Notes for Seminar ”Computational Number Theory”

Lars Quentin

1 Notes ”Modern Computer Algebra”

1.1 Fundamental Algorithms

1.1.1 Multiplication

Following our program, we first consider the product $c = a \cdot b$ of two polynomials $a, b \in R[x]$. Its coefficients are

$$c_k = \sum_{\substack{0 \le i \le n \\ 0 \le j \le m \\ i+j=k}} a_i b_j$$

for $0 \le k \le n + m$.
Algorithm 1:
Input: The coefficients of $a = \sum_{0 \le i \le n} a_i x^i$ and $b = \sum_{0 \le i \le m} b_i x^i$ in $R[x]$, where R is a (commutative) ring.
Output: The coefficients of $c = a \cdot b \in R[x]$

```
1. for i=0,...,n do d_i <- a_i x^i * b
2. return c = sum(d_i)
```

To understand the complexity let's look at an example:

$$\begin{aligned}
&(a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1)(b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1) \\
=&\sum_{i=1}^{4} a_i x^i \cdot (b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1) \\
=&\sum_{i=1}^{4} a_i x^i b_4 x^4 + a_i x_i b_3 x^3 + a_i x_i b_2 x^2 + a_i x_i a_1 x^1
\end{aligned}$$

We need n^2 multiplications since we multiply everything by everything.
We need $(n-1)$ additions per iteration, thus $(n-1) \cdot n \approx n^2$ in total.
This is because of the following:
The multiplication of $a_i x^i \cdot b$ is realized as the multiplication of each b_j by a_i plus a shift by i places. The variable x serves us just as a convinient way to write polynomials, and there is no arithmetic involved in ”multiplying” by x or any power of it.
See the formal definition of polynomial rings for more intuition.

1.2 Fast Multiplication

1.2.1 Karatsuba's multiplication algorithm

We start with the multiplication of two polynomials $f, g \in R[x]$ of degree less than n over a Ring R. As usual, "ring" means a commutative ring with 1.

Definition. A *ring* $(R, +, \cdot)$ is a set R with 2 binary operations $+$ and $\cdot$ which satisfy the following axioms.

1. $(R, +)$ is an abelian group under addition, meaning that:
   - $+$ is associative.
   - $+$ is commutative.
   - There is an neutral element $0 \in R$.
   - Every element as an additive inverse.
2. $(R, \cdot)$ is a semigroup under multiplication, meaning that $\cdot$ is associative.
3. Multiplication is distributive with respect to addition.

Definition. An *commutative ring with* 1 $(R, +, \cdot)$ is a ring that satisfies the following:

1. $(R, \cdot)$ is not only a semigroup but also a monoid, meaning that:
   - $(R, \cdot)$ is associative (i.e. a semigroup).
   - There exists a neutral element $1 \in R$ over multiplication (i.e. the multiplicative identity).
2. $(R, \cdot)$ is commutative.

A ring with 1 is also called a unitary ring.

Definition. Let $D \subseteq \mathbb{Z}, f, g : D \to \mathbb{C}$.
A *discrete convolution* of f and g is defined as

$$(f * g)(n) = \sum_{m=-\infty}^{\infty} f(m)g(n-m)$$

When g has finite support (i.e. has finitely many elements in the domain producing a nonzero value) over the set $\{-M, -M+1, \ldots, M\}$ then it can be simplified to

$$(f * g)(n) = \sum_{m=-M}^{M} f(n-m)g(m)$$

Definition. A *polynomial ring* $R[X]$ is a commutative ring $(R^{(\mathbb{N}_0)}, +, \cdot)$ with 1 defined as

- $R^{(\mathbb{N}_0)}$ is the set
  $$R^{(\mathbb{N}_0)} := \{(a_i)_{i\in\mathbb{N}_0} | a_i \in R, a_i = 0 \text{ for almost all } i\}$$
  "Almost all" means that it holds for all but a negligible amount.
- $+$ is defined as the compontentwise addition, meaning that
  $$(a_i)_{i\in\mathbb{N}_0} + (b_i)_{i\in\mathbb{N}_0} := (a_i + b_i)_{i\in\mathbb{N}_0}$$

