Chapter 2 Fundamental algorithms

2.1 Algebraic structures

Basic requirements

- Precise representation of algebraic structures (integers, polynomials,...)
- Precise arithmetic with algebraic structures (GCD, factor,...)
- Other analytical operations with these structures (differentiation, integration)

Number domains

- $\{\mathbb{Z}, +, -, \times\}$ integers with operations of addition, subtraction, multiplication;
- $\{\mathbb{Z}_m, +_m, -_m, \times_m\}$ integers under modular arithmetic where m is a positive integer;
- $\{\mathbb{Q}, +, -, \times, /\}$ rational numbers with operations of addition, subtraction, multiplication and division;

- $\{\{x = a + ib, a, b \in \mathbb{Z}\}, +, -, \times\}$ Gaussian integers, i.e., complex numbers which have integer real and imaginary parts;

- $\{\mathbb{Q}(\alpha), +, -, \times, /\}, p(\alpha) = 0$ algebraic extension field; the algebraic number is defined by the polynomial with integer coefficients which can be represented precisely; the algebraic number is a root of the polynomial p;

- $\{\mathbb{R}^n_f, +, -, \times, /\}$ floating point numbers with a precision of decimal digits where is an arbitrary positive integer–it can be 100 or 1000.

Algebraic expression domains

- ring of polynomials $O[x_1, x_2, \ldots, x_n]$ in n variables with operations of addition, subtraction, multiplication and exponentiation by a nonnegative integer; polynomial coefficients can be numbers from a number domain.

- power series, other kinds of series;

- rational functions $K(x_1, \ldots, x_n)$ (extension of polynomials by the operation of division) with operations of addition, subtraction, multiplication, division and exponentiation by an integer;

- extension of rational functions by radicals (rational exponents), with operations of addition, subtraction, multiplication, division and exponentiation by a rational number;

- algebraic functions $\{y_i, p_i(X, y_1, \ldots, y_m) = 0, i = 1, \ldots, m\}$ implicitly defined by polynomials with integer coefficients p_i which depend on algebraic functions y_i and variables from $X = \{x_1, \ldots, x_n\}$, e.g., the algebraic function defined by the polynomial $x^2 + y^2 - 1$.

- elementary transcendental functions $\exp, \log$, extension of rational functions by elementary transcendental functions;
- transcendental functions: e.g., $\sin, \cos$; extension of rational functions by transcendental functions
- matrix rings
- differential fields
- finite groups

2.2.1 Representation and addition of integers

Assume we have a 64-bit processor:

1. Single precision integer is represented by 64-bit words: $[0, 2^{64} - 1]$

2. Multiprecision integer is represented by an array of 64-bit words, where the first one encodes the sign of the integer and the length of the array:
$$a = (-1)^s \sum_{0 \le i \le n} a_i \cdot 2^{64i}, \tag{1}$$

where

$$s \in \{0, 1\}, \qquad 0 \leq n + 1 \leq 2^{63},$$
$$a_i \in \{0, \ldots, 2^{64} - 1\}.$$

Standard representation array ($a_n \neq 0$ if $a \neq 0$):

$$s \cdot 2^{63} + n + 1, a_0, \ldots, a_n$$

Length $\lambda(a) = \lfloor \log_{2^{64}} |a| \rfloor + 1 = \lfloor \frac{\log_2 |a|}{64} \rfloor + 1 = O(\log_2 |a|)$

1. Addition of single precision integers:

$$a + b + \gamma = \gamma^* \cdot 2^{64} + c$$

where $\gamma, \gamma^* \in \{0, 1\}$ are carry flags.

2. Addition of multiprecision integers $a = \sum_{0 \le i \le n} a_i 2^{64i}$, $b = \sum_{0 \le i \le m} b_i 2^{64i}$:

$$c = \sum_{0 \le i \le \max\{n,m\}} (a_i + b_i) 2^{64i}$$

Algorithm 2.1 Addition of multiprecision integers.

