Application of Discrete Models

Adam Zlehovszky

October 24, 2023

1 Representation of Integers

1.1 Euclidean division

If $a, b \in \mathbb{Z}$ with $b \neq 0$ then $\exists! q, r \in \mathbb{Z}$ such that $a = qb + r$ where $0 \leq r < |b|$. This is the *Euclidean division* or *long division* of the *dividend* a with the *divisor* b. The results of the division are the *quotient* q and the *remainder* r. The standard notation for the remainder is $a \bmod b$. In algorithmic setting we use $q, r \leftarrow \mathbf{divmod}\,(a, b)$.

1.2 Number systems

Let $1 < b \in \mathbb{Z}$ be the *base* of the *number system*. For each $0 \leq n \in \mathbb{Z}$ there exists a unique $1 \leq d \in \mathbb{Z}$ and a unique set of *digits* $0 \leq n_1, n_2, \ldots, n_{d-1} < b$ all integers, such that

$$n = \sum_{k=0}^{d-1} n_k b^k.$$

If $n = 0$, then $d = 1$ and $n_0 = 0$. Otherwise $d = \lfloor \log_b n \rfloor + 1$ and we can extract the digits of n with long division, since

$$\begin{aligned} n &= n_{d-1} b^{d-1} + \cdots + n_2 b^2 + n_1 b + n_0 \\ &= \left(n_{d-1} b^{d-2} + \cdots + n_2 b + n_1\right) b + n_0 \end{aligned}$$

where the quotient $n_{d-1} b^{d-2} + \cdots + n_2 b + n_1$ is a $d-1$ digit number and n_0 is the extracted digit.

We call n_0 the *least significant digit* and n_{d-1} the *most significant digit*. The storage order of digits is called *little endian* if we start at the least significant digits and move towards the most significant one. Otherwise it is called *big endian*.

1.3 Operations on Integers

1.3.1 Addition

Let us assume that we have two unsigned integers stored as digits in a number system with base b:

$$n^{(i)} = \sum_{k=0}^{d^{(i)}-1} n_k^{(i)} b^k,$$

for $i = 1, 2$. The following algorithm computes the digits of the sum $s = n^{(1)} + n^{(2)} = \sum_{k=0}^{d^{(s)}-1} s_k b^k$:

Algorithm 1 Standard addition

procedure $\textsc{StandardAddition}(n^{(1)},\, n^{(2)})$
  $d^{(s)} \leftarrow \max\left(d^{(1)}, d^{(2)}\right)$
  $c \leftarrow 0$
  for $k = 0, \ldots, d^{(s)} - 1$ **do**
    $c,\ s_k \leftarrow \textbf{divmod}\left(n_k^{(1)} + n_k^{(2)} + c, b\right)$
  end for
  return s
end procedure

In Algorithm 1 we assume that $n_k^{(i)} = 0$ if $k \geq d^{(i)}$ for $i = 1, 2$. The time complexity of the standard addition is $O\left(d^{(s)}\right)$.

1.3.2 Multiplication

Let $n^{(i)}$'s defined same as above for $i = 1, 2$. We will compute the digits of the product $p = n^{(1)} \cdot n^{(2)} = \sum_{k=0}^{d^{(p)}-1} p_k b^k$ with the naive multiplication method:

The time-complexity of Algorithm 2 is $O(d^{(1)} \cdot d^{(2)}) = O(d^2)$, where $d = \max\left(d^{(1)}, d^{(2)}\right)$.

Karatsuba's idea for faster multiplication can be demonstrated on two-digit numbers. Let

$$x = x_1 b + x_0, \text{ and } y = y_1 b + y_0$$

with $0 \leq x_i, y_i < b$ integers. Naive multiplication of x and y is

$$\begin{aligned} z = xy &= (x_1 b + x_0)(y_1 b + y_0) \\ &= x_1 y_1 b^2 + (x_1 y_0 + x_0 y_1) b + x_0 y_0 \\ &= z_1 b^2 + z_1 b + z_0. \end{aligned}$$

This is 4 multiplication and 1 addition.

