2 Basic algebraic operations

2.1 Division with remainder

Let R be a (commutative) ring. Given $a, b \in \mathsf{R}$, with b nonzero, express $a = qb + r$, where $|r| < |b|$. Note that we assume here that our ring R has some reasonable notion of size (and a few other properties we will discuss below).

For example, over $\mathbb{Z}$: Given $a = 32115$ and $b = 123$:

$$\begin{aligned} a &= qb + r \text{ with } |r| < |b| \\ 32115 &= (200 * b) + 7515 \\ &= (200 * b + 60 * b) + 135 \\ &= (200 * b + 60 * b + 1 * b) + 12 \\ &= 261 * b + 12 \end{aligned}$$

Over $\mathbb{Z}[x]$ things are similar (and actually easier). Consider

$$\begin{aligned} a &= 3x^5 + 2x^4 + 7x^3 - 2x^2 + 3x - 1 \\ b &= x^2 + 3x - 1. \end{aligned}$$

Then $a = qb + r$ with $\deg r < \deg b$ as follows:

$$\begin{aligned} 3x^5 + 2x^4 + 7x^3 - 2x^2 + 3x - 1 &= (3x^3)b - 7x^4 + 10x^3 - 2x^2 + 3x - 1 \\ &= (3x^3 b - 7x^2 b) + 31x^3 - 9x^2 + 3x - 1 \\ &= (3x^3 b - 7x^2 b + 31xb) - 102x^2 + 34x - 1 \\ &= (3x^3 b - 7x^2 b + 31xb - 102b) + 340x - 103 \end{aligned}$$

We will need the leading coefficient of b (which we denote $\mathrm{lc}(b)$) to be a *unit* in R. That is, it has an inverse. Why?

Look at some examples in Maple.

$\mathbb{Z}$: Operations are $+$, $-$, $*$, iquo, irem. What about $/$?

$\mathsf{R}[x]$: Operations are $+$, $-$, $*$, quo, rem. Again, look at $/$ (which is not a ring operation, but is available). In Maple, the ring R typically consists of (the field of) rational functions in some other variables, with coefficients in $\mathbb{Q}$ (though other coefficient fields are possible).

2.2 Naive cost model

In lecture we considered the standard "school" algorithms for integer and polynomial arithmetic.

Naive upper bound on cost (up to a multiplicative constant)		
operation	nonzero $a,b \in \mathsf{R}[x]$ $n = \deg a$, $m = \deg b$ operations in R	$a,b \in \mathbb{Z}$ count word operations
$a+b$	$n+m+1$	$\lg a + \lg b$
$a-b$	$n+m+1$	$\lg a + \lg b$
$a \times b$	$(n+1)(m+1)$	$(\lg a)(\lg b)$
$a = qb + r$	$(m+1)(n-m+1)$	$(\lg q)(\lg b)$

Notes:

- Here we define
$$\lg a = \begin{cases} 1 & \text{if } a = 0, \\ 1 + \lfloor \log_2 |a| \rfloor & \text{if } a \neq 0 \end{cases}$$
so that $\lg a$ corresponds to the number of bits required to represent an integer a. Note that $\lg a$ is proportional to the number of words.
- Over $\mathsf{R}[x]$, for $a = qb + r$ we assume that $\mathrm{lc}(b)$ is a unit.
- For $sa + tb = \gcd(a,b)$ we assume R is a field.

2.3 Common operation: reduction modulo many primes

The following operation is common in many algorithms. Given $a \in \mathbb{Z}_{>1}$ and $a < p_1 p_2 p_3 \cdots p_k$, where each p_i is a prime. What is the cost of computing $a \operatorname{rem} p_1$, $a \operatorname{rem} p_2$, $\cdots$, $a \operatorname{rem} p_k$?

$$\begin{aligned} a = 581869302 \qquad P &= 30 \times 17017 \times 12673 \\ &= 2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17 \times 19 \times 23 \times 29 \\ &= 6469693230 \end{aligned}$$

$$a \mapsto (0,0,2,0,3,1,0,2,7,20)$$

- Note that we want a cost estimate that's independent of how P is factored.
- Since there are k remainder operations and all numbers are bounded by P, we know that the cost is bounded by $O(k(\lg P)^2)$ word operations.
- We can do much better by looking more closely at the analysis.

Cost is actually bounded by

$$\sum_{i=1}^{k} c(\lg a/p_i)(\lg p_i) \leq c \sum_{i=1}^{k} (\lg a/p_i)(\lg p_i)$$

We make the simplification $\lg(a/p_i) \leq \lg a \leq \lg P$ and get

$$
\begin{aligned}
\sum_{i=1}^{k} c(\lg a/p_i)(\lg p_i) &\leq c \lg P \sum_{i=1}^{k} \lg p_i \\
&\leq c(1+\log_2 P) \sum_{i=1}^{k} (1+\log_2 p_i) \\
&\leq 4c(\log_2 P) \sum_{i=1}^{k} \log_2 p_i \\
&\leq 4c(\log_2 P)(\log_2 p_1 p_2 \cdots p_k) \\
&= 4c(\log_2 P)(\log_2 P).
\end{aligned}
$$

Thus the total cost is $O((\log P)^2)$, independent of k.

