Lecture 2: Algorithms

Basic Algorithms for Numbers and Graphs

Angsheng Li

BeiHang University

Computational Theory
24, Sept., 2019

Outline - Basics of Algorithms

1. **Numbers**: Fundamental algorithms
2. **Graphs**:
   2.1 Algorithm ideas
   2.2 Depth first search
   2.3 Width-first search
3. Primal and Dual
4. **Convex optimization**

Computation

1. $+$
2. $-$
3. $\times$
4. $\div$

Operations of numbers are basic to all math. Operations are actually algorithms

Definition

Definition

For integers a, b, $a \neq 0$, we say that a **divides** b, written $a|b$, if there is an integer c such that

$$b = a \cdot c$$

holds.

In this case, we say that

- a is a **factor** of b,
- b is a **multiple** of a, and
- b is **divisible** by a.

Understanding of division

For a positive integer d, the numbers that are divisible by d, are:

$$\cdots, -4d, -3d, -2d, -d, 0, d, 2d, 3d, 4d, \cdots.$$

For an integer a,

- $\lfloor \frac{a}{d} \rfloor$: The greatest integer x such that $x \cdot d \leq a$.
- $\lceil \frac{a}{d} \rceil$: The least integer y such that $y \cdot d \geq a$.
- $\lfloor \frac{a}{d} \rfloor \cdot d$ is the integer part of $\frac{a}{d}$
- $a - \lfloor \frac{a}{d} \rfloor \cdot d$ is the fractional part of $\frac{a}{d}$.
- If d is a factor of a and b, then d is a factor of any linear combination of a and b, with integer coefficients.

Intuition of division

For a positive integer d, the numbers that are divisible by d, are:

$$\cdots, -4d, -3d, -2d, -d, 0, d, 2d, 3d, 4d, \cdots.$$

The division algorithm

Theorem

Let a be an integer and d a positive integer (or a natural number). Then there exist unique integers q and r satisfying:

(1)

$$a = q \cdot d + r,$$

(2)

$$0 \leq r < d.$$

The division algorithm - intuition

Suppose that d is a positive integer and a be an integer. Let q be the integer satisfying:

$$q \cdot d \leq a < (q+1) \cdot d. \quad (1)$$

Clearly, q is unique.
Then for $r = a - qd$, then

$$\begin{aligned} a &= q \cdot d + r \quad (2) \\ 0 &\leq r < d. \end{aligned}$$

Uniqueness

Suppose that q, r and q', r' satisfy:

$$\begin{align} a &= q \cdot d + r \tag{3} \\ 0 \leq r &< d \tag{4} \\ a &= q' \cdot d + r' \tag{5} \\ 0 \leq r' &< d. \tag{6} \end{align}$$

By (1) and (3),

$$(q - q') \cdot d = (r' - r), \text{so that } d | (r' - r). \tag{7}$$

By (2) and (4),

$$-d < r' - r < d. \tag{8}$$

(5) and (6) together give $r' - r = 0$, so that $r' = r$ and $q' = q$.

Existence of q, r

In a one-dimensional axis with unit d, there are two cases:

Case 1 a is at some integral point $q \cdot d$.
Then $q = q$, and $r = 0$.

Case 2. $q \cdot d < a < (q + 1)d$.
Then $q = q$ and $r = a - q \cdot d$.

Notations

Assume $d > 0$, a, d, q and r satisfy:

(i) $a = q \cdot d + r$, and

(ii) $0 \le r < d$.

We call:

- d: *divisor*
- a: *divident*
- q: *quotient*, written $q = a \operatorname{div} d$
- r: *remainder*, written $r = a \mod d$.

For fixed $d > 0$,
$x \operatorname{div} d$:
$x \mod d$:
are both functions.

Congruence

Definition

Given integers a, b and natural number m, we say that ***a* is congruent to *b* modulo *m***, if:

$$m|(a-b).$$

In this case we write

$$a \equiv b \pmod{m}.$$

in which,
m is called *modulus* (moduli, for pl)

Remark

- $a \equiv b \pmod{m}$: a relation
- $a \mod m$: a function, if m is fixed, and a varies.

Basic properties - I

Theorem

Let a, b be integers and m be natural number. Then

$$a \equiv b \pmod{m} \iff a \mod m = b \mod b.$$

Intuition: $a \equiv b \pmod{m}$ if and only if a **and** b **have the same remainder divided by** m.

Thinking of a one-dimensional axis with unit m!.

Basic properties - II

Theorem

Let a, b be integers and m be natural number. Then:
a, b are congruent modulo m if and only if there is a k such that

$$a = b + mk.$$

$\mathbb{Z}_m$

$$\mathbb{Z}_m = \{0, 1, 2, \cdots, m-1\}.$$

Here $i \in \mathbb{Z}_m$ represents a congruence class modulo m, which is the set consists of all the numbers of the form

$$i + km$$

for all integers k.

Arithmetic in $\mathbb{Z}_m$

For natural number m, in

$$\mathbb{Z}_m = \{0, 1 \cdots, m-1\}$$

We define *addition* $+$

$$a + b = a + b \bmod m,$$

and *multiplication* $\cdot$

$$a \cdot b = a \cdot b \bmod m.$$

Arithmetic in $\mathbb{Z}_m$ - result

Theorem

Both addition $+$ and multiplication $\cdot$ are well-defined.

Properties of $\mathbb{Z}_m$

For $\langle \mathbb{Z}_m, +, \cdot \rangle$, we have

- Closure: if $a, b \in \mathbb{Z}_m$, then so are $a + b$ and $a \cdot b$
- Associativity: $(a + b) + c = a + (b + c), \ (ab)c = a(bc)$.
- Commutativity: $a + b = b + a$, and $ab = ba$.
- Identity elements: $0 + a = a$, and $1 \cdot a = a$.
- Additive inverse: $a + (-a) = 0$, $-a = m - a$.
- Distributivity: $a(b + c) = ab + ac$.

