

3 The Fundamentals: Algorithms, the Integers, and Matrices

3.1 Algorithms

Definition 1 An **algorithm** is a finite set of instructions for performing a computation or solving a problem.

Algorithm 1 Finding the maximal element in a finite sequence

procedure max (a_1, a_2, \dots, a_n : integers)	$n \geq 1$
$max := a_1$;	$max = max(a_1)$
for $i := 2$ to n do	
if $max < a_i \rightarrow max := a_i$ fi	$max = max(a_1, \dots, a_i)$
od	$max = max(a_1, \dots, a_n)$

$max, i := a_1, 2;$

do $i \leq n \rightarrow$

if $max < a_i \rightarrow max = a_i$

 | $max \geq a_i \rightarrow skip$

fi;

$i := i+1$

od

$max = max(a_1, \dots, a_{i-1}) \wedge (i=n+1) = max(a_1, \dots, a_n)$

$max = max(a_1) \quad (n \geq 1)$

$max = max(a_1, \dots, a_{i-1})$

Algorithm 2 *The linear search algorithm*

$i := 1$; **do** $(i \leq n \wedge x \neq a_i) \rightarrow i := i+1$ **od**

$[(i \leq n) \Rightarrow ((1 \leq \forall k < i: x \neq a_k) \wedge (x = a_i))] \vee [(i = n+1) \Rightarrow (1 \leq \forall k \leq n: x \neq a_k)]$

if $i \leq n \rightarrow loc := i$ | $i = n+1 \rightarrow loc = 0$ **fi**

$[(loc \leq n) \Rightarrow ((1 \leq \forall k < loc: x \neq a_k) \wedge x = a_{loc})] \vee [(loc = 0) \Rightarrow (1 \leq \forall k \leq n: x \neq a_k)]$

Another implementation for the linear search

$i, loc := 1, 0;$ $(n \geq 0)$

do $((i \leq n) \wedge (loc = 0)) \rightarrow$ **if** $x = a_i \rightarrow loc := i;$

| $x \neq a_i \rightarrow i := i+1$

od **fi**

$[(loc \leq n) \Rightarrow ((1 \leq \forall k < loc: x \neq a_k) \wedge x = a_{loc})] \vee [(loc = 0) \Rightarrow (1 \leq \forall k \leq n: x \neq a_k)]$

Algorithm 3 The binary search algorithm

procedure **binary search**(x : integer; a_1, a_2, \dots, a_n : increasing integers)

$i, j, loc := 1, n, 0$;

do $((i \leq j) \wedge (loc = 0)) \rightarrow m := \lfloor (i+j)/2 \rfloor$;

if $x > a_m \rightarrow i := m+1$

| $x = a_m \rightarrow loc := m$

| $x < a_m \rightarrow j := m-1$

fi

od

$(1 \leq \forall k < n: a_k \leq a_{k+1}) \wedge (n \geq 0)$ is the given condition (constant)

$[(loc \leq n) \Rightarrow (x = a_{loc})] \vee [(loc = 0) \Rightarrow (1 \leq \forall k \leq n: x \neq a_k)]$

$i, j := 1, n; loc := 0;$
do $(i \leq j) \rightarrow m := \lfloor (i+j)/2 \rfloor;$
 if $x > a_m \rightarrow i := m+1$
 | $x = a_m \rightarrow loc := m; i, j := m, m-1;$
 | $x < a_m \rightarrow j := m-1$
 fi
od
 $[(loc \leq n) \Rightarrow (x = a_{loc})] \vee [(loc=0) \Rightarrow (1 \leq \forall k \leq n: x \neq a_k)]$

procedure **bubble sort**(a_1, a_2, \dots, a_n : distinct integers with $n \geq 2$)

for $i := 1$ to $n-1$ **do**

for $j := 1$ to $n-i$ **do** \rightarrow

if $a_j \geq a_{j+1} \rightarrow a_j, a_{j+1} := a_{j+1}, a_j$

 | $a_j \leq a_{j+1} \rightarrow$ **skip**

fi

od

od

$i = 1$ $a_n = \max(a_1, \dots, a_n)$ $(n-1)$ iterations

$i = 2$ $a_{n-1} = \max(a_1, \dots, a_{n-1})$ $(n-2)$ iterations

...

i $a_{n-i+1} = \max(a_1, \dots, a_{n-i+1})$ $(n-i)$ iterations

...

