{K}$ be a field. We say that $\bar{\mathbb{K}}$ is an algebraic closure of $\mathbb{K}$ if for all $f \in \bar{\mathbb{K}}[X]$ of degree d, f has d roots in $\bar{\mathbb{K}}$.*

The fields $\mathbb{Q}$ and $\mathbb{R}$ are not algebraically closed and $\mathbb{C}$ is an algebraic closure of $\mathbb{Q}$ (and of $\mathbb{R}$).

Definition 4. *Let R be a ring and $f \in R[X]$ be a* monic *polynomial (i.e. its leading coefficient is 1). Let also $(s, t) \in R[X] \times R[X]$.*

One says that s and t are equivalent modulo f (and we write $s \sim_f t$) if f divides $s - t$.

The set of class of equivalences is denoted by $\frac{R[X]}{\langle f \rangle}$; we call it quotient ring.

Division algorithm for polynomials in $\mathbb{K}[X]$. Let $\mathbb{K}$ be a field, $A = a_p X^p + \cdots + a_0$ and $B = b_q X^q + \cdots + b_0$ in $\mathbb{K}[X]$.

Recall that $\mathbb{K}[X]$ is a Euclidean domain ; actually we take for the map h used to define the division, is the degree function (the one which associates to a polynomial in $\mathbb{K}[X]$ its degree).

Using it, one can define the Euclidean division in $\mathbb{K}[X]$. The traditional way to define it is as the output of the division algorithm below.

DivisionAlgorithm(A, B)

- **Input:** Two polynomials $A = a_p X^p + \cdots + a_0$ and $B = b_q X^q + \cdots + b_0$ in $\mathbb{K}[X]$.
- **Output:** Two polynomials (q, r) in $\mathbb{K}[X]$ such that

$$A = Bq + r \qquad \text{and } 0 \leq \deg(r) < q = \deg(B)$$

1. $r = A$ and $q = 0$, $b =$ leadingcoefficient(B);
2. while $\deg(r) \geq \deg(B)$ do
   (a) $a =$ leadingcoefficient(r);
   (b) $q = q + \frac{a}{b} X^{\deg(B) - \deg(r)}$;

(c) $r = r - \frac{a}{b} X^{\deg(B)-\deg(r)} B$

3. return (q, r).

Observe that it is a *rewriting* algorithm since it consists in applying the rewriting rule

$$X^q \to - \left(\frac{b_{q-1} X^{q-1} + \cdots + b_0}{b_q} \right).$$

Observe that when $q = 1$ this matches with the substitution operation. Hence, divising A by B when $q = 1$ consists in evaluating A at the *root* of B.

Actually, observe that when ρ is a root of B (in an algebraic closure of $\mathbb{K}$), A and r (the remainder of the Euclidean division of A by B) take the same value at ρ.

Theorem 5. *On input A and B in $\mathbb{K}[X]$ of respective degrees p and q with $p \geq q$, the above division algorithm performs $O(q(p-q))$ arithmetic operations in $\mathbb{K}$.*

Proof. The while loop performs $O(q)$ arithmetic operations. Observe also that it is performed $p-q+1$ times in the worst case. This ends the proof of the complexity statement. □

Observe that when $\simeq \frac{p}{2}$ the algorithm gets a quadratic behaviour in p while the output is an array of at most $q+1$ coefficients. Hence, the algorithm is not asymptotically optimal.

Remark. Obersve that if we assume that B is monic (i.e. its leading coefficient is 1), one can extend Euclidean division of A by B and assume that they have **coefficients in a ring**. Indeed, observe that when B is monic, the above Euclidean division algorithm does not perform any division.

Exercise. What is the bit complexity of the DivisionAlgorithm when the input has coefficients in $\mathbb{Q}$?

(A rational number a/b for a, b in $\mathbb{Z} \times \mathbb{Z} - \{0\}$ is said to be normalized when $\mathsf{gcd}(a, b) = 1$; then the height of a/b is the sum of the heights of a and b).

The DivisionAlgorithm is not satisfactory and one can obtain a much better one. Below, we describe the fast division algorithm.

Before that, let us give the rough idea underlying the algorithm. We aim at finding (q, r) such that $A = Bq + r$. Hence, one gets the following relation:

$$\frac{A}{B} = q + \frac{r}{B}.$$

Now, imagine that $\mathbb{K} = \mathbb{R}$. Observe that q corresponds to the asymptotic development of $\frac{A}{B}$ at infinity since because of the degree constraints on r, the fraction $\frac{r}{B}$ tends to 0 at infinity.

Hence, one can obtain q by computing the Taylor development at infinity of the fraction $\frac{A}{B}$. Since it is more convenient to compute Taylor developments at 0, one just performs the change of variables $Y \leftarrow \frac{1}{X}$ which sends ∞ to 0. One obtains the following identity:

$$\frac{X^p A(1/X)}{X^q B(1/X)} = X^{p-q} q(1/X) + \frac{X^p r(1/X)}{X^q B(1/X)}.$$

We describe now more formally the algorithm.