Question Is there multiplicative inverse?

Group

Definition

A group is a set G in which an operation, denoted $*$, is defined, such that the following properties are satisfied:

(1) Closure: if $a, b \in G$, then $a * b \in G$.

(2) Identity element: there is an 1 such that for every $a \in G$, $1 * a = a * 1 = a$.

(3) Inversive: For every $a \in G$, there is a $b \in G$ such that $a * b = b * a = 1$.

(4) Associativity: $(a * b) * c = a * (b * c)$.

Furthermore, if $a * b = b * a$ holds for all a, b, then G is called commutative.

Examples of groups

- $\langle \mathbb{Z}, + \rangle$ is a commutative group.
- For every natural number m, $\langle \mathbb{Z}_m, + \rangle$ is a finite, commutative group.

Ring

Definition

A ring is a set R with two operations $+$ and $\cdot$, satisfying the following properties:

(1) $\langle R, +\rangle$ is a commutative group.

(2) $\langle R, \cdot\rangle$ is associative.

(3) $\langle R, +, \cdot\rangle$ is distributive.

Furthermore, if $\langle R, \cdot\rangle$ is commutative, we say that $\langle R, +, \cdot\rangle$ is a commutative ring.

Examples of rings

- $\langle \mathbb{Z}, +, \cdot \rangle$ is a commutative ring.
- The set of all $n \times n$ matrices with matrix addition and multiplication is a ring, but not commutative.
- For every natural number m, $\langle \mathbb{Z}_m, +, \cdot \rangle$ is a finite, commutative ring.

Representations

- Decimal: base 10
- Binary: base 2
- Octal: base 8
- Hexadecimal: base 16

Why not base 1?

Theorem

Let b be a natural number greater than 1. Then, for every positive integer n, there exists a unique base b representation of n, that is, there is a unique $k+1$-tuple $(a_0, a_1, \cdots, a_k)$ satisfying:

1. *for each j, $0 \leq a_j < b$, and $a_k \neq 0$,*
2. $$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0.$$

Uniqueness

Suppose that $(\alpha_0, \alpha_1, \cdots, \alpha_l)$ and $(\beta_0, \beta_1, \cdots, \beta_r)$ are two representations of n. Therefore,
(1) for each $0 \leq i < l$, $0 \leq \alpha_i < b$, and $0 < \alpha_l < b$,
(2) For each j with $0 \leq j < r$, $0 \leq \beta_j < b$, and $0 < \beta_k < b$,
(3) $n = \alpha_l b^l + \cdots + \alpha_1 b + \alpha_0$, and
(4) $n = \beta_r b^r + \beta_{r-1} b^{r-1} + \cdots + \beta_1 b + \beta_0$.

Uniqueness - Continued

By $(3) - (4)$, $b | (\alpha_0 - \beta_0)$, implying $\alpha_0 = \beta_0$.
Using this, we have:

$$\alpha_l b^l + \cdots + \alpha_1 b = \beta_r b^r + \cdots + \beta_1 b$$

so that

$$\alpha_l b^{l-1} + \cdots + \alpha_2 b + \alpha_1 = \beta_r b^{r-1} + \cdots + \beta_2 b + \beta_1.$$

This shows that $b | (\alpha_1 - \beta_1)$ so that $\alpha_1 = \beta_1$.
Repeating the process, we have that $l = r$, and for each i with $0 \leq i \leq l$, $\alpha_i = \beta_i$.

Existence

Suppose that

$$
\begin{aligned}
n &= q_0 b + a_0, \ 0 \le a_0 < b \\
q_0 &= q_1 b + a_1, \ 0 \le a_1 < b \\
&\cdots \\
q_{k-1} &= q_k b + a_k, \ 0 < a_k < b \\
q_k &= 0.
\end{aligned}
$$

Then

$$
n = a_k b^k + a_{k-1} b^{k-1} + \cdots a_1 b + a_0. \tag{9}
$$

Representation Algorithm $\mathcal{R}$

The algorithm $\mathcal{R}$ for finding representation of n with base $b > 1$.

(1) Set $q = n$, and $k = 0$.
Suppose that $a_0, a_1, \cdots, a_{k-1}$ are all defined, and a_k is undefined.

(2) If $q = 0$, then output the representation $(a_{k-1}, \cdots, a_1, a_0)$.

(3) Otherwise. Then:
   - (3a) define $a_k = q \bmod b$,
   - (3b) set $q \leftarrow q \operatorname{div} b$
   - (3c) set $k \leftarrow k + 1$, and
   - (3d) go back to step (2).

Correctness and Complexity of an Algorithm

Correctness:

Time complexity: The number of **steps** used in the running of the algorithm

Space complexity: The **cells** used in the running of the algorithms

Bounds of Complexity

Represented by a function of n, where n is the **length** of **the binary representation of the input**.

- $f(n) = O(g(n))$, if there is a constant c such that for all n, $f(n) \leq c \cdot g(n)$.
- $f(n) = o(g(n))$, if $\frac{f(n)}{g(n)}$ goes to 0 as n goes to ∞
- $f(n) = \Omega(g(n))$, if there is a constant c such that for all n, $f(n) \geq c \cdot g(n)$.
- $f(n) = \Theta(g(n))$, if both $f = O(g(n))$ and $f = \Omega(g(n))$ hold.
- $f(n) = \omega(g(n))$, if $\frac{g(n)}{f(n)}$ goes to 0 as n goes to ∞.
- $f(n) = \widetilde{O}(g(n))$, if $f(n) = O(g(n) \cdot \mathrm{poly}(\log_2 g(n)))$.