- $\cdot$ is defined as the discrete convolution, meaning that

$$(a_i)_{i\in\mathbb{N}_0} \cdot (b_i)_{i\in\mathbb{N}_0} := \left(\sum_{i=0}^{k} a_i b_{k-i}\right)_{k\in\mathbb{N}_0} = \left(\sum_{i+j=k} a_i b_j\right)_{k\in\mathbb{N}_0}$$

  Since we have a finite sequences, it is also called the *Cauchy Product*

- Let X be defined such that
  - $X \in R^{(\mathbb{N}_0)}$ is defined as
  $$X = X^1 := (0, 1, 0, 0, \dots)$$
  - $1 \in R^{(\mathbb{N}_0)}$ is defined as
  $$1 := X^0 = (1, 0, 0, \dots)$$
  - Every power $X^k \in R^{(\mathbb{N}_0)}$, $k \in \mathbb{N}_0$ is defined as
  $$X^k := \underbrace{X \cdot X \cdot \dots \cdot X}_{k \text{ times}} = (\underbrace{0, 0, \dots, 0}_{k \text{ zeros}}, 1, 0, 0, \dots)$$

  With X defined we can write any polynomial $f \in R[X]$ as

$$f = \sum_{i=0}^{n} a_i X_i$$

  where n is the largest non-zero index.

Note. Let $p, q \in R[X]$ be 2 polynomials. When written in polynomial notation the multiplication is equivalent to the intuitive multiplication of both terms, i.e.

$$p + q = s_0 + s_1 \cdot X + \dots + s_{(\deg(p)+\deg(q))} X^{(\deg(p)+\deg(q))}$$

where

- $s_i = p_0 q_i + p_1 q_{i-1} + \dots + p_i q_0$
- $\deg(p)$ notes the degree of p, i.e. it's largest exponent

If $f_i, g_j, h_k \in R$ are the coefficients of f, g and $h = fg$, respectively, the classical multiplication algorithm uses $O(n^2)$ operations in R to compute the h_k from the f_i and g_j:
n^2 multiplications $f_i g_j$ plus $(n-1)^2$ additions for all $h_k = \sum_{i+j=k} f_i g_j$.
For instance, multiplying $(ax + b)(cx + d) = acx^2 + (ad + bc)x + bd$ uses four multiplications ac, ad, bc, bd.

Suprisingly, there is an easy method of doing better. We compute ac, bd, $u = (a + b) + (c + d)$ and $ad + bc = u - ac - bd$, with three multiplications and four additions and subtractions. The total has increased to seven operations, but a recusrive application will drastically recuce the overall cost (See Figure 8.2). To explain the general approach, we assume that $n = 2^k$ for some $k \in \mathbb{N}$ (otherwise we can just fill up with zeros), set $m = n/2$, and rewrite f and g in the form $f = F_1 x^m + F_0$ with $F_0, F_1 \in R[x]$ of degree less than m and similarly $g = G_1 x^m + G_0$. (If $\deg(f) < n + 1$, then some of the top coefficients are zero.) Now $fg = F_1 G_1 x^n + (F_0 G_1 + F_1 G_0) + F_0 G_0$. In this form, multiplication of f and g has been reduced to four multiplications of polynomials of degree less than m. Multiplication by a power of x does not count as a multiplication, since it corresponds merely to a shift of the coefficients.

So far we have not really archieved anything. But the method by Karatsuba in Karatsuba & Ofman (1962), explained for $n = 1$ above, shows how this expression for fg can be rearranged to reduce the number of multiplications of the smaller polynomials at the expense of increasing the number of additions. Since multiplication is slower than addition, a saving is obtained when n is sufficiently large. We rewrite this product as

$$fg = F_1 G_1 x^n + ((F_0 + F_1)(G_0 + G_1) - F_0 G_0 - F_1 G_1) x^m + F_0 G_0$$

This expression shows that multiplication of f and g requires only three multiplications of polynomials of degree less than m and some additions. The same method is now applied recursively to the smaller multiplications. If $T(n)$ denotes the time necessary to multiply two polynomials of degree less than n, then

$$T(2n) \leq 3T(n) + cn$$

for some constant c. The linear term comes from the observation that addition of two polynomials of degree less than d can be done with d operations in R.