2.4 Greatest Common Divisors

Let $a,b,c \in \mathsf{R}$. Recall the definition of the *Greatest Common Divisor (GCD)*: $c \in \mathsf{R}$ is the GCD of a and b if

(i) $c \mid a$ and $c \mid b$;
(ii) if $d \mid a$ and $d \mid b$, then $d \mid c$ for all $d \in \mathsf{R}$.

The definition of the *least common multiple* (LCM) is similar.

- Note that GCDs need not always exist (it depends on the ring)
- Even if R is a ring in which $\gcd(a,b)$ exists for all $a,b \in \mathsf{R}$, there does not necessarily exist an algorithm for computing the GCD in terms of ring operations in R.
- We can always find GCDs in a *Euclidean Domain*, essentially a ring in which the usual Euclidean algorithm works.
- Many common kinds of rings are Euclidean domains, including $\mathbb{Z}$ and $\mathsf{F}[x]$ for a field F.

Recall a few more definitions. R will always be some (commutative) ring.

- A *zero divisor* in R is an element $a \in \mathsf{R}$ such that there exists a $b \in \mathsf{R} \setminus \{0\}$ with $ab = 0$. For example, in $\mathbb{Z}_6$, we have $2 \times 3 \equiv 0 \bmod 6$, so 2 and 3 are zero divisors.
- A *unit* in R is an element $a \in \mathsf{R}$ such that there exists a $b \in \mathsf{R}$ with $ab = 1$. That is, a has an inverse in R. Units in $\mathbb{Z}$ are ± 1. Units in $\mathsf{F}[x]$ are $\mathsf{F} \setminus \{0\}$.
- An *integral domain* is a ring with no nonzero zero divisors.
- A *field* is an integral domain in which every nonzero element is a unit.
- Elements $a,b \in \mathsf{R}$ are associates if there exists a unit $u \in \mathsf{R}$ such that $a = ub$. For example, 3 and -3 are associates in $\mathbb{Z}$.

We can now define an *Euclidean domain* R as an integral domain with a Euclidean function $\delta : \mathsf{R} \to \mathbb{N} \cup \{-\infty\}$ such that for all $a,b \in \mathsf{R}$ with $b \neq 0$, there exist q,r such that

$$a = qb + r \quad \text{and } \delta(r) < \delta(b).$$

Note: $q = a \operatorname{quo} b$ and $r = a \operatorname{rem} b$.

For example, if $\mathsf{R} = \mathbb{Z}$ the $\delta(a) = |a|$ and $\delta(0) = 0$. The operators quo and rem are *not* unique (think about it). GCD's are only be unique up to associates.

For $\mathsf{R} = \mathsf{F}[x]$ with F a field (like $\mathsf{F} = \mathbb{Q}$ or $\mathsf{F} = \mathbb{Z}/(p)$), we have $\delta(a) = \deg a$ and $\delta(0) = -\infty$. Here quo and rem are unique. GCDs are only unique up to associates.

2.5 The Extended Euclidean Algorithm

We can now take a slightly different look at the extended Euclidean algorithm you probably know well.

Input: $a, b \in \mathsf{R}$ with $b \neq 0$ and R a Euclidean domain.
Output: $s, t, g \in \mathsf{R}$ such that $sa + tb = g$, where $g \in \mathsf{R}$ is a GCD of a and b.

Not only do we compute a GCD, but express it as a linear combination of a and b.

For example, compute $\gcd(91, 63)$:

$$\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}}_{Q_1} \begin{pmatrix} 91 \\ 63 \end{pmatrix} = \begin{pmatrix} 63 \\ 28 \end{pmatrix} \qquad 28 = 91 \operatorname{rem} 63, \quad 1 = 91 \operatorname{quo} 63$$

$$\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}}_{Q_2} \begin{pmatrix} 63 \\ 28 \end{pmatrix} = \begin{pmatrix} 28 \\ 7 \end{pmatrix} \qquad 7 = 63 \operatorname{rem} 28, \quad 2 = 63 \operatorname{quo} 28$$

$$\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & -4 \end{pmatrix}}_{Q_3} \begin{pmatrix} 28 \\ 7 \end{pmatrix} = \begin{pmatrix} 7 \\ 0 \end{pmatrix} \qquad 0 = 28 \operatorname{rem} 7, \quad 4 = 28 \operatorname{quo} 7$$

We note that

$$Q_3 Q_2 Q_1 = \begin{pmatrix} -2 & 3 \\ 9 & -13 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -2 & 3 \\ 9 & -13 \end{pmatrix} \begin{pmatrix} 91 \\ 63 \end{pmatrix} = \begin{pmatrix} 7 \\ 0 \end{pmatrix} \implies -2 \times 91 + 3 \times 63 = 7$$

Now we can formalize this algorithm over a Euclidean domain R.