$i = n-1$ $a_2 = \max(a_1, a_2)$ (1) iterations

$1 \leq \forall k < i < n: a_k \leq a_{k+1}.$

procedure insertion sort(a_1, a_2, \dots, a_n : distinct integers with $n \geq 2$)
for $j := 2$ *to* n *do*
 $i := 1$; *do* ($a_j > a_i$) \rightarrow $i := i + 1$ *od*
 $m := a_j$;
 for $k := 0$ *to* $j - i - 1$ *do* $a_{j-k} := a_{j-k-1}$ *od*;
 $a_i := m$
od

Dijkstra's mini language

$$S ::= x_1, \dots, x_n := E_1, \dots, E_n \mid S; S \mid \mathit{skip} \mid \mathit{abort}$$

$$\mid \mathit{if} B_1 \rightarrow SL_1 \mid \dots \mid B_n \rightarrow SL_n \mathit{fi}$$

$$\mid \mathit{do} B_1 \rightarrow SL_1 \mid \dots \mid B_n \rightarrow SL_n \mathit{od}$$

$$\mathit{if} x \geq y \rightarrow m := x$$

$$\mathit{if} x > y \rightarrow m := x$$

$$\mathit{if} x \geq y \rightarrow m := x$$

$$\mid x < y \rightarrow m := y$$

$$\mid x \leq y \rightarrow m := y$$

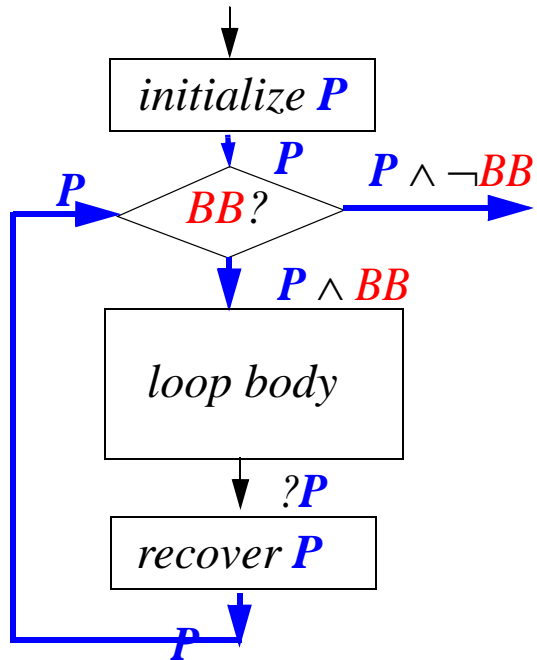
$$\mid x \leq y \rightarrow m := y$$
fi***fi******fi***

$$(m=x \vee m=y) \wedge m \geq x \wedge m < y$$

$$(m=x \vee m=y) \wedge m > x \wedge m \geq y$$

$$(m=x \vee m=y) \wedge m \geq x \wedge m \geq y$$

do $B_1 \rightarrow SL_1$ '|' ... '|' $B_n \rightarrow SL_n$ **od**



Let's assume that loop invariance is P .

and $BB = B_1 \vee B_2 \vee \dots \vee B_n$.

$\therefore \neg BB = \neg(B_1 \vee B_2 \vee \dots \vee B_n)$

$= \neg B_1 \wedge \neg B_2 \wedge \dots \wedge \neg B_n$.

Loop invariance P

Loop invariance P should be **initialized** before the loop,

Loop invariance P must be **recovered** in the loop, if destroyed, and

Loop invariance P is still **valid** after(termination of) the loop.

$P \wedge \neg BB$ is the **final** condition that you want **after the loop!**

$a, b, c, d := A, B, C, D;$
do $a > b \rightarrow a, b := b, a$
 $| b > c \rightarrow b, c := c, b$
 $| c > d \rightarrow c, d := d, c$
od

$$\neg BB = (a \leq b) \wedge (b \leq c) \wedge (c \leq d)$$

$$= a \leq b \leq c \leq d.$$

What is the loop invariance of the above do-od loop?

$P = a, b, c, \text{ and } d$ are **permutation** of $A, B, C, \text{ and } D$.

Loop invariance P should be **strong** enough to get **final result** with and $\neg BB$;

whereas **weak** enough

to be easily **initialized** and be **still valid** inside the loop.

