Fund Algor | Pro. Spencer.

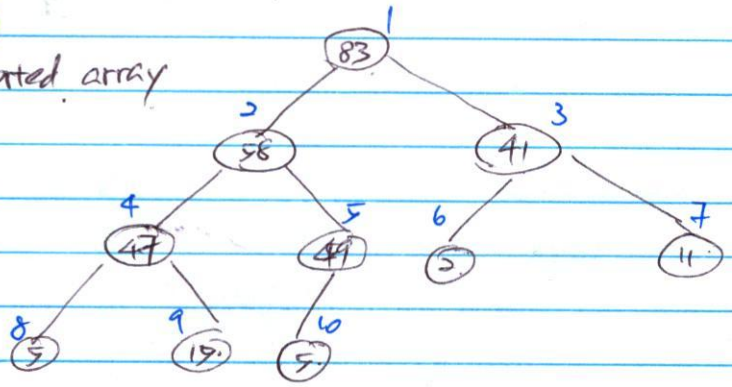
www.cs.nyu.edu/faculty/spencer/fundalg/index.html.

Office Hr Tuesday 1:30 p.m. 809

Chapter 6.

HEAPS.

semi-sorted array



1	2	3	4	5	6	7	8	9	10
83	58	41	47	49	5	11	5	15	5

Heap A.

Heap-size[A] size="n" Asymptotic.

Length[A] (length of array in the pc computer)

(In Binary-tree, left, right child cannot be the same in general)

Binary Tree Root[A] = 1 ← index

parent[I] = ⌊I/2⌋ (I ≠ 1) Root has no parent.

Left[I] = 2I (or NIL)

Right[I] = 2I + 1 (or NIL)

Max-heap property

For All I ≠ 1

A[parent[I]] ≥ A[I]

Min-heap property

For All I ≠ 1

A[parent[I]] ≤ A[I]

→ Binary tree rooted at I

I and All descendents

→ left tree of I.

Bin Tree. rooted at Left[I]

→ Right tree of I

Bin Tree rooted at Right[I]

LEAF: No children (I) > n/2.

Height[I] = length of longest path I to leaf.

Height[A] = Height[1] = ⌊lg n⌋ (lg means $\log_2$)

$N = 1023 = 2^{10}-1$ Height = 9

$N = 1048575 = 2^{20}-1$ " " = 19

$N = 2^{30}-1 \sim 10^9$ " " = 29

$N = 10^{12}$ " " = 39

MAX-HEAPFY (A, I) $O(\lg N)$

Build-max-heap (A) $O(N)$

HEAP-Sort $O(N \lg N)$

Priority Queue

- Insert $O(\lg N)$
- Extract-max $O(\lg N)$
- increase key $O(\lg N)$
- MAX $O(1)$

MAX-heapfy (A, I)

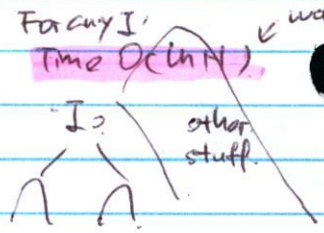
Input: left Tree of I heap
Right Tree of I heap

Output: Tree rooted at I: Heap

Time $O(\text{HEIGHT}(I))$

For any I: worst-case.

Time $O(\lg N)$



1-7 largest ← that of I, left[I], right[I]
(index) with largest A-value (not index, real value)

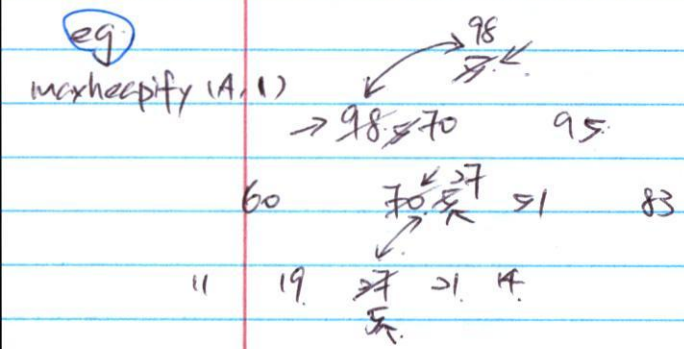
8 IF largest ≠ I (*Else STOP!*)

A[I] ⇔ A[largest]

MAX-HEAPFY (A, largest)

eg:

maxheapify (A, 1)



- why it works
- How long it takes

(partial sorting algorithm)

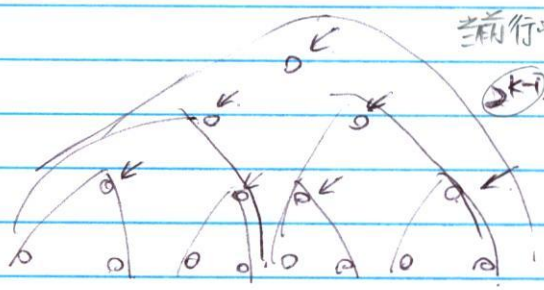
Build Max Heap (A)

input: Array A[1 ... N]

output: HEAP.

1. Heapsize[A] ← length[A]
2. For I = ⌊length[A]/2⌋ down to 1.
   max-heapify (A, I)

Time O(N)



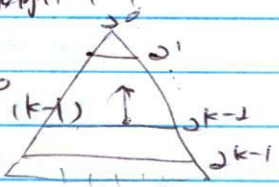
滿行的數 $H = 2^k - 1$

$2^{k-1} \cdot 0 \quad 2^{k-2}(1) + 2^{k-3}(2) + 2^0(k-1)$

$= 2^{k-1}\left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \cdots\right)$

$< 2^{k-1} \cdot 2 = 2^k = O(N)$

Height = k-1



$\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} \cdots = 1$

$\frac{1}{4} + \frac{1}{8} + \frac{1}{16} \cdots = \frac{1}{2}$

$\frac{1}{8} + \frac{1}{16} \cdots = \frac{1}{4}$

$\frac{1}{16} \cdots = \frac{1}{8}$

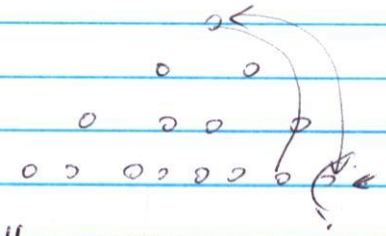
Formula: $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots \sum_{n=1}^{\infty} \left(\frac{1}{2}\right)^n = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = 1$

Heapsort [A]

Input: Array A[1 ... N]

Output: Sorted (array)

1. Build-max-Heap [A] O(N)
2. For I = length[A] down to 2.
   A[I] ↔ A[1]
   Heapsize[A] --
   Max-HEAPIFY (A, 1)



$\sum_{I=2}^{H} O(1 \lg I) = O(N \lg H)$

Priority Queue

insert into a MaxHeap

insert [A, x]

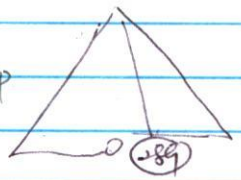
HEAPSIZE[A] ++ (I = Heapsize[A])

A[HEAPSIZE] = x

while I ≠ 1 AND key[parent[I]] < key[A[I]]

A[I] ↔ A[parent[I]]

I ↔ parent[I]



Time O(lnN)

Extract_MAX [A]

max ← A[1]

A[1] ⇔ A[HEAPSIZE[A]]

HEAPSIZE[A]--

Max_heapify (A, 1)

return max

O(lgH)

1/30.

Sorting Chapter 7&8.

Quicksort.

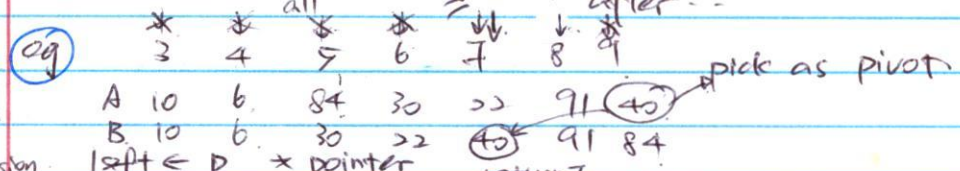
Partition [A, P, r]

input A[p ····· r] arbitrary order 隨意排列

output. pivot $x = A[r]$

all A[i] < x are before x

all - - - ≥ - - after - -



book method*

prof's version (Axillary B)

left ← p * pointer

right ← r * pointer.

x = A[r] * pivot *

For j = p to r-1

do if A[j] < x

B[left] ← A[j]

left++

else

B[right] = A[j]

right--

B[left] ← x

For j = p to r

A[j] ← B[j]

return left.

Time O(l) where

l = r - p + 1 (# items)

#/Take 3 random elements → then take median → pretty near the center.

Quicksort [A, p, r] (Assume $p \le r$)

If $p < r$ (* else done! *)

then $q \leftarrow$ Partition [A, p, r]

Quicksort [A, p, q-1]

Quicksort [A, q+1, r]

△ How much time does QuickSort take on n items?

worst case e.g. A already sorted $O(n^2)$

$F(n)$ = Time (# comparisons)

actually $F(n-1)$

If pivot always median $F(n) + 2F(\frac{n}{2})$

$F(n) = O(n \lg n)$

Randomized Partition [A, p, r]

$I \leftarrow$ Random [p, r] (in this range)

A[I] ↔ A[r]

return Partition [A, p, r]

Randomized QuickSort (A, p, r) (Expected) Avg Amount of Time.