Algorithm 2 Naive multiplication

procedure NaiveMultiplication($n^{(1)}$, $n^{(2)}$)
  $d^{(p)} \leftarrow d^{(1)} + d^{(2)}$
  for $k = 0, \ldots, d^{(p)} - 1$ **do**
    $p_k \leftarrow 0$
  end for
  for $j = 0, \ldots, d^{(2)} - 1$ **do**
    $c \leftarrow 0$
    for $i = 0, \ldots, d^{(1)} - 1$ **do**
      $c,\ p_{i+j} \leftarrow \textbf{divmod}\left(p_{i+j} + n_i^{(1)} n_j^{(2)} + c, b\right)$
    end for
    $p_{d^{(1)}+j} \leftarrow c$
  end for
  return p
end procedure

Now we can express

$$\begin{aligned} z_1 &= x_1y_0 + x_0y_1 \\ &= x_1y_0 + x_0y_1 - x_1y_1 + x_1y_1 - x_0y_0 + x_0y_0 \\ &= (x_1 + x_0)\, y_1 + (x_1 + x_0)\, y_0 - x_1y_1 - x_0y_0 \\ &= (x_1 + x_0)\,(y_1 + y_0) - x_1y_1 - x_0y_0 \\ &= (x_1 + x_0)\,(y_1 + y_0) - z_2 - z_0. \end{aligned}$$

This is 3 multiplication and 3 additions. By extending this idea to more than two digits recursively, the multiplication algorithm performs $O(d^{\log_2 3}) \approx O(d^{1.58})$ single-digit multiplication.

Fast Fourier Transform based algorithms can achieve $O(d \log d)$ complexity.

1.4 Exponentiation

We want to compute x^n for some $1 \leq n \in \mathbb{Z}$ and x that has multiplication as an operation.

Naive exponentiation By repeated multiplication, we can compute

$$x^n = \underbrace{x \cdot x \cdots x}_{n \text{ times}}.$$

This method requires $n - 1$ multiplications.

Repeated squaring If $n = 2^s$ for $0 < s \in \mathbb{Z}$, then

$$x^{(2^s)} = \left(x^2\right)^{\left(2^{s-1}\right)}.$$

This way we can compute x^n with $\log_2 n = s$ multiplications with the algorithm below:

Algorithm 3 Repeated squaring

procedure REPEATEDSQUARING(x, s)
    $y \leftarrow x$
    for $k = 0, \ldots, s-1$ **do**
        $y \leftarrow y^2$
    end for
    return y
end procedure

Fast exponentiation If we write $n = \sum_{k=0}^{d-1} n_k 2^k$ in binary, then

$$\begin{aligned} x^n &= x^{\left(\sum_{k=0}^{d-1} n_k 2^k\right)} \\ &= \prod_{k=0}^{d-1} x^{\left(n_k 2^k\right)} \\ &= \prod_{k=0}^{d-1} x^{\left(2^k\right)^{n_k}}. \end{aligned}$$

Since $y^{n_k} = y$ if $n_k = 1$ and $y = 1$ otherwise, we arrive at the following algorithm:

Algorithm 4 Fast exponentiation

procedure FASTEXP(x, n)
    $y \leftarrow 1$
    while $n > 0$ **do**
        $n, r \leftarrow$ **divmod** $(n, 2)$
        if $r = 1$ **then**
            $y \leftarrow y \cdot x$
        end if
        $x \leftarrow x^2$
    end while
    return y
end procedure

This algorithm requires $O(\log_2 n)$ multiplication.

2 Number Theory and its Applications

2.1 Divisibility

If $aq = b$, then we say that a is a *divisor* of b, or b is a *multiple* of a. The notation is $a \mid b$. Otherwise $a \nmid b$. If $a \mid b$, then long division has remainder of 0 and in case of integers $\frac{b}{a}$ is an integer as well.

The following properties are natural consequences of the definition.

