Asymptotic Notation

2021年6月14日 上午 11:18

$O(n)$

$f(n) \in O(n)$ if $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

$\Omega(n)$

$f(n) \in \Omega(n)$ if $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

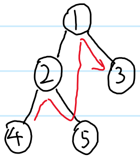
$\Theta(n)$

$f(n) \in \Theta(n)$ if $f(n) \in O(n)$ and $f(n) \in \Omega(n)$

Tree Traversal

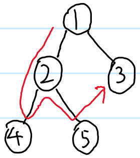
2021年6月14日 上午 11:31

inorder – left, root, right



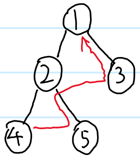
42513

preorder root, left, right



12453

postorder left, right, root



45231

Binary Search Tree

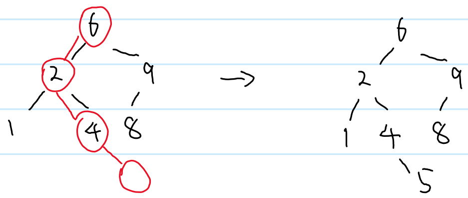
2021年6月14日 上午 11:33

- a rooted ordered tree in which every node has at most two children
- each child in a binary tree is either a left child or a right child

Insertion

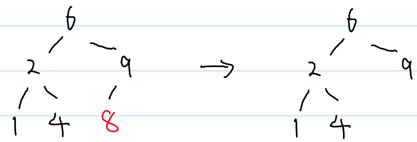
1. perform a search for the key and insert at the node reached

example insert 5

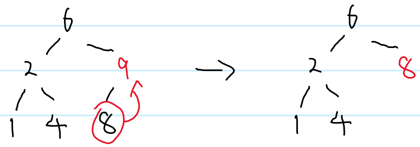


Deletion

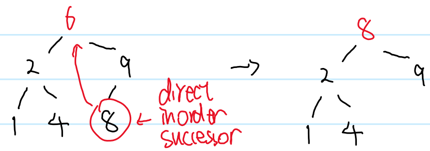
case 1: no child - directly remove



case 2. one child - replace with child



case 3: two children - replace with direct inorder successor



BST analysis

BST analysis

space: $O(n)$

search: $O(h)$ ← height

height: $O(n)$ worst case

$O(\log n)$ best case

NP-complete

a NP problem which is also NP-hard

example circuit-SAT is NP-complete

Proof: 1. circuit-SAT is NP proved above

2. circuit-SAT is NP-hard:

every NP problem can be computed using a boolean circuit (i.e. a computer); this circuit has polynomial number of elements hence can be constructed in polynomial time. Therefore $\forall L \in NP \Rightarrow L \xrightarrow{poly}$ circuit-SAT, it is NP-hard.

Therefore, circuit-SAT is NP-complete

example CNF-SAT is NP-complete

Conjunctive normal form (CNF) is a boolean formula in the form of $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ where each C_i is in the form of $x_1 \vee x_2 \vee \cdots \vee x_n$ (e.g. $\bar{x}_1 \vee \bar{x}_2 \vee x_3 \vee \overline{x_4}$).

CNF-SAT: given a boolean formula in CNF, is there an input so that the output is 1?

example 3-SAT is NP-complete

a special case of CNF where each C_i is in the form of $x_1 \vee x_2 \vee x_3$

3-SAT: is there an input so that the output is 1?

example 3-COL is NP-complete

given a graph G, is G 3-colorable? (every two adjacent vertices have different color)

example vertex cover is NP-complete

1. convert the problem into a decision problem
2. use a non-deterministic algorithm to generate possible proposals
3. verify the proposals in polynomial time

Example circuit-SAT is NP

Suppose we are given a boolean circuit C and we want to find whether it is possible for it to output 1 ①. We can first use a non-deterministic algorithm to repeatedly generate inputs to the circuit ②, and simply verify it by running the circuit. If the output is 0, we return "no". If the output is 1, we return "yes" and stop. Such computation clearly runs in polynomial time ③.

Example vertex cover is NP

Suppose we're given a graph G and we want to find whether there is a vertex cover with at most k vertices ①. We can first use a non-deterministic algorithm to repeatedly generate proposals of vertex cover with k vertices ②. We can then verify the certificate by checking through each edge, which clearly runs in polynomial time ③.

Polynomial-time reducibility

whether one problem instance can be transformed into another problem instance

take two languages A and B, defining some decision problem.

A is polynomial-time reducible to B if:

there is a function $f(s)$ such that $s \in A$ and $f(s) \in B$

If A is polynomial-time reducible to B, it is denoted as $A \xrightarrow{poly} B$

B is at least as hard as A

NP-hard

a problem which all NP problems can be reduced to

let M be a language defining some decision problem

M is NP-hard if $\forall L \in NP \Rightarrow L \xrightarrow{poly} M$

Prove NP-hardness by one of the following methods:

1. reduce all NP problems to the problem
2. reduce a known NP-hard problem to the problem

AVL Tree

2021年5月11日 下午9:55

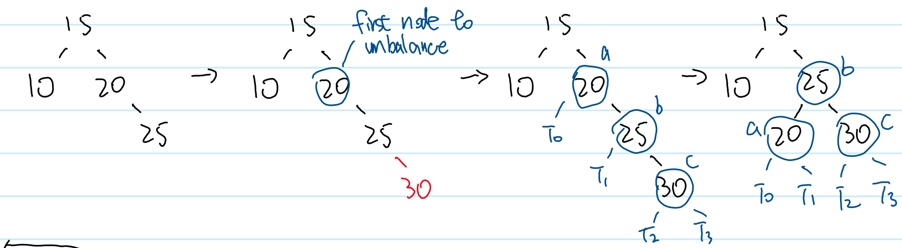
AVL Tree

- for any node, the height difference between the left and right subtree is at most 1
- $O(\log n)$ height
- uses rotations to maintain balance

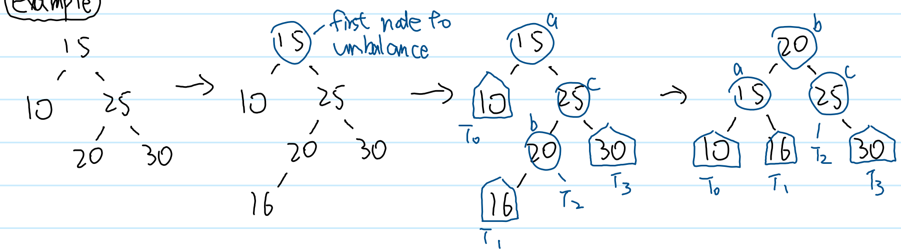
Insertion

1. if an insertion causes the tree to be unbalanced, we travel up the new node to find the first node x such that its grandparent z is unbalanced
2. let (a, b, c) be an inorder listing of x, y, z, and (T_0, T_1, T_2, T_3) be an inorder listing of their subtrees
3. replace a with b, whose children are now a and c, and let T_0, T_1, T_2, T_3 be the subtrees of a and c from left to right (inorder)

Example



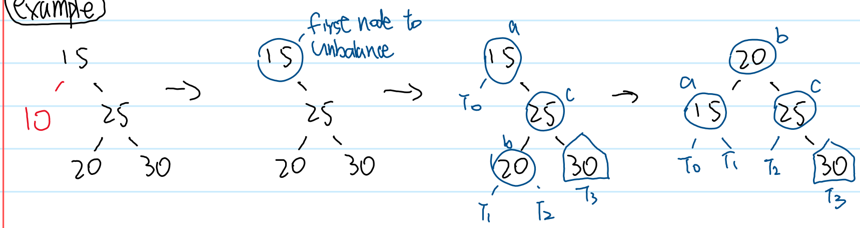
Example



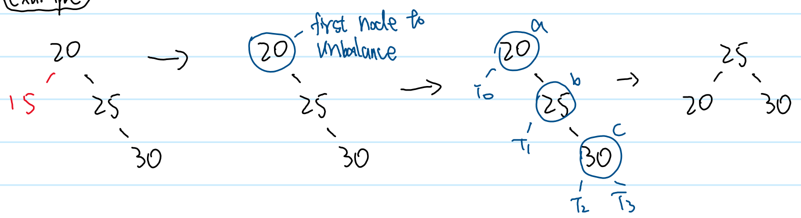
Removal

- similar

example



example



AVL tree analysis

rotation: $O(1)$

search, insertion, removal : $O(\log n)$

height: $O(\log n)$

proof suppose an AVL tree of height h can store a minimum of $n(h)$ nodes.