Correctness and Complexity of an Algorithm

Correctness:

Time complexity: The number of **steps** used in the running of the algorithm
Space complexity: The **cells** used in the running of the algorithms

Addition

Given

$$\begin{aligned} a &= a_{n-1} \cdots a_1 a_0 \\ b &= b_{n-1} \cdots b_1 b_0. \end{aligned} \tag{10}$$

Suppose that

$$\begin{aligned} a_0 + b_0 &= 2c_0 + s_0 \\ a_1 + b_1 + c_0 &= 2c_1 + s_1 \\ &\cdots \\ a_{n-1} + b_{n-1} + c_{n-2} &= 2c_{n-1} + s_{n-1} \\ s_n &= c_{n-1}. \end{aligned} \tag{11}$$

Then

$$a + b = s_n s_{n-1} \cdots s_1 s_0.$$

Addition algorithm $\mathcal{A}$

(1) set $c_{-1} = 0$.

(2) For j with $0 \leq j \leq n-1$,

  (2a) set $c_j = \lfloor (a_j + b_j + c_{j-1})/2 \rfloor$,

  2b) Set $s_j \leftarrow a_j + b_j + c_{j-1} - 2c_j$.

(3) Let $s_n = c_{n-1}$.

(4) Output

$$s_n s_{n-1} \cdots s_1 s_0.$$

Both **Time complexity** and **Space complexity**:
$O(n)$, where n is the maximal length of a and b.

Multiplication

Given

$$\begin{aligned} a &= a_l \cdots a_1 a_0 \\ b &= b_r \cdots b_1 b_0. \end{aligned} \tag{12}$$

For every $j = 0, 1 \cdots , l$, set S_j

$$S_j = \begin{cases} b0 \cdots 0(j \text{ zeros appended}), & \text{if } a_j = 1 \\ 0, & \text{Otherwise} \end{cases} \tag{13}$$

Multiplication algorithm $\mathcal{M}$

(1) Set $S = 0$ and $j = 0$.

(2) If $j = l + 1$, then output S.
Suppose that $j \leq l$.

(3) If $a_j = 1$, then
– set $S \leftarrow S + b0 \cdots 0$ (j zeros),
– set $j \leftarrow j + 1$, and
– go back to step (2).

The time complexity: $O(l(l + r)) = O(l \cdot r)$, if $l \leq r$.
The space complexity: $O(l \cdot r)$.

Subtraction

Suppose that

$$\begin{aligned} a &= a_l a_{l-1} \cdots a_1 a_0 \\ b &= b_k b_{k-1} \cdots b_1 b_0 \end{aligned}$$

with $l \geq k$ and $a \geq b$.
If $a_0 \geq b_0$, then
- $d_0 = 0$ and
- $s_0 = 2d_o + a_0 - b_0$.
If $a_0 < b_0$, then
- $d_0 = 1$,
- $s_0 = 2d_0 + a_0 - b_0$

Subtraction -continued

Suppose that s_j and d_j are both defined.
Case 1 If $a_{j+1} - d_j \geq b_{j+1}$, then
– $d_{j+1} = 0$, and
- $s_{j+1} = a_{j+1} - d_j - b_{j+1}$.
Case 2. Otherwise, then
- define $d_{j+1} = 1$, and
- set $s_{j+1} = a_{j+1} + 2d_{j+1} - d_j - b_{j+1}$.
The output is

$$a - b = s_l \cdots s_1 s_0.$$

The **time complexity** is $O(l)$, the **space complexity**: $O(\log l)$.

Algorithm for div and mod - idea

Suppose that

$$\begin{aligned} a &= a_l a_{l-1} \cdots a_1 a_0 \\ b &= b_k b_{k-1} \cdots b_1 b_0 \end{aligned}$$

are binary representations with $l \geq k$ and $a \geq b$.
Find q and r such that

$$a = qb + r$$

$$0 \leq r < b$$

Basic idea:
Get an algorithm with time complexity $O(n^3)$.

Algorithm for div and mod

Precisely, the algorithm proceeds as follows:

(1) Determine the highest digit of the quotient by reading the highest k or $k+1$ digits of a that is no less than b.
Let c be the shortest initial binary string that is greater than or equal to b.
Suppose that $a = c\hat{}d_1 d_2 \cdots d_m$.
Define $q_m = 1$. Let $\alpha = c - b$.

Algorithm for div and mod - continued

(2) If $\alpha d_1 \geq b$, then $q_{m-1} = 1$, and $\alpha_1 = \alpha d_1$.
If $\alpha d_1 < b$, let i be the least such that $\alpha d_1 \cdots d_i \geq b$, for each j with $1 \leq j < i$, set $q_{m-j} = 0$, $q_{m-i} = 1$, and set $\alpha_1 = \alpha d_1 \cdots d_i$.
Set $\alpha = \alpha_1 - b$.

(3) Reporting the procedure above.

The time complexity

The time complexity of the algorithm $\mathcal{E}$ is:

$$O(\log a \cdot \log b).$$

Note that if $a < 0$ and $b > 0$. Then let

$$-a = qb + r,\ 0 \leq r < b.$$

If $r = 0$, then $a = -qb$.
If $0 < r < b$, then

$$a = -(q+1)b + (b - r), 0 < b - r < b.$$

Modular exponentiation

Given integer a, natural numbers m, n, compute

$$a^n \bmod m.$$

Let

$$n = \alpha_l 2^l + \alpha_{l-1} 2^{l-1} + \cdots + \alpha_1 2 + \alpha_0.$$

Then

$$a^n = a^{\alpha_l 2^l} \cdot \cdots \cdot a^{\alpha_1 2} \cdot a^{\alpha_0}.$$

The algorithm

(1) Set $c_0 \leftarrow a \bmod m$,
Suppose that c_j is defined.

(2) Set $c_{j+1} \leftarrow (c_j)^2 \bmod m$.
Suppose that $c_0, c_1 \cdots, c_l$ are all defined.

