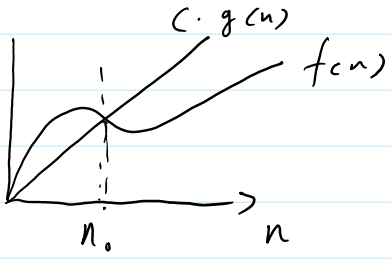
20170911

2017年9月11日　11:17

Big - Oh　　if and only if
upper bound

$O(g(n)) = \{ f(n) : \exists$ exist positive constants c & n_0 s.t. $0 \le f(n) \le c \cdot g(n) \quad \forall \ n \ge n_0 \}$



$\frac{1}{2}n^2 - 3n = O(n^2)$

prove:

$\frac{1}{2}n^2 - 3n \le c \cdot n^2$

$\frac{1}{2} - \frac{3}{n} \le c$

works for $c \ge \frac{1}{2}$

$n_0 = 6$

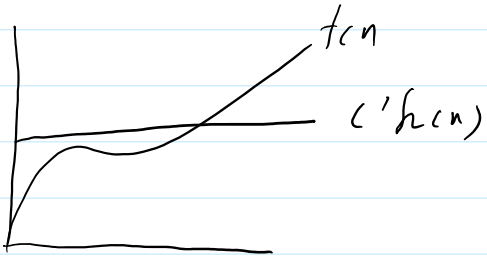
$n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot (n-1) \cdot n$

$\le n^n = O(n^n)$

$100000n^3 = O(n^3)$

Big - Omega
Lower bound

$\Omega(h(n)) = \{ f(n) : \exists$ pos constants c' & n_1 s.t. $0 \le c' \cdot h(n) \le f(n) \quad \forall \ n \ge n_1 \}$



$f(n) = 1 + 2 + 3 + \cdots + n$

$\ge \left[\frac{n}{2}\right] + \left(\frac{n}{2} + 1\right) + \left[\frac{n}{2} + 2\right] + \cdots + n$

$\geq \lfloor \frac{n}{2} \rfloor \cdot \lceil \frac{n}{2} \rceil \geq \frac{n^2}{4}$

$= \Omega(n^2)$ $c' = \frac{1}{4}$

$n_1 = 1$

$\frac{1}{2}n^2 - 3n = \Omega(n^2) = O(n^2)$

$c'n^2 \leq \frac{1}{2}n^2 - 3n$

$c' \leq \frac{1}{2} - \frac{3}{n}$

$n_1 = 7$

$c' = \frac{1}{14}$

Theta

Tight bound

$\Theta(g(n)) = \{f(n) : f(n) = O(g(n)) \text{ \& } f(n) = \Omega(g(n))\}$

Properties

$n^a \in O(n^b)$ if $a \leq b$

$\log_a n \in O(\log_b n)$ $\forall a, b$

$c^n \in O(d^n)$ if $c \leq d$

if $f(n) \in O(f'(n))$ & $g(n) \in O(g'(n))$ then

$f(n) \cdot g(n) \in f'(n) \cdot g'(n)$

$f(n) + g(n) \in \max\{g'(n), f'(n)\}$

Average Case

Assume about the input, on an average day

Expected Case

Amortized Case

Randomized alg.

Vs

Probabilistic Analysis

Ex)

show $f(n) = \sum_{i=1}^{n} i^k = \Theta(n^{k+1})$

Prove: $f(n) = O(n^{k+1}) = \Omega(n^{k+1})$

① $f(n) = \sum_{i=1}^{n} \geq n \cdot n^k = n^{k+1} = O(n^{k+1})$

② $2f(n) = \underbrace{1^k + 2^k + 3^k + \cdots + n^k}_{n^k + (n-1)^k + \cdots + 1^k} + \cdots$

$= \sum_{i=1}^{n} i^k + \sum_{i=1}^{n} (n-i+1)^k$

$= \sum_{i=1}^{n} i^k + (n-i+1)^k$

all larger than $(\frac{n}{2})^k$ ⟶

$\geq \sum_{i=1}^{n} \lfloor \frac{n}{2} \rfloor^k$

$= n \cdot \lfloor \frac{n}{2} \rfloor^k$

$= \frac{1}{2^k} (n^{k+1})$

$\therefore f(n) = (\frac{1}{2})^{k+1} (n^{k+1}) = \Omega(n^{k+1})$

20170912

2017年9月12日　11:14

Mathematical Induction

Prove property $P(n)$

$$P(1) \wedge \forall n \, (P(n) \rightarrow P(n+1)) \Rightarrow \forall n \, P(n)$$

Prove $1+2+3+\cdots+n = \frac{n(n+1)}{2}$

Basic Prove for $P(1)$ — sometimes induction does not work for small numbers

$1 = \frac{1 \cdot 2}{2}$

Hypothesis assume $P(n) = \frac{n(n+1)}{2}$

Step $P(n+1) = P(n) + n + 1$

$$= \frac{n(n+1)}{2} + n + 1$$

$$= \frac{n(n+1)}{2} + \frac{2n+2}{2}$$

$$= \frac{n^2+n+2n+2}{2}$$

$$= \frac{n^2+3n+2}{2}$$

$$= \frac{(n+1)(n+2)}{2}$$

Prove that sum of first n odd positive integer is n^2

$1+3+5+\cdots+2n-1$ ⇐ this is n

Prove $P(1) = 1 = 1^2$

assume $P(n) = n^2$

step $P(n+1) = n^2+2n-1+2$

$= n^2+2n+1$

$= (n+1)^2$

Q.E.D

Prove that $n < 2^n$

Prove that $n < 2^n$

Prove $n = 1 \quad 1 < 2^n$

assume $n < 2^n$

Step $(n+1) < 2^n + 1 < 2^n + 2^n < 2^{n+1}$

Prove that a power set of a set of n element has 2^n subsets

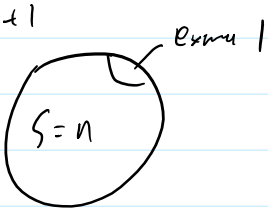
Powerset of $\{A, B, C\}$ is $\{0, A, B, C, AB, AC, BC, ABC\}$

$n=3$ $\quad 2^3 = 8$

Basic $S=0$ $2^0 = 1$

assume $S=n$ is true

Step $S = n+1$



{with extra, without extra.}

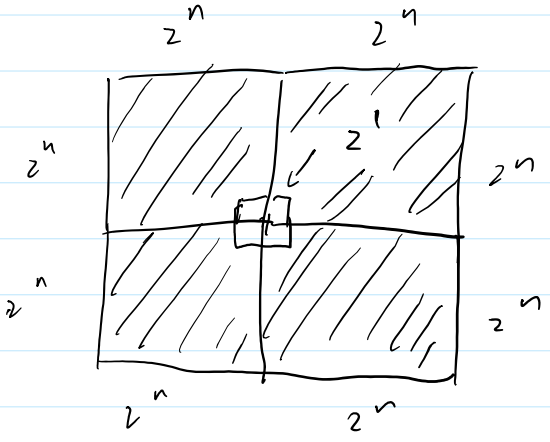
$P(n+1) = 2^n + 2^n = 2^{n+1}$

Q3 show any $2^n \times 2^n$ chess board can be filled with L-shaped tiles plus any 1 square left empty

Basis $P(0)$ ▨

assume $P(n)$ is true

Step $P(n+1)$



Q.E.D

Logarithm

$a = b^c \Rightarrow \log_b a = c$

$a = b^{\log_b a}$

$\log_a(ab) = \log_a a + \log_a b$

$\log_b a^c = c \cdot \log_b a$

$\log_b \frac{1}{a} = -\log_b a$

$\log_b \frac{a}{c} = \log_b a - \log_b c$

$a^{\log_b n} = n^{\log_b a}$

$\log^{(i)} n = \begin{cases} n & \text{iff } i = 0 \\ \log(\log^{(i-1)} n) & \text{if } i > 0 \end{cases}$

$\Rightarrow \log^{(4)} n = \log\log\log\log n$

$\log^* n = \min \{ i : \log^{(i)} n \leq 1 \}$

20170914

2017年9月14日　10:18

Contradiction

You want prove P(n)　(yes/no)

Assume ¬P(n) => ..... => ..... => false

Prove if $x^2-5x+4<0$ then $x>0$

Assume towards a contradiction that

$x^2-5x+4<0$ but $x \le 0$

$x^2 < 5x-4$ but $x \le 0$

⇓

negative　$\therefore x^2 < 0$

a contradiction, hence $x>0$

Prove if $3n+2$ = odd then n = odd

ATaC　$3n+2$ = odd but n = even = $2k$

$3 \cdot 2k+2 = 2(3k+1)$ = even

a contradiction. Hence n = odd

Show that $\sqrt{2}$ is irrational

ATaC　$\sqrt{2}$ = rational = $\frac{a}{b}$ where a & b have no common factors

$2 = \frac{a^2}{b^2} \Leftarrow a^2 = 2b^2 <=> a^2$ = even = $2 \times c$

1 (lemma) we will prove a^2 = even => a = even

<=> $a = 2c$ <=> $2b^2 = 4c^2$ <=> $b^2 = 2c^2$ <=> b^2 = even

∴ b = even <=> contradiction

Lemma:

ATaC　a^2 = even but a = odd = $2k+1$

$(2k+1)^2$ <=> $a^2 = (4k^2+2k)+1$ = odd

a contradiction

Summations

$$\sum_{k=1}^{n} k = 1+2+3+\cdots+n = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} x^k = 1+x+x^2+\cdots+x^n = \frac{x^{n+1}-1}{x-1}$$

$$(x+y)^r = \sum_{i=0}^{r} \binom{r}{i} x^i y^{r-1}$$

$\binom{r}{i}$: r choose i, $\frac{r!}{(r-i)!\,i!}$

Fibonacci #s

$F_0 = 0 \quad F_1 = 1$

$F_i = F_{i-1} + F_{i-2}$

$F_i = \frac{\phi^i - \hat{\phi}^i}{r_5} \qquad \phi = \frac{1+r_5}{2} \qquad \hat{\phi} = \frac{1-r_5}{2}$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}, \text{ if } |x|<1$$

Prove $\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2}$

$$\frac{d \sum_{k=0}^{\infty} x^k}{dx} = \frac{d\left(\frac{1}{1-x}\right)}{dx}$$

$$\sum_{0}^{\infty} k x^{k-1} = \frac{1}{(1-x)^2}$$

$$\Rightarrow \sum_{0}^{\infty} k x^k = \frac{x}{(1-x)^2}$$

Telescoping

$a_n - a_{n-1} +$

$a_{n-1} - a_{n-2} +$

$a_{n-2} - a_{n-3} +$

$\vdots$

$a_2 - a_1 +$

$a_1 - a_0 = a_n - a_0$

20170918

2017年9月18日　11:08

Recurrence

mergesort (A, p, r)

if p < r

q = ⌊(p+r)/2⌋

Mergsort (A, q+1, r)

Mergsort (A, p, q)

Mergsort (A, p, q, r)

Mergsort (X, y)

serially merge

Sorted list

x & y into Z

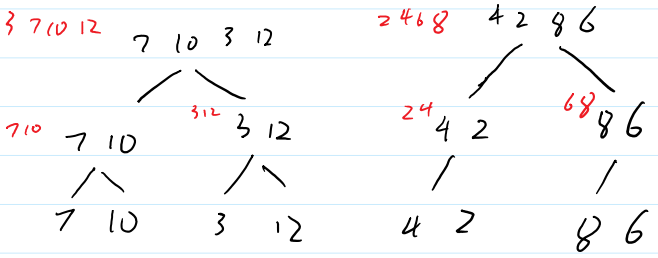
To run T(n), must run T(n/2) first twice

∴ T(n) = T(n/2) + T(n/2) + O(n)

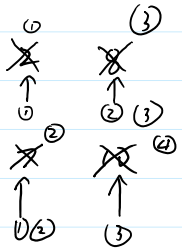
= 2T(n/2) + O(n) = Θ(n log a)

Ex)

7 10 3 12 4 2 8 6 → 2 3 4 6 7 8 10 12



Merge:



2 7 8 10

① ② ③ ④

Master Method

Let a ≥ 1 & b ≥ 1 and f(n) a function. Then recurrence

T(n) = aT(n/b) + f(n) has solution

a) if f(n) = O(n^(log_b a − ε)) for ε > 0 then T(n) = Θ(n^(log_b a))

b) if f(n) = Θ(n^(log_b a)) then T(n) = Θ(n^(log_b a) log n)

c) if f(n) = Ω(n^(log_b a + ε)) for ε > 0 and a f(n/b) ≤ c f(n) for 0 < c < 1

then T(n) = Θ(f(n))

T(n) = 2T(n/2) + O(n)

a = 2, b = 2, f(n) = O(n)

B) O(n) = f(n) = Θ(n^(log_2 2))

T(n) = Θ(n log n)

T(n) = 9T(n/3) + n

a = 9 b = 3 f(n) = n

log 9 − ε

$a=9 \quad b=3 \quad f(n)=n$

a) $f(n) = n = O\left(n^{\log_3 9 - 1}\right) \Rightarrow f(n) < n^{\log_3 9 - \varepsilon}$, case 1

$\therefore T(n) = \Theta(n^2)$

$T(n) = 3T\left(\frac{n}{4}\right) + n\log n$

$a=3 \quad b=4 \quad f(n) = n\log n$

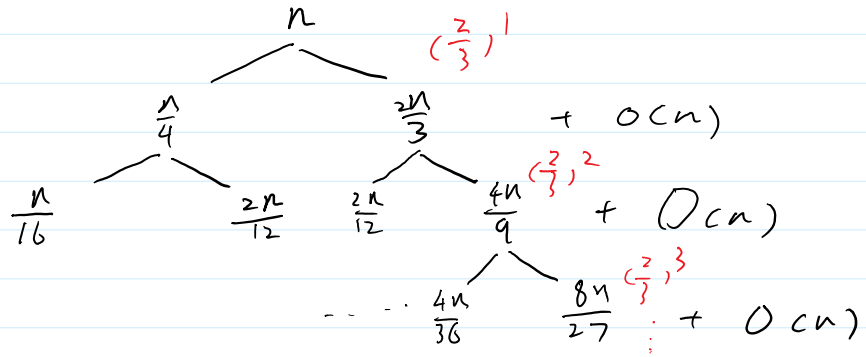
c) $n^{\log_4 3} = O(n^{0.7 n 3})$

Hence $f(n) = \Omega\left(n^{\log_4 3 + 0.2}\right)$ ✓ ε

finally $3\left(f\left(\frac{n}{4}\right)\right) = 3\frac{n}{4}\log\frac{n}{4} \le \frac{3}{4} n\log n$

Recursion Tree

$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{2n}{3}\right) + n$



$T(n) = \left(\begin{array}{l}\text{work} \\ \forall \text{ tree level}\end{array}\right) \times \left(\begin{array}{c}\#\ \text{of levels in tree} \\ (\text{height})\end{array}\right)$