$P \wedge \neg BB$ is the **final condition** that you want.

Greedy algorithm**Optimization problem**

to find a solution to the given problem that

either minimizes or maximizes the value of some parameters

(global) optimal

to select the best choice considering all sequences of steps

local optimal (greedy algorithm)

to select the best choice at each step

Algorithm 6 Greedy Change-Making Algorithm

procedure change(n_1, n_2, \dots, n_r : number of coins are r and

$c_1 > c_2 > \dots > c_r$ is the value for each coin $_i$;

n : positive integer is the amount that you want to make change)

$n_1, n_2, \dots, n_r := 0, 0, \dots, 0; i := 1;$

do ($i \leq r$) \rightarrow do ($n \geq c_i$) $\rightarrow n_i := n_i + 1; n := n - c_i$ od; $i := i + 1$ od

The Halting Problem

Unsolvable problems

Halting problem

where program will be halt or not(infinite loop)

Alan Turing, 1936

No algorithm that computes it!

Three cases

i) (totally) solvable(computable; recursive)

always terminate

algorithms

ii) partially solvable(partially computable;recursively enumerable)

algorithm(program) exits

but may halts or loops forever

iii) unsolvable(uncomputable; Not recursively enumerable)

No algorithm(program) exists

proof Assume a procedure $H(P: \text{program}, I: \text{input data})$ exists where

If P runs with I and halts \rightarrow print “halt” as output.

| P with I loops forever \rightarrow print “loops forever” as output.

Program also can be a input data(compiler)

Consider procedure $K(P)$ as follows

procedure $K(P: \text{program})$

if $H(P, P)$ prints “halts” \rightarrow loops forever

| $H(P, P)$ prints “loops forever” \rightarrow halts

fi

Now consider $K(K)$

if $H(K, K)$ print “loops forever” $\rightarrow K(K)$ halts

| $H(K, K)$ print “halts” $\rightarrow K(K)$ loops forever

Contradiction

No algorithm for halting program!

Three cases for problem(programs) (Turing-Church's thesis)

- i) Always terminate **algorithm**
totally solvable
 - ii) Terminate or loop forever **program**
partially solvable
programmable
 - iii) Program does not exists **non-programable**
halting program
Russel's Paradox
- $i) \subset ii)$ $ii) \cap iii) = \emptyset$.

3.2 The Growth of Functions

Big-O Notation

Definition 1 Let $f: \mathbf{N} \rightarrow \mathbf{R}$ or $\mathbf{R} \rightarrow \mathbf{R}$.

We say $f(x)$ is $O(g(x))$, if $\exists c, k \in \mathbf{R} .\exists. |f(x)| \leq c|g(x)| \forall x > k$.

We also write $f(x) = O(g(x))$ or even $f(x) \in O(g(x))$.

The constant c and k are called the **witnesses** of ... $f(x)$ is $O(g(x))$.

Assume c and k are **smallest** witnesses, Then $\forall c', k' .\exists. c < c', x < x',$
 c' and k' are also **witnesses**

$g(x)$ is **faster or equal(not slower)** growing than $f(x)$
 when x becomes **large enough**
 and **ignoring** multiplying some **constant** c .

Example 1, 2, 3, 4

Theorem 1 Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$. Then $f(x)$ is $O(x^n)$.
proof easy with witness $k=1$ and $c = \sum |a_i|$.

Example 6 $n!$

$n! \leq n^n$. $n!$ is $O(n^n)$ with witness $k=1$ and $c=1$.

$\log n! \leq \log n^n = n \log n$

$1, \log n, n, n \log n, n^2, \dots, 2^n, n!$

Definition Let $g: \mathbf{N}$ or $\mathbf{R} \rightarrow \mathbf{R}$.

$O(g) = \{f: \mathbf{N}$ or $\mathbf{R} \rightarrow \mathbf{R} \mid \exists c, k > 0 \forall x > k: |f(x)| \leq c|g(x)|\}$

If $f \in O(g)$, f grows **at most** order of g .

$O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset \dots \subset O(2^n) \subset O(n!)$

Theorem 2, 3 Let $f_1 \in O(g_1)$, $f_2(x) \in O(g_2)$. Then

$$f_1 + f_2 \in O(g_1 + g_2) = O(\max(|g_1|, |g_2|)) \text{ and}$$

$$f_1 f_2 \in O(g_1 g_2).$$

Colollary 1 Let $f_1, f_2 \in O(g)$. Then $f_1 + f_2 \in O(g)$.