(3) Set $s = 1$.

(4) For j from 0 to l, in increasing order, if $\alpha_j = 1$, then
– set $s \leftarrow (s \cdot c_j) \bmod m$.

The time complexity

- Step (1): $\log|a|\log m$ (where $|a|$ is the absolute value of a.)
- Step (2):
  $\log n$ rounds,
  each round: $\log^2 m$
- Step (3): The same as step (2)

The total time complexity is:

$$O(\log|a|\log m + \log n \log^2 m).$$

The fundamental theorem of arithmetic

Definition

We say that a natural number n is *prime*, if there are no $a, b < n$ such that $n = a \cdot b$, and *composite*, otherwise.

Intuition: Primes are the “atomics" or “building blocks" of numbers.

Theorem

Every integer greater than 1 can be uniquely represented by the following form

$$n = p_1 p_2 \cdots p_k,$$

where p_j's are primes in increasing order.

Existence

We prove by induction on n. It is clearly for $n = 2$. For $n > 2$. Suppose by induction that the theorem holds for all $n' < n$.

Case 1 n is prime.

Done.

Case 2. Otherwise.

In this case, there is a prime p such that $n = p \cdot n_1$.

By the inductive hypothesis, there is a prime factoring $q_1 q_2 \cdot q_l$ of n_1.

Reordering $p, q_1, \cdots, q_l$ in increasing order, gives rise to a prime factoring of n.

Uniqueness

Suppose by induction that the result holds for all $n' < n$.
Suppose that

$$n = p_1 p_2 \cdots p_l$$

$$n = q_1 q_2 \cdots q_r$$

satisfying:

- all p_i, q_j's are primes
- $p_1 \leq p_2 \leq \cdots \leq p_l$
- $q_1 \leq q_2 \leq \cdots \leq q_r$.

Then both p_1 and q_1 are the least prime factor of n, giving $p_1 = q_1 = p$.
Dividing n by p, the same proof shows that $p_2 = q_2$.
Continuing the procedure, we have that $l = r$, and for each j from 1 to l, $p_j = q_j$.

Intuition

Whenever we see a natural number n, we can think of a prime factoring of n of the following form:

$$n = p_1 p_2 \cdots p_k,$$

for primes $p_1 \leq p_2 \leq \cdots \leq p_k$.

A great Challenge

The proof above describes an effective mechanism to find the prime factoring of a natural number *n*. However, it is a great challenge to find an efficient algorithm that

- runs in time complexity $\log^{O(1)} n$, and that
- finds the unique prime factoring of *n*.

Significance: The current cryptosystem depends on the hardness of prime factoring.
Good: There exists a quantum algorithm that finds prime factors of *n* in time polynomial of $\log n$.
Bad: Quantum computers are hard to build.

Roles in science

Gödel and Turing used the fundamental theorem of arithmetic to encode symbolic reasoning and the computation of Turing machines to natural numbers, so that reasoning and computation becomes a theory of natural numbers.
This actually encodes all discrete objects to natural numbers, allowing us to understand the discrete objects by the theory of numbers.
The reason is that both the encoding and decoding based on the prime factoring are computable by mechanisms.
The Theorem plays a fundamental role in both Gödel and Turing's theories, in the last century.

Questions

The fundamental questions about primes are:

1. Is there a polynomial time algorithm that decides, for a given natural number n, whether or not n is prime?
2. Is there a polynomial time algorithm that finds a prime factor of a given natural number?

The two questions are essential to Computer science.

Basic property

Theorem

If n is composite, then there exists a prime $p \leq \sqrt{n}$ such that $p|n$.

Towards a contradiction. If $n = q_1 q_2 \cdots q_l$, for $l \geq 2$ and for primes q_j. Then each $q_j > \sqrt{n}$, implying $q_1 q_2 > n$. A contradiction.

Significance: Anyway, the theorem reduces somehow the search space for a prime factor of a natural number, leading to some algorithms.

The sieve

The basic theorem in the last page suggests the method of sieving for finding all the primes not exceeding a fixed natural number n, 100 say.
The sieving proceeds as follows:

1. Find all primes less than or equal to $\sqrt{n}$. Suppose that $p_1, p_2, \cdots, p_l$ is the list of all such primes, in increasing order. Let $L = \{2, 3, \cdots, n\}$.
   Let $j = 1$
2. (Sieving cycle S_j) Cycle S_j:
   - (2a) For each $x \in L$, is p_j is a proper prime factor, i.e., $p_j | x$ and $p_j \neq x$, then sieve x out from L
   - (2b) If $j = l$, then output L and terminate.
   - (2c) If $j < l$, then set $j \leftarrow j + 1$, and go back to step 2 above.

By the theorem, L is the set of all the primes less than or equal to n.

A question

How good is the sieving? What is the limit of sieving? Chen and other Chinese mathematicians in 1960 - 1980 achieved significant results (Chen's so called $1 + 2$)

Primes are infinitely many

Theorem

There are infinitely many primes.

Theorem

There are infinitely many primes of the form $3k + 2$.

Primes of particular form

(1) (Dirichlet) For a, b with $(a, b) = 1$, there are infinitely many primes of the form $ak + b$.

(2) Erdös conjecture (1930):
For any n, there are a, b with $(a, b) = 1$ such that $ak + b$ for all $k \in \{1, 2, \cdots, n\}$ are primes.
Green, Tao, 2006, proved the conjecture. Tao was awarded the Fields Medal due to this progress.

The Prime Number Theorem

For every natural number x, let N_x be the number of primes less than or equal to x.