$= O(n) \cdot h \quad \left(\frac{2}{3}\right)^h n = 1$

$= O(n\log n) \quad h = \log_{\frac{3}{2}} n = O(\log n)$

20190919

2017年9月19日　11:30

$T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n \quad T(1) = 1$ substitution

Induction

we guess $T(n) = cn \cdot \log n = O(n \log n)$

Hypothesis $T(\lfloor \frac{n}{2} \rfloor) \le c \lfloor \frac{n}{2} \rfloor \log \lfloor \frac{n}{2} \rfloor$ assume true

Step $T(n) \le 2T(\lfloor \frac{n}{2} \rfloor) + n \le 2c \lfloor \frac{n}{2} \rfloor \log \lfloor \frac{n}{2} \rfloor + n$

$= cn \log n - cn \log 2 + n$

$= cn \log n - cn + n$

$= cn \log n + (1-c)n$ c is chosen arbitrarily

$\le cn \log n$ true for $c > 1$

$T(1) = 1$

$T(2) = 2T(1) + 2 = 4 \le 2c \log 2 \le 2c$

$T(3) = 2T(1) + 3 = 5 \le 3c \log 3 \le 3c \cdot \log 3$

$c \ge 2$ satisfy all conditions

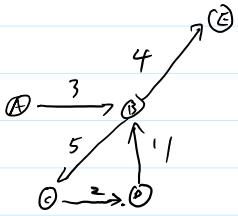
Graphs

$G = (V, E)$

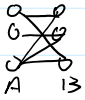
vertices　edges

- directed & undirected
- weighted or not
- path
  - simple path: does not repeat a V

    B - C - D　　B - C - D - B - E ⇐ not a simple path
  - cycle (simple cycle)
- connected graph ∀ two vertices, u & V ∃ Path p $u \xrightarrow{p} v$
- induced subgraph $G' = (V', E')$ where $V' \subseteq V$



Bipartite G



two groups, only path between different groups

- DAG - directed acyclic graph

- clique (complete graph)

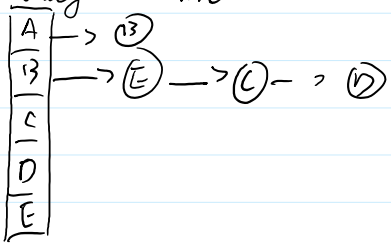
  ∃ edge between every 2 V

  # $\frac{n(n-1)}{2}$ edges

- vertex degree = # edges adjacent
  - in/out for directed

Graph Representations

Adjacent list



Adj Matrix

	A	B	C	D	E
A		3			
B			5	11	4
C				2	
D					
E					

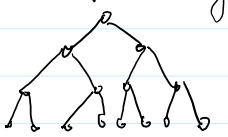
transpose
reverse direction

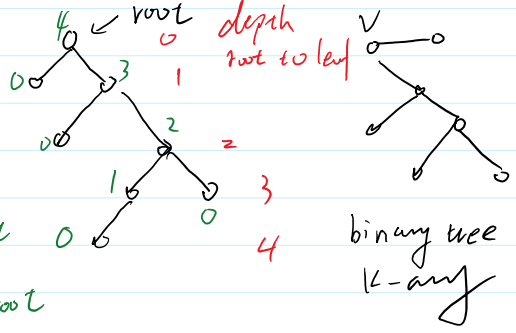
if undirected, matrix will be symmetric

	AL	AM
time	$O(E)$	$O(1)$
Memory	$O(V+E)$	$O(V^2)$

Tree = a connected acyclic undirected graph

Complete binary tree





equivalent statements:

A) G is a tree

B) ∀ two vertices of G are connected by a unique simple path

C) $G=(V,E)$ is connected but removing any edge disconnect it

D) G is connected & $|E|=|V|-1$

E) G is acyclic & $|E|=|V|-1$

F) G acyclic & adding any new edge creates a creates a cycle

20170921

2017年9月21日　10:12

If a event can happen "m-ways" and b "n-ways"

Rule of product

$n \times m$ - ways A & B can occur together

Rule of sum

$n + m$ ways either A or B can occur

Ex) 5 latin, 7 greek, 10 French book

# of ways to choose 2 books. diff language

$5 \times 7 + 7 \times 10 + 5 \times 10$

# of ways to choose 2 books

22×21

Permutations

$P(n,r)$ = the way to arrange r out of n - distinct objects where order matters

$$P(n,r) = \frac{n!}{(n-r)!}$$

# of ways n ppl can sit on a round table

$$\frac{P(n,n)}{n}$$

If not all object distinct but you got

q_1 1^{st} kind

q_2 2^{nd} kind

q_t t^{th} kind

# of ways to arrange them

$$\frac{n!}{q_1! \cdot q_2! \cdots q_t!}$$

Five dashs & 8 dots can be arranged in

$$\frac{13!}{5! \cdot 8!} \text{ ays}$$

show $(k!)!$ is divisible by $(k!)^{(k-1)!}$

assume have $k!$ objects

k of 1st kind

k of 2nd kind

$\vdots$

k of $(k-1)!$ kind

# of ways to arrange:

$$\frac{(k!)!}{\underbrace{k! \cdot k! \cdots k!}_{(k-1)! \text{ times}}} = \frac{(k!)!}{(k!)^{(k-1)!}}$$

ways to arrange r-objects selected out of n-object with unlimited repetion

n^r

Among 10 billion #s $1 \ldots 10,000,000,000$

how many contain "1" and how many do not?

do not have 1 $\underbrace{9, 9, \ldots 9}_{9^{10}}$

$9^{10} - 1$

$\therefore 10^{10} - (9^{10} - 1)$ contain 1

Combonation

$C(n,r) =$, order does not matter

$$C(n,r) = \frac{P(n,r)}{r!}$$

$$= \frac{n!}{r!(n-r)!} = \binom{n}{r}$$

How many diagonals in a 10-gon has?

In how many ways can three #s be selected from $[1, 2, 3 \ldots 300]$ so that their sum is divisible by 3

$\exists$ 100 #s that leave remainder of 1

$\exists$ 100 #s $= // =$ 2

$\exists$ 100 #s $= // =$ 0

$C(100,3) + C(100,1)\,C(100,2) + 100^3 =$ □

111 222 1000 1120

Eleven crazy scientists are working on a secret project. They want lock

documents in a cabinet. At least 6 scientists are present to open the cabinet
- what is the smallest # locks needed
- what is the smallest # keys each scientist need have

A) $C(11,5)$ = 462

∀ groups of 5 ∃ one lock they cannot open
∀ two diff groups of five, this lock need be diff
(Otherwise 6 or more ppl we been able to open it)

B) $C(10,5)$ = 252 keys any single scientist should join any group of 5 of the remain 10 to open the lock.

20170925

2017年9月25日 11:09

Probability

an experiment over a sample space S with outcomes being events

Axiom

$P_r(a \in S) \geq 0$

$P_r(S) = 1$

$P_r(A \cup B) = P_r(A) + P_r(B) - P_r(A \cap B)$

$P_r(A \cap B) = P_r(A) \cdot P_r(B)$ if $A \& B$ are independent

Uniform Probability Distribution.

when $P_r(s) = \frac{1}{|S|}$ for any event s occuring in S

two coin tossed

- flip n times

$P_r(\text{exactly } k\text{-tails}) = \binom{n}{k} \frac{1}{2^n}$

Bayes Theorem

Conditional Prob:

$$P_r(A|B) = \frac{P_r(A \cap B)}{P_r(B)}$$

$$P_r(A \cap B) = P_r(A|B) P_r(B) = P_r(B|A) P_r(A)$$

$$P_r(A|B) = \frac{P_r(B|A) P_r(A)}{P_r(A) P_r(B|A) + P_r(\bar{A}) P_r(B|\bar{A})}$$

A: both coins are H

B = atleast 1 H

$$P_r(A|B) = \frac{P_r(A \cap B)}{P_r(B)} = \frac{\frac{1}{4}}{\frac{3}{4}} = \frac{1}{3}$$

- two coins, one fair, one biased to always be H. Pick one at random & flip twice

$$P_r(\text{picked biased} \mid \text{you get two H's}) = \frac{P_r(B|A) P_r(A)}{P_r(A) P_r(B|A) + P_r(\bar{A}) P_r(B|\bar{A})} = \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{4}} = \frac{4}{5}$$

$$P_r\left(\begin{array}{c}\text{picked}\\ \text{biased}\\ \underline{A}\end{array} \Big/ \begin{array}{c}\text{you got}\\ \text{two H's}\\ \underline{B}\end{array}\right) = \frac{P_r(B|A)P_r(A)}{P_r(A)P_r(B|A)+P_r(\bar{A})P_r(B|\bar{A})} = \frac{\frac{1}{2}}{\frac{1}{2}+\frac{1}{2}\cdot\frac{1}{4}} = \frac{4}{5}$$

Discrete Random

They map sample space S to discrete results:

Random variable (r.v.) X

$P_r(X = x) = \sum P_r(s)$

$(s \in S, X(s) = x)$

Pair of Dice

X = max of two rolls

$P(X = 3) = \sum P_r\{1\text{-}3, 2\text{-}3, 3\text{-}3, 3\text{-}1, 3\text{-}2\}$

$= 5 \cdot \frac{1}{36}$

Expected Value $= E[X] = \sum_{x \in X} x\, P_r(X = x)$

$E[X+Y] = E[X]+E[Y]$

$E[\alpha X] = \alpha E[X]$

$E[X \cdot Y] = E[X] \cdot E[Y]$ bayes hold

Flip 2 coins, win \$3 on H, lose \$2 on T

$E[X = \text{your earnings}] = 6\,P_r(H\text{-}H) + 1\,P_r(H\text{-}T) + 1\,P_r(T\text{-}H) + (-4)P_r(TT)$

$= 1\,\$$

20170926

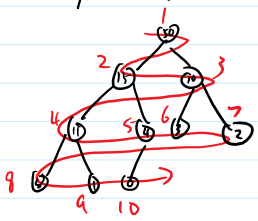
2017年9月26日　11:14

Heaps & Heapsort : In-place sorting algorithm
it uses only O(1) additional space than original array A to be sorted

Heap structure :

Heap shape : complete binary tree except last level leaves all pushed to left

Heap order : (key parent) > (key children)



1	2	3	4	5	6	7	8	9	10
50	15	10	11	4	3	2	8	1	10

index parent i
- index left child 2i
- index right child 2i+1

(Heapify)

bubble-down(i)

compare A[i] with A[2i] & A[2i+1] and swap with largest item.

Repeat if a swap occurred

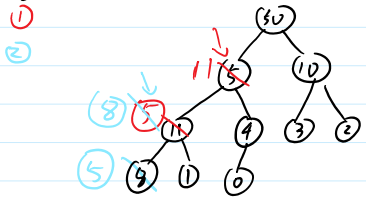
O(h)

Build-Heap

for i := ⌊ size(A) / 2 ⌋ -> 1

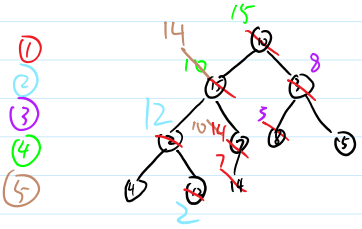
bubble-down(i) O(nh)

for a random tree



1	2	3	4	5	6	7	8	9	10
10	15	3	2	7	8	5	4	12	14

A



(Key parent) < (key children) : Min Heap

Delete max(A)

swap A(1) with A(size(A))

A(size) := A(size) - 1

O(h)

swap A(1) with A(size(A))
A(size) = A(size) - 1
bubble-down(1)

$O(h)$

Heapsort(A)
Build-Heap(A)
for i = size(A) - 1
  delete-max(A)

$O(nh)$

$O(h) = O(\log n)$

Lemma. There are at most $\lceil \frac{n}{2^{h+1}} \rceil$ nodes of height h in a heap

Tight bound: build-heap takes $O(n)$ time

$$\#\text{work} = \sum_{h=0}^{\log n} \lceil \frac{n}{2^{h+1}} \rceil \cdot h \le O\left(n \sum_{h=0}^{\log n} \lceil \frac{h}{2^h} \rceil\right) \le n \cdot \frac{1/2}{(1-\frac{1}{2})^2} = 2n = O(n)$$

20170928

2017年9月28日　10:12

```
QS ( in, left, right )
  QS (in, left, right)
    pivot = Partition ( in, left, right )
      if pivot > left
        QS ( in. left, pivot )
      if pivot < right
        QS ( in, left, pivot )

int Partition (in, left, right)
    ls = left
    for j = left+1 ..... right
      if (in(i) ≤ pivot )
        ls = ls + 1
        swap(in(i), in(ls))
    swap(in(left), in(ls))
    return ls
```

Worst case　　← partition

$T(n) = T(n-1) + \Theta(n)$

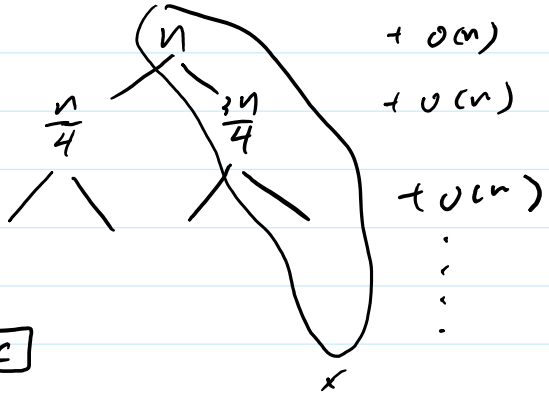
$= T(n-2) + \Theta(n-1) + \Theta(n)$

$= T(n-3) + \Theta(n-2) + \Theta(n-1) + \Theta(n)$

$\Theta \sum_{k=1}^{n} k = \Theta(n^2)$

$n/2$	$n/2$	best
$O(1)$	$O(n)$	worst
$n/4$	$3n/4$	average

$T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + O(n)$



Total work $= \underbrace{h \cdot O(n)}_{\left(\frac{3}{4}\right)^h n = 1} = O(n \log n)$

total work $= n \cdot U(n) = U(n \log n)$

$(\frac{3}{4})^h n = 1$

$h = \log_{\frac{4}{3}} n = U(\log n)$

Worst Case (formal)

$$T(n) = \max_{1 \le q \le n-1} \{ T(q) + T(n-q) \} + \Theta(n)$$

Substitution

Guess $T(n) \le cn^2$

Step $T(n) \le \max_{1 \le q \le n-1} \{ cq^2 + c(n-q)^2 \} + \Theta(n)$

$= c \max_{1 \le q \le n-1} \{ q^2 + (n-q)^2 \} + \Theta(n)$

achieves maximum at end points $q = 1$ or $q = n-1$

$T(n) = cn^2 - 2c(n-1) + \Theta(n) \le cn^2$

20171002

2017年10月2日 11:13

Quick Sort

Rand-Partition → key

Quick sort (left, key) /* ≤ Pivot */

Quick sort (key+1, right) /* Pivot < */

Best $O(n\log n)$ worst $O(n^2)$

Expected $O(n\log n)$ (×) Formal proof

Expected Time of Rand-QS

$T(n) = \frac{1}{n}\left[T(1)+T(n-1)+\sum_{q=1}^{n-1}\left(T(q)+T(n-q)\right)\right] + \overbrace{\Theta(n)}^{\text{rand partition}}$