1. For every a, we have that $a \mid a$.
2. $a \mid 0$, for every a.
3. If $0 \mid a$, then $a = 0$.
4. If $a \mid b$ and $b \mid c$, then $a \mid c$.
5. If $a \mid b$ and $c \mid d$, then $ab \mid cd$.
6. If $a \mid b$, then $ac \mid bc$ for every c.
7. If $ac \mid bc$ and $c \neq 0$, then $a \mid b$.
8. If $a \mid b_i$ for some finite indices i, then $a \mid \sum_i c_i b_i$ for every c_i.

If $\varepsilon \mid a$ for every a, then we call ε a *unit element.* The unit elements of $\mathbb{Z}$ are ± 1.

If $a \mid b$ and $b \mid a$ and $a \neq b$, then we call a and b *associated elements.* Two elements a and b are associated if and only if $a = \varepsilon b$ for some unit element ε. Consequently, a and b are associated integers if and only if $|a| = |b|$.

Let $p \neq 0$ be a non-unit element. We say that p is an *irreducible element* if $p = ab$ implies that a or b is an associated element of p (and the other is a unit). If an element is not irreducible, then it is *composite.* We call p a *prime element* if $p \mid ab$ implies that $p \mid a$ or $p \mid b$. If p is an irreducible element, then it is also a prime element. In case of integers, the reverse is also true, i.e. every prime element is irreducible.

Theorem 1 (The Fundamental Theorem of Arithmetic)**.** *If $a \neq 0$ is not a unit element, then it is a product of irreducible elements. The product is unique (up to ordering and up to multiplication with unit elements).*

If $1 < n \in \mathbb{Z}$, then the *canonical form* of

$$n = p_1^{\alpha_1} \cdots \; p_r^{\alpha_r}$$

where p_i's are different prime numbers (positive prime elements of $\mathbb{Z}$) and $\alpha_i > 0$ for all $i = 1, \dots, r$.

Theorem 2 (Euclid)**.** *There are infinitely many prime numbers.*

Proof. Let us assume that there are only finite many primes $p_1, \dots, p_n$. In this case the long division of $n = p_1 \cdots p_n + 1$ with p_i yields a remainder of 1 for every prime. This means that n does not have canonical form, which contradicts Theorem 1. $\square$

Theorem 3 (Distribution of prime numbers)**.** *The following statements illustrate some properties of the distribution of prime numbers:*

1. *If $N > 1$, then $\exists a > 2$ such that $a+1, a+2, \dots, a+N$ are all composite numbers.*

2. *For every $M > 2$, there is a prime number between M and $2M$.*

Let $\pi(x)$ denote the number of positive prime numbers below x.

Theorem 4 (Prime Number Theorem)**.** *An approximation of $\pi(x)$ is $\frac{x}{\ln x}$. In other words*

$$\lim_{x \to +\infty} \frac{\pi(x)}{x/\ln x} = 1.$$

2.2 Greatest common divisor

The *greatest common divisor* of $a_1, \dots, a_n$ elements is d, if

- $d \mid a_i$ for all i, i.e. d is a common divisor of the elements and
- if $d' \mid a_i$ for all i, then $d' \mid d$ that is d is maximal with respect to divisibility.

From the definition, it is clear that if d is a greatest common divisor and ε is a unit element, then εd is also a greatest common divisor. We usually fix a greatest common divisor for integers by nominating the positive one. We will use the notation $\gcd(a_1, \dots, a_n)$.

From the definition we can see that

1. $\gcd(a, 0) = a$,
2. $\gcd(0, 0) = 0$,
3. $\gcd(a, b) = \gcd(b, a)$,
4. $\gcd(a, b) = \gcd(a - b, b)$ or in general $\gcd(a - qb, b) = \gcd(a, b)$ for any q. Specifically $\gcd(a \bmod b, b) = \gcd(a, b)$.