we can get from observation that

$n(1) = 1$

$n(2) = 2$

when $n > 2$, $n(h) = \overbrace{1}^{\text{root}} + \overbrace{n(h-1) + n(h-2)}^{\text{two subtrees}}$

we have: $n(h-1) > n(h-2)$

therefore, $n(h) > 2n(h-2)$

$n(h) > 2n(h-2)$

$> 2(2n((h-2)-2)) = 2^2 n(h-4)$

$> 2^3 n(h-6)$

$> \cdots$

$> 2^i n(h-2i)$ where $n(h-2i) = n(1)$ or $n(2)$

so $h-2i = 1$ or $h-2i = 2$ $\Rightarrow$ $i = \frac{h-1}{2}$ or $i = \frac{h-2}{2}$

hence $n(h) > 2^{\frac{h-1}{2}}$ or $n(h) > 2^{\frac{h-2}{2}}$

taking logarithms, we get $\log n(h) > \frac{h-1}{2}$ or $\log n(h) > \frac{h-2}{2}$

that is, $h < 2\log n(h) + 1$ or $h < 2\log n(h) + 2$

therefore the height of an AVL tree is $O(\log n)$

yes | no

S is a composite | undecided

the non-deterministic algorithm takes n as input and generates a, the possible proposals of factors of n, which is called certificate.
the certifier runs in polynomial time, takes n and a as input, and outputs whether the a is a factor of n. Therefore, if the answer is yes, the algorithm will terminate in polynomial time

It is clear that if the non-deterministic algorithm runs in polynomial time, then the whole algorithm runs in polynomial time too, hence NP. However, these type of algorithms don't currently exist.

example 0-1 knapsack is NP

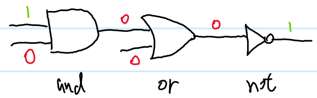
let $I = \{i_1, i_2, \cdots, i_n\}$ be a collection of items where each item has a weight w and benefit b. Given a maximum weight W.

decision is there a subset of I that values at least k?

optimization find the subset of I with the maximum value.

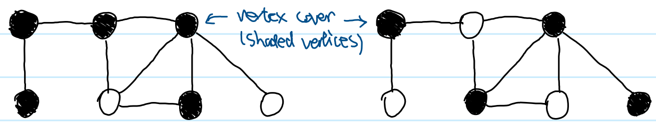
example Circuit-SAT

given a boolean circuit, is there an input so that the output is 1?



example Vertex cover

given a graph $G = (V, E)$, a vertex cover is a subset $C \subseteq V$ such that for every edge $(u, v) \in E$, $u \in C$ or $w \in C$.



decision is there a vertex cover containing at most k vertices?

optimization find the vertex cover with the minimum number of vertices.

Prove that a problem is NP

1. convert the problem into a decision problem

2. [illegible]

NP Completeness

2021年6月1日 下午 9:04

Desicion vs optimization problem

a decision problem is a problem to which the answer is either yes or no

an optimization problem is a problem to which the answer is a maximum/minimum value

If a decision problem is hard then its related optimization problem is also hard

example

decision problem

let G be a weighted connected graph, does G have a minimum spanning tree of weight at most k?

optimization problem

let G be a weighted connected graph, find the minimum spanning tree with the minimum weight.

Complexity class P - polynomial time

the set of decision problems that can be solved in $O(p(n))$ ← a polynomial on n

by some deterministic algorithm

(polynomial solvable)

example

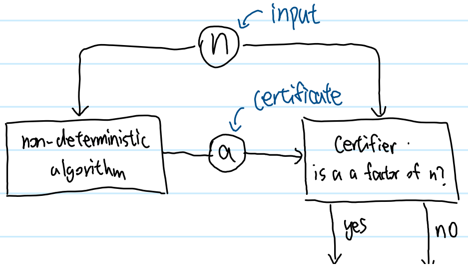
- single-source shortest path
- fractional knapsack
- task scheduling

Complexity class NP - non-deterministic polynomial time

the set of decision problems for which there exists an efficient certifier

(polynomial verifiable) ← and polynomial solvable if such a non-deterministic algorithm can be found

example given a number n, is it a composite (not prime)?



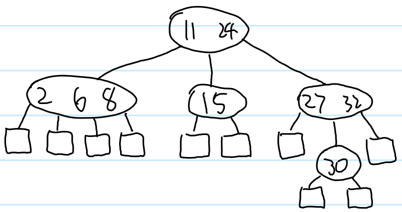
that is, $h < 2\lg n(h) + 1$ or $h < 2\lg n(h) + 2$

therefore the height of an AVL tree is $O(\lg n)$

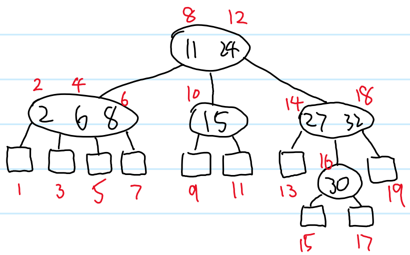
(2, 4) Tree

2021年5月11日　　下午 9:58

Multi-Way Search Tree

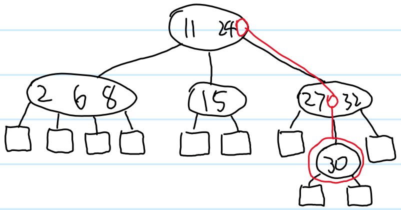


Inorder traversal



Multi-way searching

example search for 30

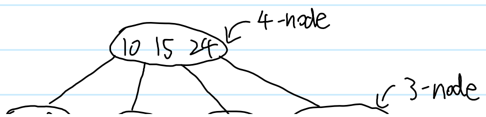


(2,4) Tree

- Multi-way search tree with the following properties:

Node-size property: each node has at most 4 children

Depth property: all the external nodes have the same depth.



therefore, 9 decrypts to 59.

init

1. choose two prime numbers p, q ; let $n = pq$
2. calculate $\varphi(n)$. $\varphi(n) = (p-1)(q-1)$
3. choose e relatively prime to $\varphi(n)$ i.e., $\gcd(e, \varphi(n)) = 1$
4. calculate d. $d = e^{-1}$ in $Z_{\varphi(n)}$
5. message M in Z_n

keys

public key : $K_E = (n, e)$

private key : $K_D = d$

encryption

$C = M^e \bmod n$

decryption

$M = C^d \bmod n$

example suppose $p=5$, $q=17$, $e=13$, first find d, then encrypt 37, finally decrypt 9.

$n = pq = 85$

$\varphi(n) = (p-1)(q-1) = 64$

to find d, we first calculate e^{-1} in $Z_{\varphi(n)}$

q	r_1	r_2	r	t_1	t_2	t
4	64	13	12	0	1	-4
1	13	12	1	1	-4	5
12	12	1	0	-4	5	-64
	1	0		5 ← e^{-1}		

since $5 \in [5]_{64}$ and $5 \in Z_{\varphi(n)}$, e^{-1} in $Z_{\varphi(n)}$ is 5.