If $p < r$.

$q \leftarrow$ Randomized Partition (A, p, r)

Randomized Quicksort (A, p, q-1)

Randomized QuickSort (A, q+1, r)

equally val.

Get a data and shuffle the data first. then → QuickSort.

(elements!)

Difficult point know

$T(n)$ = Expected # comparisons with n items.

$T(0) = T(1) = 0$

$T(n) = n-1 + \frac{1}{n}\sum_{i=1}^{n}[T(i-1) + T(n-i)]$ recursive equation.

↑ comp with pivot.

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

$T(2) = 2-1 + \frac{2}{2}(0) = 1$

$T(3) = 3-1 + \frac{2}{3}(T(1) + T(2)) = \frac{8}{3}$

$T(4) = 4-1 + \frac{2}{4}(T(1) + T(2) + T(3)) =$

Math →

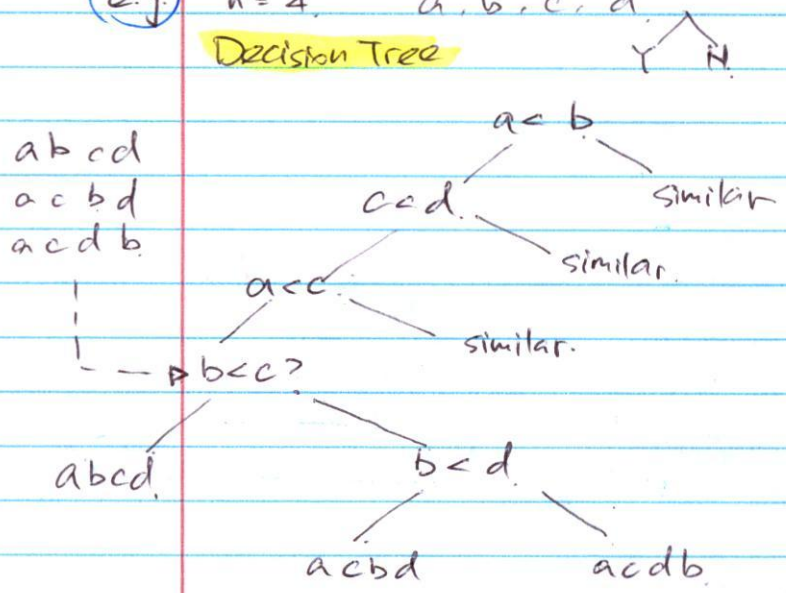
$T(n) = \Theta(n \lg n)$

Comparison Sorts

puzzle: Find order on n objects By comparison.

e.g. $n = 4$ a, b, c, d

Decision Tree



If m possibilites and h Questions

Decision Tree lower Bound $m \le 2^h$

$h \ge \log_2 m$

Sorting n elements $m = n!$

comparison $\ge \lceil \lg n! \rceil$

n	$n!$	DT LB
4	24	5
5	120	7

(? To P. 30 Q)

$\lg n! = \lg 2 + \cdots + \lg n \le n \lg n$.

$\ge \lg \frac{n}{2} + \cdots + \lg n \ge \frac{n}{2}(\lg n - 1)$

$\lg n! = \Theta(n \lg n)$

stirling formula $n! \sim n^n e^{-n} \sqrt{2\pi n}$

Linear Time Sorts

require special data

Counting Sort

only when data is in good form

Input: $A[1 \cdots n]$ All $0 \le A[i] \le k$ integers

If $k = O(n)$ Linear time Time $O(\max(n, k))$

Input: $A[1 \cdots n]$

output $B[1 \cdots n]$

Auxiliary $C[0 \cdots k]$

initialize. $C[i] = 0$ $0 \le i \le k$. (to count ~~of~~ occurance of value)

$O(n)$ For $i = 1$ to n $C[A[i]]++$ (*counting*)

$O(k)$ For $i = 1$ to k $C[i] = C[i] + C[i-1]$ (*cumulative counting*)

B. 0. 2 2 2

A. 1 3 0 2 3 1 3 2 0

eg C. 0 1 2 3

2 2 3 3

2 4 7 10

For J = N Down to 1.

Value ← A[J]

O(N) Place ← C[value]

B[place] ← value.

C[value] − −.

Radix Sort.

input A[1 ... N]

$0 \le A[J] < k^D$ integers written in base k.

D digits (count from left).

Tradeoff of k and D.

N numbers → base N.

eg All $0 \le A[J] \le N^3$?

Base N. Each count $O(\max(N, N)) = O(N)$

count = D = 3

For. J = D Down to 1.

Apply counting sort to. J^{th} Digit.

eg. 51 62 31 72 53

51 31 62 72 53

31 51 53 62 72.

Time O(D · Max(N, k))

3/6

Bucket Sort.

Buckets

0 1 ... 99

input Array A[1 ... N] unsorted.

Assumption $0 \le A[J] < 1$ "Random"

Output sorted Array.

Array B[0, ... (N-1)] of linkedlist

initially empty

For. I = 1 to N

O(N) Place = ⌊A[I] × N⌋

insert A[I] into. list B[Place]

For. J = 0 to N-1

sort B[J] By any sort.

Assuming no correlation ↓ independent.

(mean 1)

Poisson Distribution X

$\Pr[B[i] \text{ empty}] = (1-\frac{1}{n})^n \sim e^{-1} = 36\%$

$\Pr[B[i] \text{ has 1 item}] = n \cdot \frac{1}{n} \cdot (1-\frac{1}{n})^{n-1} \sim e^{-1} = 36\%$

$\Pr[B[i] \text{ has 2 items}] = \binom{n}{2} \frac{1}{n^2} (1-\frac{1}{n})^{n-2} \sim \frac{1}{2e} = 18\%$

" " 3 " " $\binom{n}{3} \frac{1}{n^3} (1-\frac{1}{n})^{n-3} \sim \frac{1}{6e} \sim 6\%$

$\Pr[X=k] = \frac{1}{ek!}$

$C_0^2 = 0$
$C_1^2 = 0$
$C_2^2 = 1$
$C_3^2 = 3$
$C_4^2 = 6$
$C_5^2 = 10$

Count comparison

Assume sort takes $\binom{k}{2}$ comparisons

Total # of comparasons:

$N[\frac{1}{e}(0) + \frac{1}{e}(0) + \frac{1}{2e}(1) + \frac{1}{6e}(3) + \frac{1}{24e}(6) + \frac{1}{120e}(10) + \frac{1}{720e}(15) + \cdots]$

$= CN$ (promised linear time)

Growth of Functions

$O \quad o \quad \Theta \quad \Omega \quad \sim \quad \omega$

Def: $T(n) = \Theta(g(n))$ means $\exists\, c_1 c_2 > 0$ such that $c_1 g(n) \le T(n) \le c_2 g(n)$ for n sufficiently large.

eg: $3n^3 - 5n^2 + 1576n = \Theta(n^3)$

$2n^3 \le \quad \| \quad \le 4n^3$

$F(n) \sim g(n)$ means $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$

$T(n) = O(g(n))$ means $\exists c$ $T(n) \le c\,g(n)$ for n sufficiently large

want $g(n)$ = "nice"

eg.

n	n^2
n^a	$\lg^\beta n$
$\lg n$	$n^a \lg^\beta n$
$n^{3/2}$	1 (constant, doesn't matter what's size of data.)
$n \lg n$	

$T(n) = \Omega(g(n))$ means $\exists\ c>0$ $T(n) \geq c\,g(n)$ for n suff large.

$f(n) = o(g(n))$ means

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Funny Addition

part I $T_1(n) = O(g(n))$

part II. $T_2(n) = O(g(n))$

Total $T_1(n) + T_2(n) = O(g(n))$

But Note! You can't set part into N.

★ $\lg n = o(n^\varepsilon)$ for any $\varepsilon > 0$ ★

eg: $n \lg n = o(n^{3/2})$

$\lg^k n = o(n^\varepsilon)$ for any $k, \varepsilon > 0$

Nested loops

BLOP. [N]

For I=1 to N.

Do BLIP. [I]

Suppose BLIP [I] takes $O(g(I))$

then BLOP[N] takes $\Theta(g(1) + \cdots g(N))$

```
X=0
For. I=1 to N
    For J=1 to I        } O(I)        } O(1+2+... N).
        X++
    End For
End For
```

Suppose g increasing

$g(1) + \cdots + g(N) \leq N\,g(N)$

$\geq g(\frac{N}{2}) + \cdots g(N) \geq \frac{N}{2}\,g(\frac{N}{2})$

eg: $1^3 + 2^3 + \cdots N^3 \leq N\,N^3$

$\geq \frac{N}{2}(\frac{N}{2})^3 = \frac{N^4}{16}$

$\therefore 1^3 + \cdots N^3 = \Theta(N^4)$ works for any power.

$2^1 + 2^2 + 2^3 + \cdots 2^N \leq N\,2^N$? (true but NOT actual)

$= 2^{N+1} - 2 = \Theta(2^N)$

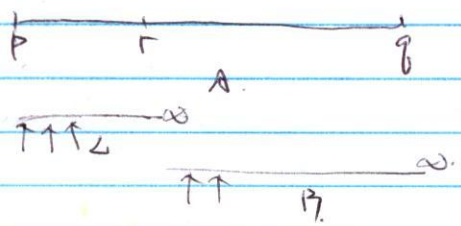
老师上课讲时混乱了 P R Q顺序

MergeSort (No assumption for the data)

Merge [A, p, q, r]

Input: A[p ... r] sorted
A[r+1 ... q] sorted

output: A[p ... q] sorted



① create Aux [1, ... r-p+2] with A[p ... r] & ∞;
R[1 ... q-r+1] with A[r+1 ... q] & ∞;

L point ← 1
R point ← 1

For k = p to q

If L[L point] ≤ R[R point]
A[k] ← L[L point]
L point ++

Else
A[k] ← R[R point]
R point ++

Linear
O(q-p+1)

Reference Book P.31.