FastDivisionAlgorithm(A, B)

- **Input:** Two polynomials $A = a_p X^p + \cdots + a_0$ and $B = b_q X^q + \cdots + b_0$ in $\mathbb{K}[X]$.
- **Output:** Two polynomials (q, r) in $\mathbb{K}[X]$ such that

$$A = Bq + r \qquad \text{and } 0 \leq \deg(r) < q = \deg(B)$$

1. Compute $\tilde{A} = X^p\ A(1/X)$ and $\tilde{B} = X^q\ Q(1/X)$

   Observe that this step does not require any arithmetic operation.

2. Compute the quotient $\tilde{A}/\tilde{B} \mod X^{p-q+1}$ by inverting a formal power series and performing a multiplication. Let Q be this quotient.

3. Deduce q by reverting the coefficients of Q.

4. Deduce r by computing $A - Bq$.

Theorem 6. *On input A and B in $\mathbb{K}[X]$ of respective degrees p and q with $p \geq q$, the* FastDivisionAlgorithm *algorithm performs $\mathsf{M}(p)$ arithmetic operations in $\mathbb{K}$ (where $\mathsf{M}(d)$ denotes the cost of multiplying two polynomials of degree $\leq d$ in $\mathbb{K}[X]$).*

Proof. The proof is immediate provided that the student can estimate the cost of inverting a series up to some power. □

Basic exercise. What is the bit complexity of the FastDivisionAlgorithm for inputs in $\mathbb{Q}[X]$?
Same question for integers.

One can now investigate some applications and consequences.

Corollary 7. *Let R be a ring and f be a monic polynomial in $R[X]$ of degree d. The quotient $\frac{R[X]}{\langle f \rangle}$ equipped with the addition and multiplication induced by the addition in $R[X]$ and the combination of multiplication and Euclidean division by f in $R[X]$ is a ring in which one can perform addition and multiplication in time $O(\mathsf{M}(d))$.*

This is also a $\mathbb{K}$-vector space of dimension $d+1$.

Proof. The proof is rather straightforward. The single subtelty is to use the assumption on monicity of f to show that Euclidean division extends naturally. □

Euclide's algorithm (and extended Euclide's algorithm. From now on, we assume that R is an integral ring.

Definition 8. *Let a and b in R. One says that g is a greatest common divisor (gcd) of a and b in R if*

- *$g \in R$;*
- *g divides a;*
- *g divides b;*
- *any common divisor of a and b divides g.*

The above definition is rather general. In particular, there exist integral rings containing couples of elements without any gcd.

Example. Let R be the ring $\mathbb{Z}[i\sqrt{3}]$, $a = 4$ and $b = 2 + 2i\sqrt{3}$. One can establish easily that $g_1 = 2$ divide both a and b; $g_2 = 1 + i\sqrt{3}$ divide both a and b but there is no gcd to a and b (because $g_1 g_2$ does *not* divide a in R).

Integral rings admitting gcds for all their couples of elements are called rings with gcd. In general, one preferes to speak about *factorial rings* (students are encouraged to read a bit about these rings in any textbook or other reference). For computational purpose, the subclass of rings with gcds are simply

Euclidean rings. Recall that a height function h is aassociated to any element in Euclidean rings. Recall also that for $0 \in R$, $h(0)$ is less than the height of any other element of R.

The core idea behind Euclide's algorithm is that on input a and b in a Euclidean ring R equipped with a height function h and for (q, r) such that

$$a = bq + r \qquad \text{with} \qquad h(r) < h(b)$$

one has $\mathsf{gcd}(a, b) = \mathsf{gcd}(b, r)$. Hence one can call recursively the algorithm on input (b, r), yielding a decreasing sequence of height values.

EuclideAlgorithm

- **Input.** Two elments a and b in a Euclidean ring R equipped with the height function h;
- **Output.** A gcd of a and b in R.

1. $r_0 = a$ and $r_1 = b$ and $i = 1$
2. while $r_i \neq 0$ do
   (a) $r_{i+1} = \mathsf{remainder}(r_{i-1}, r_i)$
   (b) $i = i + 1$
3. return r_{i-1}.

The sequence $(r_0, \ldots, r_\delta)$ is called the sequence of Euclidean remainders (or Euclidean remainder sequence).

Theorem 9. *Let $R = \mathbb{K}[X]$ where $\mathbb{K}$ is a field. The above algortihm* EuclideAlgorithm *on input two polynomials A and B in $\mathbb{K}[X]$ returns a gcd of A and B and performs $O(\deg(A)\deg(B))$ operations in $\mathbb{K}$.*

Proof. Correctness is immediate through an reasoning by induction as the one done before the description of the algorithm. We focus on the complexity statement, assuming that $\deg(A) \geq \deg(B)$.

Using the complexity of the naive algorithm performing the Euclidean division, dividing a polynomial S by a polynomial T is done using $O(\deg(T)(\deg(S) - \deg(T)))$. Hence there exists a constant K such that this is done using *at most* $K \deg(T)(\deg(S) - \deg(T))$.