Theorem

(1896)

$$\lim_{x\to\infty} \frac{N_x}{\frac{x}{\ln x}} = 1. \tag{14}$$

The Prime Number Theorem -note

- Before 1896, experimental verification
- Gauss conjectured at the age of **15 or 16**
- 1896, Hadamard, and de la Valle Poussian independently proved the result using complex analytic properties of Riemann zeta function.
- 1948, Atle Selberg gave a proof without using complex analysis
- Full proof can be found in books of classical number theory

Intuitions and applications

(i) The number of primes within x is approximately $\frac{x}{\ln x}$. Therefore, for a number n randomly and uniformly chosen within x, the probability that n is a prime is

$$\approx \frac{1}{\ln n},$$

which is nontrivially large, since $\ln n << n$.

(ii) For $l = \ln x$, if $n_1, n_2, \cdots, n_l$ is within x, each of which is randomly and uniformly chosen, then with probability greater than some constant α, one of the n_i is a prime.

Primality test

1. M. O. Rabin, Probabilistic algorithm for testing primality. J. Number Theory, 12, pp, 128 - 138, 1980.
   - A Turing award achievement, efficient algorithm, used in real applications
2. Manindra Agrawal, Neeraj Kayal, Nitin Saxena, Primality is in P, Annals of Mathematics, 2008
   Time complexity $O(n^6)$, impractical at the moment.

General view

- The most **general mathematical model** for sciences
- **Networks** - the 21st century graph theory
- 丘成桐 2017: **网络的公理化研究**

  Nevertheless, we will need a new **network theory**

The Universe

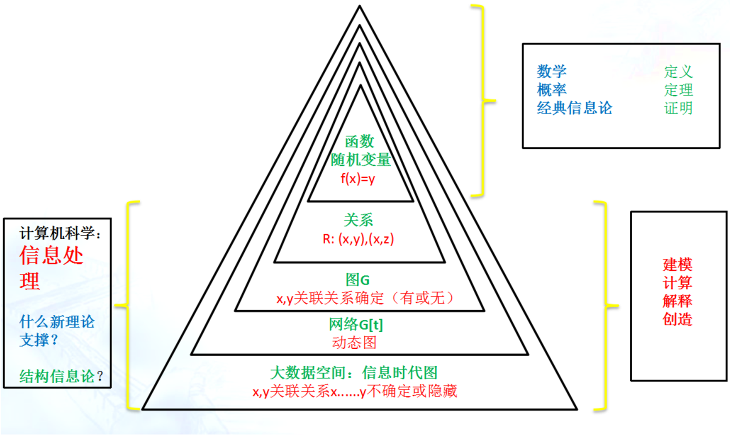

Figure: The Universe

Definition

A graph is a pair of sets, i.e., $G = (V, E)$, such that

(i) V is the set of *vertices* or *nodes*,

(ii) E is the set of *edges* or *links* of two vertices.

- An edge (u, v) has two endpoints u, v.
- If the edges are ordered pairs of vertices, then G is called a *directed graph*. Otherwise, it is an *undirected graph*.
- An edge (u, v) is called a *self-loop*, if $u = v$.
  It reflects the *reflexivity* of relations.
- An edge $e = (u, v)$ may have a weight, in which case, the graph is called a *weighted graph*.

Graph as An Extension of Relations

We know that

(i) Relation is an extension of functions, and
(ii) Graphs are extensions of relations.

Questions

1) Why?
2) Are graphs well-defined mathematical model?

Classical Graphs

Classical graphs are the mathematical model for discrete and interesting systems that are usually

- Small-scaled
- Static
- Relatively regular
- Deterministic
- Representation of the laws of links

Advantages of Graphs

- Geometric intuition
- Combinatorial characteristics
- Algebraical study

Simple Graphs

Definition

A graph $G = (V, E)$ is called *simple*, if:

(i) there is no self-loop

(ii) there is no multiple edges.

Remark: Self-loops are important for CS

Multigraphs

A graph is called a multigraph, if it contains multiple edges.

This is a special case of weighted graphs.

Directed Graphs

Definition

A graph $G = (V, E)$ is called *directed*, if the edges E are ordered pairs.

For a directed graph G, an edge $e = (u, v)$ is an ordered pair. For an undirected graph G, an edge is an unordered pair, denoted $e = \{u, v\}$, meaning that there are two directions both from u to v and from v to u.

Elements of a Graph

How to understand a graph? A graph contains

- **Syntax**
- **Semantics**
- **Structural functions**

Fundamental Questions

(1) Representations
   - Algebraic representation
   - Geometric representation

   Key: The representations must support mathematical study of graphs.

(2) Mathematical properties of graphs

(3) Operations in graphs

   Key: Graph algorithms

The Graphs in the Information Age

1) The web graph
2) Social networks
   – acquaintanceship graphs
   – friendship graphs
3) Influence graphs
4) Cooperation graphs
5) Call graphs
6) Citation graphs
7) Dependency graphs
8) Airline routes

The Graphs in the Information Age - Continued

9) Road networks
10) Protein interaction graphs
11) Computer networks
12) Mobile networks
13) Economics networks
14) Gene networks
15) Molecular topology
16) Physical systems

The Graphs in the Information Age - Continued

17) Networks that are naturally evolving
18) Engineering networks
19) Networking engineering
20) Computing systems etc

Basic Characteristics

- Large
- Dynamically evolving
- Sparse
- Laws
- Random variations
- Various interaction, communications and operations that occur in the networks

The Challenges of the New Graphs

1. What are the basic laws
2. Modelling
3. Dynamical complexity
4. Robustness and security
5. Concentration and convergence
6. New engineering
7. New economics and social sciences
8. New science in general including math, physics, CS and Information Science

Adjacency - notations

Let $G = (V, E)$ be a graph of n.

- Two vertices u and v in an undirected graph are called *adjacent* in G, if u and v are endpoints of an edge $e \in E$.
- If $e = (u, v)$ is an edge, we say that e is *incident with u and v*.
  For an edge $e = (u, v)$, we also say that
- e connects or links u and v
- u is a *neighbour of v*.