From the last property we have that

$$\begin{aligned}
\gcd(a, b) &= \gcd(a \bmod b, b) \\
&= \gcd\left(a \bmod b, b \bmod (a \bmod b)\right) \\
&= \gcd\left((a \bmod b) \bmod b \bmod (a \bmod b), b \bmod (a \bmod b)\right).
\end{aligned}$$

Let us define the recurrence relation $r_0 = a$, $r_1 = b$ and $r_{n+2} = r_n \bmod r_{n+1}$. In this case the equations above become

$$\begin{aligned}\gcd(r_0, r_1) &= \gcd(r_2, r_1)\\ &= \gcd(r_2, r_3)\\ &= \gcd(r_4, r_3).\end{aligned}$$

We can swap the arguments of gcd, so

$$\begin{aligned}\gcd(r_0, r_1) &= \gcd(r_1, r_2)\\ &= \gcd(r_2, r_3)\\ &= \gcd(r_3, r_4)\end{aligned}$$

The sequence in strictly decreasing, that is $r_n > r_{n+1}$ and $r_n \geq 0$ from $n = 2$. This means that there is an index N such that $r_N = 0$, but $r_{N-1} > 0$. From the properties and definition of recurrence relation it is clear, that

$$\gcd(a, b) = \gcd(r_{N-1}, 0) = r_{N-1}.$$

The argument above gives the *Euclidean algorithm* in recursive form:

Algorithm 5 Recursive Euclidean algorithm

procedure GCD(a, b)
  if $b = 0$ **then**
    return a
  end if
  return GCD(b, $a \bmod b$)
end procedure

We can transform the recursion into a loop. At any given step, we only need r_n and r_{n+1} to produce r_{n+2}.

Algorithm 6 Iterative Euclidean algorithm

procedure GCD(a, b)
  $r_{\text{old}},\ r_{\text{new}} \leftarrow a,\ b$
  while $r_{\text{new}} \neq 0$ **do**
    $r_{\text{old}},\ r_{\text{new}} \leftarrow r_{\text{new}},\ r_{\text{old}} \bmod r_{\text{new}}$
  end while
  return r_{old}
end procedure

For later applications we not only need the $\gcd(a, b) = d$ but two additional elements x and y, such that

$$d = ax + by.$$

We can extend Euclidean algorithm to calculate x and y. For this, we are going to ensure that during the algorithm we are updating producing x_n and y_n for each r_n, such that

$$r_n = ax_n + by_n.$$

This will be our loop invariant property, i.e. this property will be true for n and $n+1$ whenever we enter the body of the loop and we will make sure that it is true for $n+2$ after the last statement of the body. Before the first execution of the loop body, the values $x_0 = 1$, $y_0 = 0$, $x_1 = 0$ and $y_0 = 1$ will suffice. Let $q = \lfloor r_n / r_{n+1} \rfloor$. During the body of the loop we have that

$$\begin{aligned} r_{n+2} &= r_n \bmod r_{n+1} \\ &= r_n - q r_{n+1} \\ &= (ax_n + by_n) - q\,(ax_{n+1} + by_{n+1}) \\ &= a\,(x_n - qx_{n+1}) + b\,(y_n - qy_{n+1}), \end{aligned}$$

so $x_{n+2} = x_n - qx_{n+1}$ and $y_{n+2} = y_n - qy_{n+1}$ will work. Again, we only need tha last two value of x and y during the calculations. We call the procedure *Extended Euclidean algorithm*:

Algorithm 7 Extended Euclidean algorithm

procedure GCD(a, b)
    $r_{\text{old}},\ x_{\text{old}},\ y_{\text{old}} \leftarrow a,\ 1,\ 0$
    $r_{\text{new}},\ x_{\text{new}},\ y_{\text{new}} \leftarrow b,\ 0,\ 1$
    while $r_{\text{new}} \neq 0$ **do**
        $r_{\text{old}},\ q,\ r_{\text{new}} \leftarrow r_{\text{new}},\ \textbf{divmod}\,(r_{\text{old}}, r_{\text{new}})$
        $x_{\text{old}},\ x_{\text{new}} \leftarrow x_{\text{new}},\ x_{\text{old}} - qx_{\text{new}}$
        $y_{\text{old}},\ y_{\text{new}} \leftarrow y_{\text{new}},\ y_{\text{old}} - qy_{\text{new}}$
    end while
    return $r_{\text{old}},\ x_{\text{old}},\ y_{\text{old}}$
end procedure