$d = 5$

then, to encrypt 37, we have:

$C_1 = M_1^e \bmod n = 37^{13} \bmod 85$

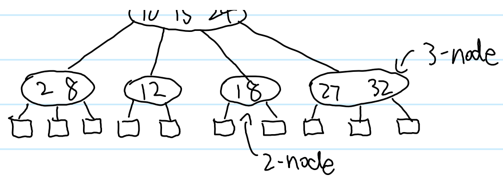
$37^{13} = 37^{12} \cdot 37$	$37^3 \bmod n = 78$
$37^{12} = (37^6)^2$	$37^6 \bmod n = (37^3 \bmod n)^2 \bmod n = 49$
$37^6 = (37^3)^2$	$37^{12} \bmod n = (37^6 \bmod n)^2 \bmod n = 21$
	$37^{13} \bmod n = (37^{12} \bmod n)(37 \bmod n) \bmod n = 12$

therefore, 37 encrypts to 12.

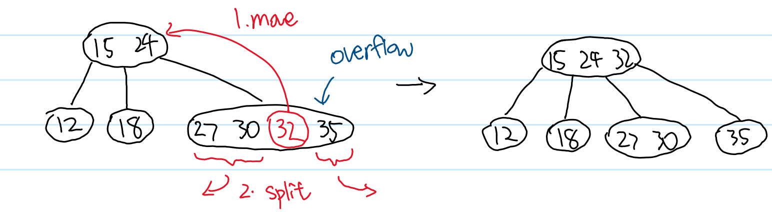
finally, to decrypt 9, we have:

$M_2 = C_2^d \bmod n = 9^5 \bmod 85 = 59$

therefore, 9 decrypts to 59.

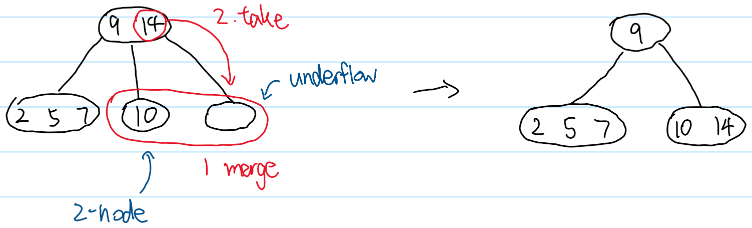


Insertion

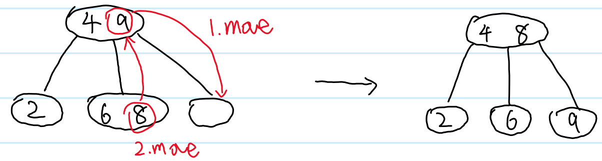


Deletion

case 1 : adjacent siblings are 2-nodes



case 2 : there exist a sibling which is a 3 or 4-node



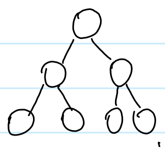
Analysis

height : $O(\log n)$

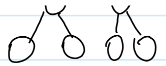
proof let h be the height of a $(2,4)$ tree containing n items

in the worst-case scenario, all nodes are 2-node

height	items
1	1
2	2
h	2^h



h 2^h



we have: $n \geq 1 + 2 + 4 + \cdots + 2^h$

since the sum of geometric series is:

$$\sum_{k=1}^{n} (ar^k) = a\left(\frac{r^n - 1}{r - 1}\right)$$

and in our case, $a=1$, $r=2$, $n=h$

we have. $n \geq 2^h - 1$

therefore, $h \leq \log(n+1)$

the height of an (2,4) tree is $O(\log n)$

Cryptography

2021年6月1日 下午 7:53

Substitution cipher

replace each letter with another letter: $y = \pi(x)$

when decrypting, replace back: $x = \pi^{-1}(y)$

example

A B C D

A B C D

BAD CAB → ACD BCA

example the Caesar cipher

move each letter by k

A B C D

A B C D

← $k=3$

BAD CAB → ADC BDA

Problem

vulnerable to statiscal attacks

One-time pad

1. a key as long as the message is shared
2. perform XOR to the message with the key to encrypt / decrypt
3. only secure if the key is used once

example

message	0 1 0 0 1 1 1 1
pad	1 1 0 1 0 1 1 1
cipher	1 0 0 1 1 0 0 0
pad	1 1 0 1 0 1 1 1
message	0 1 0 0 1 1 1 1

RSA

plaintext → $\boxed{\text{encrypt}}$ (public key) → ciphertext → $\boxed{\text{decrypt}}$ (private key) → plaintext

Algorithm

init

1. choose two prime numbers p, q; let $n = pq$

let p be a prime, and let x be a integer such that $x \bmod p \neq 0$, then:

$x^{p-1} \equiv 1 \pmod p$

example

$1^4 \bmod 5 = 1$

$2^4 \bmod 5 = 16 \bmod 5 = 1$

$4^4 \bmod 5 = 256 \bmod 5 = 1$

Corollary

let p be a prime, and let x be a nonzero residue of Z_p, then:

$x^{-1} \equiv x^{p-2} \pmod p$

Euler totient function

the size of Z_n^*, denoted as $\varphi(n)$

1. get the prime factorisation of n: $n = p_1^{e_1} \cdot p_2^{e_2} \cdots \cdot p_m^{e_m}$ the power is ignored
2. $\varphi(n) = n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_n})$

Example evaluate $\varphi(28)$

method 1: enumerate Z_{28}^*

$Z_{28}^* = \{1,3,5,9,11,13,15,17,19,23,25,27\}$

the size of Z_{28}^* is 12, hence $\varphi(28) = 12$

method 2: calculate

$28 = 2^2 \times 7$

$\varphi(28) = 28(1-\frac{1}{2})(1-\frac{1}{7}) = 28 \cdot \frac{1}{2} \cdot \frac{6}{7} = 12$

Heap

2021年5月11日　　下午 10:00

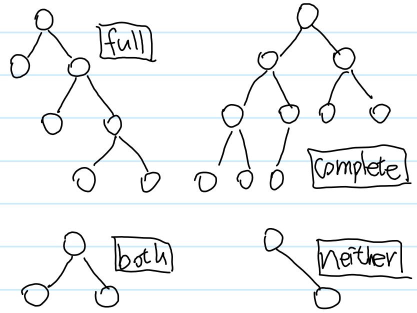
Heap

- implementation of priority queue, efficient for both insertions and deletions
- $O(\log n)$ insertion or deletion
- almost a complete binary tree

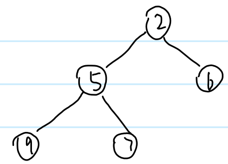
for a binary tree:

full: each node is either a leaf or has two children

complete: every level except possibly the last is full, nodes of the last level are left-alligned

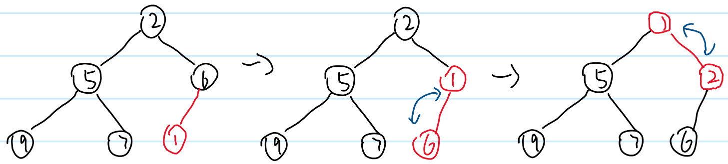


Example Min-heap: $key(v) \geq key(parent(v))$



Insertion: up-heap bubbling

1. insert to the last node
2. if $key(v) < key(parent(v))$, bubble up
3. when $key(v) \geq key(parent(v))$ or reached to root, stop.

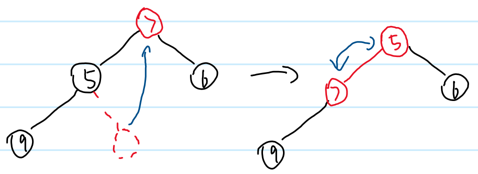


Pop (top element)

1. replace root with last node v
2. swap v downward
3. stop upon reaching a leaf or a node whose children have keys greater than v's

2. swap v downward

3. Stop upon reaching a leaf or a node whose children have keys greater than v's.



Heap analysis

Space: $O(n)$

Insert, remove: $O(\log n)$

size, isEmpty, min: $O(1)$

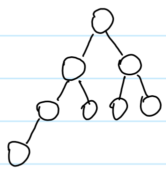
Heap-based priority queue sorting: $O(n \log n)$

height: $O(\log n)$

proof: Let h be the height of a heap sorting n keys

in the worst-case scenario, there is only 1 key at height h.

height	keys
1	1
2	2
$h-1$	2^{h-1}
h	1



we have $n \geq 1+2+4+\cdots+2^{h-1}+1$

$n \geq 2^{h-1}$

$h \leq \log n + 1$

therefore, the height of a heap is $O(\log n)$

example $10x \equiv 2 \pmod{15}$

first, calculate $\gcd(10,15)$

q	r_1	r_2	r
1	15	10	5
2	10	5	0
	5	0 ← stop	

↖ $\gcd(15,10)$

we have $\gcd(10,15)=5$.