Neighbourhood

Definition

Let $G = (V, E)$ be a graph.

(1) For a vertex v, the *neighbourhood of v* is the set of all the vertices u such that u, v are adjacent.
We use $N(v)$, or $\Gamma(v)$ to denote the set of neighbours of v in G.

(2) For a set A of vertices, define

$$N(A) = \cup_{v \in A} N(v).$$

This is also referred to as

$$\Gamma(A).$$

Degree

Definition

Let $G = (V, E)$ be a graph and $v \in V$. The *degree of v* is the number of edges that link to v.
We use $d(v)$ to denote the degree of v in G.
A vertex v is called *isolated*, if it has zero degree, i.e., $d(v) = 0$.

Theorem

Let $G = (V, E)$ be an undirected graph with m edges such that none of the edges is a selfloop. Then

$$\sum_{v \in V} d(v) = 2m.$$

Every edge contributes 2 to the total degree.
Note A selfloop of a vertex v contributes degree 1 to v.

Volume

Definition

Let $G = (V, E)$ be an undirected graph and $S \subseteq V$. The *volume* of S in G is defined by

$$\mathrm{vol}(S) = \sum_{u \in S} d(u). \tag{15}$$

Degree of Vertices of Weighted Graphs

Definition

Let $G = (V, E, W)$ be an undirected graph with weight function W from E to $\mathbb{R}^+$.
For a vertex $v \in V$, define the *degree* of v in G as follows:

$$d(v) = \sum_{e=(u,v)\in E} W(e). \tag{16}$$

Remark: Graphs may have both positive and negative weights. How can we deal with the graphs of this type?

Degrees in a Directed Graph

Given a directed graph $G = (V, E)$ and a vertex v, the *in-degree* of v in G is the number of edges arrival at v, denoted $d_{\text{in}}(v)$; the *out-degree* of v in G is the number of edges leaving from v in G, denoted $d_{\text{out}}(v)$.

Theorem

Let $G = (V, E)$ be a directed graph. Then

$$\sum_{v \in V} d_{\text{in}}(v) = \sum_{v \in V} d_{\text{out}}(v) = |E|. \tag{17}$$

Complete Graph K_n

- Denote K_n
- Simple
- For any $1 \leq i, j \leq n$, $i \neq j$, there is an (undirected) edge (i, j).

Cycle C_n

A cycle of length n for $n \geq 3$ is a graph of n nodes $v_1, v_2, \cdots, v_n$ having edges

- (v_i, v_{i+1}), $1 \leq i \leq n$, and
- (v_n, v_1).

Wheel W_n

For $n > 3$, the wheel W_n consists of a cycle C_{n-1} and a center vertex u such that for each vertex v of cycle C_{n-1}, there is an edge between u and v.

n-Cube Q_n

For each n, and n-dimensional hypercube or n-cube, written Q_n, consists of

- V consists of $(a_1, a_2, \cdots, a_n)$ where each $a_i = 0$ or 1.
- For $\alpha = (a_1, a_2, \cdots, a_n)$ and $\beta = (b_1, b_2, \cdots, b_n)$, there is an edge between α and β if and only if there is exactly one i such that $a_i \neq b_i$.

Bipartite Graphs

Definition

A graph $G = (V, E)$ is called *bipartite*, if:

(i) G is simple

(ii) there is a partition L and R of V such that all the edges of G are between nodes of L and the nodes of R.

Independent Set

Definition

Let $G = (V, E)$ be a graph and $S \subset V$, we say that S is an *independent set* of G, if there is no edges among vertices in S at all.

Independent set problem Find the independent set of maximum size in G.

Characteristics of Bipartite Graphs

Theorem

A simple graph is bipartite if and only if it is possible to assign one of two colores to each vertex of the graph such that no two adjacent vertices are assigned the same color.

Complete Bipartite Graphs

A *complete bipartite graph*, written $K_{l,r}$, is a graph that is partitioned into two subsets of l and r vertices X and Y such that

- both X and Y are independent sets, and
- for every two vertices $x \in X$ and $y \in Y$, there is exactly one edge between x and y.

Induced subgraph

Given a graph $G = (V, E)$ and a subset $S \subset V$, the *induced subgraph of S in G* is the graph with vertices S and edges of E whose two endpoints are both in S.

Operations

There are many operations for graphs and in graphs:

- Edge contraction
- removal of vertices or edges
- graph union
- graph reduction
- interaction
- communication
- transportation
- virus spreading

Applications - Local Area Networks

- star
- cycle
- star $+$ cycle
- new structures, optimal structures? what are the principles? (Big challenge)

Applications - Parallel Computation

Sequential vs **Parallel**

Basic structures of parallel computation:

- K_n
- Path
- Grid
- Q_n, n-cube

Requirements: Most works are done by individual processors, and the interactions among the different processors are small. What is the mathematical measure for these intuitive requirements? (A big challenge)

Open Questions

1. What are the principles for the network of computing systems?
2. What is the correct model of cloud computing?

Data Structure

Classic data structures:

- linear ordering
- grid
- trees
- table

Open Questions

1. What is the principle of classical data structure? Why?
2. What is the principle for the structure of big data?

Graph Reachability

A **graph** $G = (V, E)$ is a finite set V of **vertices** and a set E of **edges**, which are pairs of vertices.
The **reachability problem** is: Given a graph G and two vertices $1, n \in V$, is there a path from 1 to n?
The representation of G: as math or as input of algorithms, key factor: **the size**

1. Adjacency matrix A, where

$$A_{i,j} = \begin{cases} 1, \text{if there is an edge from } i \text{ to } j, \\ 0, \text{otherwise} \end{cases}$$

2. List of tables

Algorithm

Input: Graph $G = (V, E)$, vertices $1, n \in V$.
The algorithm proceeds as follows: Let S be a set of vertices enumerated by the algorithm

1. (Initialization) Set $M = S = \{1\}$.
2. If $M = \emptyset$, then terminate.
3. Otherwise. Then for every $i \in M$,
   (i) If all the neighbors of i are in S, then **Mark** i, and remove i from M,
   (ii) If there is a $j \in V \setminus S$, such that there is an edge from i to j, then
   – **Enumerate j's into both M and S**.