MergeSort [A, p, r] (* p < r unsorted *)

If (p < r) (* else done! *)
q = ⌊(p+r)/2⌋
MergeSort [A, p, q]
MergeSort [A, q+1, r]
Merge [A, p, q, r]

Let T(n) = time (# comp) for mergesort on n ~~items~~ items

$T(n) = O(n) + T(\frac{n}{2}) + T(\frac{n}{2})$

$= 2T(\frac{n}{2}) + O(n)$

2/13.

Recursions

$T(n) = a\,T(n/b) + f(n)$

- a: # calls
- n/b: size
- $f(n)$: overhead

initial value $T(1)$ (Not Important!)

MergeSort $T(n) = 2T(n/2) + \Theta(n)$

→ 2 recursive call of half size.

→ Just Right: $T(n) = \Theta(n \lg n)$ ∵ $f(n) = n^{\log_2 2} = n$

In **No overhead** case

$T(n) = aT(n/b)$

$T(1) = 1$

$T(b) = a$

$T(b^2) = a^2$

$\vdots$

$T(b^i) = a^i$

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

$T(1) = 1$

$T(2) = 4 \cdot 1 = 4$

$T(4) = 16$

$\vdots$

$T(2^i) = 4^i \rightsquigarrow T(n) = n^2$

generalize

$n = b^i$

$a^i = \left[b^{(\log_b a)}\right]^i = (b^i)^{\log_b a} = n^{\log_b a}$

$\Rightarrow T(n) = n^{\log_b a}$ depending on the growth rate of overhead

Master Theorem { how do we get? / how to use?

Low overhead If $f(n)$ is "substantially lower"
Than $n^{\log_b a}$
Then $T(n) = \Theta(n^{\log_b a})$

High overhead If $f(n)$ is "substantially higher"
Than $f(n)$
Then $T(n) = \Theta(f(n))$

Just right Overhead If $f(n) = \Theta(n^{\log_b a})$
Then $T(n) = \Theta(n^{\log_b a} \lg n)$

$\log_2 8 = 3$

eg:

$T(n) = 8T(\frac{n}{2}) + n^{2.9}$ — $T(n) = \Theta(n^3)$ low

— $n^{3.5}$ — $\Theta(n^{3.5})$ high

— n^3 — $\Theta(n^3 \lg n)$ JustRight.

$\frac{T(n)}{n^{\log_b a}} = \frac{aT(n/b)}{n^{\log_b a}} + \frac{f(n)}{n^{\log_b a}}$ renormalize it.

$\frac{T(n)}{n^2} = \frac{4T(n/2)}{n^2} + \frac{f(n)}{n^2}$

set $S(n) = T(n)/n^{\log_b a}$

$S(n/b) = T(n/b)/(\frac{n}{b})^{\log_b a}$

$= aT(n/b)/n^{\log_b a}$

$S(n) = S(n/b) + g(n)$

where $g(n) = f(n)/n^{\log_b a}$

Set $S(n) = \frac{T(n)}{n^2}$.

$S(n) = S(n/2) + g(n)$

where $g(n) = f(n)/n^2$.

proof:

$S(b^l) = g(b^l) + g(b^{l-1}) + \cdots + g(b) + g(1) + S(1)$

⇒ HighOver $g(n)$, $g(n/b)$, $g(n/b^2)$ drops & sum done by 1st. term, costant to $\Theta(g(n))$

$g(n)$ pospow of n

⇒ LowerOver $g(1) + g(b) + g(b^2) \cdots + g(n)$ costant to $\Theta(1)$

$g(n)$ Negpow of n

⇒ JustRightOver All $g(1) + g(2) + \cdots g(n) = \Theta(\lg n)$

$g(n) = \Theta(1)$

eg:

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

$T(1) = 0$

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

$S(1) = \frac{T(1)}{1} = 0$

$n = 2^b$. $S(n) = (1 - \frac{1}{2^b}) + (1 - \frac{1}{2^{b-1}}) + \cdots (1 - \frac{1}{2}) + 0$

STRASSEN

$T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$

$\log_2 7 = 2.807$.

MasterThm $= T(n) = \Theta(n^{\log_2 7})$

KARATSUBA

Multiply 2 N-digit #s (eg in decimal)

#s given by A[0 ... N]
B[0 ... N]

Output. C[0 ... (2N)]

Note:
这里的数值是倒装的.
98 ⇒ A=[8,9]
+49. B=[9,4]
Then Addition Algorithm makes sense.

Addition CARRY ← 0
For i=0 to N
C[i] = A[i] + B[i] + CARRY
If C[i] < 10
do CARRY ← 0
else
CARRY ← 1
C[i] = C[i] − 10
END FOR
C[N+1] = CARRY

Normal Multi $\Theta(N^2)$. ← Note:
这里的数值是正装的.
24 ⇒ A=[2,4]
×11 B=[1,1]

Initial. All. C[i] ← 0
For I=0 to N
For J=0 to N
C[I+J] = C[I+J] + A[J] × B[I]
CARRY ← 0
For k=0 to 2N
C[k] = C[k] + CARRY
CARRY ← ⌊C[k]/10⌋
C[k] = Remainder[C[k], 10]

Divide & Conquer

binary

n even.

	x			y		
α	3	1	4	1	5	9
β	2	7	1	8	1	8
	z			w		

Input α, β write $\alpha = x2^{n/2} + y$

Output $\alpha\beta$ $\beta = z2^{n/2} + w$

$\alpha\beta = (xz)2^n + (xw + yz)2^{n/2} + yw$

Bad way: calculate xz, xw, yz, yw & overhead

$T(n) = 4T(n/2) + \theta(n)$

low Overhead $T(n) = \theta(n^2)$

		$\theta(n)$
xz	$T(n/2)$	$(xz)2^n + (yw)$
yw	$T(n/2)$	$(xw+yz)2^{n/2}$
$x+y$	$\theta(n)$	⇓
$z+w$	$\theta(n)$	$T(n) = 3T(\frac{n}{2}) + \theta(n)$
$(x+y)(z+w)$	$T(\frac{n}{2})$	(a)low overhead
$(x+y)(z+w) - xz - yw$	$\theta(n)$	$T(n) = \theta(n^{\log_2 3})$
$\|\|$ $xw+yz$		

2/23 Recitation point.

$T(n) = 2T(n/2) + \theta(n\log n)$

⟹ If use Master Theorem. $\log_2 2 = 1 \Rightarrow n$ compare with $n\log n$.

It will be high Overhead $\boxed{T(n) = \theta(n\log n)}$

⟹ But in fact, in reality, is that true ??

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

let $S(n) = \frac{T(n)}{n}$

$S(n) = S(n/2) + \log n$

$= S(n/4) + \log(n/2) + \log n$

$\vdots$

$\lg n = S(1) + \underbrace{\log(1) + \cdots \log(n/2) + \lg n}_{\theta(\lg n \cdot \lg n)}$

$\because \lg(n/2) = \lg n - 1$

$\lg(n/4) = \lg n - 2$

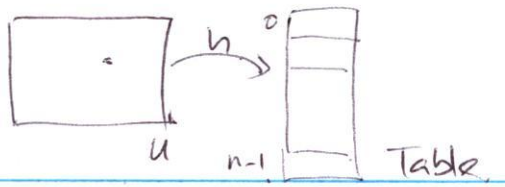
$\vdots$

$\lg(1) = \lg n - \lg n$

$\therefore S(n) = \lg^2 n + S(1) \Rightarrow T(n) = n(\lg^2 n + S(1))$

3/27

Hashing



Universe U of keys.
Dynamic ~~DATA~~ structure
Function Supported Insert
Delete.
Search

Hash Table Size m.
Hash Function h:
$U \to \{0, 1, \ldots, n-1\}$

IDEA.

Insert John into T[h[key(JOHN)]]
Search John in T[h(key(JOHN))]
Delete John from T[h(key(JOHN))]

Problem.. Collision..

h[key(John)] = h[key(Mary)]

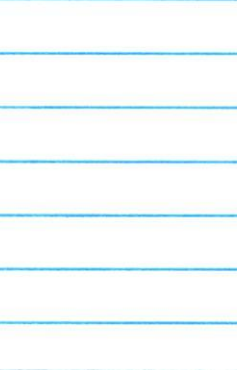
Collision Resolution by chaining

T ARRAY of Doubly linked lists
Insert 'John' at the head of T[h(key(JOHN))]
Search ~ search list T[h(key(JOHN))]
Delete ~ search - remove 'John' functionally linked list

m = size HASHTABLE
n = # records
Load Factor $\alpha = \frac{n}{m}$
Successful $1 + \frac{\alpha}{2}$
unsuccessful $1 + \alpha$

Suppose keys are nonnegative integers
e.g. $h(x) = x \bmod m$.

e.g. $m = 9$
Insert 37. $37 \% 9 = 3$
Insert $591 \% 9 = 6$
Search $200 \% 9 = 2$ No
Insert $813 \% 9 = 3$
Search 37>
Delete 813



Properties of good hash Function

1. Quick to compute.
2. Acts like Random Function } Pseudorandom
3. Deterministic

Probe Sequence

h(key, 0) h(key, 1), ...,

Insert: Try h(key, 0) if full h(key, 1) ...