$\frac{1}{n}[T(1)+T(n-1)] \le \frac{1}{n}(\Theta(1)+\Theta(n^2)) = \Theta(n)$

$\therefore \frac{1}{n}[T(1)+T(n-1)]$ will be absorbed

$T(n) \le \frac{1}{n}\sum_{q=1}^{n-1}\left(T(q)+T(n-q)\right)+\Theta(n)$

$= \frac{2}{n}\sum_{k=1}^{n-1} T(k) + \Theta(n)$

Use substitution: $T(n) = an\log n + b$ for large enough a & b ⇐ assumption

$T(n) \le \frac{2}{n}\sum_{k=1}^{n-1}(ak\log k + b) + \Theta(n) = \frac{2a}{n}\sum_{k=1}^{n-1} k\log k + \frac{2b}{n}(n-1) + \Theta(n)$

Lemma $\sum_{k=1}^{n-1} k\log k \le \frac{1}{2}n^2\log n - \frac{1}{8}n^2$

using Lemma

$T(n) \le an\log n - \frac{a}{4}n + 2b + \Theta(n)$

$= an\log n + b + [\Theta(n) + b - \frac{a}{4}n] \le an\log n + b$

⇒ can choose a large enough a so value in bracket become negative

Lemma:

$\sum_{k=1}^{\lceil \frac{n}{2}\rceil - 1} k\log k + \sum_{k=\lceil \frac{n}{2}\rceil}^{n-1} k\log k \le \log\frac{n}{2}\sum_{k=1}^{\lceil \frac{n}{2}\rceil + 1} k + \log n \sum_{k=\lceil \frac{n}{2}\rceil}^{n-1} k$

$= \log n \sum_{k=1}^{\frac{n}{2}-1} k - \sum_{1}^{\frac{n}{2}-1} k + \log n \sum_{n/2}^{n-1} k$

$= \log n \sum_{k=1}^{n} k - \sum_{k=1}^{\frac{n}{2}-1} k$

$= \frac{(\log n + \log n)n}{2}$

$= \frac{1}{2}n^2\log n + \frac{n\log n}{2} - \left(\frac{n}{2}-1+1\right)\frac{n}{2}$

$$= \frac{1}{2}n^2\log n + \frac{n\log n}{2} - \frac{n^2}{8}$$

$$\leq \frac{1}{2}n^2\log n - \frac{n^2}{8}$$

Lower bound on comparison-based sorting

$\Omega(n \log n)$

$O(n)$
- fixed range
- fixed # bits
- no comparison based

Radix Sort

n = # numbers
K = range of digits
d = # digits

6
[0 ... n]
3

- Stable Sorting algorithm ???

7 5 3 5 1

Radix Sort
For LSB to MSB
stable sort digits
↓
counting sort also $O(n)$

Ex)

5	3	1		5	3	1		10	4		104	runtime
3	3	2		5	1	1		5	1		332	$\forall$ digit $O(n)$
7	1	2	=>	3	3	2	=>	7	12	=>	511	total $= O(dn)$
5	1	1		7	1	2		6	13		531	
6	1	3		6	1	3		5	31		613	
				10	4			3	32		712	

Ex)

1000 #s 64-bits

QS: $O(n\log n)$ per # $\log(n)$ passes
$\approx$ 10 passes / #

RS: [| | |]
← 16 bits

4 passes / #

20171003

2017年10月3日　11:11

Counting Sort

Sorts #'s in range 0....k

Not in-place but stable

A = array to be sorted

B & C are auxiliary arrays

```
C[i] = 0  ∀ i ∈ [0...k]
for j = 1 .... length(A)
    C[A(j)] = C[A(j)] + 1
for i = 1 ..... k
prefix sum  C[i] = C[i] + C[i-1]
for j = length(A) ..... 1 do
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

Time

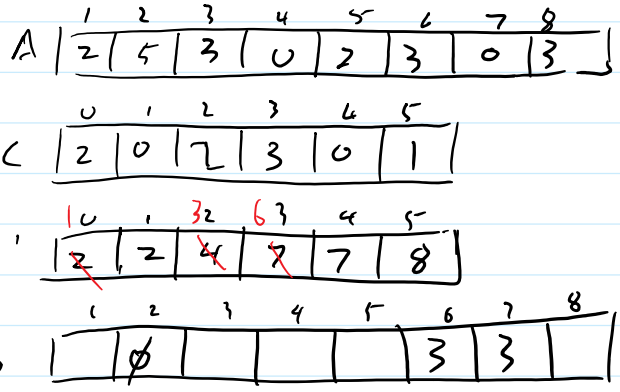
$O(n)$

$O(k)$

$O(n)$

+)

$O(n+k)$



Searching

Given n - unsorted elements, What is the x^{th} largest element?

max : $O(n)$ $\Leftrightarrow n-1$

min if you know max $(\frac{n}{2}-1)$

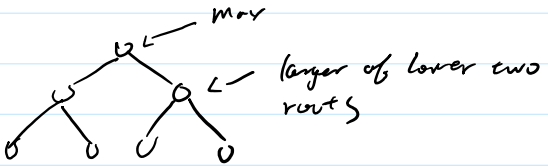
2nd max?

$(n-1)+(\log n - 1)$

$\Downarrow$

K^{th} max $= (n-1)+k\log n$

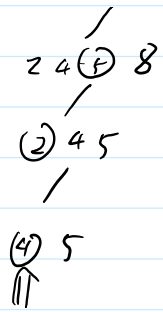
$= O(n+k\log n)$



8 1 5 9 2 10 4 17 20　　7 th largest?

Quick sort: 1 5 2 4 (6) 9 10 17 20

(1) 5 2 4 8