Since $5 \nmid 2$, there is no solution.

example $14x \equiv 12 \pmod{18}$

first, calculate $\gcd(14,18)$

q	r_1	r_2	r
1	18	14	4
3	14	4	2
2	4	2	0
	2	0 ← stop	

↖ $\gcd(14,18)$

we have $\gcd(14,18)=2$

since $2 \mid 12$, there are 2 solutions

$14x \equiv 12 \pmod{18}$

$\Rightarrow 7x \equiv 6 \pmod 9$

$\Rightarrow x \equiv 6(7^{-1}) \pmod 9$

we calculate $7^{-1} \bmod 9$

q	r_1	r_2	r	t_1	t_2	t
1	9	7	2	0	1	-1
3	7	2	1	1	-1	4
2	2	1	0	-1	4	-9
	1	0 ← stop		4 ← $7^{-1} \bmod 9$		

therefore, $7^{-1} \bmod 9$ is the element in $[4]_9$ and $\mathbb{Z}_9$, which is 4. So $7^{-1} \bmod 9 = 4$

$\Rightarrow x = 24 \pmod 9 = 6$

therefore, the answer are the elements in $[6]_9$ and $\mathbb{Z}_{18}$, which are: 6 and 15

hence $x_0=6$ and $x_1=15$.

Modular exponentiation

Fermat's little theorem

let p be a prime, and let x be a integer such that $x \bmod p \neq 0$, then:

$x^{p-1} \equiv 1 \pmod p$

Solution: since $\gcd(10, 8) = 2 \neq 1$, there is no mulplicative imerse

example

$Z_{10} = \{0,1,2,3,4,5,6,7,8,9\}$

3 and 7 $\in Z_{10}$ and $3 \times 7 \equiv 1 \pmod{10}$

therefore, 3 and 7 are multiplicative inverses of each other

Algorithm

$r_1 = n \quad r_2 = a$

$t_1 = 0 \quad t_2 = 1$

while $r_2 > 0$:

$q = r_1 / r_2$

$r = r_1 - q r_2$

$r_1 = r_2$

$r_2 = r$

$t = t_1 - q t_2$

$t_1 = t_2$

$t_2 = t$

if $r == 1$:

$a^{-1} = t_1$

example find the multiplicative inverse of 8 mod 11

q	r_1	r_2	r	t_1	t_2	t
1	11	8	3	0	1	-1
2	8	3	2	1	-1	3
1	3	2	1	-1	3	-4
2	2	1	0	3	-4	11
	1	0 ← stop		-4 ← 8^{-1}		

hence the multiplicative inverse of 8 mod 11 is x where:

$x \in [-4]_{11}$ and $x \in Z_{11}$

we have $x = 7$.

$[-4]_{11} = \{\cdots, -15, -4, 7, \cdots\}$

Multiplicative group

the set of elements in Z_n that are relatively prime to n, denoted as Z_n^*

note that $\gcd(n, 0) = n$, therefore 0 is never in Z_n^*

example

$Z_6 = \{0,1,2,3,4,5\} \quad Z_6^* = \{1,5\}$

$Z_5 = \{0,1,2,3,4\} \quad Z_5^* = \{1,2,3,4\}$

note that when n is prime, $Z_n^* = Z_n - \{0\}$

Single-variable linear equation

an equation in the form of $ax \equiv b \pmod{n}$

Number of solutions

1. calculate $\gcd(a, n)$
2. if $\gcd(a,n) \nmid b$, then no solution

   if $\gcd(a,n) \mid b$, then $\gcd(a,n)$ solution(s)

example $10x \equiv 2 \pmod{15}$

first, calculate $\gcd(10, 15)$ | we have $\gcd(10,15) = 5$

Divide and Conquer

2021年5月11日 下午 10:02

Divide and Conquer

General method:

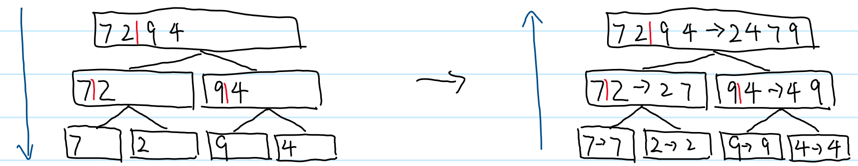
1. Divide: if the input size is small, solve directly

   else, divide the input into disjoint subsets
2. Recur: Recursively solve the subproblems associated with the subsets
3. Conquer: Take the solutions to the subproblems and merge into the solution to the original problem.

Merge Sort

- Divide: partition S into two sequences S_1 and S_2 of about $S/2$ elements each
- Recur: recursively sort S_1 and S_2
- Conquer: merge S_1 and S_2 into a sorted sequence

example



Analysis

height: $O(\log n)$

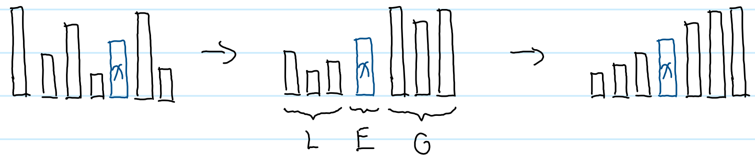
operations per level $O(n)$

time complexity: $O(n \log n)$

Quick Sort

- Divide: pick a pivot x and partition x into
  - L - elements less than x
  - E - elements equal to x
  - G - elements greater than x
- Recur: sort L and G
- Conquer: join L, E, and G

example

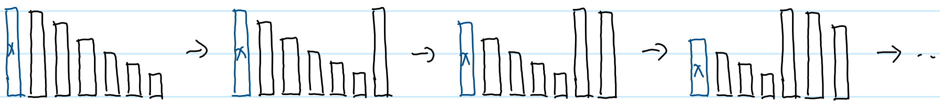


Analysis

average: $O(n\log n)$

worst: $O(n^2)$

example greatest to smallest, always choose the leftmost as pivot



Time Complexity Analysis

$$\text{given } T(n) = \begin{cases} b & \text{if } n<2 \\ 2T(n/2)+bn & \text{otherwise} \end{cases}$$

$\boxed{1}$ Substitution method

$$\begin{aligned} T(n) &= 2T(n/2)+bn \\ &= 2(2T(n/2^2)+b(n/2))+bn \\ &= 2^2\,T(n/2^2)+2bn \\ &= 2^3\,T(n/2^3)+3bn \\ &= \cdots \\ &= 2^i\,T(n/2^i)+ibn \end{aligned}$$

at the base case, we have:

$T(n) = b$

$2^i = n \Rightarrow i = \log n$

therefore:

$T(n) = bn + bn\log n$

$T(n)$ is $O(n\log n)$

$9^{12} = (9^6)^2$

$9^6 = (9^3)^2$

$9^{52} \bmod n = (9^{26} \bmod n)^2 \bmod n = 169$

$9^{53} \bmod n = (9^{52} \bmod n)(9 \bmod n) \bmod n = 25$

$9^{106} \bmod n = (9^{53} \bmod n)^2 \bmod n = 64$

$9^{107} \bmod n = (9^{106} \bmod n)(9 \bmod n) \bmod n = 15$

example prove that the remainder of an integer divided by 3 is the same as the sum of its digits divided by 3.