Linear Probing $h(key, i) = h'(key) + i \bmod m$

Problem: Primary Clustering

Double hashing

m Prime

Two Hash Funs $h_1(k) \in \{0, 1, \ldots m-1\}$

$h_2(k) \in \{1, \ldots n-1\}$

$h(k, i) = h_1(k) + i h_2(k) \bmod m$

eg $m = 11$ $h_1(k) = 5$ $h_2(k) = 3$

probe sequence 5, 8, 0, 3, 6, 9, 1, 4, 7, 10, 2

Deletes: I [dirty]

Add "Delete" to spot on hashtable

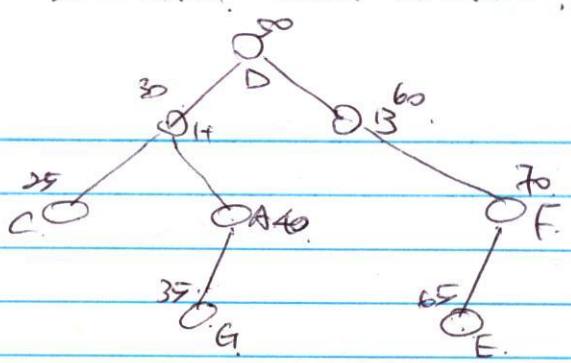
$h_1(Tom) = 4$

$h_2(Tom) = 3$

	0	Insert 'John'
	1	Insert 'Mary'.
	2	" 'Tom'
	3	
	4 John	Delete 'Mary'
	5	search 'Tom'.
	6	
	7 ~~Mary~~ Delete	either found or a blank spot.
	8.	
	9	
	10. Tom.	

Think about unsuccessful searching time.

Semisorted Data Structure.



Binary Search Tree

N nodes

H height

Root. Parent, Left, Right

Root = D

	A	B	C	D	E	F	G	H
parent	H			D		H		
left	G	NIL		NIL		NIL		
right	NIL	F		NIL				
key	40	60		25				

BST Property (Very important!)

Any Node x (Definition of BST)

Any Node y in Left Tree has key(y) ≤ key(x)

— — z — Right — — key(z) ≥ key(x)

Dynamic Data Structure [Assume all keys are distinct]

Search $O(H)$

Insert $O(H)$

Delete. $O(H)$

3/7. Midterm date confirmed as stated on Syllabus. Office Hour: Tmr. 2:30 – 3:30

Assignments + 参 Solutions. (看老师的思路和写法)

BST

N: Nodes H: Height

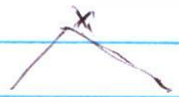
worst case $H = N-1$

Usually: $H = \Theta(\lg N)$ unlike heap, it can be long and stringing can have left child/right child or not.

TreeSearch [x, k]

↑ root.

Time $O(H)$



① If. k = x.key
return x.

② else if x = NIL
return NIL

③ Else if k < x.key [left[x]
return TreeSearch [x.left, k]

④ Else (* k > x.key *)
return TreeSearch [x.right, k]

* BST is a Good dynamic data structure.

Full-Tree-Search [T, k] 完整的 code 要抄這一段

x ← Root[T]

Tree Search [x, k]

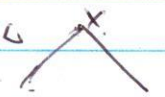
TREE-MIN [x]

while left[x] ≠ NIL

x ← left[x]

end while

return x;



TREE-MAX [x] similar.

while right [x] ≠ NIL

x ← right [x]

end while

return x;

If you are in inorder, stick in new stuff is a disaster!

In-Order TREE-Walk (x)

If x ≠ NIL (* else done! *)

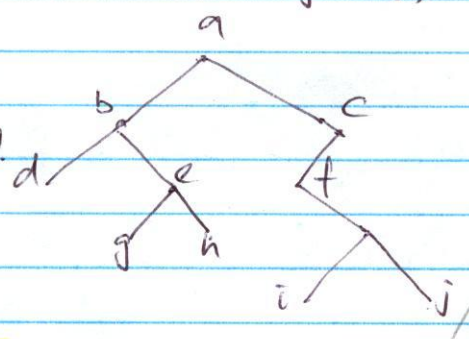
In-Order-tree-walk (left[x])

print x

In-Order tree-walk (right[x])

Time = O(N).

each Node gets hit 3 times!



Insert [z]

Time = O(H)

x ← Root[T]

p ← NIL

while x ≠ NIL

if z.key < x.key

p ← x

x ← left[x]

else p ← x

x ← right[x]

end while

parent[z] ← p

left[z] ← NIL

right[z] ← NIL

If z.key < p.key

z ← left[p]

else

z ← right[p]

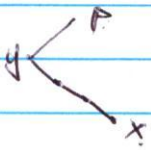
Successor [x] $O(h)$

If. Right [x] ≠ NIL.

return MIN [Right [x]]

else.

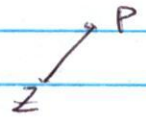
Climb from x until Reach y is the left child of P.



Deletion.

Delete [z]

Case 1. z childless



1A: No Parent.

Root ← NIL

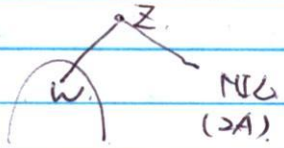
Else: If z = Right [P]

NIL ← Right [P]

else NIL ← left [P]

Case 2. z has 1 child

(say left child)



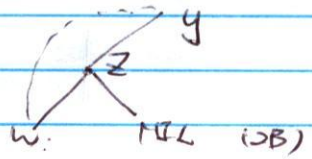
w ← left [z]

2A: z is Root.

Root [T] ← w

Parent [w] ← NIL

2B: Left [y] ← w.



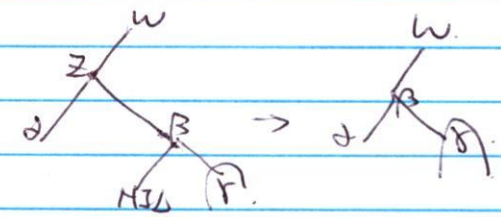
parent [w] ← y

2C: right [y] ← w

parent [w] ← y

Case 3. z has 2 children.

w ← parent [z]



3A: Left [β] = NIL

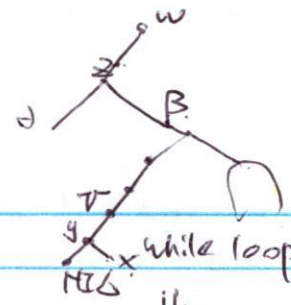
left [w] = β.

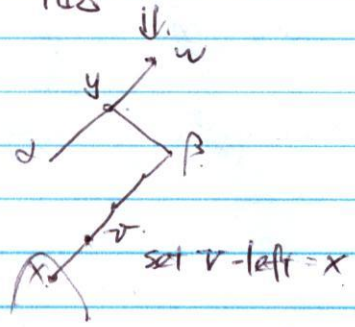
parent [β] = w.

left [β] = α.

parent [α] = β

SB: left[I[β]] ≠ NIL





Time: O(H)

(Finding y from β)

Dynamic Programming

15-1 optimization ROD-CUTTING

Total length N.

Input: Price P[I] For I UNIT ROD N $1 \le I \le N$

partition p(N)

$p(N) \sim \frac{1}{4\sqrt{3}} e^{\pi\sqrt{2n/3}}$

greater than poly time.

Want to maximize total reference.

Output: Revenue R[I] $1 \le I \le N$ FirstCut[I], $1 \le I \le N$.

set R[N] = MAX REV.

Initial: R[0] = 0 Time $O(N^2)$

R[I] = P[I], FirstCut[I] = I.

Recursive solution (*) $\Rightarrow$ $R[N] = \max_{1 \le I \le N} (P[I] + R[N-I])$ $O(2^n)$

3/20 FirstCut[N] = that I giving MAX

To make cuts of Rod of length N

Done ← 0

until Done = N

Done ← Done + FirstCut[N − Done]

"cut" at Done.

This is linear time.

Top-down & bottom up memoized cut-rod → $O(N^2)$

19.4 [Longest Common Sequence]

Input: $x[1 \cdots M]$ $(x = x_1 x_2 \cdots x_n)$

$y[1 \cdots N]$ $(y = y_1 y_2 \cdots y_n)$

$x_i = x_1 x_2 \cdots x_i$ $\quad c[i,j]$ = length of # LCS of $x_i y_j$

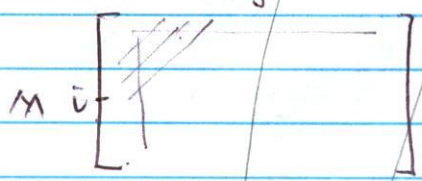
$y_j = y_1 y_2 \cdots y_j$

Thus: If $x_i = y_j$ then A LCS of $x_i y_j$ uses x_i, y_j.

and $c[i,j] = c[i-1, j-1] + 1$

(*) If $x_i \neq y_j$ $\quad x$ use T $\quad x$ use A

$c[i,j] = \max_{N}^{M} [c[i-1, j], c[i, j-1]]$



Initialization

$c[I, 0] = 0 \quad 1 \le I \le M$

$c[0, J] = 0 \quad 1 \le J \le N$

For $k = 2$ to $M+N$

For $I = MAX(1, k-M)$ to N

$J = k - I$

$c[I,J] = (*)$

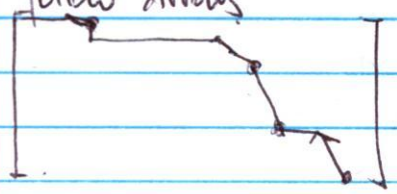
For $I = 1$ to M

For $J = 1$ to N

$c(I,J) = (*)$

Time $\Theta(M \cdot N)$

$b[I,J] =$ follow Arrows



Parenthesization.