Input: $a, b \in \mathsf{R}$, $b \neq 0$, and $\delta(a) \geq \delta(b)$.

- Let $r_0 = a$ and $r_1 = b$; Let $R_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$;
- For $i = 1, 2, \ldots$

- Compute q_i and r_{i+1} such that

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

  where $\delta(r_{i+1}) < \delta(r_i)$.
- We have

$$\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix}}_{Q_i} \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix}.$$

  Let $R_i := Q_i R_{i-1}$.
- Stop at smallest $i = \ell$ such that $r_{\ell+1} = 0$.

Why do we know it will stop? Because $\delta(r_1) > \delta(r_2) > \delta(r_3) > ... > \delta(r_\ell) > 0$ and $r_{\ell+1} = 0$. At this point we know

$$R_\ell \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = Q_\ell Q_{\ell-1} \cdots Q_2 Q_1 \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} s_\ell & t_\ell \\ s_{\ell+1} & t_{\ell+1} \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} r_\ell \\ 0 \end{pmatrix},$$

so $s_\ell r_0 + t_\ell r_1 = r_\ell$.

Claim: r_ℓ is a GCD of r_0 and r_1.

Proof: Need to show

(i) $r_\ell \mid r_0$ and $r_\ell \mid r_1$;
(ii) if $d \mid r_0$ and $d \mid r_1$ then $d \mid r_\ell$ for all $d \in \mathsf{R}$.

For part (i), observe that each Q_i is invertible over R:

$$\underbrace{\begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix}}_{Q_i^{-1}} \underbrace{\begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix}}_{Q_i}$$

This implies that each R_i is invertible over R:

$$R_i^{-1} = Q_1^{-1} Q_2^{-1} \cdots Q_i^{-1}$$

and in particular

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \underbrace{\begin{pmatrix} * & * \\ * & * \end{pmatrix}}_{R_\ell^{-1}} \begin{pmatrix} r_\ell \\ 0 \end{pmatrix}.$$

This shows (i). Why?

Part (ii) follows since $r_\ell = s_\ell r_0 + t_\ell r_1$. Why?

Cost analysis.

Consider $\mathsf{R} = \mathsf{F}[x]$ ($\mathsf{R} = \mathbb{Z}$ is similar, but requires more fiddling). Assume $\deg r_0 \geq \deg r_1$.

Cost of computing $(q_i, r_{i+1})_{1\leq i\leq \ell}$**.**

Q: How many divison steps ℓ?

A: $\ell \leq \deg r_1$ since $-\infty = \deg r_{\ell+1} < \overbrace{\deg r_\ell}^{\geq 0} < \cdots < \deg r_2 < \deg r_1$.

Division with remainder of r_{i-1} by r_i costs $c(\deg r_i + 1)(\deg q_i + 1)$ operations from F for some constant c.

Key observation:

$$\sum_{i=1}^{\ell} \deg q_i = \sum_{i=1}^{\ell} (\deg r_{i-1} - \deg r_i) = r_0 - r_\ell \leq r_0.$$

Total cost, in terms of operations in F, is thus at most

$$\begin{aligned}
&\sum_{i=1}^{\ell} c(\deg r_i + 1)(\deg q_i + 1) \\
&\leq c(\deg r_1 + 1)\sum_{i=1}^{\ell}(\deg q_i + 1) \ \ \text{(using the fact that } \deg r_i \leq \deg r_1\text{)} \\
&\leq c(\deg r_1 + 1)(\deg r_0 + \ell) \\
&\leq c(\deg r_1 + 1)(\deg r_0 + \deg r_1) \\
&= O((\deg r_0)(\deg r_1)) \quad \text{operations in } \mathsf{F}.
\end{aligned}$$

We can now extend our naive cost table to include gcd.

Naive upper bound on cost (up to a multiplicative constant)		
operation	nonzero $a, b \in \mathsf{R}[x]$ $n = \deg a$, $m = \deg b$ operations in R	$a, b \in \mathbb{Z}$ count word operations
$a + b$	$n + m + 1$	$\lg a + \lg b$
$a - b$	$n + m + 1$	$\lg a + \lg b$
$a \times b$	$(n+1)(m+1)$	$(\lg a)(\lg b)$
$a = qb + r$	$(m+1)(n-m+1)$	$(\lg q)(\lg b)$
$sa + tb = \gcd(a, b)$	$(n+1)(m+1)$	$(\lg a)(\lg b)$

Notes:

- Here we define
$$\lg a = \begin{cases} 1 & \text{if } a = 0, \\ 1 + \lfloor \log_2 |a| \rfloor & \text{if } a \neq 0 \end{cases}$$
so that $\lg a$ corresponds to the number of bits required to represent an integer a. Note that $\lg a$ is proportional to the number of words.
- Over $\mathsf{R}[x]$, for $a = qb + r$ we assume that $\mathrm{lc}(b)$ is a unit.