One deduces that in order to estimate the cost of Euclide's algorithm in $\mathbb{K}[X]$, one needs to upper bound the sum of all

$$K \deg(R_i)(\deg(R_{i-1}) - \deg(R_i)).$$

(to keep notations coherent, the sequence of remainders is denoted now by capital letters as we do for polynomials).

But all polynomials R_i have degree bounded by $\deg(B)$. Hence one needs to bound now

$$K \deg(B) \sum_{i \geq 1} (\deg(R_{i-1}) - \deg(R_i)) \leq K \deg(B) \deg(A).$$

This finishes the proof. □

The above complexity bound does reflect the practical behaviour of the algorithm when the degrees of the Euclidean remainder sequence decrease one by one.

Let us go back to the case of a Euclidean ring R and let a and b in R with $g = \mathsf{gcd}(a, b)$. Assume that $g = 1$. One then expects to find u and v in R such that

$$g = ua + vb.$$

The reason of this is rather intuitive and goes back to the case of integers. When a and b are *coprime* integers (e.g. their gcd is 1), one expects to be able to "invert" a modulo b, hence establishing that $\mathbb{Z}/p\mathbb{Z}$ is a field in p is a prime integer.

When a and b are elements of a Euclidean ring R, and when $g = \mathsf{gcd}(a, b)$ one expects to find u and v with additional height properties

$$g = ua + vb \qquad \text{and } h(ug) < h(b) \text{ and } h(vg) < h(a).$$

This has *many* applications, in particular in computer algebra, cryptography, computational geometry, etc.

The elements u and v as above are called cofactors ; again this terminology comes from the analogy with integers.

The rather intuitive idea for computing the cofactors is as follows. At each step i of Euclide's algorithm, one finds u_i and v_i such that

$$r_i = u_i a + v_i b.$$

Note that for $i = 0$, it suffices to take $u_0 = 1$ and $v_0 = 0$ which trivially yields $1.a + 0.b = a$. For $i = 1$, one also easily obtains $u_1 = 0$ and $v_1 = 1$. Next, Euclidean division enters in the game with the relation

$$r_{i-1} = q_i r_i + r_{i+1}.$$

Using the recurrence relation, one obtains

$$r_{i+1} = r_{i-1} - q_i r_i = (u_{i-1} - q_i u_i)a + (v_{i-1} - q_i v_i)b$$

which shows that one can define

$$u_{i+1} = u_{i-1} - q_i u_i \qquad \text{and} \qquad v_{i+1} = v_{i-1} - q_i v_i$$

Proving that using this definition for the cofactors satisfies the height requirements is easy to do by induction.

ExtendedEuclideAlgorithm

- **Input.** two elements a and b of a Euclidean ring R
- **Output.** the gcd of a and b as well as the cofactors.

1. $r_0 = a$, $r_1 = b$, $u_0 = 1$, $v_0 = 0$, $u_1 = 0$, $v_1 = 1$, $i = 1$
2. while $r_i \neq 0$ do
   (a) $(q_i, r_{i+1}) = \mathsf{EuclideanDivision}(r_{i-1}, r_i)$
   (b) $u_{i+1} = u_{i-1} - q_i u_i$ and $v_{i+1} = v_{i-1} - q_i v_i$
   (c) $i = i + 1$
3. return $r_{i-1}, u_{i-1}, v_{i-1}$.

Theorem 10. *When $R = \mathbb{K}[X]$, the above algorithm runs on input A and B using $O(\deg(A)\deg(B))$ arithmetic operations in $\mathbb{K}$.*

Proof. The proof is immediate from the analysis of Euclide's algorithm and taking into account the extra cost of computing the cofactors. □

Applications. The first application of Euclide-like's algorithms is the computation of the gcd of two elements in a Euclidean domain.

Theorem 11. *Let a and b two elements of a Euclidean ring R and g be the output of* EuclideAlgorithm *on input (a, b). Then g is a gcd of a and b.*

Another *super important* application of the extended Euclidean algorithm is the so-called modular inversion. For a Euclidean ring R, we say that (p_1, p_2) in $R \times R$ are coprime when their gcd is 1. Hence, by Bézout's relation, there exists a couple (u, v) in $R \times R$ such that

$$up_1 + vp_2 = 1.$$

We use this modular inversion to *solve* modular equations. Let R be a Euclidean ring, p_1 and p_2 be two coprime elements of R and a, b be elements of R. We aim at solving the *modular equations*

$$x = a \mod p_1 \qquad \text{and} \qquad x = b \mod p_2. \tag{1}$$

where x is the unknown (the mod notation means that we take the remainder of the Euclidean division).

Recall that since p_1 and p_2 are coprime, their gcd is 1 and, consequently, there exists a couple (u, v) in $R \times R$ such that

$$up_1 + vp_2 = 1.$$

In other words, we have

$$u = p_1^{-1} \mod p_2 \quad \text{and } v = p_2^{-1} \mod p_1.$$

All in all, we easily guess that one solution of the modular equations (1) is

$$x = avp_2 + bup_1 \mod p_1p_2.$$

Observe that one can apply this easily to a variety of problems such as Chinese Remainder Theorem, polynomial interpolation (i.e. Lagrange interpolation) and manipulation of algebraic parametrizations.