$A = 10 \times 100$ $\quad B = 100 \times 5$ $\quad C = 5 \times 50$

$(AB)C$ $\quad 10 \times 5 \times 50 = 2500$

$A(BC)$ $\quad 10 \times 100 \times 50 = 50000$

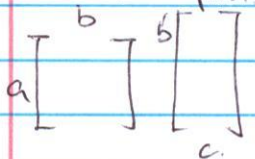
2500

Matrix Chain Mult.

Input: $p_0, p_1, \cdots p_n$ positive integers

want to find $A_1 \cdots A_n$ where A_i has size $p_{i-1} \times p_i$

Assume cost = abc



(AB)(CD) P(n) = # ways to paren. $A_1 \cdots A_n$

A I(BC)DJ Cut $A_1 \cdots A_n = (A_1 \cdots A_I)(A_{I+1} \cdots A_n)$

A[B(CD)] If Cut is btw $I, I+1$ $P[I] * P[n-I]$

I(AB)CJD Recursion $P(n) = \sum_{i=1}^{n-1} P(i) P(n-i)$

[A(BC)]D $\rightarrow P(n) = \frac{1}{2n-1}\binom{2n-1}{n-1} \sim c 4^n n^{-3/2}$

Input $P_0, P_1, \cdots P_n$.

let $m(i,j)$ = min cost to find $A_i \cdots A_j$ $(i \le j)$

initial $m(i,j) = 0$

$m(i, i+1) = P_{i-1} P_i P_{i+1}$

$A_i \cdots A_k$ $P_{i-1} * P_k$

$A_{k+1} \cdots A_j$ $P_k * P_j$

$$m(i,j) = \min_{i \le k < j} \left[m(i,k) + m(k+1,j) + P_{i-1} P_k P_j \right]$$

Triple loop Time $O(N^3)$.

```
For L=3 to N   (* length of interval *)
  For I=1 to N-L+1
      J = I+L-1   (* intervals with L terms)
    m(I,J) = (*)
```

Mid-term 6 problems.

write program + comment to explain.

Sorting Algorithm (all) understand Time + reason.

Give the analysis.

Recursion

Optimal BST.

1, 2, ... N

Input frequency

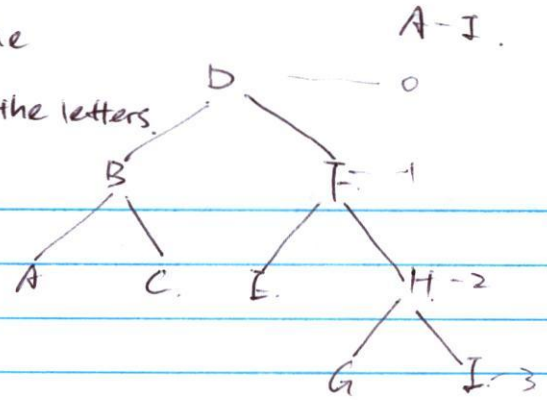
P[I], $1 \le I \le N$

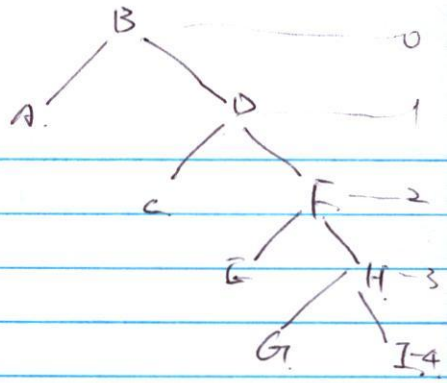
want BST that minimizes $\sum_{I=1}^{N} d(I) p(I)$.

depth of Tree.

4/3

Depend on the frequency of the letters





OPT BST

keys 1 ... , N.

given frequencies: p[1], ... , p[N]

want to minimize the sum over all keys

$$\text{Cost} = \sum_{I=1}^{N} p[I]\, d[I]$$

$d[I]$ ↑ depth in BST

Idea: let C[I, J] be the minimal cost for a BST on keys I ... J

Initialize C[I, I] = 0

C[I, I-1] = 0 (*useful*)

(*) $C[I,J] = \min_{I \le k \le J} \left(C[I,k-1] + C[k+1,J] + \sum_{s=I}^{J} p(s) - p(k) \right)$

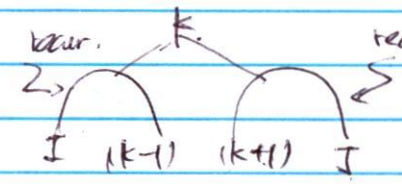
Each k in time O(1)

for all k ≤ J O(N)

O(N) pre process

O(N) Initialize -

$O(N^3)$ use (*)



$\sum_{s=I}^{J} p(s) - p(k)$ = cum[J] − cum[I−1] − P[k]

pre process

cum[0] = 0.

cum[I] = P[I]

For S = 2 to N

cum[S] = cum[S−1] + P[I]

eg For D = 2 to N − 1

For I = 1 to N − D

J = I + D

C[I, J] = (*)

Greedy Algorithm

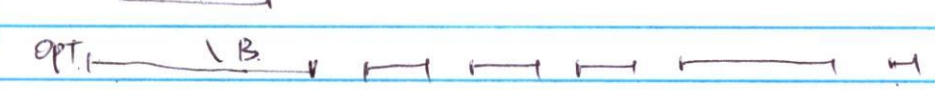
Activity selector.

input: $A_I = [S[I], F[I]]$ $1 \le I \le N$

problem: Optimize the number of activities.

Theorem: Some Optimal Solution uses the activity that finishes first.

First finish A.



replace B by A.

∴ Might as well as A.

1) Order by FIJ $O(n \lg n)$

$O(n)$
- (a) select $a_1 = (s_1, f_1)$
- b) TempF $\leftarrow f_1$
- c) For $I = 2$ To n
  - If $S_I >$ TEMPF. (*else skip a_I)
    - select a_I
    - TempF $\leftarrow f_I$

✓ Huffman decoding

Given: Alphabet Ω (eg: {a -- z}). (order doesn't matter)

Frequencies $f(x) > 0 \quad \forall x \in \Omega$.

$\sum_{x \in \Omega} f(x) = 1$

			$\zeta(x)$
0.55	a	0	0
0.2	b	01	110
0.24	c	10	10
0.01	d	11	111
		2	(x0.55 3x0.2 2x0.24 3x0.01 =1.66

want each letter x to code word. (0-1 string)

cost $= B[T] = \sum_{x \in \Omega} f(x)\, d_T(x)$ — depth of x in Tree = length of codeword

prefix code: No codeword is the prefix of another codeword.

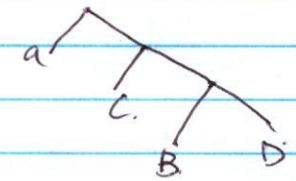
$\Rightarrow$ Unambiguous parsing

eg: 111|10|11|00|11|01|10|

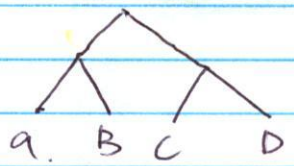
D C B A B B

We want prefix code with minimum cost.

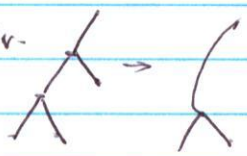
(*)



(†)



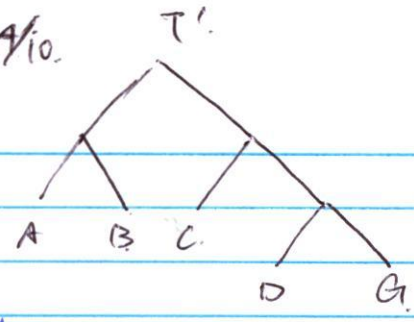
keypoints ① Mover.

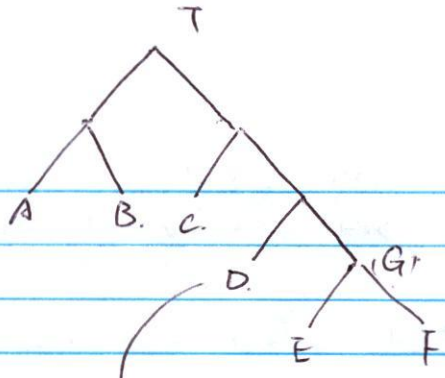


② given shape of Tree. place x Bottom up.

③ Any shape., the two least frequent letter appear on bottom level. We concord will assure they're siblings

4/10.