Proof: $a = a_0\times10^0 + a_1\times10^1 + a_2\times10^2 + \cdots + a_n\times10^n$

e.g. $6371 = 1\times10^0 + 7\times10^1 + 3\times10^2 + 6\times10^3$

we have:

$$\begin{aligned} &(a_0\times10^0 + a_1\times10^1 + \cdots + a_n\times10^n) \bmod 3 \\ &= ((a_0\times10^0)\bmod 3 + (a_1\times10^1)\bmod 3 + \cdots + (a_n\times10^n)\bmod 3) \bmod 3 \\ &= ((a_0 \bmod 3)(10^0 \bmod 3) \bmod 3 + \cdots + (a_n \bmod 3)(10^n \bmod 3) \bmod 3) \bmod 3 \\ &= (a_0 \bmod 3 \bmod 3 + a_1 \bmod 3 \bmod 3 + \cdots + a_n \bmod 3 \bmod 3) \bmod 3 \\ &= (a_0 \bmod 3 + a_1 \bmod 3 + \cdots + a_n \bmod 3) \bmod 3 \\ &= (a_0 + a_1 + \cdots + a_n) \bmod 3 \end{aligned}$$

Additive inverse

In Z_n, two numbers a and b are additive inverses of each other if:

$a+b \equiv 0 \pmod n$

example

$Z_6 = \{0,1,2,3,4,5\}$

2 and 4 $\in Z_6$ and $2+4 \equiv 0 \pmod 6$

therefore, 2 and 4 are additive inverses of each other

Mulplicative inverse

In Z_n, two number a and b are multiplicative inverses of each other if:

$ab \equiv 1 \pmod n$

the multiplicative inverse of $a \pmod n$ is denoted as $a^{-1} \pmod n$

a and b must be relative prime to n

example find the mulplicative inverse of 8 in Z_{10}

Solution: since $\gcd(10, 8) = 2 \neq 1$, there is no mulplicative inverse the set of all possible results by calculating $n \% a$ is the set of lease residues modulo n, denoted as Z_n

$Z_n = \{0, 1, 2, 3, \cdots, (n-1)\}$

example

$Z_2 = \{0, 1\}$, $Z_6 = \{0, 1, 2, 3, 4, 5\}$

Congruence (同余)

if $a \bmod m = b \bmod m$, a and b are congruent, denoted as $a \equiv b \pmod n$

example

$8 \equiv 13 \pmod 5$

$13 \equiv 18 \pmod 5$

$18 \equiv 23 \pmod 5$

$\Rightarrow \underbrace{\cdots \xleftarrow{-5} -2 \xleftarrow{-5} 3 \xleftarrow{-5} 8 \xrightarrow{+5} 13 \xrightarrow{+5} 18 \xrightarrow{+5} \cdots}_{\% 5 \; = \; 3}$

Residue class (同余类)

the set of integers congruent modulo n, denoted as $[a]_n$

example

$[3]_5 = [8]_5 = [13]_5 = \cdots = \{\cdots, -2, 3, 8, 13, 18, \cdots\}$

Properties of the mod operator

1. $(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
2. $(a-b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$
3. $(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$

example

$136 \bmod 6 = (37+99) \bmod 6 = ((37 \bmod 6) + (99 \bmod 6)) \bmod 6 = (1+3) \bmod 6 = 4$

example $9^{107} \bmod 187$

$9^{107} = 9^{106} \cdot 9$

$9^{106} = (9^{53})^2$

$9^{53} = 9^{52} \cdot 9$

$9^{52} = (9^{26})^2$

$9^{26} = (9^{13})^2$

$9^{13} = 9^{12} \cdot 9$

$9^{12} = (9^6)^2$

$9^6 = (9^3)^2$

let $n = 187$

$9^3 \bmod n = 168$

$9^6 \bmod n = (9^3 \bmod n)^2 \bmod n = 174$

$9^{12} \bmod n = (9^6 \bmod n)^2 \bmod n = 169$

$9^{13} \bmod n = (9^{12} \bmod n)(9 \bmod n) \bmod n = 25$

$9^{26} \bmod n = (9^{13} \bmod n)^2 \bmod n = 64$

$9^{52} \bmod n = (9^{26} \bmod n)^2 \bmod n = 169$

$T(n) = bn + bn \log n$

$T(n)$ is $O(n \log n)$

2 The master method

given $T(n)$ in the form of

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{otherwise} \end{cases}$$

where $d \geq 1$, $a > 0$, $c > 0$, $b > 1$

Case 1: If $f(n)$ is polynomially smaller than $n^{\log_b a}$ ($f(n)$ is $O(n^{\log_b a - \varepsilon})$), then $T(n)$ is $\Theta(n^{\log_b a})$

Case 2: If $f(n)$ is asymptotically close to $n^{\log_b a}$ ($f(n)$ is $\Theta(n^{\log_b a})$), then $T(n)$ is $\Theta(n^{\log_b a} \lg n)$

Case 3: If $f(n)$ is polynomially larger than $n^{\log_b a}$ ($f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$) and $af(n/b) \leq cf(n)$ for $c < 1$ and $n >$ some constant, then $T(n)$ is $\Theta(f(n))$

example $T(n) = 4T(n/2) + n$

we have $a = 4$, $b = 2$, $f(n) = n$

$n^{\log_b a} = n^{\log_2 4} = n^2$

hence $f(n)$ is $O(n^{2-\varepsilon})$ for $\varepsilon = 1$, we're in case 1.

therefore, $T(n)$ is $\Theta(n^2)$

example $T(n) = T(2n/3) + 1$ not $\frac{2}{3}$!

we have $a = 1$, $b = 3/2$, $f(n) = 1$

$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

hence $f(n)$ is $\Theta(1)$, we're in case 2.

therefore, $T(n)$ is $\Theta(\lg n)$

example $T(n) = T(n/3) + n$

we have $a = 1$, $b = 3$, $f(n) = n$

$n^{\log_b a} = n^{\log_3 1} = n^0 = 1$, hence $f(n)$ is $\Omega(n^{0+\varepsilon})$ for $\varepsilon = 1$

$af(n/b) = \frac{n}{3} = \frac{1}{3} f(n) < cf(n)$ when $c = \frac{2}{3}$ and $n > 1$.

$af(n/b) = \frac{n}{3} = \frac{1}{3}f(n) < cf(n)$ when $c = \frac{2}{3}$ and $n > 1$.
we are in case 3.
therefore, $T(n)$ is $\Theta(n)$

(example) $T(n) = 2T(n/2) + n\lg n$
we have $a=2$, $b=2$, $f(n) = n\lg n$
$n^{\log_b a} = n^{\log_2 2} = n$
however, $f(n)$ is not $\Omega(n^{1+\varepsilon})$
$T(n)$ does not satisfy any of the cases
the master method is not applicable.

(example) $T(n) = 2T(n^{1/2}) + \log n$
first, we try to reform $T(n)$ to be in the form of the master method.
let $k = \log n$, then $n = 2^k$
we have $T(2^k) = 2T(2^{k/2}) + k$
let us take the substitution $S(k) = T(2^k)$, we have:
$S(k) = 2S(k/2) + k$
now, we have $a=2$, $b=2$, $f(k)$
$k^{\log_b a} = k^{\log_2 2} = k$
hence $S(k)$ is $\Theta(k)$, we're in case 2.
therefore, $S(k)$ is $\Theta(k\log k)$
substitute back for $T(n)$, we get
$T(n) = \Theta(\log n \log\log n)$

Algorithm

$\boxed{a = qb + r}$
$r_1 = a$ $r_2 = b$ (init)
while $r_2 > 0$:
  $q = r_1 / r_2$
  $r = r_1 - qr_2$
  $r_1 = r_2$
  $r_2 = r$
$\gcd(a,b) = r_1$

(Example) $\gcd(25, 15)$

q	r_1	r_2	r
1	25	15	10
1	15	10	5
2	10	5	0

5 ← $\gcd(25,15)$
0 ← $r_2 > 0$ is false, stop

Extended Euclidean Algorithm

Algorithm

$\boxed{\gcd(a,b) = sa + tb}$
$r_1 = a$ $r_2 = b$
$s_1 = 1$ $s_2 = 0$
$t_1 = 0$ $t_2 = 1$ (init)
while $r_2 > 0$:
  $q = r_1 / r_2$
  $r = r_1 - qr_2$
  $r_1 = r_2$
  $r_2 = r$
  $s = s_1 - qs_2$
  $s_2 = s$
  $t = t_1 - qt_2$
  $t_1 = t_2$
  $t_2 = t$
$\gcd(a,b) = r_1$
$s = s_1$
$t = t_1$