```
Rand-select (A, p, r, i)
    if p = r  return A[p]
    q = rand-partition (A, p, r)
    k = q - p + 1
    if i ≤ k
        Rand-select (A, p, q, i)
    else
        Rand-select (A, q+1, r, i-k)
```

$$T(n) \le$$
$$\le \frac{1}{n}(T(1)+T(n-1)) + \sum_{k=1}^{n-1} \max(T(k), T(n-k)) + O(n)$$
$$= O(n) \text{ expected}$$

20071005

2017年10月5日　10:19

Successor　O(1)*　Threaded Tree

run inorder

"on the fly"

Predecessor　O(1)*

"reverse" inorder

"on the fly"

Post order:	Pre order:
visit left (-r)	print key
visit right (-r)	visit left (-r)
print key	visit right (-r)

Search (key)

probe root go

left or right based

on key - value

Insert (key)

search & insert as leaf

ⓐ key = leaf

delete it

ⓑ key has one children

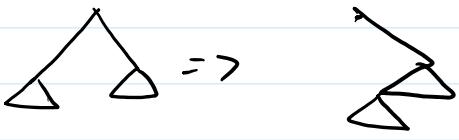
delete it & uplifted child

ⓒ key has two children

i) move left children partial tree

on left child of smallest of right

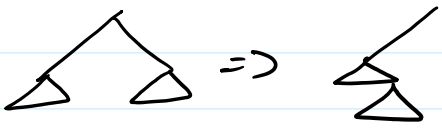
children partial tree



ii) move right children partial tree

on right child of largest of left

children partial tree



Tree structure depends on key order

tree structure depend on [illegible]

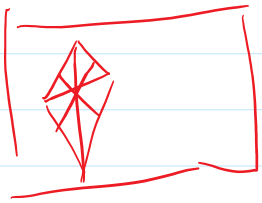
BST height

$O(\log n) \le h \le O(n)$

and all Insert / Search / Delete is of $O(h)$

"Structured / Balanced BST":

- Red Black, AVL, 2-3,
- $h_{average} = O(\log n)$



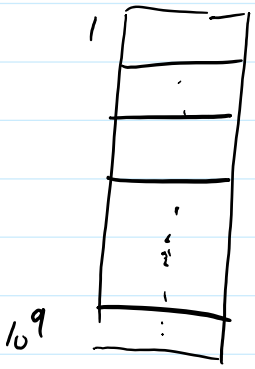
20171010

2017年10月10日　11:19

Hash Table: Insert, Dete, Search

Store n-elements

SIN xxx xxx xxx



	Time	Memory
Array	O(1)	96% waste
Linked list	O(n)	O(n)

Hash Table

n-elements

m-storage

$\alpha = \frac{n}{m}$ = load factor

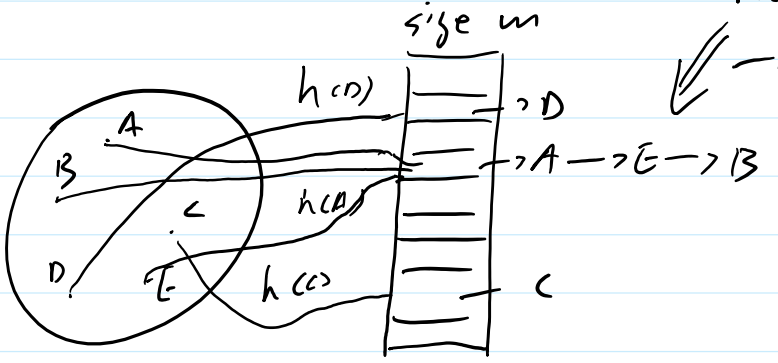
$\alpha \leq 1$ (expected, wish, want)

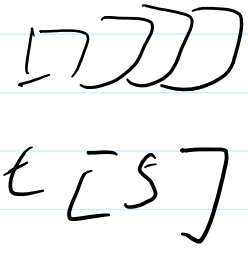
⇓

$m \geq n$

Collision Resolution:

→ by chain





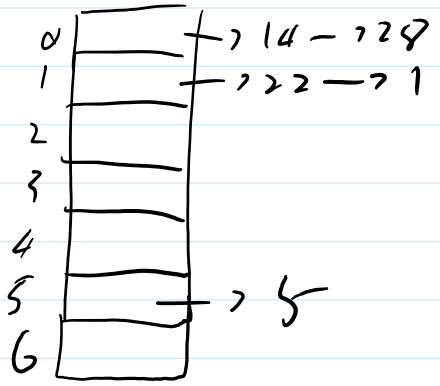
Division Method

h(k) = key mod m

good value for m = prime

worst 2^p

m=7



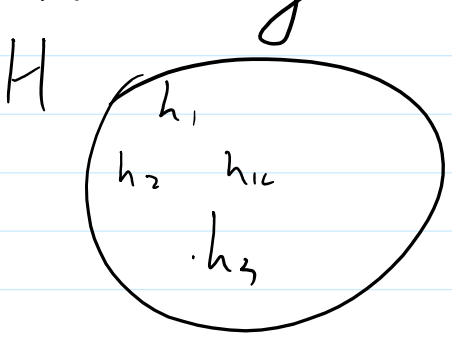
Multiplication Method

$h(key) = \lfloor m \cdot ((key \cdot A) \bmod 1) \rfloor$

where $0 < A < 1$

- 2^p = good for m
- $\hat{\phi} = \frac{\sqrt{5}-1}{2}$ = good for A

Universal Hashing



$O(1+\alpha)$ expected time

Open addressing

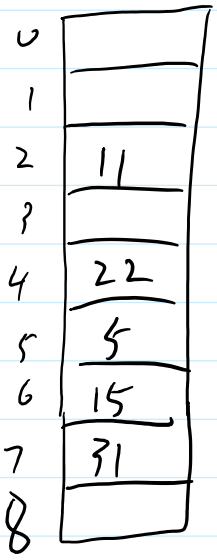
(Linear Probing)

$h_0 = key \bmod m$

$h_{j+1} = (h_j + 1) \bmod m$

↑ probe

$h = key \bmod 9$: $m = 9$ $c = 1$ (linear



when deleting, delete and insert a flag, indicating something

was there

Clustering

"Double Hashing"

Secondary hash function

$i_{j+1} = \left(i_j + h_2(key)\right) \bmod m$

α = load factor $= n/m \leq 1$

Chaining $1 + \frac{\alpha}{2}$

Linear Probe $\frac{1}{2}\left(1 + \frac{1}{1-\alpha}\right)$

Double Hash $\frac{1}{\alpha} \ln\left(\frac{1}{1-\alpha}\right)$

20171016

2017年10月16日　11:10

Dynamic Programming

"divide & conquer"

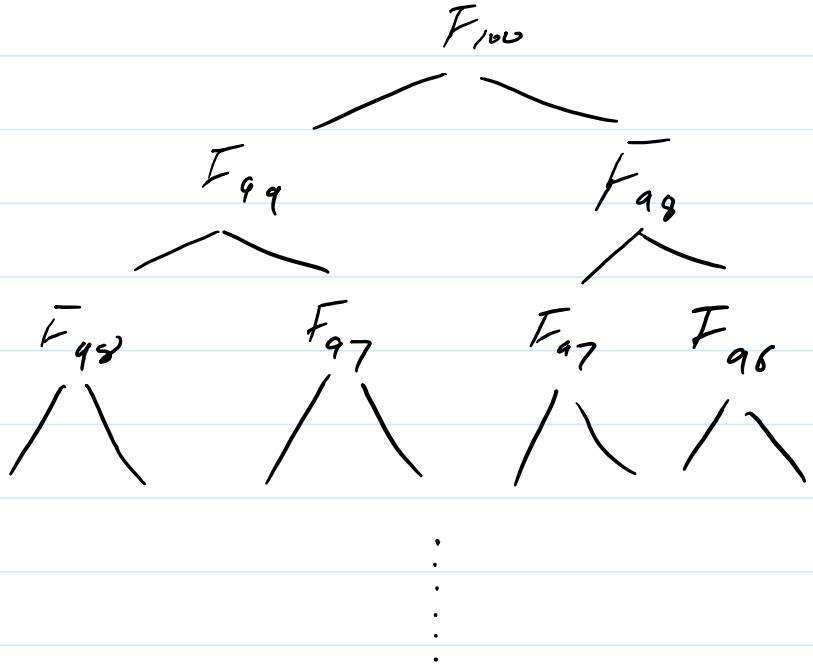
(A) optimal sub-problems
(like greedy)

(B) overlapping sub-problems
- Memorization

Fibonacci #s

$F_0 = 0 \quad F_1 = 1$

$F_i = F_{i-1} + F_{i-2}$



Matrix Parenthesization
↳ Multiplication

Matrice, $A_1 A_2 \cdots A_n$ to multiply in this order. Best way to reduce # of scalar multiplications?

$A_{n \times m}$. $B_{m \times k}$

scalar multiply $= n \times m \times k$

A_1 A_2 A_3

10×100 100×5 5×50 7500

$(A_1 A_2) A_3 = (10 \times 100 \times 5) + 10 \times 5 \times 50$ 11

$A_1 (A_2 A_3) = (100 \times 5 \times 50) + 10 \times 100 \times 50$ 100×50 11 75000

$P(n) =$ # scalar mult

$= \sum_{k=1}^{n-1} P(k) \cdot P(n-k)$

Naive: $= \Omega\left(4^n / n^{3/2}\right)$ computation

$A_1 A_2 A_3 \cdots A_k A_{k+1} \cdots A_n$

If A_i is $p_{i-1} \times p_i$

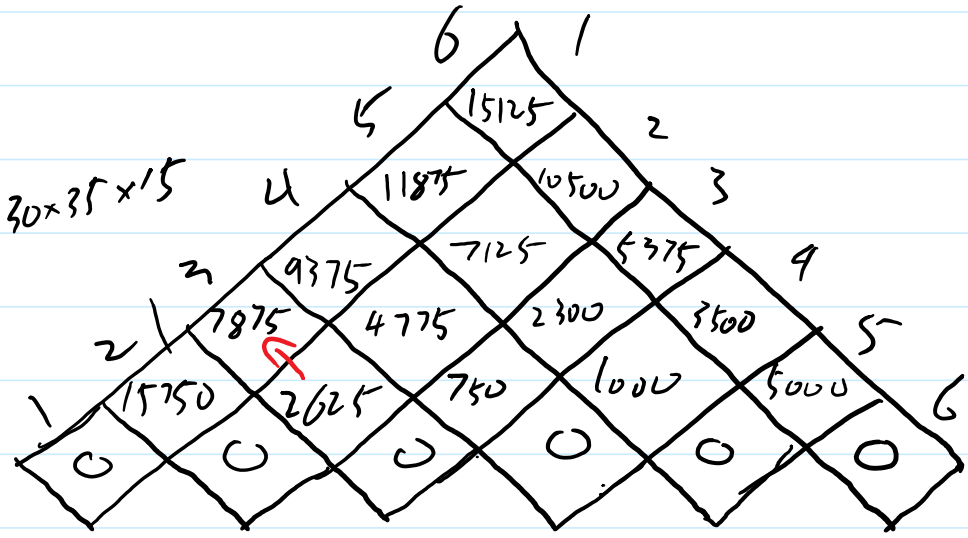
If $A_{ij} = A_i A_{i+1} \cdots A_j$ product

$m(i,j) = \begin{cases} 0 & \text{if } i = j \\ & \text{if } i < j \end{cases}$

$C[i,j]$ = optimal # of scalar mult to get A_{ij}

$$= \begin{cases} 0 & \text{if } i=j \\ \min\{m(i,k)+m(k+1,j)+P_{i-1}\times P_k \times P_j\} & \text{if } i<j \end{cases}$$

A_1 30×35
A_2 35×15
A_3 15×5
A_4 5×10
A_5 10×20
A_6 20×25



A_1 A_2 A_3

$(A_1 A_2) A_3 = 15750 + 30\times15\times5$
30×15 15×5 ✓ smaller

$A_1 (A_2 A_3) = 2625 + 30\times35\times5$
30×35 35×5

① $A_1(A_2A_3A_4) = 4775 + 30\times35\times10$

② $(A_1A_2)(A_3A_4) = 15750 + 750 + 30\times13\times10$

③ $(A_1A_2A_3)A_4 = 7875 + 30\times5\times10$

best: 9375

Print "via" k-index

Running time:

Running time:

$O(n) \cdot O(n^2)$

$= O(n^3)$

20171017

2017年10月17日　11:20

Longest Common Subsequence (not necessarily consecutive) (LCS)

Given the two seqs $X = x_1 \dots x_m$ & $Y = y_1 \dots y_n$ find longest subsequence (not necessarily consecutive) common to both (but in that order)

s p r i n g t i m e

p i o n e e r

Naive approach

Assume $m \le n$

$\Omega(n \cdot 2^m)$ time

Notation: $X_k = x_1 \dots x_k$ prefix

THM:

if $Z = z_1 \dots z_k$ is a LCS of X and Y

(a) if $x_m = y_n$ then $x_m = y_n = z_k$ and Z_{k-1} is a LCS of X_{m-1} & Y_{n-1}

(b) if $x_m \neq y_n$ then $z_k \neq x_m$ implies Z is LCS of X_{m-1} & Y_n

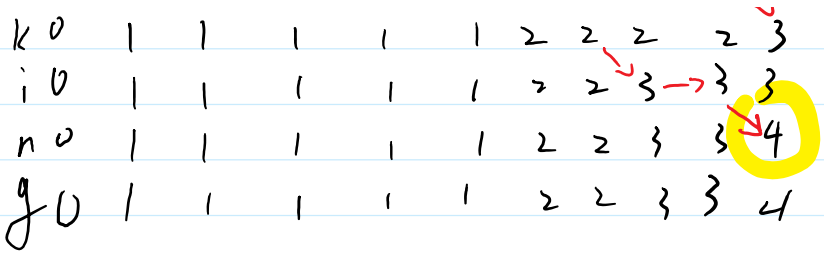
(c) if $x_m \neq y_n$ then $z_k \neq y_n$ implies Z is LCS of X_m & Y_{n-1}

$c[i,j]$ = length of LCS X_i & Y_j

$$c[i,j] = \begin{cases} 0 & \text{if } i = \phi \text{ or } j = 0 \\ c[i-1,j-1]+1 & \text{if } x_i = y_j \\ \max(c[i-1,j], c[i,j-1]) & \text{if } x_i \neq y_j \end{cases}$$

$i-1,j-1$	$i, j-1$
$j-1, j$	i,j

		a	m	p	u	t	a	t	i	o	n
	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0
p	0	0	0	1	1	1	1	1	1	1	1
a	0	1	1	1	1	1	2	2	2	2	2
n	0	1	1	1	1	1	2	2	2	2	3
k	0	1	1	1	1	1	2	2	2	2	3
i	0	1	1	1	1	1	2	2	3	3	3



Time

$O(m \cdot n)$

20171019

2017年10月19日　10:22

Greedy Algorithm

Principles:

- Optimal sub-problems
- Greedy choice: the local optimal (greedy) choice leads to a global optimal solution

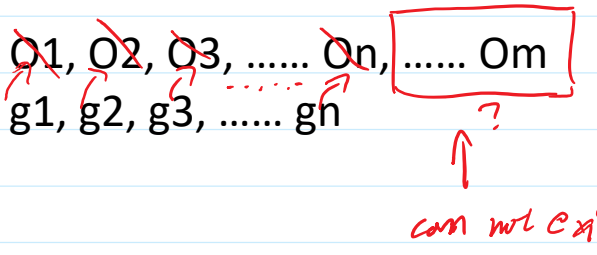
Class Scheduling

You got many (sometime overlapping) lectures, and one lecture hall. Maximize # lectures, scheduled in one day

- Sort by finish time
- Schedule by earlier finish time with no overlap