A .45
B .13
C .12
D .16
E .09 > G .14
F .05

✓ Optimal BST

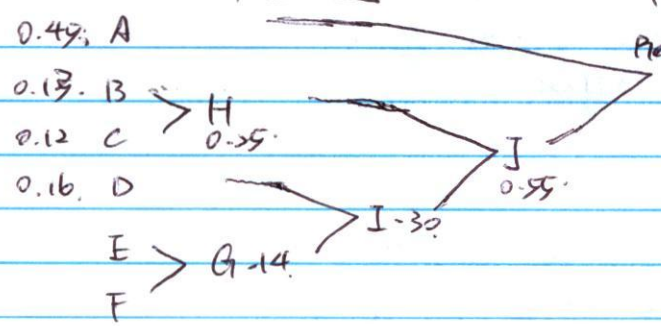
p[1] ... p[M] frequencies

Minimize $\sum_{I=1}^{M} p[I] \cdot d[I]$

$0.45(2) + 0.13(2) + 0.12 \cdot 2 + 0.16 \cdot 3 + 0.09 \cdot 4 + 0.05 \cdot 4 + 0.14 \cdot 3$

$\Delta = 0.09 + 0.05 = 0.14$

THM: We may assume two letters of smallest frequency are siblings.



Result: 0 / 1

A 0
B 100
C 101
D 111
E 1110
F 1111

Min Heap Implementation ($|Q| = n$)

① Build Min heap Q (for Q under p). $O(n)$

part II / $n-1$ Times

when $|Q| = i$

$O(\lg i)$ α = Extract-Min[Q]

$O(\lg i)$ β = Extract-Min[Q]

$O(1)$ { Allocate new γ.
left[γ] ← α; right[γ] = β; parent[α] ← γ; parent[β] ← γ
$p(\gamma) \leftarrow p(\alpha) + p(\beta)$.

$O(\lg i)$ insert [Q, γ]

Time when $|Q| = i$ is $O(\lg i)$

part II time is $O(\sum_{i=2}^{n} \ln i) = O(n \lg n)$

∴ Total time $O(n) + O(n \lg n) = O(n \lg n)$

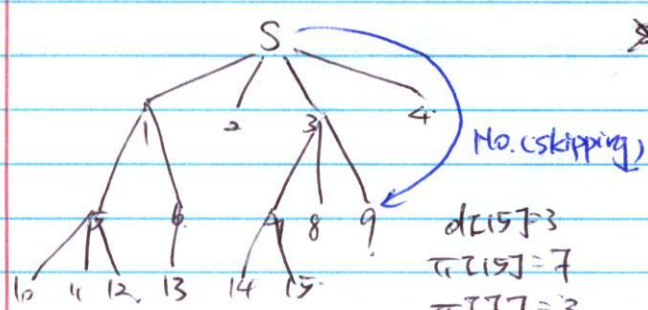
V = set of nodes
E = set of (directed) ~~edg~~ adjacency

Adjacency list representation
for each Node v, Adj[v] is linked list of neighbors

✓ BreathFirst Search BFS [G, s]
Find All v readable from s ↑ source. (Given)
d[v] = distance [s, v] (shortest distance from s to v
BFS Tree $\pi[v]$ = parent of v
(Root s).

colors white grey black
untouched in progress processed
Initial: All white except s grey
Queue Q initially Q = (s)
while $Q \neq \phi$
$u \leftarrow$ dequeue [Q]
for $v \in Adj[u]$
if color[v] = white
color[v] $\leftarrow$ grey
enqueue [v]
$\pi[v] \leftarrow u$
$d[v] = d[u] + 1$
end for (* v *)
color [~~u~~] $\leftarrow$ Black

Vertices
Edge
Time O(V+E)



~~x~~ 1
2 ~~x~~
3 ~~x~~
4 4
5
6
7
8
9

d[15] = 3
$\pi[15] = 7$
$\pi[7] = 3$
$\pi[3] = s$

Huffman code - BST - order matters

prefixcode - BST - order does not matter.

Depth First Search DFS[G]

white grey black

Initially all white

Given. Ordering of V.

"Outside"

DFS[G]
time = 0
for all $v \in V$.
  if color[v] = white
    → Do DFS-visit[G, v]

DFS-visit[u]

color[u] ← grey

time ++

d[u] ← time.

for each $v \in Adj[u]$

  if color[v] = white (*else ignore*)

    $\pi[v] \leftarrow u$.

    → DFS-visit[v]

color[u] ← black

time ++

f[u] ← time.

d[u] = discovery time u becomes grey

f[u] = final time u becomes black.

Time $\Theta(V+E)$

parentheses THM

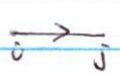
let $v, w \in V$ with $d[v] < d[w]$

w not found in DFS-visit[v]

	(	)	(	)	
either	d[v]	f[v]	d[w]	f[w]	Reach DFS-visit[v]
	(	(	)	)	v grey, w white
or	d[v]	d[w]	f[w]	f[v]	w found in DFS-visit[v]
	(	(	)	)	
But NOT!	d[v]	d[w]	f[v]	f[w]	

Dag - directed Acyclic Graph

Topsort



means must do i before j

input: Dag

output: Ordering. $v_1\ v_2\ v_3\ v_4\ \dots\ v_n$.

so that NO. $v_i \leftarrow v_j$

Topsort. [G]

1) call DFS [G] Time $O(V+E)$

2) place vertices in reverse order of finish time

$f(v_1) > f(v_2) > \dots > f(v_n)$.

Time $O(V)$ when v finishes: place at front of list.

Q: Why does Top-Sort work?

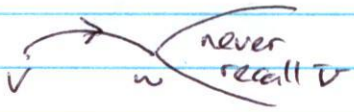
Need: if  then $f(v) > f(w)$

proof: Case I: $d[v] < d[w]$ During DFS-visit.[v] find w.

DFS-visit.[w] subroutine of DFS-visit [v]

$f[w] < f[v]$

Case II $d[w] < d[v]$.



DFS-visit [w] misses v

$f[w] < d[v] < f[v]$

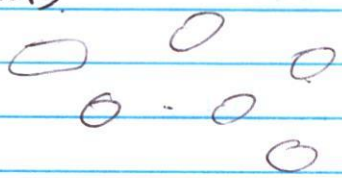
$\therefore f[w] < f[v]$

Strongly connected components

directgraphs.  $v \rightsquigarrow w$

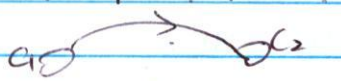
$\exists$ dir path v to w

Def. $v \equiv w$ if $v \rightsquigarrow w$ AND $w \rightsquigarrow v$

clusters 

super graph

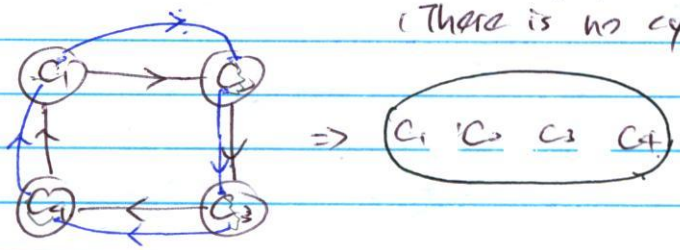
cluster → points

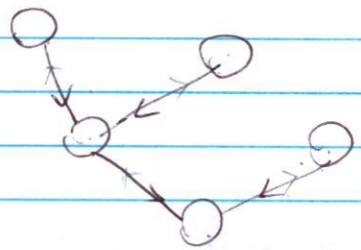


directly graph $C_1 \to C_2$ if

cluster graph is DAG

(There is no cycles!)

proof. 



Input: Directed Graph G

Output: One cluster

(Book: All clusters)

v finished last.

claim: v must be in a top cluster. (nothing above it)

① Apply DFS to G

let v be the vertex that finishes last

v must be lie in a top cluster

Lemma: if $w \rightsquigarrow v$. But not $v \rightsquigarrow w$. then $f(w) > f(v)$

PF. Case 1: $d(v) < d(w)$

in DFS-visit(v), w not discovered

so $d(v) < f(v) < d(w) < f(w)$

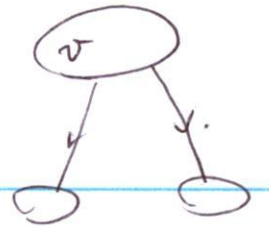
Case 2: $d(w) < d(v)$

there is a path in $w \to v$. thus v is discovered then

During DFS-visit(w). As $w \rightsquigarrow v$ v is discover—

∴ DFS-visit(v) is subroutine of DFS-visit(w)

∴ $f(v) < f(w)$.

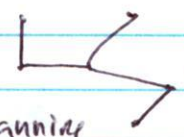


② let G^T be G with arrow reversed. $\Theta(V+E)$

③ Apply DFS-visit(v) to G^T $\Theta(V+E)$

④ Output. all points that you can reach that turn grey/black.
reach all w in the cluster.
AND nothing else as v in Top cluster.