Proof - I

Let S be the set generated by the algorithm.

Lemma

For any $i \in V$, i is marked if and only if there is a path from 1 to i in G.

Proof.

(For $\Rightarrow$). By induction on the time order that an enumerated in S. Let $S[t]$ be the set constructed by the algorithm at the end of time step t. Suppose that the result holds at the end time step t, and that i is enumerated into S at time step $t+1$. By the algorithm there is a vertex i' such that $i' \in S[t]$ and there is an edge from i' to i. Then there is a path from 1 to i. $\square$

Proof - II

Let S be the set generated by the algorithm.

Lemma

For any $i \in V$, $i \in S$ if and only if there is a path from 1 to i in G.

Proof.

(For $\Leftarrow$). Suppose that there is a path from 1 to i. We prove by induction on the the distance from 1 to i.
Let k be the distance from 1 to vertices. $k = 0$. In this case, the only vertex is 1.
Suppose that all vertices with distances from 1 within k are in S. Let t be the least time step at the end of which all vertices within distance k from 1 are in S, that is, $S[t]$.
By the algorithm for any vertex i, if $d(1, i) = k + 1$, then either $i \in S[t]$ or there is a time step $t' > t$ such that i is enumerated into S at time step t'.

$\square$

The Time Complexity

The time complexity is:

$$O(|E|)$$

Proof.
For every vertex i, the time of steps the algorithm runs at most $d(i)$ times. Therefore, the number of steps the algorithm runs is at most the number of edges, which is at most n^2. $\square$

Depth-First Search

At any stage t, suppose that M is the following sequence

$$x_1, x_2, \cdots, x_l$$

with the ordering as they are listed in M.
The implementing of the algorithm always chooses $i = x_l$, the element most recently enumerated in M.

Width-First Search

In the implementation of the algorithm, the algorithm always chooses i to be the earliest enumerated into M.

A tree with root v:

Any branch of the tree is a shortest path.

Polynomial Time Algorithms

$$O(n^k),$$

where k is a constant independent of n, and n the size of the input.

Space Complexity of Reachability

Linear Programming

A **Linear Programming, LP and Dual** in general form:

$$
\begin{array}{lllr}
\text{Primal} & & \text{Dual} & \\
\min c' \cdot x & & \max \pi' b & (18) \\
a_i' x = b_i, & i \in M & \pi_i \lessgtr 0 & \\
a_i' x \geq b_i, & i \in \bar{M} & \pi_i \geq 0 & \\
x_j \geq 0, & j \in N & \pi' a_j \leq c_j & \\
x_j \lessgtr 0, & j \in \bar{N} & \pi' a_j = c_j &
\end{array}
$$

Primal and Dual

Theorem

If an LP has an optimal solution, so does its dual, and at optimality their costs are equal.

Maximum Flow

A **network** $N = (V, E, s, t, c)$ is a graph $G = (V, E)$ with two specific vertices s, referred to as **source** and t, referred to as **sink**, and a **capacity** function c such that for every edge (i, j), the capacity $c(i, j) \geq 0$. The source s has no incoming edges, and the sink t has no outgoing edges.

A **flow** f of network N is an assignment of edges satisfying:

(i) For every edge (i, j), $f(i, j) \leq c(i, j)$,

(ii) For every vertex $x \in V$, if $x \neq s,\ x \neq t$,

$$\sum_{y \in V} f(y, x) = \sum_{z \in V} f(x, z). \tag{19}$$

The **value** of the flow f is $\sum_{x \in V} f(s, x) = \sum_{y \in V} f(y, t)$.

The maximum flow problem is: Given a network N, find a flow of the largest possible value.

Linear Programming

Let $|V| = n$ and $|E| = m$. Let the flow in arc (x, y) be $f(x, y)$. The maximum flow problem is the following **Linear Programming (LP)** of the maximum flow:

$$\begin{aligned} \max v \\ Af + dv = 0 \\ f \leq c \\ f \geq 0 \end{aligned} \tag{20}$$

where A is the node-arc matrix, $d \in \mathbb{R}^n$ defined by

$$d_i = \begin{cases} -1, & \text{if } i = s \\ +1, & \text{if } i = t \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

An $s-t$ Cut

Definition

An $s-t$ **cut** is a partition $(W, \bar{W})$ of the nodes of V into sets W and the complement $\bar{W}$ of W such that $s \in W$ and $t \in \bar{W}$. The **capacity** of the $s-t$ cut is

$$c(W, \bar{W}) = \sum_{(x,y) \in E \text{ such that } x \in W \text{ and } y \in \bar{W}} c(x, y). \tag{22}$$

The Dual Programming of the Max Flow

We introduce n variables $\pi(x)$ for $x \in V$, and m variables $\gamma(x, y)$ for all arcs (x, y) in E.
The **dual programming** of Equation (20) is:

$$\min \sum_{(x,y)\in E} \gamma(x, y) \cdot c(x, y) \tag{23}$$

$$\begin{aligned} \pi(x) - \pi(y) + \gamma(x, y) \geq 0, \text{ for all } (x, y) \in E \\ -\pi(s) + \pi(t) \geq 1 \\ \pi(x) \geq 0 \\ \gamma(x, y) \geq 0. \end{aligned}$$