Input: $a = (-1)^s \sum_{0 \le i \le n} a_i 2^{64i}$, $b = (-1)^s \sum_{0 \le i \le n} b_i 2^{64i}$

Output: $c = (-1)^s \sum_{0 \le i \le n+1} c_i 2^{64i}$ such that $c = a + b$.

1. $\gamma_0 \longleftarrow 0$, for $i = 0, \ldots, n$ do
   $c_i \longleftarrow a_i + b_i + \gamma_i, \gamma_{i+1} \longleftarrow 0$
   if $c_i \ge 2^{64}$ then $c_i \longleftarrow c_i - 2^{64}, \gamma_{i+1} \longleftarrow 1$

2. $c_{n+1} \longleftarrow \gamma_{n+1}$,
   return $(-1)^s \sum_{0 \le i \le n+1} c_i 2^{64i}$

“Big Oh” notation

In many situations in algorithmics it is convenient to express the cost of an algorithm only “up to a constant factor” or “rate of growth”.

Let $g : \mathbb{N} \longrightarrow \mathbb{R}$ be eventually positive. Then $O(g)$ is the set of all eventually positive functions $f : \mathbb{N} \longrightarrow \mathbb{R}$ for which there exist constant number $c \in \mathbb{N}$ such that $f(n)$ and $g(n)$ are defined and $f(n) \leq cg(n)$ for all $n \geq \mathbb{N}$.

The addition of two n integers can be done in time $O(n)$, or cost $O(n)$ word operations. Adding two multiprecision integers is a linear operation in the sense that doubling the input size also approximately doubles the running time.

If $f(n)$ denotes the cost for matrix multiplication as above and $g(n) = n^3$, we may write $f \in O(g)$. In the literature, one often finds $f = O(g)$ or $f(n) = O(n^3)$.

Two advantages from “big Oh” notation:

- a shorter and more intuitive notation
- independence of particular machine

The cost measure that is widely used for algorithms dealing with integers is the number of bit operation which can be rigorously defined as the number of steps of a Turing or register machine or the number of gates of a Boolean circuit implementing the algorithm.

2.2.2 Representation of polynomials

- Prefix representation
- Dense representation
- Sparse representation
- Recursive representation

Prefix representation

Polynomials are represented by lists using e.g.,

- the prefix operators PLUS for addition
- DIFFERENCE for subtraction (or unary MINUS)
- TIMES for multiplication
- EXPT for exponentiation;

The first element of a list is the prefix operator and the rest of the elements are its arguments. For example

$$4x^3 + 2x - 5$$

is represented by the list

(PLUS (TIMES 4 (EXPT X 3)) (TIMES 2 X) (MINUS 5))

This representation can be used for arbitrary algebraic expressions. However, algorithms using this representation are not particularly fast.

Dense representation

A polynomial in one variable x

$$a = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = \sum_{0 \leq i \leq n} a_i x^i$$

is represented by the array of all its coefficients $a_i, i = 0, \ldots, n$.

In this representation, one needs for the polynomial $2x^{1000} - 1$ to store 1001 coefficients even though only 2 nonzero coefficients are really necessary; for polynomials in more variables, the situation is even worse.

Sparse representation

A polynomial in one variable x

$$a = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = \sum_{0 \leq i \leq n} a_i x^i$$

is represented by pairs of corresponding exponents and coefficients (i, a_i) for each term of the polynomial, i.e., all coefficients a_i in this representation are nonzero.