2.3 Linear Diophantine equation

Let $a, b, c \in \mathbb{Z}$ and we search for the integers solutions $x, y \in Z$ for the equation

$$ax + by = c.$$

If $d = \gcd(a, b)$ then $d \mid ax + by$ there $d \mid c$. Otherwise there is no solution. With Extended Euclidean algorithm, we can compute $d = ax' + by'$. Since $d \mid c$, there exists $m \in \mathbb{Z}$ for which $c = d \cdot m$. In summary

$$a\,(x'm) + (x'm) = d \cdot m = c,$$

so $x_0 = x'm$ and $y_0 = y'm$ is a pair of solution.

If there is a pair of solution, then there are infinitely many solutions in the form of

$$x = x_0 + k\frac{b}{d}, \quad y = y_0 - k\frac{a}{d}$$

where $k \in Z$ and there are no other solutions.

2.4 Modular arithmetic

Let $m > 1$. We say that a is *congruent to* $b \pmod{m}$, if $m \mid a-b$. The notation for this relation $a \equiv b \pmod{m}$. Otherwise we say, that a is *incongruent to* b, or $a \not\equiv b \pmod{m}$.

For a fixed $m > 1$, $a \equiv b \pmod{m}$ defines an equivalence relation over the $\mathbb{Z}$. The equivalence classes are called *residue class*es. The residue class of a is

$$\overline{a} = \{a + km : k \in \mathbb{Z}\}.$$

Let us denote

$$\mathbb{Z}_m = \left\{\overline{0}, \overline{1}, \dots, \overline{m-1}\right\}$$

the set of residue classes.

Theorem 5 (The compability of integer operations with the residue classes)**.** *If* $a \equiv b \pmod{m}$ *and* $c \equiv d \pmod{m}$*, then*

- $a + c \equiv b + d \pmod{m}$ *and*
- $ac \equiv bd \pmod{m}$.

This can be stated on residue class level as well:

1. $\overline{a} + \overline{b} = \left\{a' + b' : a' \in \overline{a}, b' \in \overline{b}\right\} = \overline{a+b}$ *and*
2. $\overline{a} \cdot \overline{b} = \left\{a'b' : a' \in \overline{a}, b' \in \overline{b}\right\} = \overline{ab}$.

Theorem 5 enables us to work on the *residue system* level, i.e. on

$$\mathbb{Z}_m = \left\{\overline{0}, \overline{1}, \dots, \overline{m-1}\right\} \simeq \{0, 1, \dots, m-1\}$$

The operations on the residue system can be defined as

$$a +_m b = (a + b) \bmod m, \quad a \cdot_m b = ab \bmod m.$$

Theorem 6 (Structure of $\mathbb{Z}_m$)**.** *Let* $m > 1$ *and* $a \neq 0$.

1. *If* $\gcd(a, m) \neq 1$*, then* $ax \equiv 0 \pmod{m}$ *has non-zero solution.*
2. *If* $\gcd(a, m) = 1$*, then* $ax \equiv 1 \pmod{m}$ *has a solution.*

Proof. 1. If $d = \gcd(a, m) > 1$, then

$$\frac{a}{d}m = a\frac{m}{d} \equiv 0 \pmod{m},$$

so $x = m/d$ is nonzero solution.

2. With the extended Euclidean algorithm, we can compute $1 = ax + my$. Rearranging this, we have that $ax - 1 = -ym$. So $m \mid ax - 1$, which is by definition means that $ax \equiv 1 \pmod{m}$.

$\square$

If $ax \equiv 1 \pmod{m}$, then we call x the *modular inverse* of a, denoted by $a^{-1} \pmod{m}$.