Primal-dual

Theorem

Every s-t cut determines a feasible solution with cost $C(W, \bar{W})$ to the dual of max-flow as follows:

$$\gamma(x, y) = \begin{cases} 1, & \textit{if } (x, y) \textit{ such that } x \in W \textit{ and } y \in \bar{W}, \\ 0, & \textit{otherwise.} \end{cases} \tag{24}$$

$$\pi(x) = \begin{cases} 0, & x \in W \\ 1, & \textit{otherwise} \end{cases} \tag{25}$$

Max-flow min-cut theorem

Theorem

The value v of any s-t flow is no greater than the capacity $C(W, \bar{W})$ of any s-t cut. Furthermore, the value of the maximum flow equals the capacity of the minimum cut, and a flow f and cut $(W, \bar{W})$ are jointly optimal if and only if

(i) *For each $(x, y) \in E$ with $x \in \bar{W}$ and $y \in W$, $f(x, y) = 0$.*

(ii) *For each $(x, y) \in E$ with $x \in W$ and $y \in \bar{W}$, $f(x, y) = c(x, y)$.*

Augmenting Path

Given a flow network $N = (V, E, s, t, c)$ and a feasible s-t flow f, and **augmenting path** P is a path from s to t in the **undirected** graph resulting from $G = (V, E)$ by ignoring arc directions, satisfying:

(a) For every arc $(x, y) \in E$ that is traversed by P in the forward direction, referred to as **forward arc**, we aways have $(f(x, y) < c(x, y)$.

(b) For every arc $(y, x) \in E$ that is traversed in the reverse direction, referred to as **backward arc**, we have $f(y, x) > 0$.

Updating flow along an augmenting path

Suppose that P is an augmenting path. We can **increase** the flow from s to t while maintaining the **flow conservation** at every node as follows:

(1) Increase the flow on every forward arc on P, if any

(2) Decrease the flow of every backward arc on P, if any.

The **maximum amount of flow augmentation possible along** ***P*** **is**:

$$\delta = \min_{\text{arcs of P}} \{c(x,y) - f(x,y) \text{ for forward arcs, } f(y,x) \text{ for backward arc}\} \quad (26)$$

Definition of Labelling and Scanning

Every node x will be assigned a **label** of the form:

$$\mathrm{label}(x) = (L_1[x], L_2[x]),$$

where $L_1[x]$ denotes the node from which x is labelled, and $L_2[x]$ denotes the amount of extra flow that can be brought to x from s.

Labelling and Scanning

The process of **labelling** from x is called **scanning** x, which proceeds as follows:
Case 1: Node y is unlabelled, succeeds x (forward arc (x, y)) and $f(x, y) < c(x, y)$

$$\begin{aligned} L_1[y] &:= x \\ L_2[y] &:= \min\{L_2[x], c(x, y) - f(x, y)\} \end{aligned}$$

Case 2: y is unlabelled, precedes x (backward arc (x, y)) and $f(y, x) > 0$.

$$\begin{aligned} L_1[y] &:= -x \\ L_2[y] &:= \min\{L_2[x], f(x, y)\} \end{aligned}$$

Informal Description of Algorithm

1. Starts by scanning s and adding to LIST all nodes labelled from s.
2. The process is repeated - a node x is selected from LIST and scanned, and all nodes labelled from x are added to LIST.
3. The process terminates either t gets labelled, or we can construct an augmenting path backwards from t using L_1.

Ford and Fulkerson Algorithm

1. Set $f := 0$, $\mathrm{LIST} = \{s\}$, $L_2[s] = \infty$.
2. If $\mathrm{LIST} = \emptyset$, terminate.
3. Otherwise, then let $x \in \mathrm{LIST}$,
   3.1 scan x
   3.2 delete x from LIST
4. if t is labelled, terminate with an OPT
5. Otherwise, find an augmenting path, and go back

Scann x

1. Label forward to all unlabelled nodes adjacent to x by arcs that are unsaturated, putting newly labelled nodes on LIST;
2. Label backward to all unlabelled nodes from which x is adjacent by arcs that have positive flows, putting newly labelled nodes on LIST.

Correctness

Theorem

When the Ford and Fulkerson labelling algorithm terminates, it does so at optimal flow.

Proof.

Let W be the set of all nodes **labelled**, and $\bar{W}$ the set of **unlabelled**. All arcs (x, y) directed from W to $\bar{W}$ must be saturated; otherwise, y would have been labelled when x was scanned. Similarly, all arcs (y, x) directed from $\bar{W}$ to W must be empty; otherwise y would also have been labelled when x was scanned. Therefore, $(W, \bar{W})$ is a min-cut and the flow must be optimal. $\square$

Hard Problems in Graphs

There are many hard problems in graphs, such as :

1. Graph isomorphism problem
2. Clique
3. Traveling salesman problem

Assignments - 1

(1) Show that if n and k are positive integers, then $\lceil \frac{n}{k} \rceil = \lfloor \frac{n-1}{k} \rfloor + 1$.

(2) Design an algorithm that, on binary representations of integers a and b, determines whether $a > b$, $a = b$, or $a < b$. Analyse the time and space complexity of your algorithm.

(3) How many zeros are there at the end of $100!$?

(4) Show that $\log_2 3$ is an irrational number.

(5) Show that if $2^n - 1$ is prime, then n is prime.

Assignments - 2

(6) For any simple graph of 6 vertices, either there is a clique of size 3, or there is an independent set of size 3.

(7) Let $G = (V, E)$ be a connected, undirected multigraph with n vertices. A cut C of G is a set of edges of G whose removal results in G being disconnected. The minimum cut problem is to find the cut of the least number of edges in G. Let k be the size of the minimum cut of G. Show that

0.1 The least degree of vertices of G is at least k

0.2 The number of edges in G is at least $\frac{kn}{2}$.

Define the operation of contraction as follows: Pick an edge uniformly at random and merge the two endpoints of the edge together.
Parallel edges are always kept.

(8) An edge contraction does not reduce the size of the minimum cut.

Thank You!