MESSAGES | PHONE CALLS

FRIDAY DECEMBER 17

$T_1' = T_1 + \phi_2 - \phi_1$

$T_2' = T_2 + \phi_3 - \phi$

$T_n' = T_n + \phi_{n+1} - \phi_n$

$\Rightarrow T_1' + \cdots T_n' = T_1 + \cdots + T_n + \underbrace{\phi_{n-1} - \phi_1}_{\geq 0}$ ($\phi_1 \to 0$)

? $T_1 + \cdots T_n \leq C \cdot n$.

x_i : number of the begening of i^{th} step ($\equiv i$ in binary)

ϕ_i = # 1's in the binary expansion of i ($\equiv$ # 1's in x_i)

$\phi_1 = 0$

$\phi_i \geq 0$

claim: $T_i' \leq 2 \Rightarrow T_1 + \cdots T_n \leq 2n$ as well.

$\Rightarrow \sum T_i \leq 2n$

pf: x_i (i) $\quad 110\,110 \cdots 10$ (1)

$110\,110 \cdots 11$

$\phi_{i+1} - \phi_i = 1$

$T_i = 1$

ii). $1\,01110 \cdots 0\,\overbrace{11111}^{\ell}$ (1)

$1\,00000$

$\phi_{i+1} - \phi_i = 1 - \ell$

$T_i = \ell + 1$

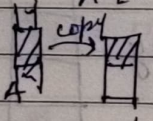
$T_i' = 2 \qquad T_1 + T_2 + \cdots T_n = 2n$

how can we define ϕ for Random function. ?? not easy.

Expandable array. Class-13.

Adding x at the end of the array

$A[++i] = x$ Amortized time: n



what is the good strategy to decide size of A'

isc {x'} {

if A is full

copy A into an array of twice the size

then add x at the end

else add x at the end

}

18 SATURDAY DECEMBER

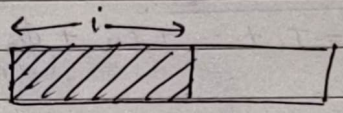
MESSAGES	PHONE CALLS

$1+2+3+1+5+1+1+1+9+1+1+\cdots \leq 2n$.

$T_i' = T_i + \underbrace{\phi_i - \phi_{i+1}}_{\sim m}$ $\quad T_i' \leq 2$ always.

when T_i is large we want it to be small. T_i is large when array is

let $\phi_i = -m_i$ $\quad m+1$



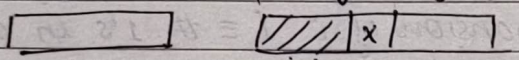
m_i = Size of theory.

i) the array size does not change.

$T_i = 1$

$\phi_i - \phi_{i-1} = 0$

ii) the array is full at the begining of element-



ϕ_i $\quad -(i-1)$ $\quad \phi_{i+1}$ $\quad -2(i-1)$

$T_i = (i-1)+1 = i$

$T_i' = i + \Delta\phi$

$= i + \phi_i - \phi_{i+1}$

$= i + (-(i-1))$.

$= 1$. OH NO!! ($\phi_i < 0$ violated).

$T_i' = i + [2-(i-1)]$

$= i + 2 - i + 1$

$= 3$.

$\underbrace{T_1' + \cdots T_n'}_{\leq kn} = \underbrace{T_1 + \cdots T_n}_{\leq kn} + \phi_n - \phi_1$ $\quad \geq 0$

if ϕ_n is too -ve then we will have -ve quantity

how to make ϕ_i +ve ? $\quad i - m_i$? still -ve $\quad 2i - m$ ok!

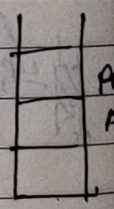
This shows that amortized complexity is ≤ 3

$T_1' + \cdots + T_n' \leq 3n \rightarrow T_1 + \cdots + T_n \leq 3$.

19 SUNDAY Resizable Array :

At each step i) Insert a new element at the end of array

ii) remove the last element



$n \leftarrow$ # elements.

i') $n++$, $A[n] = x$ } → if array is full

ii') $n--$ ↳ double the size of the array ↳ move copy element-

1999

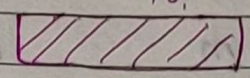
MESSAGES | PHONE CALLS

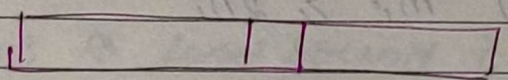
MONDAY 20 DECEMBER

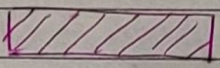
want what? not too space is wasted.

insert, Delete, Delete

insert (x) {
  if array is not full
    n++, a[n] = x
  else:
    copy the array of ~~time~~ twice of the size A(new).
    A[n] = x
    n = 2n
}