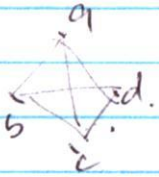
- **Minimal Spanning Tree Problem. (MST)**
undirected G. V vertices, E edges
each edge $e = \{v, w\}$ has weight w(e)
assume G connected
Spinning Tree T on V-1 Edges
if V-1 Edges & No cycle → spanning
—— & spanning → No cycle



Defin. $w(T) = \sum_{e \in T} w(e)$

✓ **MST**: Find T with minimal w(T)

V^{V-2}

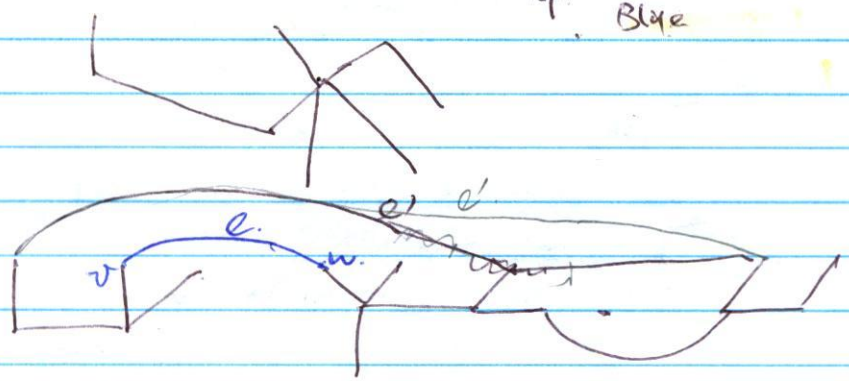


- $T + e - e' = T'$ tree. $w(T') = w(T) + w(e) - w(e')$

Thm: The edge of minimal weight is in MST

✓ Kruskal/Prim

Modified MST
T. Blue



Claim: Modified MST must include e.
with minimal w(e) amongst potential edges
between two components
If not add e delete e'

✓ higher_level Kruskal

① Order edges by weight. $w(e_1) < w(e_2) \ldots$

$\Theta(E \ln E) = \Theta(E \ln V)$ $E \le V^2$

② For $I = 1$ to E

ADD $e_I := \{x_I, y_I\}$ iff No cycle created

Union-Find $(\emptyset)$

Data Sequence $e_i = \{x_i, y_i\}$ $1 \le i \le E$

initialize $T \leftarrow \emptyset$

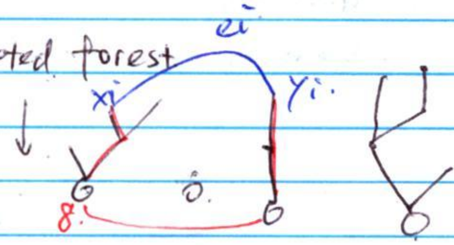
each I Accept iff x_i, y_i in different component of T

if accept $T \leftarrow T \cup \{x_i, y_i\}$

union-find: Dynamically keep rooted forest

$\pi(v)$ = parent of v

$\pi(v) = v$, v is root.



want: size[v] = size of tree if v is root

↳ : Components of forest of accepted edges = components in the rooted forest

Union-Find

Initialization All $\pi(x) = x$ size$(x) = 1$

for $I = 1$ to E $e_I = [x_I, y_I]$

$x \leftarrow x[I]$ $y \leftarrow y[I]$

$O(E \ln V)$ while $x \ne \pi(x)$

$O(\ln V)$ $x \leftarrow \pi(x)$ Down to roots

while $y \ne \pi(y)$

$y \leftarrow \pi(y)$

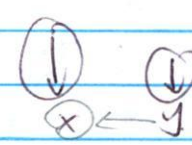
(indent to align with while)

If $x \ne y$ (*else do nothing *)

reset size(x) ≥ size(y)

$O(1)$ $\pi(y) \leftarrow x$

size(x) ← size(x) + size(y)



size[v] = size of components of v when v stops being a root.

接前面的 claim

Claim: if $\pi(v) = w$ Then size[w] ≥ 2 size[v]

At moment that set $\pi[v] = w$.

before size[v] $\leq$ size[w]

size[w] = size[w] + size[v]

New size[w] ≥ 2 size[v]

later size[v] stays same size[w] doesn't go down (still hold)

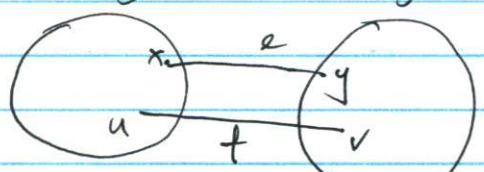
If $a \leq b$ then $a+b \geq 2a$

★ size[$\pi^i(x)$] $\geq 2^i$ size[x]

corollary Bannister length $\leq \lg V$

4/24 Prim MST S^c S complement.

Thm: Any split $V = S \cup \bar{S}$, let $e = \{x, y\}$ be the edge between S and $\bar{S}$ of minimal weight. Then e is in the MST.



pf: Take MST without e

Take path x to y in MST

That path has some "crossing" edge f. Replace f by e →←

指定 Designate Source {w}

High Level Prim

$S \leftarrow \{w\}$.

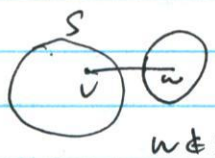
$V-1$ times.

Find. $e = \{x, y\}$, $x \in S$. $y \notin S$.

of. MIN WEIGHT

ADD e to MST

$S \leftarrow S \cup \{y\}$



Data Structure

MinHeap Q.

$w \in S'$ $\pi[w]$ that $\in S$ with cost $[v, w]$ minimal.

key[w] = cost $[v, w]$

initialization

for each $z \in Adj[w]$ (places can go directly)

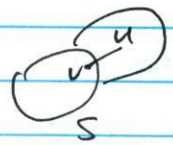
$\pi[z] \leftarrow w$

$key[z] \leftarrow cost[w, z]$

build MIN-HEAP Q

T — $V-1$ times

$O(\ln V)$ → $u \leftarrow$ Extract-MIN $[Q]$



Add edge $\{u, \pi(u)\}$

Add u to S

update

$O(E \ln V)$ — for $z \in Adj[u]$ (This occurs $2E$ times)

if $z \in S$ forget about it

if $z \notin S$

if $w(z, u) < key(z)$

$\pi[z] \leftarrow u$

$O(\ln V)$ → $key[z] \leftarrow cost(z, u)$

L

Number Theory: Numbers N with n digits

primality

2002: Agarwal / Kayal / Saxena — Polynomial Alg. determines a number is a prime. (Black box polynomial Algo)

Chapter 31. 8.9 X.

31.1 skim **Euclid's Algorithm**

input: a, b

output: $\gcd(a, b)$

Division: given $b \neq 0, a$

(*) FIND q, r $a = qb + r$ AND $|r| < b$

eg: $a = 37$ $b = 10$

$37 = 3 \times 10 + 7$

$= 4 \times 10 - 3$

THM: $\gcd(a, b) = \gcd(b, r)$ [Default = $\gcd(b, 0) = b$]

$d \mid a$ & $d \mid b \Rightarrow d \mid r = a - qb$

$a \mid b$ & $d \mid r \Rightarrow d \mid a = qb + r$.

EUCLID (a, b). (*a > b*)

If b = 0 return a

else Find q, r (*)

return EUCLID (b, r).

Euclid (100, 41) $100 = 2(41) + 18$

- - - (41, 18) $41 = 2(18) + 5$

- - - (18, 5) $18 = 3(5) + 3$

- - - (5, 3) $5 = 3 + 2$

- - - (3, 2) $3 = 2 + 1$ $1 = 3 - 2$

- - - (2, 1) $2 = 2 \cdot 1 + 0$

- - - (1, 0) = 1

when a, b have $\leq n$ digits.

Time. $O(n^2)$

Extended Euclidean Algo

THM: $\exists x, y \quad ax + by = \gcd(a, b)$

EXTEUC (a, b) returns (d, x, y)

$O(n^2)$ $d = \gcd(a, b)$ $ax + by = d$

If b = 0 return (a, 1, 0)

else find q, r (*),

(d, x', y') ← EXTEUC (b, r)

return (d, y', x' − qy')

$a = bq + r$

$d = bx' + ry'$

$= bx' + (a - bq)y'$

$= ay' + b(x' - qy')$

Z_n elements 0, 1, ... n−1

a has a multi inverse a^{-1}

iff $\gcd(a, n) = 1$

To find a^{-1}.

use EXTEUC (a, n)

if d ≠ 1 No inverse

if d = 1, x, y, Then $1 = ax + ny$

$\therefore$ In Z_n $ax = 1$

eg) In Z_{100}

$16(100) - 39(41) = 1$

In Z_{100} $41(-39) = 1$

$41^{-1} = -39 = 61$

Equation $Ax \equiv B \bmod N$

Assume $\gcd(A, N) = 1$

$x \equiv A^{-1} B \bmod N$

Chinese Reminder Theorem

Assume $\gcd(M, N) = 1$

system $x \equiv A \bmod M$

$x \equiv B \bmod N$

Has solution $x \equiv C \bmod MN$

① Parameterize $x = MY + A$

$MY + A \equiv B \bmod N$

$MY \equiv (B - A) \bmod N$

$Y = M^{-1}(B - A) \bmod N$

eg. $x \equiv 13 \bmod 41$

$x \equiv 27 \bmod 100$

$x \equiv 41y + 13 = 41(54) + 13 = 2227 \bmod 4100$

$41y + 13 = 27 \bmod 100$

$41y = 14 \bmod 100$

$y = \underbrace{(61)}_{1}(41)y = 61(14) = 854 = 54 \bmod 100$

Fermat If p Prime than $a^{p-1} \equiv 1 \bmod p$ P95

(more: $a^{p-1} \equiv 1 \bmod p$, $1 \le a \le p-1$) Theorem 31.31

Modular Exponentiation.