(example) $a = 134$ $b = 52$

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
2	134	52	30	1	0	1	0	1	-2
1	52	30	22	0	1	-1	1	-2	3
1	30	22	8	1	-1	2	-2	3	-5
2	22	8	6	-1	2	-5	3	-5	13
1	8	6	2	2	-5	7	-5	13	-18
3	6	2	0	-5	7	-26	13	-18	67

2 → $\gcd(134, 52)$ 0 ← stop 7 → s -18 → t

validation:
$7 \times 134 - 18 \times 52 = 2$

Modular Arithmetic

Set of residues (剩余系)
the set of all possible results by calculating $n \% a$ is the set of lease residues

Number Theory

2021年5月31日　　下午 10:45

Set of integers

$\mathbb{Z} = \{ \dots, -2, -1, 0, 1, 2, \dots \}$

Integer division

$a = q \times n + r$

dividend　result　divisor　remainder

$1 \leftarrow q$

$n \rightarrow 3 \overline{)5} \leftarrow a$

3

$2 \leftarrow r$

Divisibility

if $r = 0$, then a is divisible by n, $n \mid a$

if $r \neq 0$, then a is not divisible by n, $n \nmid a$

example

$5 \mid 25 \quad 3 \mid -12 \quad -4 \nmid 7 \quad 6 \nmid 21$

properties　　　　　　　　　　　　example

1. if $a \mid 1$, then $a = \pm 1$　　　$-1 \mid 1 \Rightarrow -1 = -(1)$
2. if $b \mid a$ and $a \mid b$, then $a = \pm b$　　　$3 \mid -3$ and $-3 \mid 3 \Rightarrow 3 = -(-3)$
3. if $b \mid a$ and $c \mid b$, then $c \mid a$　　　$2 \mid 6$ and $6 \mid 12 \Rightarrow 2 \mid 12$
4. if $a \mid b$ and $a \mid c$, then $a \mid (mb + nc)$　　　$2 \mid 4$ and $2 \mid 6 \Rightarrow 2 \mid (3 \times 4 + 7 \times 6)$

Greatest common divisor GCD

the gcd g of two integers a and b is the greatest integer such that $g \mid a$ and $g \mid b$

$\gcd(a, 0) = a$

Relatively prime

if $\gcd(a, b) = 1$, they are relatively prime

Euclidian Algorithm

if $a = qb + r$, then $\gcd(a, b) = \gcd(b, r)$

example

$25 = 1 \cdot 15 + 10$, $\gcd(25, 15) = \gcd(15, 10)$

$15 = 1 \cdot 10 + 5$, $\gcd(15, 10) = \gcd(10, 5)$

$10 = 2 \cdot 5 + 0$, $\gcd(10, 5) = \gcd(5, 0)$

$\gcd(5, 0) = 5 \Rightarrow \gcd(25, 15) = 5$

Optimization Problems

2021年4月22日　　下午 7:02

Optimization Problems

Greedy algorithm

- makes the choice that looks the best at the moment
- does not always lead to the optimal solution

example Fractional Knapsack Problem (FKP)

- Let S be a set of items, where each item i has a positive weight w_i and benefit b_i
- Allowed to take arbitary fractions of each item x_i

Goal: find the maximum benifit subset such that it does not exceed total weight W

Solution: calculate benefit / value ratio as value index

use a heap-based max queue

i	1	2	3	4
b_i	7	9	9	2
w_i	3	4	5	2
value index	2.33	2.25	1.8	1

Use greedy strategy to chose item until knapsack full.

Analysis: Time complexity: $O(n \log n)$

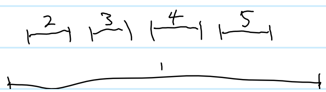
n items　　for a heap-based priority queue, taking one item is $O(\log n)$

example Interval scheduling

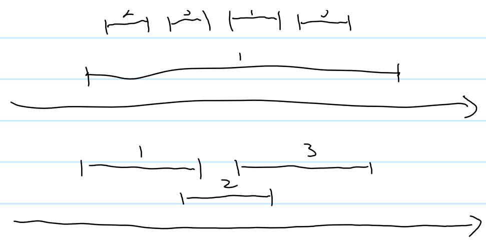
- Given a set of tasks and a machine that can run one task at a time
- Each task has a start time s_i and a finish time f_i

Goal: Select a subset of the tasks to maximize the number of tasks that we can schedule on this machine

example



(example)



Solution: Use greedy strategy to chose the task that finishes first.
Analysis: $O(n \log n)$ again

Dynamic Programming
- similar to divide-and-conquer
- stores intermediate results
- primarily used for optimization problems
- useful only when the problem has a certain structure that can be exploited

▷ characteristic
1. Optimal substructure: optimal solution consists of optimal solutions to subproblems
2. Overlapping subproblems: down the substructure, many subproblems solved more than once.

▷ idea
1. Solve problem bottom-up, building a table of solved subproblems
2. table can be in various forms

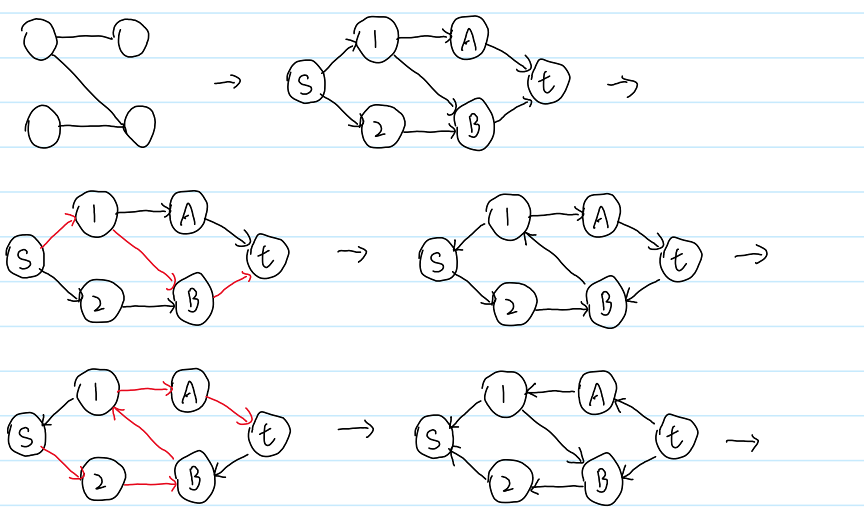
(example) 0-1 Knapsack Problem
- taking fractional items is forbidden

maximize $\sum_{i \in T} b_i$ subject to $\sum_{i \in T} w_i \leq W$

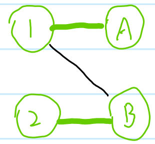
Solution: ① Enumerate.
② Dynamic programming

Let S_k be the subset of S, only containing the first k items
we can build a table $B[k, w]$ such that each cell represents the maximum benefit of selecting from S_k with weight limit w.