  - (O(nlogn) time)

Proof of correctness

A ta C that greedy g1, g2,.....gn is not optimal but some other solution o1, o2, …… om is (that is m>n)



Can you replace O_1 with g_1?

Yes. g_1 by definition

finish(g_1) ≤ finish(O_1)

If oi can be replaced by gi ∀ I ≤ n

And oj, m ≥ j ≥ n ∄ , hence greedy is optimal

A thief breaks into a store with n-items where item I weights wi and values vi. Thief can carry at most W weight (knapsack problem)

<u>O-1 (integral)</u>

Take or leave a while item

Dynamic

<u>Fractional</u>

Can take a fraction of an item

Greedy

Fractional

Can take a fraction of an item Greedy

Wants max. his profit:

0-1 (dynamic) -> assume some order 1. n to items

c[i, w] = optimal solution for items 1 .... i with w lbf overweight

$$c[i,w] = \begin{cases} \phi & \text{if } w = \phi \text{ or } i = \phi \\ c[i-1,w] & \text{if } w_i > w \\ \max\{c[i-1,w],\ c[i-1,w-w_i] + v_i\} & \text{if } w \geq w_i \end{cases}$$

20171024

2017年10月24日 11:11

Greedy Principles:
- Greedy Choice
- optimal sub problem

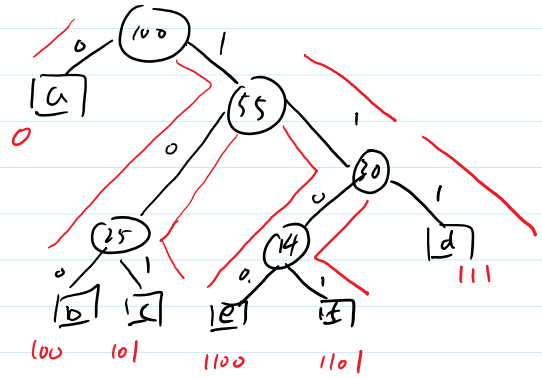
Huffman Code for Data Compression

char	G	a	b	c	d	e	f
f(c)	Frequency	45	13	12	16	9	5
d(c1)	Fixed size	000	001	010	011	100	101
d(c2)	variable size	0	101	100	111	1101	1100

B(T) = size of encoded file
$= \sum_c f(c)\, d(c)$ $\forall c$
= # of bits used

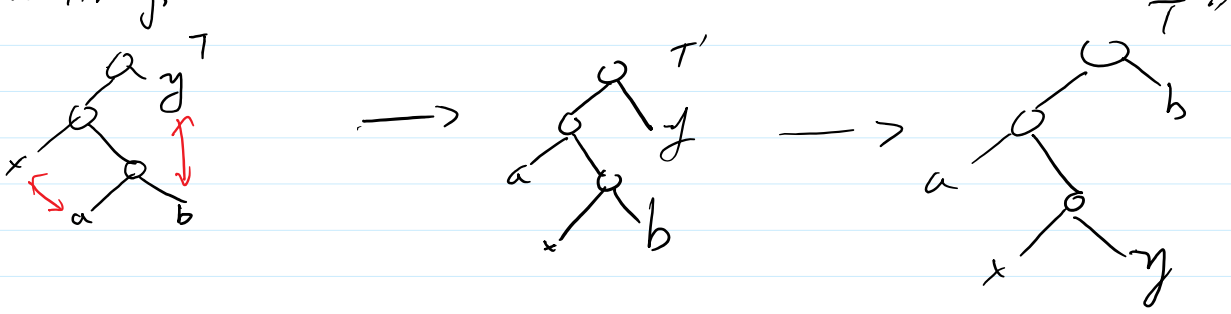
Greedy algorithm
- Unite the two lowest frequency chars & replace with new pseudo-char with sum of their frequencies
- Repeat forming a tree
- label tree edges

use min-heap



Greedy Principle: "every optimal coding tree has the two lowest frequency x, y as siblings of highest depth different in one single bit"

Proof: A to C It's not true but an optimal tree T has a & b at lowest depth as siblings



=> prove T' will be at least as optimal as T

$B(T) - B(T') = f(x)\cdot d_T(x) + f(a)\, d_T(a) - f(x) d_{T'}(x) - f(a) d_{T'}(a)$

but $d_T(x) = d_{T'}(x)$

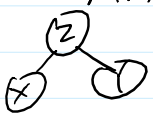
$d_T(a) = d_{T'}(c)$

$[f(a) - f(x)] \cdot [d_T(a) - d_T(x)] \geq 0$

≥ 0 ≥ 0 $\therefore \geq 0$

$\Leftarrow > B(T) \geq B(T')$

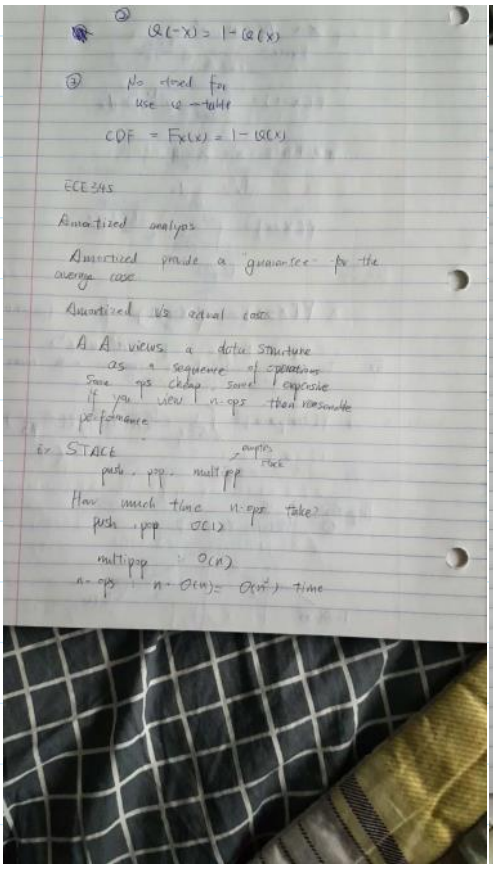
Optimal Substructure

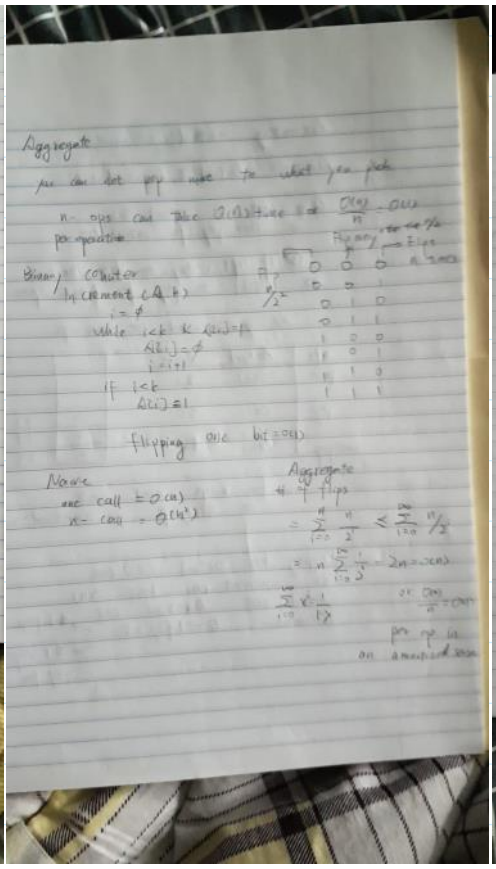
Let  be part of and optimal for character set C

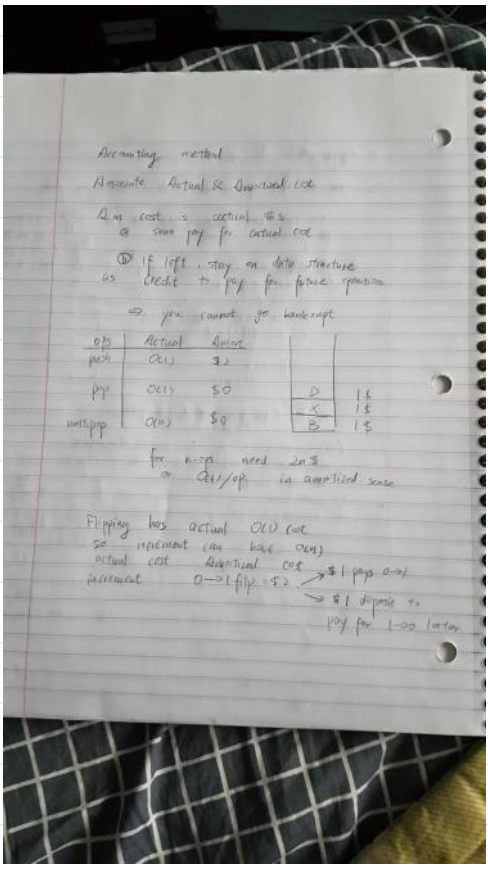
Then $T' = T - \{x, y\} + \{z\}$ is optimal for prefix code $C' - \{x, y\} + \{z\}$
where $f(z) = f(x) + f(y)$

20171026

2017年10月28日　21:19







20171030

2017年10月30日　11:10

Splay (x)

while x ≠ root

if P(x) = root

rotate p(x)　] ZIG

if P(x) & x both left or right children

rotate P(P(x))

rotate P(x)　] ZIG ZIG

else

rotate P(x)

rotate new P(x)　] ZIG ZAG

Splay trees

- they are BSTs

they are "balanced" tree in armotized sense

- they achieve the info theoretical optimal bound for the weighted dictionary problem

Keys A, B, C, ... Z

Freq 15% 12% 5% .....

search: splay (x)

delete: splay (parent (x))

The structure becomes better on air for more frequent seaches

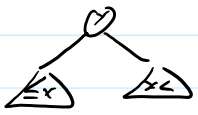
Search (x)

Like BST & Splay(x)

Insert (x) - ↗ same

Log n Delete (x): ↗ same & splay on parent delete item

Log n Join:



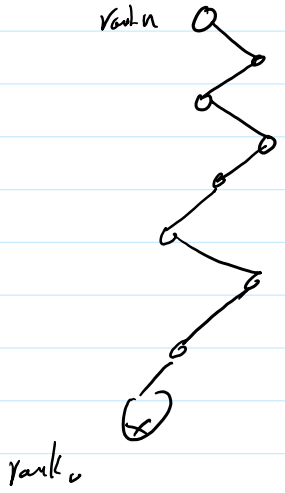
Log n split:

splay (x) & returns 3 component

Let weight $W(x)$ be the # of nodes in subtree of x incl. x
Define $rank(x) = \lceil \log W(x) \rceil$

Credit Invariant
Every node has $rank(x)$ $\$$s
($\Rightarrow$ we need show that splay costs $O(\log n)$ $\$$s to pay for rotations but also maintain the invariant

Claim (for tomorrow)
every ZIG or ZIG ZIG or ZIG ZAG costs $3(newrank(x) - oldrank(x))$ except from final zig that may require one ($\#$ extra



How much splay costs?

$3(rank_1 - rank_0) + 1$
$3(rank_2 - rank_1) + 1$
$3(rank_3 - rank_2) - 1$
$\vdots$
$3(rank_n - rank_{n-1}) + 1$
$= 3(rank_n - rank_0) + 1$
$\leq 3(\log n - \log 0) + 1 = O(\log n)$ $\$$s

20171031

2017年11月5日　21:44

Splay
2017年10月30日 11:13

Splay (x)

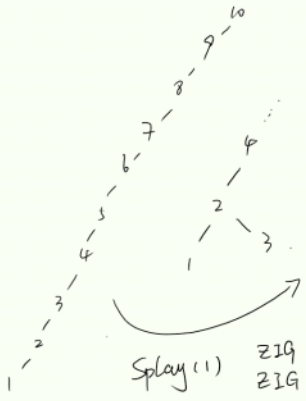
SPLAY (x)　　cost of splay(x) O(log n) $s

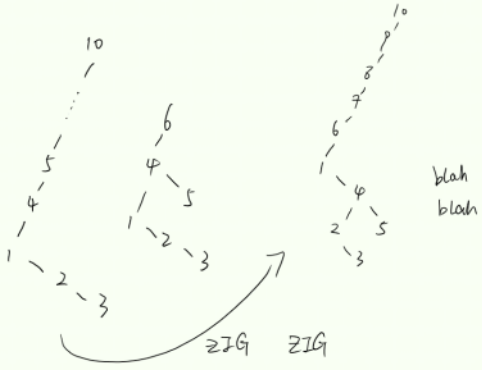
while x ≠ root
　if p(x) = root　] ZIG
　　rotate p(x)
　if p(x) & x both left or right children　] ZIG
　　rotate p(p(x))
　　rotate p(x)
　else
　　rotate p(x)　] ZIG
　　rotate new p(x)　] ZAG

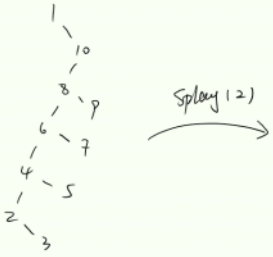
每次把新 insert 的都放到 root

SPLAY TREES

- They are BSTs
- They are "balanced" trees in amortized sense.
- They achieve the info theoretical optimal bound for the weighted dictionary problem.







- Search (x)　O(log n) $s
　like BST & splay(x)

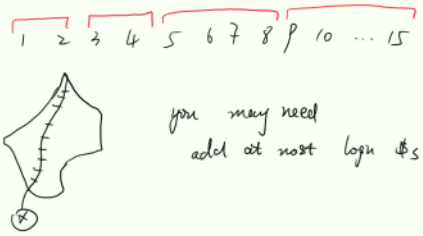
O(log n)$s Insert (x) : same as splay(x)

O(log n)$s Delete (x) : same ↑ and splay on parent of delete item

O(log n) Join :

O(log n) Split :

splay(x) & returns 3 components



you may need added at most log n $s

Let weight wT(x) be the # nodes in subtree of x include x　(wT(x) = # children of subtree plus x)
Define rank(x) = ⌈Log wT(x)⌉

Credit Invariant
　Every node has rank(x) $s.
　(⇒ we need show that splay costs O(log n) $s

Credit Invariant

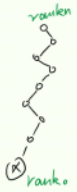
Every node has rank(x) $s.

(⇒ we need show that splay costs O(log n) $s to pay for rotation but also maintain the invariant)

Claim

Every zig, or zig-zig or zig-zag costs $\leq 3[\text{newrank}(x) - \text{oldrank}(x)]$

except from final zig that may require one $ extra.



How much a splay costs ?

$3(\text{rank}_1 - \text{rank}_0) +$

$3(\text{rank}_2 - \text{rank}_1) +$

⋮

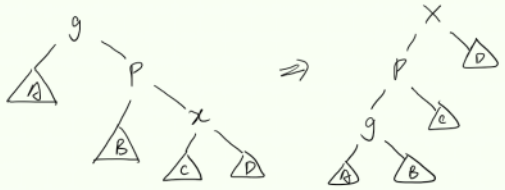
$3(\text{rank}_n - \text{rank}_{n-1}) + 1$

$= 3(\text{rank}_n - \text{rank}_0) + 1 \leq 3(\log n - \log 0) + 1 = O(\log n)\$s$

Thm: Splay costs O(log n) $s

↳ time in amortized sense

ZIG ZIG



only g & p & x may changes ranks

You have

oldrank(x) + or(g) + or(p)

and you need

newrank(x) + nr(p) + nr(g)

To fix ranks you need

$nr(x) + nr(p) + nr(g) - or(x) - or(p) - or(g) \leq 2[nr(x) - or(x)]$ to fix rank

$nr(p) + nr(g) \leq 2nr(x)$; $or(x) - or(p) - or(g) \leq or(x)$

Observations:

$nr(p) \leq nr(x)$