Island Hopping Algo Calculate $A^B \bmod N$

zero time

① Write $B = 2^{s_1} + 2^{s_2} + \cdots 2^{s_T}$ $\quad s_1 > s_2 > \cdots s_T$

② Compute $x_T = A^{2^T} \bmod N$, $0 \le T \le S$,

$x_0 = A$

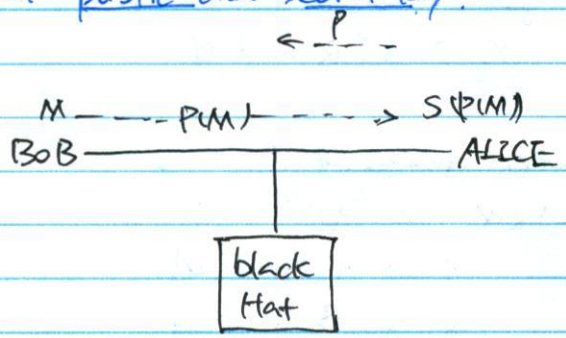
$x_T = x_{T-1}^2 \bmod N$ $\quad O(d^3)$ with d = # digits

③ $A^B = A^{2^{s_1}} \cdot A^{2^{s_2}} \cdots A^{2^{s_T}} = x_{s_1} \cdot x_{s_2} \cdots x_{s_T}$

9/1.

Publickey Encryption.

Each participant has both a public and secret key.



Message M.
ENCRYPTION P(M)
DECRYPTION S
S(P(M)) = M.

One-Way Function → Alice picks P.S
Alice → sends P to BoB
keeps S secret.
Only Alice can compute $S_{A.I}$

Thm. RSA cryptosystem P963. proof.
let p.q be distinct primes $\sim 10^{100}$.
$n = pq$
equation: 31.∞
$\phi(n) = n\cdot(1-\frac{1}{p})(1-\frac{1}{q})$
$= pq(1-\frac{1}{p})(1-\frac{1}{q})$
$= (p-1)(q-1)$

For any a relatively prime p&q P95 Theorem 31.31
$a^{p-1} \equiv 1 \bmod p \rightarrow a^{(p-1)(q-1)} \equiv 1 \bmod p$
$a^{q-1} \equiv 1 \bmod q \rightarrow a^{(p-1)(q-1)} \equiv 1 \bmod q$
$\Big\} a^{(p-1)(q-1)} \equiv 1 \bmod pq = n$ (Theorem 31.29)

if $ef \equiv 1 \bmod (p-1)(q-1)$ d. $ef = k(p-1)(q-1)+1$.
then $a^{ef} \equiv a^1 \equiv a \bmod (p-1)(q-1)$?
$a^1 \cdot a \bmod n$.

RSA.

Alice [1] selected primes p.q ($\sim 10^{100}$) (secret)
[2] set n = pq (public)
[3] Select e. $\gcd(e,(p-1)(q-1)) = 1$ Public P=(e,N)
[4] compute d., $ed \equiv 1 \bmod (p-1)(q-1)$ Secret S=(d,N)

MESSAGE. M $0 < M < n$ $\gcd(M,p) = \gcd(M,q) = 1$
$P[M] = M^e \bmod N$ (Public!)
$S[C] = C^d \bmod N$ (secret)
$S[P(M)] = (M^e)^d = M^{ed} = M \bmod n$

P. NP. NP-Complete

Property ~ YES/NO Question
~ Yes/No Decision Procedure
~ Set L (Language) of instances with property

P?

✓ Connected Graph (G).
✓ Shortest Path (G, V, w, k) — integer; V: vertices

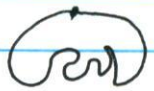
finite problems

Yes ⟺ p(v,w) ≤ k

✗ CLIQUE (G, k) ∉ NP

Yes ⟺ clique of size ≥ k

✗ HAMILTONIAN CYCLE (G) ∉ NP



We say L ∈ P (polynomial time) if there is an algorithm

(1) Always answers YES/NO
(2) Answer always correct.
(3) How much time it takes.

Time $O(n^c)$ with n = inputsize
c constant.

3-SAT ∈ NP (Truth Table ?)

Boolean $x_1, \ldots x_n$

CLAUSE. $C = \bar{x}_i \vee x_j \vee \bar{x}_k$

INPUT $C_1 \wedge C_2 \wedge \cdots \wedge C_m$

Output Yes if. $\exists\ x_i \leftarrow T$ or F
$\ni C_1 \wedge \cdots \wedge C_m \leftarrow T$

3-SAT ∈ P ? Probably NOT

NP – Nondeterministic Polynomial Time

VERIFIER (Poly Time)

ORACLE

$L \in NP$ iff → Input: x

Oracle: Certificate y → when $x \in L$ ORACLE can supply certificate to VERIFIER. Who then knows $x \in L$

$P = NP$?

~~$NP = coNP$~~ ?

Reducibility $L_1 \leq L_2$ (L_1 reducible to L_2)

given blackbox for L_2 (in unit time)

then $\exists$ poly time algorithm for L_1

So: If $L_2 \in P$ then $L_1 \in P$

4-SAT $\leq$ 3-SAT

for each clause $x_1 \vee x_2 \vee x_3 \vee x_4$

Add "new" z. Replace clause by

$x_1 \vee x_2 \vee z_1$

AND $x_3 \vee x_4 \vee \bar{z}_1$

Big big Theorem:

Any $L \in NP$, has $L \leq$ 3-SAT

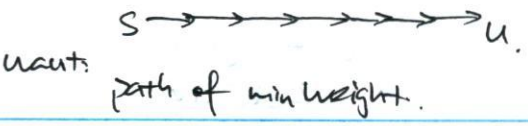
Corollary. If 3-SAT $\in P$ then $P = NP$

Def:

$L \in$ NP Complete

i) $L \in NP$

ii) Every $L' \in NP$ has $L' \leq L$



want: path of min weight.

24.5. DIJKSTRA

Input Directed Graph $G = (V, E)$

Source $s \in V$

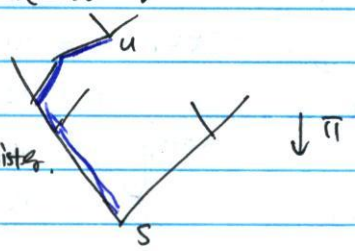
weight Fun. $w(x, y)$ for $(x, y) \in E$ | weight Fun always ≥ 0

$\delta(u, v) :=$ min w. of path u to v

Output: $\delta(s, u)$ for all $u \in V$.

π = "parent"

MINPATH: slide down the bannister. and then reverse.



~~not that efficient.~~

Data.

$d[v]$: shortest length path "for now"

$\pi[v]$: parent "for now"

S = vertices processed

Q = Min heap on $\bar{S}$ under d.

Initial.

$S = \{s\}$

For $v \in ADJ(s)$ $\pi[s] = s$

$d[v] = w(s, v)$

else $\pi[v] = NIL$

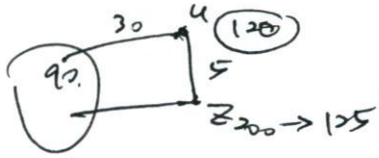
$d[v] = \infty$

Build heap. ~ linear time

Middle: All $v \in S$ $d[v] = \delta[s, v]$ and slide down Bannister for Path

For $v \notin S$ $d[v]$ is min 'cost' v to $z \in S$ (1 Edge) + Then down to Bannister.

Claim: $v \notin S$ with $d[v]$ minimal is correct.



$O(E \lg V)$

- DIJKSTRA.

① Initialize

while $Q \neq \phi$

$u \leftarrow$ heap-Extract min $[Q]$ $\lg V$

$S \leftarrow S \cup \{u\}$

其实是指整个while loop.

Total E-time ← For $z \in ADJ[u]$ AND $z \notin S$

*** { If $d[u] + w[u,z] < d[z]$ } (*else nothing.)

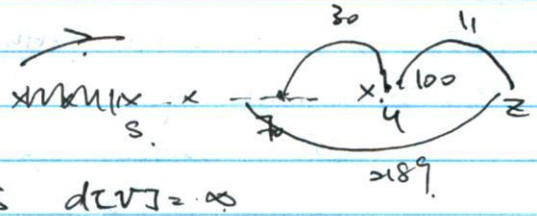
$d[z] \leftarrow d[u] + w[u,z]$

$\pi[z] = u$

Update Q } $O(\lg n)$.

☆ (always remember to update the data structure)

24.2 G is DAG.



① TopSort. $(E+V)$

ignore v before s

initialize $d[s] = 0$ $d \neq s$ $d[v] = \infty$

For each u after s in $\pi[v] = NIL$ TopSort Order]

$O(E)$.

(* $d(u)$, $\pi(u)$ correct *).

For $z \in ADJ[u]$

update ***.

☆ Dynamic Problem always base on previous step

Text:

Input: length $\ell_1, \cdots \ell_n$.

Line length L

Penalty Function $P(x)$

✓ Text Alignment: $P(x) = x^3$

$L=10$	Gap	P
Mow Sing__	2	8
A Melody__	2	8
Mow Sing A	0	0
Melody wwww	4	64.

$m(\ell)$ = min penalty for $\ell_1 \cdots \ell_i$

initialization: while fit on one line, put all in one line.

given $m(1), \cdots m(i-1)$

For each k where $\ell_k, \cdots, \ell_i$ fit on one line

constant time → calculate $F[k]$ = Penalty for line $\ell_k \cdots \ell_i$ + $m[k-1]$ (takes 0 time already calculated)

Set $m(i)$ = Min of those $f(k)$

set $\pi[i]$ = that k

① Mid-term 前也要考 + Mid-term 題. [有一題來自作業]

② proof : words. at least 1 proof

③ 七問題全分化為小問