In this representation, the polynomial $2x^{1000} - 1$ would be represented by the list ((1000 2) (0 -1))

Recursive representation

- Recursive representation is sparse representation with exponent ordering used to represent multivariate polynomials;

- For recursive representation, an ordering of variables has to be chosen, e.g., alphabetically;

- The variable which is chosen first in the ordering is called the main variable of the polynomial;

- Coefficients of the powers of the main variable will be polynomials in the other variables; For simplicity, we will consider a polynomial in two variables, with variable ordering $x > y$, so that the main variable is x

$$\sum_{i=0}^{n}\sum_{j=0}^{m} a_{i,j}x^i y^j = \sum_{i=0}^{n} c_i x^i, \text{where } c_i = \sum_{j=0}^{m} a_{i,j} y^j$$

- The polynomial $x^3(y^2+8y) - x^2(7y+3) - 5xy + 4$ is then represented by the list

$$((x\ 3((y\ 2\ 1)(y\ 1\ 8)))(x\ 2((y\ 1\ 7)(y\ 0\ 3)))(x\ 1((y\ 1\ 5)))(x\ 0\ 4))$$

2.2.3 Addition and multiplication of polynomials

Assume we have a commutative ring or field R, such as $\mathbb{Z}$ or $\mathbb{Q}$, a polynomial $a \in R[x]$ in x over R is:

$$a = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = \sum_{0 \le i \le n} a_i x^i. \tag{2}$$

where $a_n \neq 0$, n is the degree of a. $a_n = \mathrm{lc}(a)$ is the leading coefficient. If $\mathrm{lc}(a) = 1$, then a is monic. Take $\deg 0 = -\infty$.

Algorithm 2.2 Addition of polynomials.

Input: $a = \sum_{0 \le i \le n} a_i x^i$, $b = \sum_{0 \le i \le n} b_i x^i$ in $R[x]$.

Output: The coefficients of $c = a + b \in R[x]$.

1. for i $=$0, . . . , n do $c_i \longleftarrow a_i + b_i$
2. return $c = \sum_{0 \le i \le n} c_i x^i$

The arithmetic operations($+$,-,$\times$, $/$) in the addition algorithm is $O(n)$.

Algorithm 2.3 Multiplication of polynomials.

Input: $a = \sum_{0 \le i \le n} a_i x^i$, $b = \sum_{0 \le i \le m} b_i x^i$ in $R[x]$.

Output: The coefficients of $c = a \cdot b \in R[x]$.

1. for $i = 0, \ldots, n$ do $d_i \longleftarrow a_i x^i \cdot b$

2. return $c = \sum_{0 \le i \le n} d_i$.

The cost is $2nm + n + m + 1$ or $O(mn)$ arithmetic operations, or in quadratic time.

Algorithm 2.4 Multiplication of multiprecision integers .

Input: $a = (-1)^s \sum_{0 \le i \le n} a_i 2^{64i}$, $b = (-1)^t \sum_{0 \le i \le m} b_i 2^{64i}$

Output: $c = a \cdot b$.

1. for $i = 0, \ldots, n$ do $d_i \longleftarrow a_i 2^{64i} \cdot |b|$

2. return $c = (-1)^{s+t} \sum_{0 \le i \le n} d_i$

The cost is $O(mn)$.

2.2.4 Division with remainder

1. Given integers a, b with b nonzero, find a quotient q and a remainder r both integers so that

$$a = q \cdot b + r, \quad |r| < |b|$$

2. Given polynomials $a, b \in R[x]$ with b nonzero, find $q, r \in R[x]$ so that:

$$a = q \cdot b + r, \quad \deg r < \deg b.$$

Algorithm 2.5 Polynomial division with remainder

Input: $a = \sum_{0 \le i \le n} a_i x^i$, $b = \sum_{0 \le i \le m} b_i x^i$ in $R[x]$, b_m a unit, $n \ge m \ge 0$.

Output: $q, r \in R[x]$ with $a = q \cdot b + r$ and $\deg r < m$.

1. $r \longleftarrow a$, for $i = n - m, n - m - 1, \ldots, 0$ do
   if $\deg r = m + i$ then $q_i \longleftarrow \mathsf{lc}(r)/b_m$, $r \longleftarrow r - q_i x^i b$
   else $q_i \longleftarrow 0$

2. return $q = \sum_{0 \le i \le n-m} q_i x^i$ and r.

The cost is $(2m+1)(n-m+1)$ or $O(mn)$ arithmetic operations.

For the division of two integers, we refer to the comprehensive discussion by Knuth.