$nr(g) \leq nr(x)$

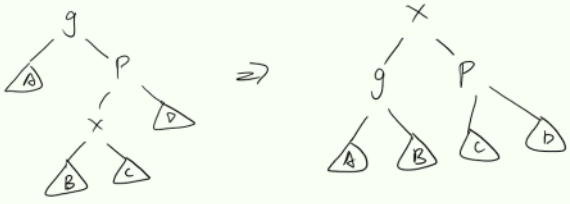
$nr(x) = or(g)$

$or(p) \geq or(x)$

I allocated x3 this amount (claim). Use x2 from this to fix ranks and 1x to pay for rotation ... which works unless $nr(x) = or(x)$. In that case I don't need to fix any ranks but who pays for rotation ?

The tree ... if $or(x) = nr(x) \Rightarrow$ more than $\frac{1}{2}$ tree nodes were in C & D under $x \Rightarrow < \frac{1}{2}$ tree nodes in A & B $\Rightarrow$ after rotation g drops in rank releasing $ to pay for rotation

ZIG - ZAG



Exactly as ZIG-ZIG unless

$nr(x) = or(x)$ in that case

B & C have $> \frac{1}{2}$ tree nodes

Case 1

B >> C

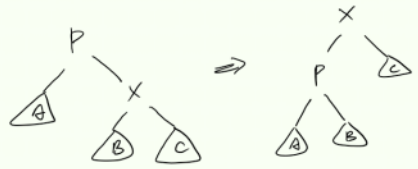
Case 2

Case 1
B >> C
then p drops in rank releasing $s to pay rotation

Case 2
both g & p drop in rank

Case 0
B << C

ZIG



$or(x) \le nr(x)$

$or(p) \ge nr(p)$

$nr(x)+nr(p)-or(x)-or(p) \le nr(x)-or(x)$

1x of claim fixes ranks & x2 remaining pay for rotation

But if $nr(x)=or(x)$, use the extra 1$ to pay rotation.

20171102

2017年11月5日　21:43

BFS , DFS

2017年11月2日 10:11

Breadth First Search (BFS)

$\forall u \in V \quad d[u] = \infty$

$d[s] = \phi \ , \ Q = \phi$

enqueue (Q , s)

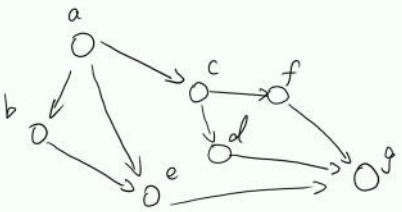
while $Q \neq \phi$

$\forall$ v adjacent to u

if $d[v] = \infty$

$d[v] = d[u] + 1$

enqueue (v)



Q :

d : $a_0\ c_1\ b_2\ e_3\ d_4\ f_5\ g_6$

time $O(V+E)$

Depth First Search (DFS)

DFS(V, E)

$\forall u \in V$

color (u) = white

time = ϕ

for each $u \in V$

if color (u) = white

DFS - visit (u)

DFS - visit

color (u) = gray

time = time + 1

$d[u]$ = time

$\forall$ v adjacent to u

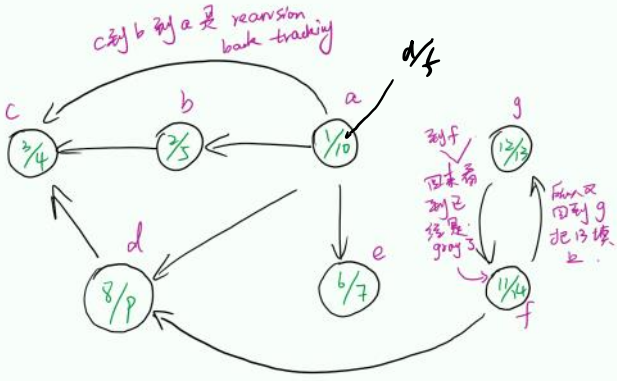
if color (v) = white

DFS - visit (v)

color (u) = black

time = time + 1

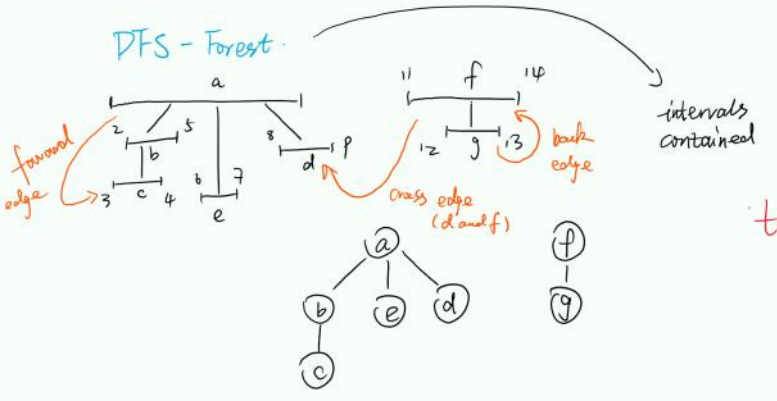
$f[u]$ = time



white : not discovered

gray : discovered but not finished

black : finished



time (V+E) visit every nodes twice

Thm In a DFS of an undirected graph

Thm In a DFS of an undirected graph every edge is a tree or back edge

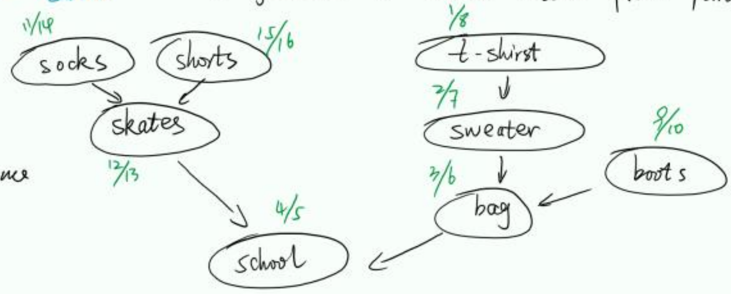
No Fwd = back - edge (no Fwd 就是 back edge)

No cross = tree

Topological Sort → generate a total order from partial order

↓

given a DAG generate a serial sequence of events



TS:

- run DFS
- output vertices in decreasing finish time

short → socks → skates → books → shirt → sweater → bag → school

图的意思是 skates 发生在这2个之后 需要满足这个前提

Correctness

show that if $(u,v) \in E \Rightarrow f(v) < f(u)$

(u) → (v) when I visit u->v what is the color of v?

u = gray

- v = white $f(v) < f(u)$
- v = black $f(u) < f(u)$
- v = gray

Lemma A directed G is a DAG → directed acyclic graph has direction no cycle

iff DFS yields no back edges.

⇒ A Ta C G = DAG but ∃ back edge impossible as you get a cycle

⇐ DFS no back edge but G ≠ DAG → G has a cycle ⇒ ∃ a DFS generating a back edge

20171106

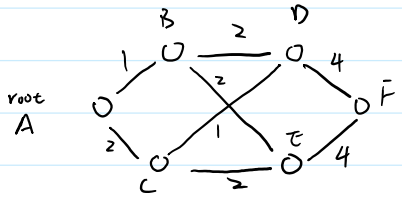
2017年11月6日　11:12

Minimum Spanning Tree

Undirected positively weight connected Graph G = (V,E) weight of edge (u,v) denoted as w(u,v) find a tree T that connects all vertices with minimum weight

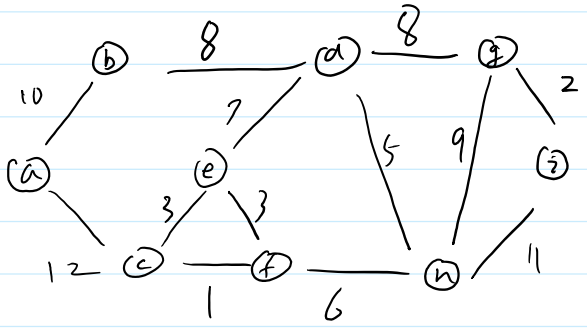
minimize $w(T) = \sum_{(u,v) \in T} w(u,v)$

Note: MST may not be unique



MST1: A-B-E-F (10 units)
D-C

MST2: A-B-C-D-F (10 units)
E



Assume you have a "partial" MST
An edge is called <u>safe</u> if it can be added to T_P creating a "larger" MST

A <u>cut</u> V_1 & V_2 in a G_T is a partition of V s.t. $V_1 \cap V_2 = \phi$ & $V_1 \cup V_2 = V$

Edge (u,v) crosses a cut if $u \in V_1$ & $v \in V_2$

Edge (u,v) crossing a cut is called <u>light</u> iff it's of minimum weight

Generic Solution

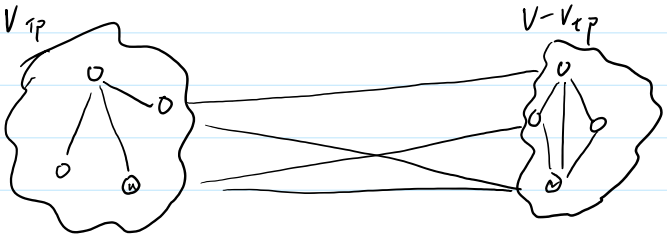
T_p = root vertex (any vertex)

while T_p does not contain all vertices,

  if (u,v) is a safe edge

    let $T_p = T_p \cup (u,v)$

THM Let T_p be a subset of a MST on vertices V_{T_p}. Let light (u,v) edge crossing cut $(V_{T_p}, V - V_{T_p})$. Then (u,v) = safe edge



AtoC (u,v) ≠ safe but some other edge (x,y) is, $w(x,y) > w(u,v)$. Add (x,y), grow the MST incl $V - V_{T_p}$ partition. In the MST add now (u,v). It creates a cycle. Break cycle by removing (x,y) $w(\text{new tree}) < w(\text{MST})$ a contradiction

PRIMS

Insert every vertex in queue Q decrease(Q, root, ∅)

while(Q ≠ ∅)

  extract-min(Q) → u

  ∀ v adjacent to u

    if w(u,v) < key(v)

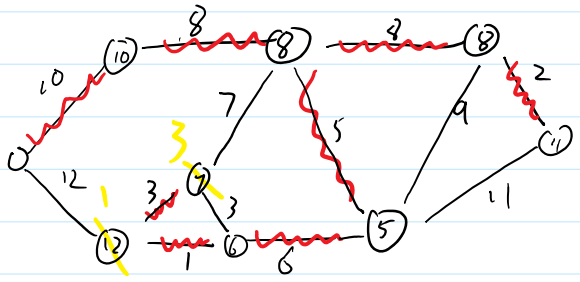
V log V

decrease-key (key(v), w(u,v))

E log V

change key(v) to w(u,v), bubble up

O(1) O(log V)



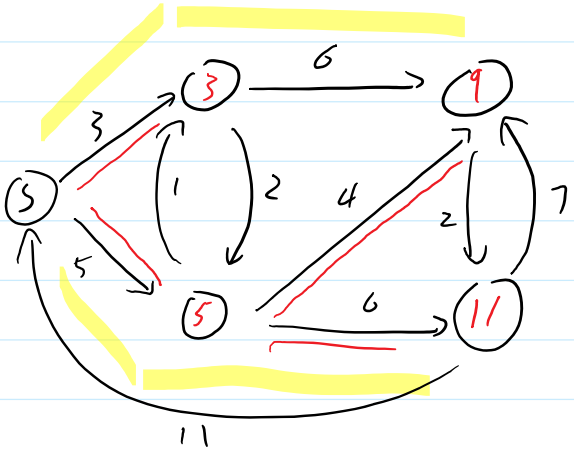
use priority queue with |V| elements

Heap

20171107

2017年11月7日　11:13

Single Source Shortest Paths



- Find the shortest path from a single source to every city considering the weights are mileage
- Weights can be
  - Penalties
  - Cost
  - Profit etc.
- Shortest path may not be unique
- Weights can be positive or negative but cannot have negative cycles

- Dijkstra (positive only weights)
- Bellman-Ford(positive or negative weights, also reports negative cycles)

Variations

- Single source SP(SSSP) / Dijkstra = greedy
  - We focus
- Single destination SP
  - Reverse edges & run SSSP
- Single pair SP
  - SSSP again
- All pair SP
  - Dynamic programing

Formal definition

Given a weighted directed G, find $\delta(s, v)$ where

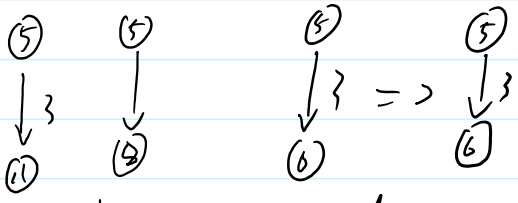
$$\delta(s,v) = \begin{cases} \min w(p): u \overset{p}{\rightsquigarrow} v & \text{if } \exists\, u \overset{p}{\rightsquigarrow} v \\ \end{cases}$$

$$\delta(s,v) = \begin{cases} \\ \infty \quad \text{if} \quad \nexists\ u \overset{p}{\rightsquigarrow} v \end{cases}$$

Both algorithms keep an estimate $d[v]$ of $\delta(s,v)$
where $d[v] \geq \delta(s,v)$ and initially estimate $d[v] = \infty\ \forall v \in V$

They iterate keep improving the estimate by "relaxing" edges

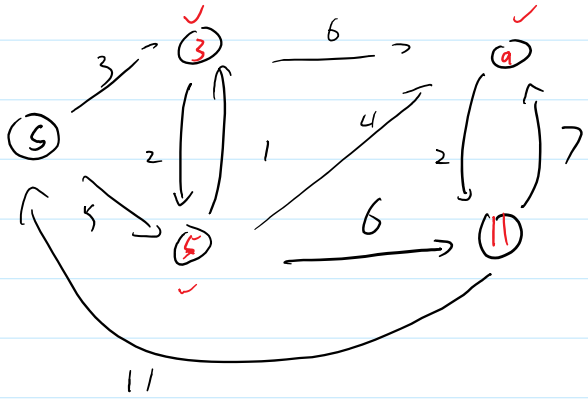
```
RELAX (u,v) /* (u,v) ∈ E */
    if d[v] > d[u] + w(u,v)
        d[v] = d[u] + w(u,v)
        π(v) = u   /* parent of v is now u */
```



DIJKSTRA (no neg-weights)