Big-Omega and Big-Theta Notation

Definition 2 $\Omega(g) = \{f: \mathbf{N} \text{ or } \mathbf{R} \rightarrow \mathbf{R} \mid \exists c, k > 0 \forall x > k: |f(x)| \geq c|g(x)|\}$

If $f \in \Omega(g)$, f grows *at least* order of g .

Definition 3 $\Theta(g) = \{f: \mathbf{N} \rightarrow \mathbf{R} \mid \exists c_1, c_2, k > 0, \forall x > k: c_1|g(x)| \leq |f(x)| \leq c_2|g(x)|\}$

If $f \in \Theta(g)$, f grows (exactly) *same* order of g .

$f \in \Theta(g)$, iff $f \in O(g) \wedge f \in \Omega(g)$.

Theorem 4 Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ with $a_n \neq 0$. Then

$$f(x) \in \Theta(x^n).$$

3.3 Complexity of Algorithms

Computational complexity

time complexity

space complexity

Worst-case analysis

Average-case analysis

$\Theta(1)$	<i>constant complexity</i>
$\Theta(\log n)$	<i>logarithmic complexity</i>
$\Theta(n)$	<i>linear complexity</i>
$\Theta(n^k)$	<i>polynomial complexity</i>
$\Theta(k^n)$	<i>exponential complexity</i>
$\Theta(n!)$	<i>factorial complexity</i>

(totally) solvable

tractable

polynomial

intractable

exponential

class P

polynomial, tractable

class NP ($\mathbf{P} \subseteq \mathbf{NP}$)

Nondeterministic polynomial

lower bound?

exponential(upper bound)

tractable or intractable

NP-complete problems

subset of class NP

If any of these problems can be solved in polynomial time,

*all the problems in class in **NP** also in class **P** ($\mathbf{P} = \mathbf{NP}$).*

3.4 Integer and Division

Definition 1 Let $a, b \in \mathbf{Z} \wedge a \neq 0$. We say a **divies** b , if $\exists c \in \mathbf{Z} . \exists. b = ac$.

We say a is a **factor**(약수) of b , and b is a **multiple**(배수) of a .

$a \mid b$ denotes a **divides** b , and $a \nmid b$ to denote a **does not divide** b .

Theorem 1 $\forall a, b, c \in \mathbf{Z}$.

$$i) (a \mid b) \wedge (a \mid c) \Rightarrow a \mid (b+c).$$

$$ii) (a \mid b) \Rightarrow a \mid bc.$$

$$iii) (a \mid b) \wedge (b \mid c) \Rightarrow a \mid c.$$

Corollary 1 $\forall a, b, c \in \mathbf{Z}$. (Extension of **Thm 1. i)** and **ii)**)

$$(a \mid b) \wedge (a \mid c) \Rightarrow a \mid (mb+nc) \quad \forall m, n \in \mathbf{Z}.$$

Theorem 2 The Division Algorithm

$\forall a \in \mathbf{Z}, \forall d \in \mathbf{Z}^+ \exists_1 q, \exists_1 r \in \mathbf{Z}: 0 \leq r < d \text{ and } a = dq + r.$

Definition 2 d is called the **divisor**, a is called **divided**,
 q is called the **quotient**, and r is called the **remainder**.
 $q = a \text{ div } d, r = a \text{ mod } d.$

Modular Arithmetic

Definition 3 Let $a, b \in \mathbf{Z}, m \in \mathbf{Z}^+$. Then a is **congruent** to b modulo m ,
 written $a \equiv b \pmod{m}$ or $b \in [a]_{\text{mod } m}$, if $m \mid (a - b)$.

Theorem 3 $a \equiv b \pmod{m}$, iff $a \text{ mod } m = b \text{ mod } m$.
 or $(a - b) \text{ mod } m = 0$.

Theorem 4 $a \equiv b \pmod{m}$, iff $\exists k \in \mathbf{Z}: a = b + km$.

Theorem 5 If $a \equiv b \pmod{m} \wedge c \equiv d \pmod{m}$, then
 $a + c \equiv b + d \pmod{m}$ $ac \equiv bd \pmod{m}$.

