3 The Fundamentals:Algorithms, the Integers, and Matrices

3.1 Algrithms

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

Dijkstra's mini language

$S ::= x_1, \ldots, x_n := E_1, \ldots, E_n \mid S; S \mid$ ***skip*** $\mid$ ***abort***

$\mid$ ***if*** $B_1 \to SL_1$ '$\mid$' $\ldots$ '$\mid$' $B_n \to SL_n$ ***fi***

$\mid$ ***do*** $B_1 \to SL_1$ '$\mid$' $\ldots$ '$\mid$' $B_n \to SL_n$ ***od***

if $x > y \to m := x$
$\mid x \le y \to m := y$
fi

if $x \ge y \to m := x$
$\mid x < y \to m := y$
fi

if $x \ge y \to m := x$
$\mid x \le y \to m := y$
fi

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

Algorithm *1 Finding athe maximal element in a finite sequence*

procedure ***max***$(a_1, a_2, \ldots, a_n$*: integers)*

$max := a_1;$

do $i \leq n \rightarrow$ ***if*** $max \leq a_n \rightarrow max = a_n;\ i := i+1$

$\quad \mid max \geq a_n \rightarrow i := i+1$

fi

od

Algorithm *2 The linear search algorithm*

procedure ***linear search****(x: integer;* $a_1, a_2, \ldots, a_n$*: distinct integers)*

$i := 1;$

do $(i \le n \wedge x \ne a_i) \rightarrow i := i+1$ ***od***

if $i \le n \rightarrow location := i$

$\mid i=n+1 \rightarrow location = 0$

fi

$i := 1;\ location := 0;$

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

$\mid x \ne a_i \rightarrow i := i+1$

fi

od

Algorithm *3 The binary search algorithm*

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

$i, j := 1, n;\ location := 0;$

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

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

$| \; x = a_m \rightarrow location := m;$

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

fi

od

do $(i \le j) \rightarrow m := \lfloor (i+j)/2 \rfloor ;$

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

$| \; x = a_m \rightarrow location := m;\ i, j := m, m-1;$

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

fi

od

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

$i := 1$;
do $(i \leq n\text{-}1) \rightarrow j := 1$;
    do $(j \leq n\text{-}i) \rightarrow$ ***if*** $a_j \geq a_{j+1} \rightarrow a_j, a_{j+1} := a_{j+1}, a_j; j := j+1$
        $\mid a_j \leq a_{j+1} \rightarrow j := j+1$
        fi;
    od;
    $i := i + 1$
od

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

j := 2;
do *($j \leq n$)* $\rightarrow$ *i := 1;*
do *($a_j > a_i$)* $\rightarrow$ *i := i+1;*
m := a_j;
k := 0; ***do*** *($k \leq j - i - 1$)* $\rightarrow$ *$a_{j-k} := a_{j-k-1}$;* ***od****;*
$a_i := m$
od*;*
i := i + 1
od

Greedy algorithm

Optimization problem

to find a solution to the given problem that

either ***minimizes*** *or* ***maximizes*** *the value of somr parameters*

*(**global**)* ***optimal***

to select the best choise considering all sequences of steps

local optimal**(greddy algorithm)**

to select the best choise at each step

Algorithm *6 Greedy Change-Making Algorithm*

procedure change(n_1, n_2, …, n_r: number of coins where

$c_1 > c_2 > \ldots > c_r$ is the value for each $coin_i$; n: positive integer)

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

do $(i \le r) \rightarrow$ **do** $(n \ge c_i) \rightarrow n_i := n_i + 1;\ n := n - c_i$ **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: program, I: input data) exists where*

If P runs with I and halts → print "halt" as output.

| P with I loops forever → print "loops forever" as output.

Program also can be a input data(compiler)

Condider procedure K(P) as follows

procedure K(P: program)

if H(P, P) prints "halts" → loops forevar

| H(P, P) prints "loops forever" → halts

fi

Now consider K(K)

if H(K, K) print "loops forever" → K(K) halts

| H(K, K) print "halts" → K(K) loops forever

Contradiction

No algorithm for halting program!