```
Initialize d[v] = ∞ ∀ v ∈ V - {s}
d[s] = φ ;   Q = V /* priority queue */
while Q ≠ φ
    u = extract min(Q)
    S = S ∪ {u}
    ∀ vertex v adjacent to u do
        Relax(u,v)
```

time
$O(E \log V)$ w/ heap



20171109

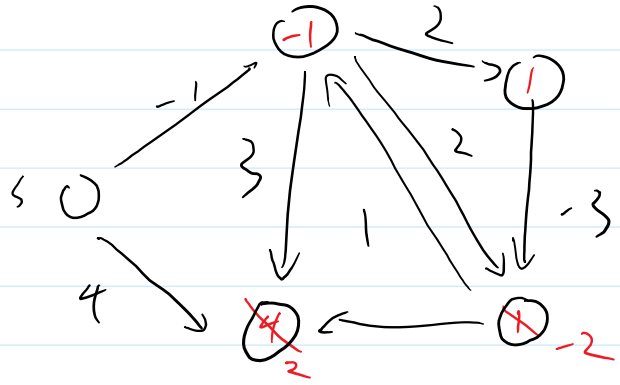
November 9, 2017 10:12 AM

SSSPs

$$\delta(u,v) = \begin{cases} \infty & \text{if } \nexists\ s \overset{p}{\rightsquigarrow} v \\ \min(w(p) : s \overset{p}{\rightsquigarrow} v) & \text{if } \exists\ s \overset{p}{\rightsquigarrow} v \end{cases}$$

Estimate

Bellman-Ford



∀ v d[v] = ∞ ; d[s] = 0

For i ... |V-1|

relax (u,v) ∀(u,v) ∈ E time O(VE)

For ∀edge(u,v) ∈ E

if d[v] > d[u] + w(u,v)

return NEG_cycle

Why does it report NEG-cycle

Assume ∃ neg cycle $v_0, v_1, v_2, \ldots, v_k$ ($v_k = v_0$)

ATOC it does not report a neg cycle

$\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$

$d[v_1] \le d[v_0] + w(v_0, v_1)$

⋮

+) $d[v_k] \le d[v_{k-1}] + w(v_{k-1}, v_k)$

$0 \le w(v_0,v_1) + w(v_1,v_2) + w(v_2,v_3) + \cdots + w(v_{k-1}, v_k)$

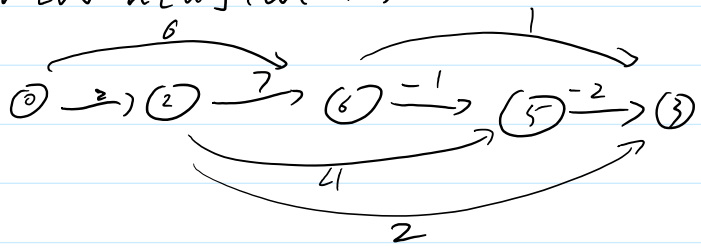
Positive

SSPs on DAGs

$d[v] = d[u] + w(u,v)$ Topological Sort V times

SSPs on DAGs

$d[v] = d[u] + w(u,v)$



- Topological Sort Vertices
- Relax edges in that order

$O(V+E)$

Objective Function

minimize (or max)

$[c_1 \dots c_n] \begin{bmatrix} x_1 \\ \vdots \\ x_2 \end{bmatrix}$

$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$

Integer programming

Given Constraint

$Ax \leq B$

find values for x that

- Ellipsoid (poly but slow)
- Simplex (exp. but fast)

Difference Constraints

$x_1 - x_2 \leq 5$

$x_1 - x_3 \leq 6$

$x_2 - x_4 \leq -1$

$x_3 - x_4 \leq -2$

$x_4 - x_1 \leq -3$

we reduce DCs to a SSSP problem

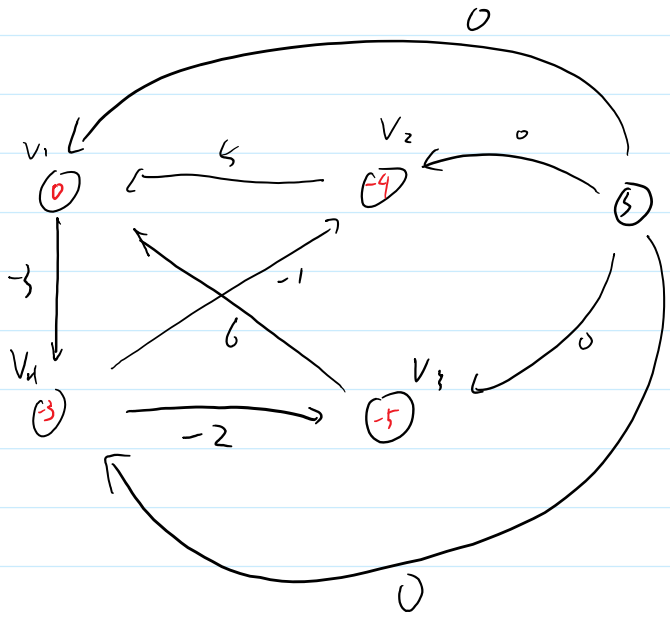
algorithm

Build a "constraint" graph

- v_i vertex corresponds to a variable x_i
- Add Pseudo source v_0 with edges ϕ to everybody
- Add edge $v_i \xrightarrow{b_k} v_j$

if $x_j - x_i \leq b_k$

- run Bellman Ford



20171113

November 13, 2017 12:16 PM

Computability & Complexity

"Logicomix" Christos Papadimitriou

"Emperor's New Mind" by Penrose

Computational Models

Regular Languages (r.e)

Given an alphabet Σ, regular expressions are defined as follows

- ϵ (empty string) is a r.e
- $\forall\, a \in \Sigma$, a is a r.e
- if r & s are r.e, then $r|s$ is a r.e
- if =/= , then $r \cdot s$ (concatenation) is a r.e
- if =/= , $r^* = r \cdot r \cdots r$ is a r.e
  - 0 or more times
  - ↖ empty string

given r a r.e

$L(r)$ = language of r (that is, the set of strings generated by r)

$L\{(0|1)(0|1)\} = \{00, 01, 10, 11\}$

$L\{(0|1)^*1\}$ = every odd binary numbers

compiler :

for (i=1 ; i<10 ; i++) ⟶ string x → M → yes, $x \in L(m)$ / No, $x \notin L(m)$

recognizer

DFAs ≡ r.e ≡ NFAs

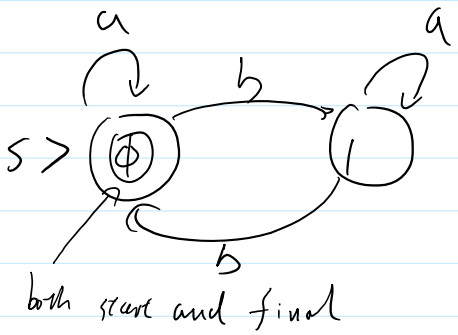
Deterministic Finite Automata

Σ : finite alphabet	$\Sigma = \{a, b\}$
K : finite # of states	$K = \{0, 1\}$
s : start state	$s = \{\phi\}$
f : set of finite states	$f = \{0\}$
δ : transition function $K \times \delta \to K$	

a a

$K \times \Sigma \rightarrow K$

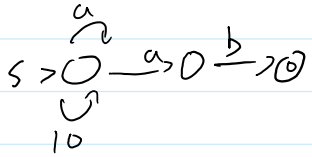
Current state	next state	
	a	b
0	0	1
1	1	0



L(DFA) = a string with even number of "b"s

$= a^* (b\ a^*\ b)^* a^*$

Non-deterministic FA



$L(0^n 1^n)$ ← impossible for DFA since DFA has no memory

DFA: Limited machines

20171114

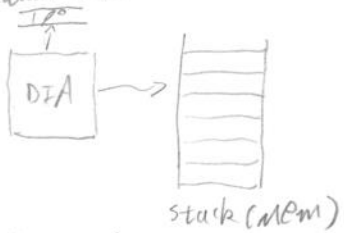
November 16, 2017 11:40 PM

$L = \{0^n 1^n\}$

$L = \{ww\}$ → no PDA

$L = \{$

Pushdown Automator



nPDA ≠ PDA

Context Free Grammers

E → E+E

E → E*E

E → E-E

E → (E)

E → E/E

E → number

Ex: E → E+F → *E) + E → (E*E) + E → (num * num) + num

Turing machine

I/O tape

...	x	a	b	b	c	d	...

DFA

Current State	Next State	action
1	4	
2	1	
3	2	

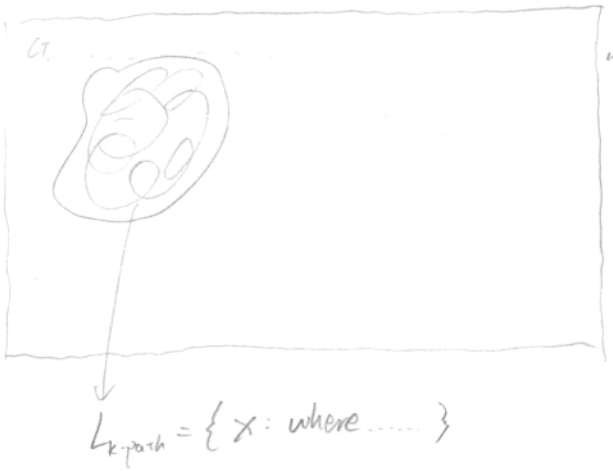
Alphabet

$\Sigma = \{0, 1\}$

Godelization

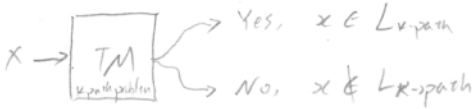
- ∀ problem can be encoded as sequence of 0's & 1's
- You can also code the specification of any TM as a sequence of 0's & 1's

Given G is there a path of length $\leq k$ between vertices u & v?



$L_{k\text{-path}} = \{ x : \text{where} \ldots\ldots \}$

TM decides



TM accept

TM decides: you have a guarantee TM will halt eventually with a yes/no answer

TM accepts: if it halt, answer is correct but as long as it runs no guarantee it will ever halt

Decision Problems vs Optimization Problems

Is there a path of k edges from u to v? → Decision

What is the best path from u to v? → Optimization

$OPT \geq DEC$

Every TM's Specification can be encoded as a sequence of 0's and 1's $\{0,1\}^*$ $M_\alpha(x)$

TM with spec α running on input x, $\alpha, x \in \{0,1\}^*$

universal TM:

$\forall x, \alpha \in \{0,1\}^* \ \exists$ a UTM U s.t. $U(x,\alpha) = M_\alpha(x)$

if $M_\alpha(x)$ halts in T-steps, then U ~~halts~~ holes in $CT\log T$ steps

There is a function $UC: \{0,1\}^* \to \{0,1\}^*$ that is not computable by any TM

Proof:

$UC: \forall \alpha \in \{0,1\}^*$: if $M_\alpha(\alpha)=1$ then $UC(\alpha)=0$

if $M_\alpha(\alpha)=0$ or does not halt then $UC(\alpha)=1$

Aim: $\exists$ TM M computes UC, that is $M(\alpha)=UC(\alpha)$ $\forall \alpha \in \{0,1\}^*$

Then $UC(M) = UC(M)$ but this impossible since $M(M)=1$ iff $UC(M)=0$, vice versa

Halting Problem

$HALT(A, X) = 1$ there is M_A halts on input X is undecidable

A to C $\exists M_{HALT}$. I use it to build machine computing VC)

M_{VC} build as follows

- on input w, M_{VC} runs $M_{HALT}(a, a)$
  - if result 0, then output 1
  - if result 1, then run UTM simulating M_{VC}
    - if U=1 then output 0 for M_{VC}
    - if U=0 then output 1 for M_{VC}

20171116

November 16, 2017 10:19 AM

g

Complexity Theory

Complexity class P

$P = \{ L \in \{0,1\}^* : \exists$ algorithm to decide $x \in L$ in poly time

Thm

$P = \{L$ can be accepted in poly time$\}$

if it's accepted in cn^k time

simulate this machine in $cn^k \log cn^k$ time to decide it

Verification

Algorithm A verifies a problem if given instance $x \in L$ of problem there exist a witness (or certificate, "candidate solution") y such that $A(x,y) = 1$ Language verified is: $L \{ x \in \{0,1\}^* : \exists y \in \{0,1\}^*$ s.t. $A(x,y)=1 \}$

Complexity class NP

Informally: NP is the class of problems that can be verified in poly time

Formally: $L \in NP$ if $\exists$ poly time algorith A & constant c s.t.

$L = \{ x \in (0,1)^* : \exists$ certificate y where $|y| = O(|x|^c)$ s.t. $A(x,y)=1$ in poly time $\}$

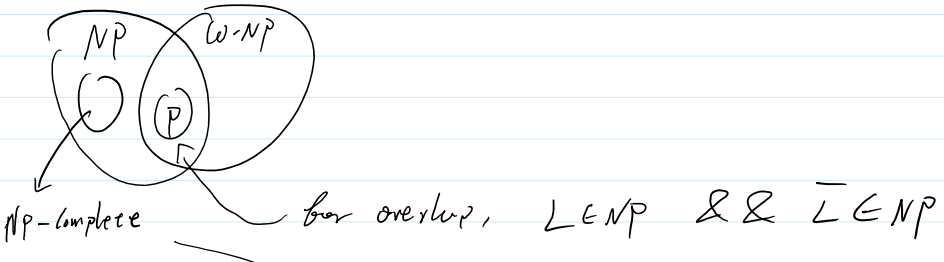
Thm $P \subseteq NP$

if L can be solved in poly time, just decide it & compare the solutions

Class con-np

if $L \in NP$ then $\overline{L} \in co\text{-}np$

Thm: if $L \in P$ then $\overline{L} \in P$



Open Questions

$NP \stackrel{?}{=} co\text{-}NP$

$P \stackrel{?}{=} NP \cap co\text{-}NP$

$NP \stackrel{?}{=} P$

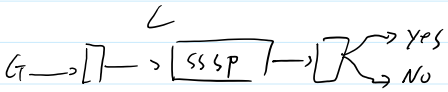
NP-Complete

Any NP problem can be reduced in poly time to a NPC problem

NP-complete

↓

NPC problem


G
SSSP
K
yes
No

20171120

November 28, 2017 8:12 PM

Reducibility

- Instances of problem Q can be reduced to instances of problem Q' if a solution to Q' gives a solution to Q (informal)
- (formal)
  - Language L_1 is poly-time reducible to L_2 denoted
    a) $L_1 \leq_P L_2$ iff $\exists$ function f that takes poly time to compute s.t. $x \in L_1$ iff $f(x) \in L_2$

L_1 not harder than L_2

THM

if $L_1 \leq_P L_2$ & $L_2 \in P$ then L_1 belongs to P

given M_{L_2}

Q: if $x \in L_1$?

=>

NP-Complete Class

- We say $L \in NPC$ iff $L \in NP$ (can be verified in poly time)
- $\forall L' \in NP$, $L' \leq_P L$
  (NP-hard)

THM

① if $\exists L \in NPC$ & $L \in P$ then $NP = P$

② $NP = co\text{-}NP$ iff $\exists L \in NPC$ s.t. $\bar{L} \in NP$

$\Leftarrow$ let $L \in NPC$ s.t. $\bar{L} \in NP$. Pick $L' \in NP$.

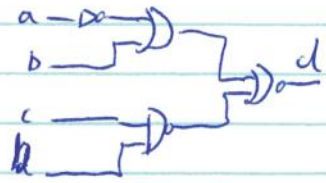