2. find a path from s to t, reverse each edge on the path
3. repeat 2 until no such path can be found
(example)



no more path can be found →



is a maximum bipartite matching

Bipartite Matching

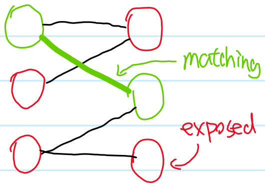
2021年5月31日　下午 7:47

A graph $G=(V,E)$ is bipartite if it can be split into 2 parts, where each edge one endpoint in each set

Terminology

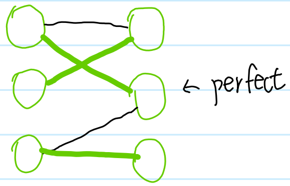
Matching

a matching $M \subseteq E$ is a set of edges with no endpoints in common



Exposed

a vertex is exposed if it is not incident to any of the edges in M



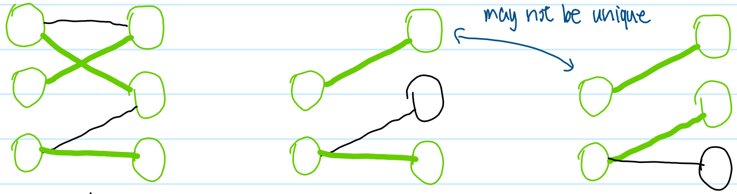
Perfect

a matching is perfect if no vertex is exposed

Maximum bipartite matching

the largest bipartite matching in the graph

△ a perfect matching is always a maximum bipartite matching



Perfect and Maximum　　Maximum　　Maximum

Solution

The maximum bipartite matching problem can be solved by reducing it to a simplified maximum flow problem, where the edge capacity is 1 everywhere:

1. construct the source node and the sink node
2. find a path from s to t, reverse each edge on the path
3. repeat 2 until no such path can be found

B[k,w] is calculated as:

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w < w_k \\ \max(B[k-1,w],\ b_k + B[k-1, w-w_k]) & \text{otherwise} \end{cases}$$

can't even fit in an empty bag

do not put in item k　or　clear up the space to put in item k

Example

Given:

i	1	2	3	4
b_i	25	15	20	36
w_i	7	2	3	6

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	25	25	25	25
2	0	0	15	15	15	15	15	25	25	40	40
3	0	0	15	20				35	35	40	45
4	0	0	15	20				36	51	56	56

try it yourself

Graphs

2021年5月31日　　下午 12:09

Graph

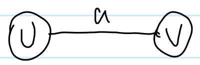
defined by $G=(V,E)$

V is a set of vertices

E is a collection of pairs of vertices from V, called edges

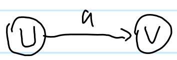
Edge type

undirected



$a=(U,V)$

directed



$a=(U,V)$

origin　destination

Terminology

End vertices (endpoints)

- U and V are the endpoints of a

Adjacent vertices

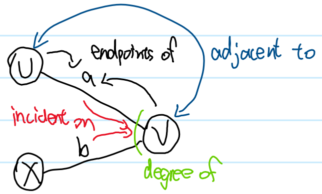
- U and V are adjacent

Edges incident

- a and b are incident on V

Degree of vetex

- V has degree 2

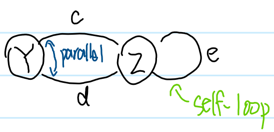


Parallel edges

- c and d are parallel edges

Self-loop

- e is a self-loop



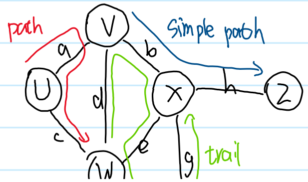
Path (walk)

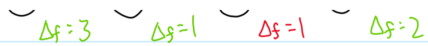
a sequence of alternating vertices and edges

begins with a vertex

ends with a vertex

- UaVaUcW is a path





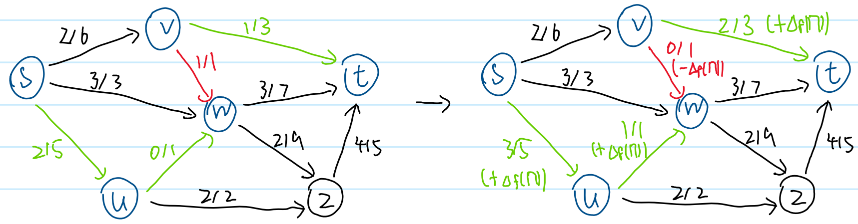
$\Delta_f(\Pi) = \min(\Delta_f(s,u), \Delta_f(u,w), \Delta_f(w,v), \Delta_f(v,t)) = 1$

$\Delta_f(\Pi)$ indicates how much the flow can be increased along the augmenting path

Procedure:

1. find an augmenting path
2. calculate $\Delta_f(\Pi)$
3. for each edge along the path:
   if it is a forward flow, increase the flow by $\Delta_f(\Pi)$
   if it is a backward flow, decrease the flow by $\Delta_f(\Pi)$
4. repeat 1-4 until no more augmenting path can be found

example



$\Delta_f(\Pi)=1$　　　$|f|=7 \rightarrow |f|=8$

Theorem

1. when the network has no augmenting path, then it is a maximum flow
2. there is a cut X such that $|f|=c(X)$ (the bottleneck)
   when f is the maximum flow, X is called the minimum cut

Minimum cut

for a network with maximum flow, a minimum cut is where:

flow of forward edge is: $f(e)=c(e)$

flow of backward edge is: $f(e)=0$

Forward edge

an edge from V_s to V_t

Backward edge

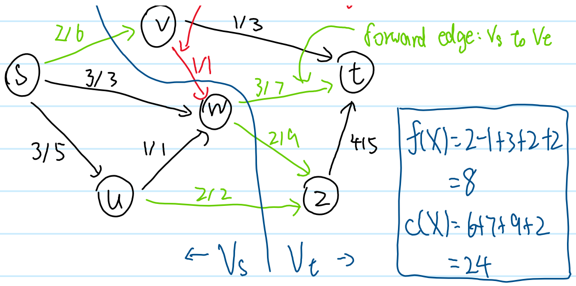
an edge from V_t to V_s

Capacity $c(X)$

forward edges total capacity

Flow $f(X)$

forward edges total flow − backward edges total flow



Properties

1. for any cut X, the flow of the network f = the flow across the cut $f(X)$

$|f| = f(X)$

2. for any cut X, the flow of the network is no greater than the cut's capacity $c(X)$

$|f| \leq c(X)$

Maximum flow

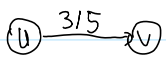
A flow of a network is said to be maximum if its value is the largest of all flows of the network.

Augmenting path

A path that can increase the flow of the graph

Goal: increase forward flow, decrease backward flow

for the flow $u \to v$ on the right:



if it is a forward flow, it can be increased by:

$\Delta f(u,v) = c(e) - f(e) = 5-3 = 2$

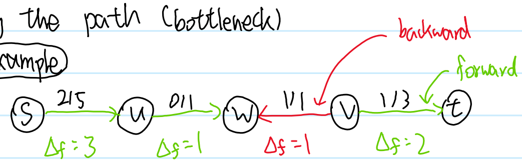
if it is a backward flow, it can be decreased by:

$\Delta f(v,u) = f(e) = 3$

this is called residual capacity

let Π be a path from s to t, $\Delta f(\Pi)$ is equal to the smallest residual capacity along the path (bottleneck)

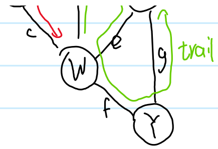
example



$\Delta f(\Pi) = \min(\Delta f(s,u), \Delta f(u,w), \Delta f(w,v), \Delta f(v,t)) = 1$

ends with a vertex

− UaVaUcW is a path



Simple path

a path with no repeated edge or vertex

− VbXhZ is a simple path

Trail

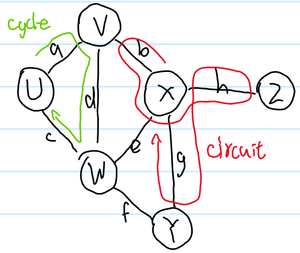
a path with no repeated edge

Circuit

a path with the same start and end vertex

Cycle

a circuit with no repeated vertex (except the start and end vertex)



Directed path, walk, circuit, etc.