3.2 The Groth of Functions

Big-O Notation

Definition *1 Let f:* **N** $\rightarrow$ **R** *or* **R** $\rightarrow$ **R**.

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

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 witnesses, $\forall c', k'$ *.ə.* $c < c', x < x'$,

c' and k' are also ***witnesses***

g(x) is ***faster or equl(not slower)*** *glowing than f(x)*

when x becomes ***large enough***

and ***ignoring*** *multiplying some* ***constant*** *c.*

Example 1, 2, 3, 4

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

Example 6 n!

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

$\log n! \le \log n^n = n \log n$

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

Definition *Let* g: **N** *or* **R** $\rightarrow$ **R**.

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

we may write $f \in O(g)$.

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

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

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

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

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

Definition *3*

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

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

We say f is ***order*** *of g, if* $f \in \Theta(g)$.

Theorem *4 Let* $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ *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

$\Theta(1)$	***constant*** *complexity*
$\Theta(\log n)$	***logarithmic*** *complexity*
$\Theta(n)$	***linear*** *complexity*
$\Theta(n^k)$	***polinomial*** *complexity*
$\Theta(k^n)$	***exponential*** *complexity*
$\Theta(n!)$	***factorial*** *complexity*

(totally) solvable

tractable

polynomial

intractable

exponential

***class* P**

polynomial, tractable

class* NP*(*P ⊆ NP*)

Nondeterministc polynomial *exponetial(upper bound)*

lower 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*(*P = NP*)*.

3.4 Integer and Division

Definition *1 Let* $a, b \in \mathbf{Z} \wedge a \neq 0$. *We say a* ***divies*** *b, if* $\exists c \in \mathbf{Z}$.ə. $b = ac$. *We say a is a* ***factor*** *of b and b is a* ***multiple*** *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* $\forall a, b, c \in \mathbf{Z}$.

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

Theorem** 2 **The Division Algorithm

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

***Definition** 2 d is called the **divisor**, a is called **dividened**,*
*q is called the **quotient**, and q is called the **remainder**.*
$q = a$ ***div*** d, $r = a$ ***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 $b \in [a]_{mod\ m}$, if $m \mid (a - b)$.

***Theorem** 3 $a \equiv b \pmod m$, iff $a \bmod m = b \bmod m$.*
or $(a - b) \bmod m = 0$.

***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*
$a + c \equiv b + d \pmod m$ $\quad$ $ac \equiv bd \pmod m$.

3.5 Primes and Greatest Common Divisors

Definition *1* $p \in \mathbf{Z}^+$ *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 SQRT(n).*

Theorem *3 There are* ***infinitely*** *many primes.*
proof *proof by contradiction*
Assume there are only finitelt many primes, $p_1, p_2, \ldots p_n$.
Let $Q = p_1 p_2 \ldots p_n + 1$.

Q is prime or product of two or more primes.

If $p_j \mid Q$, $p_j \mid Q - p_1p_2 \ldots p_n = p_j \mid 1$.

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

$\therefore$ *There exist other prime not in the list* $p_1, p_2, \ldots p_n$ *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.*

Graetest 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* $a_1, a_2, \ldots a_n$ *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.*

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

Let $a = p_1^{a1} p_2^{a2} \dots p_n^{an}$ *and* $b = p_1^{b1} p_2^{b2} \dots p_n^{bn}$.

$$gcd(a, b) = p_1^{min(a1, b1)} p_2^{min(a2, b2)} \dots p_n^{min(an, bn)}.$$

$$lcm(a, b) = p_1^{max(a1, b1)} p_2^{max(a2, b2)} \dots p_n^{max(an, bn)}.$$

Theorem *5 Let a and b be positive integer.*

$$ab = gcd(a, b)\, lcm(a, b)$$

3.6 Integer and Algorithms

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

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

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

unique *repesentation*

base expansion *of n*

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

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) Dijkstra'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