$\bar{L}' \in co\text{-}NP$. $L' \leq_P L \Leftrightarrow \bar{L}' \leq_P \bar{L}$

but $\bar{L} \in NP$ hence $\bar{L}' \in NP$

Methodology

- To show that $L \in NPC$
  - (a) show $L \in NP$ (verified in poly time)
  - (b) pick known $L' \in NPC$ and show $L' \leq_P L$
    - (b1) $x \in L'$ iff $f(x) \in L$
    - (b2) $f()$ takes poly time to compute

Circuit - SAT $\in NPC$



Given a circuit with AND, OR, NOT gates, is there an input assignment that makes output = 1 ?

Formula - SAT

- Let formula ϕ with $x_1, \ldots x_n$ vars & connectives $\neg, \wedge, \vee, (,), \Rightarrow, \Leftrightarrow$

Decision Problem

- Is there an assignment to $x_1, \ldots x_n$ that make it = 1 ?

Solution:

- (a) just plug the values, and evaluate in poly time
- (b) Circuit SAT $\leq_P$ Formula SAT: given any circuit C I will generate a ϕ in poly time s.t. Output = 1 $\Leftrightarrow$ ϕ = 1

20171121

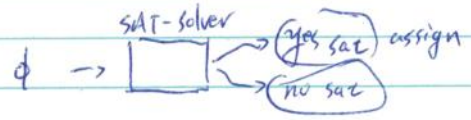
November 28, 2017 8:12 PM

3-CNF-SAT (conjuctive normal form)

Decision problem: given variables $x_1 \dots x_n$ as a disjunction of conjunctions of clauses of 3-literals, is there a satisfying assignment?

$\phi = (x_1 \vee x_3 \vee \bar{x}_2) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_5 \vee \bar{x}_3 \vee \bar{x}_2) \wedge (\cdots)$

clauses; literals



Formula-SAT $\leq_P$ 3-SAT (CLRS)

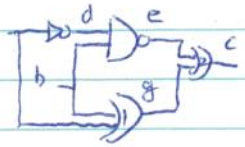
circuit-SAT $\leq_P$ 3-SAT

(a) show 3-SAT $\in$ NP

Given assignment $\forall i, x_i$

plug binary values and evaluate ϕ

(b) circuitSAT $\leq_P$ 3-SAT



given a crt I will generate a ϕ 3-SAT s.t. crt output = 1 $\Leftrightarrow$ 3-SAT is satisfiable

characteristic Function

a	b	$a \overset{g}{\vee} b$	T/F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Max term

$\Phi_{or} = (\bar{a} \wedge \bar{b} \wedge g) \vee (\bar{a} \wedge b \wedge \bar{g}) \vee (a \wedge \bar{b} \wedge g) \vee (a \wedge b \wedge \bar{g})$

$\bar{\Phi}'_{or} = (\bar{a} \wedge \bar{b} \wedge g) \vee (\bar{a} \wedge b \wedge \bar{g}) \vee (a \wedge \bar{b} \wedge \bar{g}) \vee (a \wedge b \wedge g)$

$\phi'_{or} = (a \vee b \vee \bar{g}) \wedge (a \vee \bar{b} \vee g) \wedge (\bar{a} \vee b \vee g) \wedge (\bar{a} \vee \bar{b} \vee g)$

$\phi = \phi'_{or} \wedge \phi^2_{or} \wedge \phi_{normal} \wedge \phi_{not} \wedge$ (cvx1vx2) (cvx1vx2) (cvx1vx2) (cvx1vx2)

$(a \vee b) \equiv (a \vee b \vee x) \wedge (a \vee b \vee \bar{x})$

↑

so $c = 1$, ϕ is working

20171127

November 27, 2017 12:39 PM

NPC

1) $L \in NP$

2) $L' \in PL$ for all $L' \in NP$

⇕

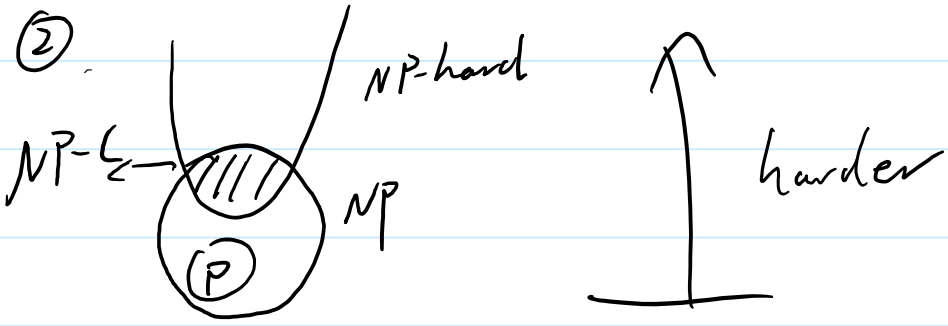
1) $L \in NP$

2) $L \in NP\text{-hard}$

① $A \leq_P B$

↳ B at least as hard as A

②



Subset - SUM

given set S is there a subset s.t. sum = t

a) $SS \in NP$

certificate: set of numbers C

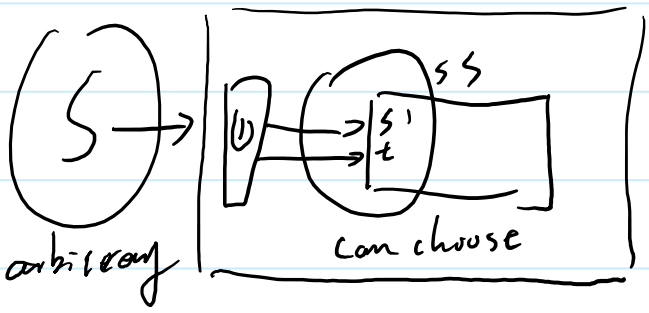
1) $S \cup C = S$

2) $\sum_{c \in C} = t$ can be done in poly time

b) SET-PARTITION (SP) $SP \leq PSS$

b) SET-PARTITION (SP)　　$SP \leq PSS$

given S, is there σ_1, σ_2 SS s.t. $\sigma_1 = \sum \sigma_2$



$S' = S$

$t = \frac{\sum S}{2}$

20171128

November 28, 2017 11:20 AM

To show $L \in NPC$

(a) show $L \in NPC$ (verified in poly time)

(b) Pick known $L' \in NPC$

(b1) show $x \in L'$ iff $f(x) \in L$

(b2) show $f(\cdot)$ takes poly time

} $L' \leq_p L$

Clique

a **clique** is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent

Decision Problem

Does G have a clique of size k?

(a) in $O(V^2)$ time I can verify if the witness is a clique or not

(b) $3\text{-SAT} \leq_p$ clique

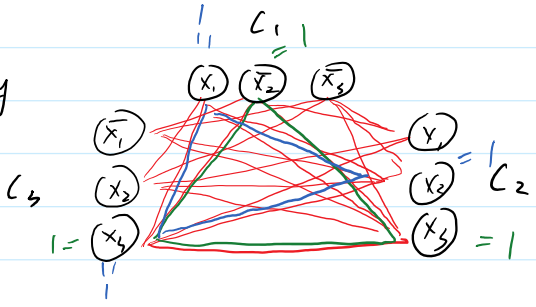
$\phi_1 = C_1 \wedge C_2 \wedge C_3$

$C_1 = x_1 \vee \bar{x}_2 \vee \bar{x}_3$

$C_2 = x_1 \vee x_2 \vee x_3$

$C_3 = \bar{x}_1 \vee x_2 \vee x_3$

- introduce a vertex per literal
connect vertices from different clauses if they are not complementary



CNF is satisfied

↑ iff ↓

$\exists$ clique of size # of clauses

Vertex Cover

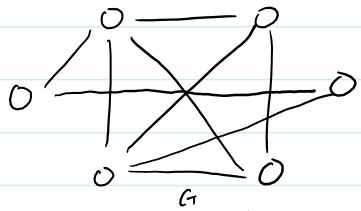
VC = a subset $V' \subseteq V$ of G s.t. if $(u, v) \in E$ then at least one of u or v is in V'

Optimization: what's the min VC?

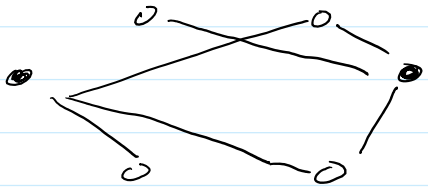
Decision: is there a VC of k?

(a) easy

(b) clique $\leq_p$ VC



G



G have a clique of size k (=4)

iff

.... has a VC of size 4

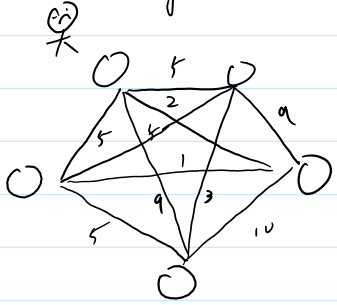
iff

$\overline{G}$ have a VC of size $|V|-K$

Traveling Sales Man Problem

you are give a complete graph (clique) which is weighted.

find a Ham cycle of minimum weight (optimization)



Decision: is there a TSP of size K

@ easy.

(b) HAM $\leq_P$ TSP

add remaining edges ϕ weight if original

1 .... if new edge

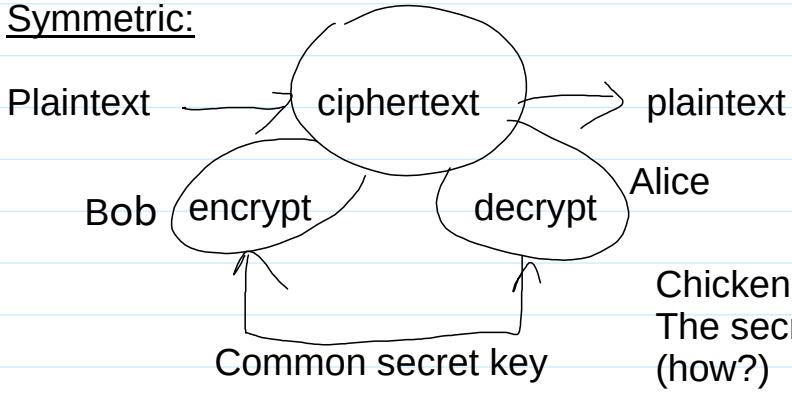
Is there a TSP of size ϕ?

20171204

2017年12月4日　11:15

Cryptography
- Symmetric crypto
- Asymmetric crypto
- Crypto hash functions
- Digital signature

Symmetric:



Chicken and egg problem:
The secret key has to be transferred (how?)

Notations:
S(C) = decrypt ciphertext C with secret key S

P(M) = encrypt plaintext M with public key P

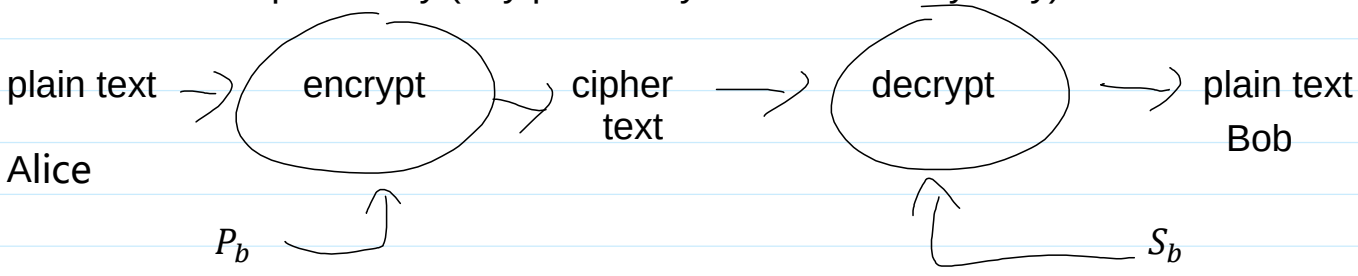
Alice: $P_a \& S_a$
Bob: $P_b \& S_b$

Asymmetric:

Every participant has two keys:
public & private

Alice: $P_a \& S_a$
Bob: $P_b \& S_b$

Network knows public key (any public key is known to anybody)

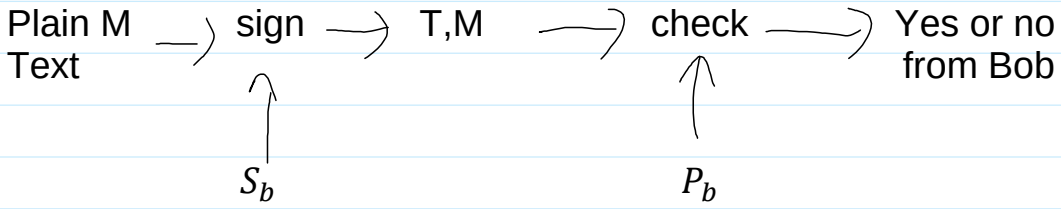


Digital signatures:

Use model:
They cryptographically guarantee that
- A specific person sends a message, and
- A message received comes from that

specific person

M = plaintext, T = signature
Property:
$[T = sign(M, S)] => [check(T, M, P) = 1]$



RSA cryptography
How public & secret keys are decided?

1. Select two large prim #s p & q (2048 bits or more)
2. Compute $n = p * q$
3. Compute $\varphi(n) = (p-1) * (q-1)$
4. Select small integer e which is relative prime to $\varphi(n)$
   $(gcd(\varphi(n), e) = 1)$
   Good $e = 2^{16} + 1$
5. Find d s.t.
   $d * e \equiv 1(mod\ \varphi(n))$ ($d * e$ is relative prime to $\varphi(n)$)
6. Public key $P = (e, n)$
   Secret key $S = (d, n)$

n is very hard to factor

In-Multiplicative inverse,
$d * e \equiv 1(mod\ \varphi(n))$ is equivalent to $d * e\ mod\ \varphi(n) = 1$

Encrypt M compute $M^e\ mod\ n = C$
Decrypt C compute $C^d\ mod\ n$
Why it works? => $M^{e*d} = M(mod\ n)$

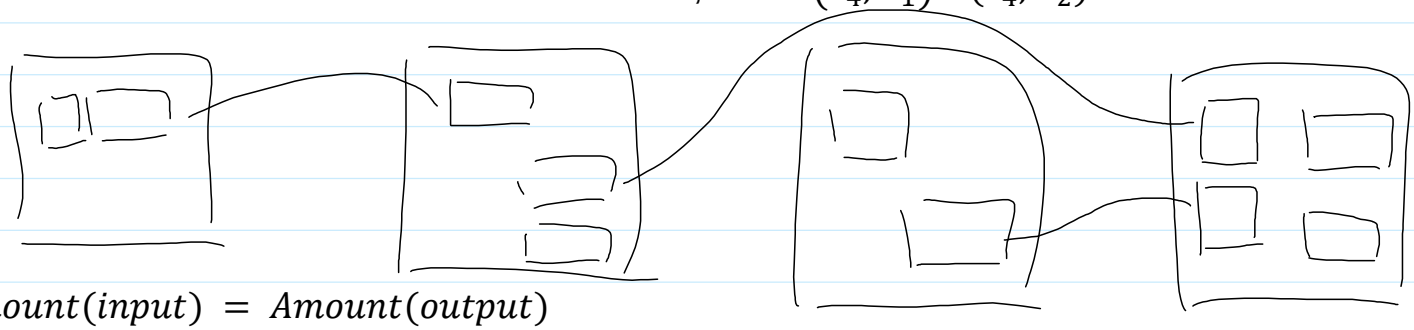
Cryptographic hash functions:
$h(M) \rightarrow M'$

But it's intractable to find h^{-1}(i.e given M' is intractable to find M)
=> one way functions,
non-reversible functions

BITCOIN
Satoshi Nakamoto (2009)
Transactions are put in blocks & they contain sender / recipient / amount sent

Let
$(T_i, O_j) = j^{th}\ output\ of\ i^{th}\ transaction$
Every transaction is an IOU

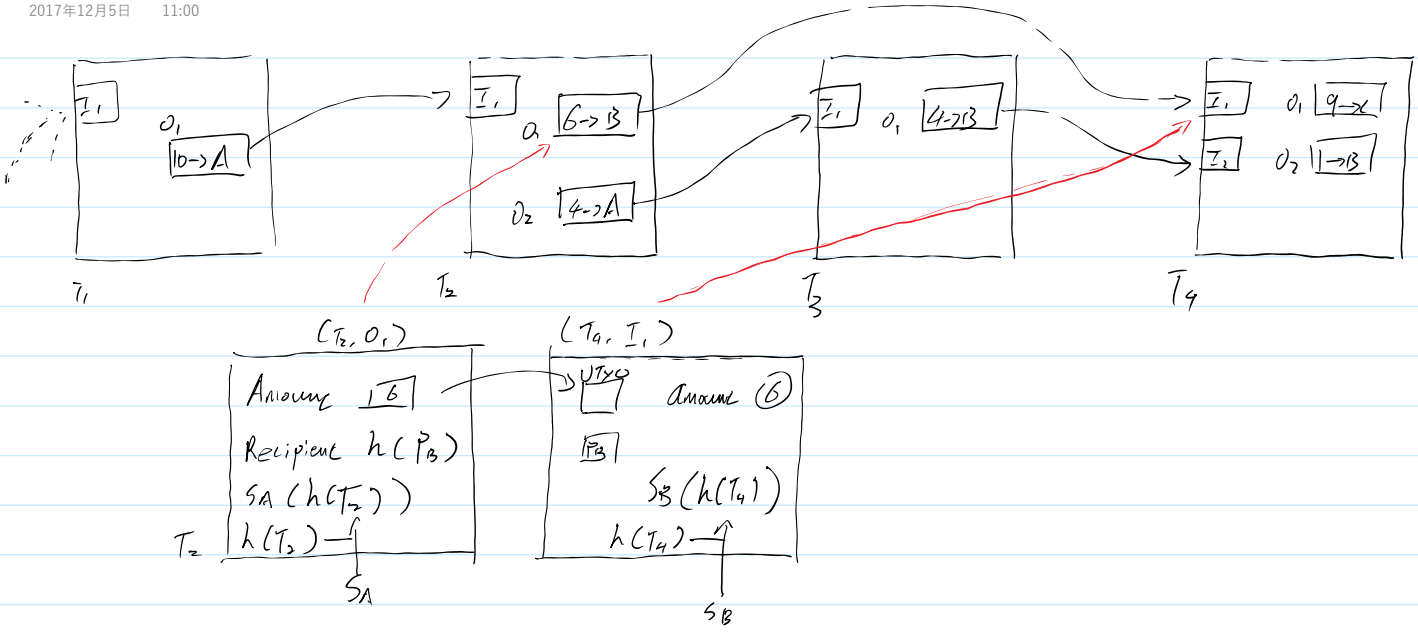
Alice received 10 B in (T_1, O_1)
Now Alice sends 6 B to Bob and 4 B to herself, call these $(T_2, O_1) and\ (T_2, O_2)$
Then she wants to send another 4 B to Bob, call these (T_3, O_1)
And Bob send 9 B to Charlie & 1 B to himself, call it (T_4, O_1)& (T_4, O_2)



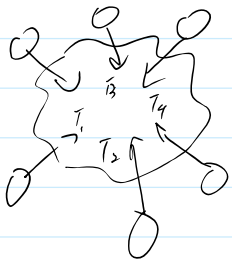
$Amount(input) = Amount(output)$
ALWAYS

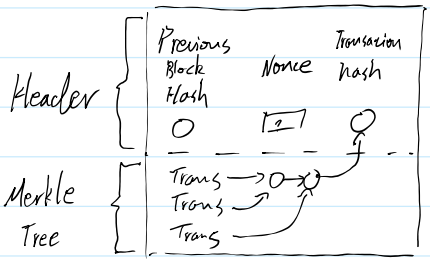
20171205

2017年12月5日　11:00

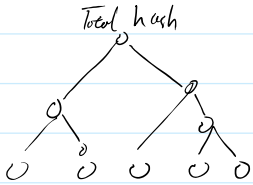


Bitcoin Block





Merkle Tree.



Mining: solve hard crypto puzzle

$h($ Previous Block$_t$ + Nounce + Trans. hash $) <$ specific # → difficulty target

- by "plugging random nounces"
- on avg 200 quantillion nounces tried before block is verified
- Every 14 days, DT gets re-adjusted

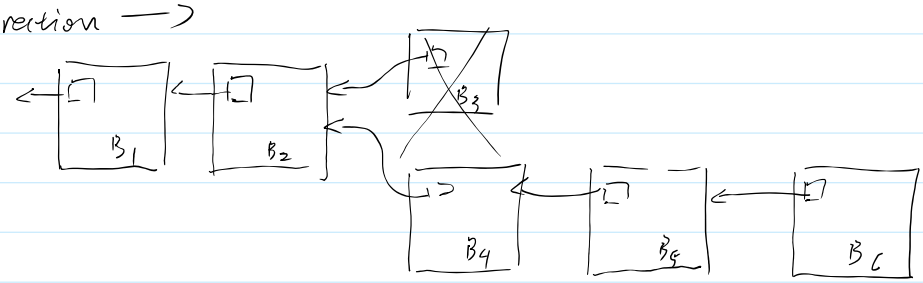
Proof of Work: originally from Hashcash

- solution found, → broadcast to Internet to verify

→ Incentive. Block & transaction fees

→ Proof of Stake

Direction →



Coin Trigger

shelling point

(Game Theory)

Scalability

# transactions per sec	Bitcoin	Ether	Visa	Paypal
	3-7	20	1000	200

Ethereum

Smart Contracts

Plasma