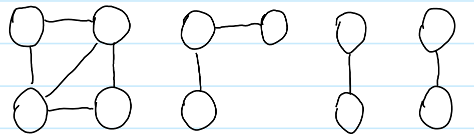
all edges are directed

Subgraph

a graph whose edges and vertices are subsets of another graph

Spanning subgraph

a subgraph which contains all the vertices of the original graph



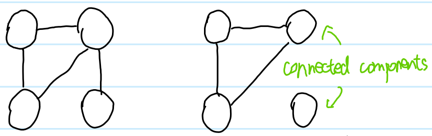
Graph　　Subgraph　　Spanning Subgraph

Connected

for any two distinct vertices, there is a path between

Connected components

maximal connected subgraphs of a not connected graph



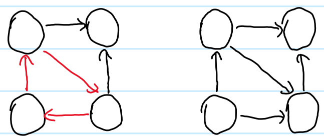
Connected　　Not Connected

Cyclic

a graph that contains cycles

Acyclic

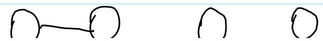
a graph that contains no cycle



Cyclic　　Acyclic

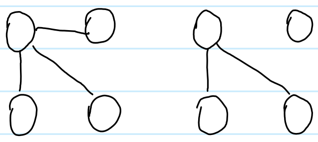
Tree

an undirected graph that is



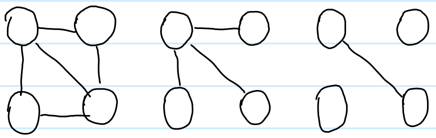
Tree
an undirected graph that is connected and acyclic

Forest
an undirected graph that is acyclic
the maximal connected subgraphs of a forest are trees



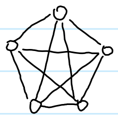
Spanning tree
a spanning subgraph of a graph that is a tree

Spanning forest
a spanning subgraph of a graph that is a forest



Properties

1. total degree = 2x total edges
   $\sum_v \deg(v) = 2m$
   proof: each edge is counted twice
2. total edges ≤ total vertices x (total vertices - 1) / 2
   $m \leq n(n-1)/2$
   proof. the degree of each vertex is at most n-1



3. If a graph is connected, then total edges ≥ total vertices - 1
   $m \geq n-1$
   explanation. allow cycles
4. If a graph is a tree, then total edges = total vertices - 1
   $m = n-1$
   explanation: no cycle
5. If a graph is a forest, then total edges ≤ total vertices - 1
   $m \leq n-1$
   explanation: allow not connected

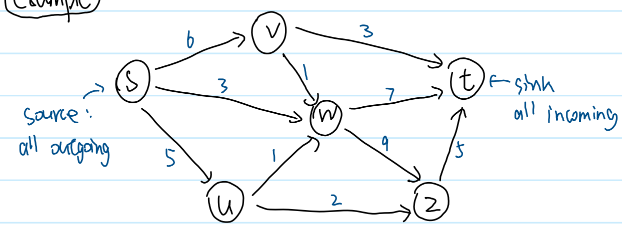
Depth first search DFS (stack based)
1. start at the start node
2. visit the node

Flow Network

2021年5月31日 下午 4:32

A weighted graph with nonnegative integer weights such that:
the weight of an edge e is called the capacity c(e) of e
has a source vertex s which has no incoming edge
has a sink vertex t which has no outgoing edge

example

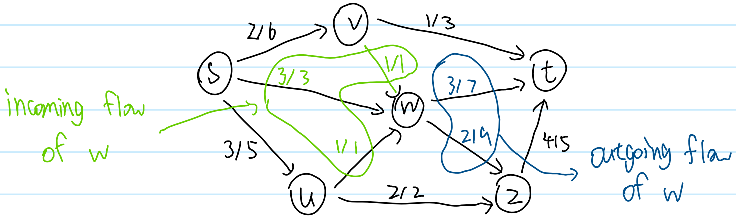


Flow
the flow through the network
Capacity rule: for each edge e, $0 \leq f(e) \leq c(e)$
not negative
no greater than capacity

Conservation rule: for each non-source vertex $v \neq s$,
the incoming flow is equal to the outgoing flow.

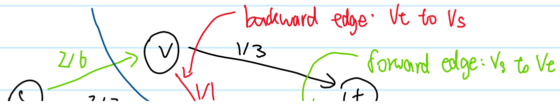
$$\sum_{e \in E^-(v)} f(e) = \sum_{e \in E^+(v)} f(e)$$

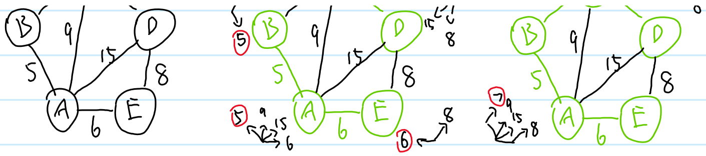
incoming flow　　outgoing flow



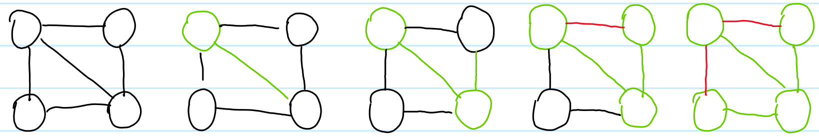
Cut
A partition of the vertices of the graph into two parts $X = (V_s, V_t)$ such that
$s \in V_s$ and $t \in V_t$

Forward edge

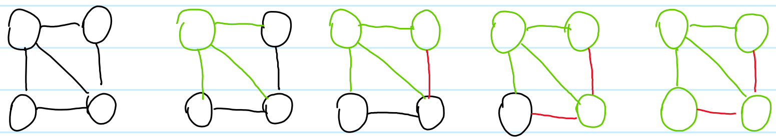




1. start at the start node
2. visit the node
3. for each ajacent node.
    if it is not visited, visit it and repeat 2-3.
    if it is visited, remove the incident edge(s)

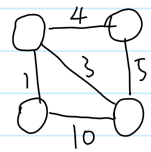


Bredth first search (queue based)
1. start at the start node
2. visit the node
3. for each adjacent node:
    if it is not visited, add into queue
    if it is visited, remove the incident edge(s)
4. pop the next node in queue and repeat 2-4.



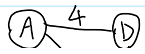
DFS is better for ansering complex connectivity questions
BFS finds the shortest path of a graph

Weighted graph
  a graph that has numerical weights
  associated with each edge

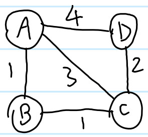


Single-source shortest path
  for a fixed vertex, find the shortest
  path from it to all other vertices

Dijkstra's algorithm



to A B C D visited



to	A	B	C	D	visited
	0	∞	∞	∞	∅
	0	1	3	4	A
	0	1	2	4	A B
	0	1	2	4	A B C
	0	1	2	4	A B C D

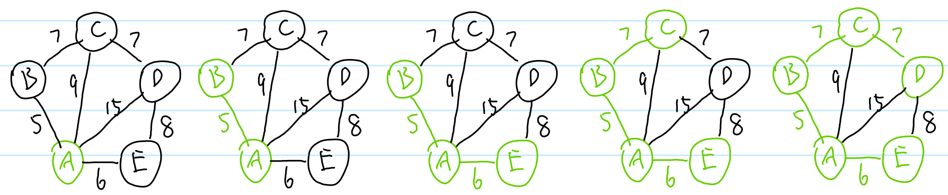
Minimum Spanning Tree

2021年5月31日 下午 4:06

A spanning tree of a weighted graph which minimizes the sum of the weights

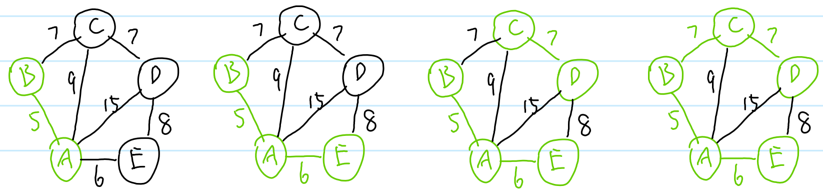
Prim's algorithm

1. start from one node
2. grow the tree by adding the edge with minimal weight and does not create a cycle, and add the incident node
3. repeat 2 until all vertices reached



Kruskal's algorithm

1. sort all edges by weight
2. add edges one by one, skip the edges that creates cycles
3. repeat 2 until all edges reached



Borůvka's algorithm

1. treat the graph as a forest and each node as a tree
2. for every tree, connect to the ajacent tree by the smallest weighted edge
3. repeat 2 until all edges reached