3.5 Primes and Greatest Common Divisors

Definition $p \in \mathbf{Z}^+$ is **prime**, $\Leftrightarrow p > 1 \wedge \neg \exists a \in \mathbf{Z}^+ : (1 < a < p \wedge a \mid p)$.
 Otherwise **composite**.

Theorem 1 The Fundamental Theorem of Arithmetic

Every positive integer has a **unique** representation
 as the product of nondecreasing series of **zero or more primes**.

Theorem 2 If n is composite integer, then n has a prime divisor less than or equal to $\text{SQRT}(n)$.

Theorem 3 There are **infinitely** many primes.

proof proof by contradiction

Assume there are only finitelt many primes, p_1, p_2, \dots, p_n .

Let $Q = p_1 p_2 \dots p_n + 1$.

Q is prime or product of two or more primes.

If $p_j \mid Q$, $p_j \mid Q - p_1 p_2 \dots p_n = p_j \mid 1$.

$\therefore \neg \exists j: 1 < j < n, p_j \mid Q$.

\therefore There exist other prime not in the list p_1, p_2, \dots, p_n or Q is a prime.

\therefore There are infinitely many primes.

Conjectures and Open Problems about Primes

Goldbach's conjecture

Every even integer n , $n > 2$, is the sum of two prime numbers.

$$2 \cdot 10^{17}.$$

The Twin Prime conjecture

*primes that **differs two***

*$16,8699,8733,9975 \cdot 2^{17,1960} \pm 1$ are **primes** with 5,1779 digits.*

Greatest common divisors and Least common multiples

Definition 2 Let $a, b \in \mathbf{Z}$ and not both zero.

$$d = \gcd(a, b) = \max(d: d \mid a \wedge d \mid b) \Leftrightarrow, \\ d \mid a \wedge d \mid b \wedge \forall e \in \mathbf{Z}, (e \mid a \wedge e \mid b) \rightarrow d \geq e.$$

Definition 3 The integers a and b are **relatively prime (coprime)** if $\gcd(a, b) = 1$.

Definition 4 The integers a_1, a_2, \dots, a_n are **pairwise relatively prime**, if $\gcd(a_i, a_j) = 1$ whenever $1 \leq i < j \leq n$.

Definition 5 Let $a, b \in \mathbf{Z}$ and not both zero.

$$d = \text{lcm}(a, b) = \min(m: a \mid m \wedge b \mid m) \Leftrightarrow, \\ a \mid m \wedge b \mid m \wedge \forall n \in \mathbf{Z}, (n \mid a \wedge n \mid b) \rightarrow m \leq n.$$

Let $a = p_1^{a_1} p_2^{a_2} \dots p_n^{a_n}$ and $b = p_1^{b_1} p_2^{b_2} \dots p_n^{b_n}$.

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \dots p_n^{\min(a_n, b_n)}.$$

$$\text{lcm}(a, b) = p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \dots p_n^{\max(a_n, b_n)}.$$

Theorem 5 Let a and b be positive integer.

$$ab = \gcd(a, b) \text{lcm}(a, b)$$

3.6 Integer and Algorithms

Theorem 1 Let $b \in \mathbf{Z}^+$, $b > 1$. Then $\forall n \in \mathbf{Z}^+$,

$$n = a_k b^k + a_{k-1} b^{k-1} + \dots + a_1 b + a_0.$$

$$k \geq 0, 0 \leq a_0, a_1, \dots, a_k < b, a_k \neq 0.$$

unique representation

base expansion of n

written $n = (a_k a_{k-1} \dots a_1 a_0)_b$.

Lemma 1 Let $a = bq + r$, $a, b, q, r \in \mathbf{Z}$. Then $\gcd(a, b) = \gcd(b, r)$

proof $\forall d, d \mid a \wedge d \mid b$, then $d \mid a - bq = r$. (Corollary 1)

$\forall d, d \mid b \wedge d \mid r$, then $d \mid bq + r = a$.

Algorithm 6 Euclid algorithm

procedure gcd(a, b: positive integer)
 do $b \neq 0 \rightarrow r := a \bmod b; a := b; b := r$ *od;*
 return a

procedure gcd(a, b: positive integer) Euclid's algorithm
 do $a > b \rightarrow a := a - b$
 | $a < b \rightarrow b := b - a$
 od;
 return a

3.7 Application of Number Theory**3.8 Matrices**