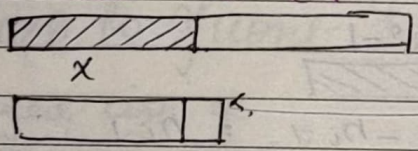






remove last {
  if (# elements ≤ n/2)
    copy the elements into an array of half the size & update n ← n/2
}

Let's take one eg.



insert- DD → I I DD (x, a)

→ Again increase

what is the potential fn?

m_i : size of array at time step i

n_i : # ~~steps~~ elements in the array at step i (or earlier $n_i = i$)

$\phi_i = 2n_i - m_i$

$T_i' = T_i + \phi_i - \phi_{i-1}$

when is T_i' is a lot? → we want potential to ↓

$-(2n_i - m_i)$ → when we are deleting & $n_i = m_i/4$

$c_i = \max(2n_i - m_i, m_i)$

$c_i = \begin{cases} 2n_i - m_i & \text{i) if the array is more than half full} \\ m_i/2 - 2n_i & \text{ii) if the array is less than half full} \end{cases}$

$n_i = m_i/2$

1999 if array is half ⇒ both of them match = 0

21 TUESDAY DECEMBER

MESSAGES

PHONE CALLS

change in pot. & del threshold 1/3

$\phi_i \geq 0$

i) $2n_i \geq m$

ii) $m_i \geq 2n_i$

Insert opⁿ {

(a) if array is not full

$T_i = 1.$

$T_i' = 1 + [2] \leq 3$

b) if the array is full.

$T_i = n_i + 1$

$T_i' = n_i + 1 + [2 - (n_i - n_{i-1})]$

$= n_{i-1} + 1 + [2 + n_{i-1}]$

$= [2[n_{i-1}+1] - 2n_{i-1}].$

$n_{i-1} = m_{i-1}$

$n_i - 1$

$2n_i - 1$

Delete opⁿ

(a) If array is more than 1/4 - full $\leq$ 1 }

$T_i = 1.$

$T_i' = T_i + \Delta\phi \leq 3.$

≤ 2

(b) If the array is 1/4 full.

$T_i = n_{i-1} + 1$

$T_i' = n_{i-1} + 1 + \Delta\phi$

$-2n_{i-1}$

$\leq 1 - n_{i-1} \leq 3.$

$n_{i-1} = 2$

$4n_{i-1}$

Initial pot $3n_{i-1}$

final pot

$2n_{i-1} - n_{i-1} = n_{i-1}$

CLRS. : Book.

Class 14 : 30/9/21.

Queue from two stacks.

enque.(x) {

S_1. push ( ) }

①

Dequeue }

if S_2 is empty not

(S_2.pop ( ))

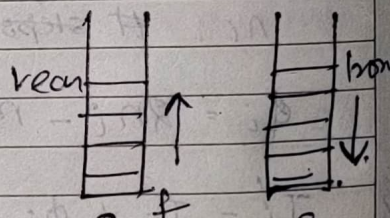
else

while (S_1 ! empty)

{ S_2. push (S_1. pop). } } tot of oper.

S_2. pop }

①



Amortized Complexity

potential fn : $\phi_i = 2$ (# elements in stack 1.)

eqn = $1 + 2 = 3$

Deque : $T_i = 1$ (case 1)

(case 2)

1999

MESSAGES | PHONE CALLS

WEDNESDAY DECEMBER 22

potential drop needed = $2m$.

case 2: $T_i = 2m+1$. $\Phi_{i-1} = 2m$

$T_i' = 1$ $\Phi_i = 0$.

[T_i': artificial thing that we created to ~~find~~ bound $T_1 + \dots T_n$]

$T_1 + \dots T_N \le T_1' + \dots T_N' \le CN$

$+ \Phi(N)$ → we can't make it -ve.

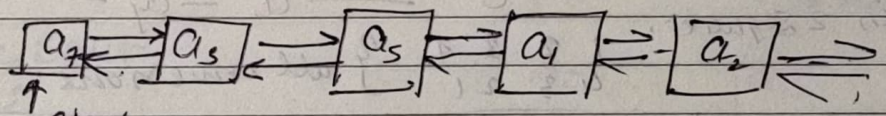
Amortize Case: $O_1, O_2 \dots O_N$ we say that amortized compl. of C is the total running time $\le C.N$. (kind of worst-case)

Average Case: $i_1, i_2 \dots i_L$ (All possible i/p). $\frac{rt(i_1) + runtime(i_2) + \dots (i_L)}{L}$

Worst Case: max running time in i/p i

Move to front heuristic (Slacter Tarjan '82)

Linked list of n distinct elements



Input $a_5, a_7, a_3, a_5, a_3, a_5, a_1 \dots a_{tN}$

3 + 1 + 2 + 3.

Cost = total time of access the element.

At time i, the alg see the next request a_{ti} (this is called online alg)

offline algorithm: know the entire sequence b4 hand.

↓

OPT: the optimal cost of an offline algorithm.

Alg A: Move to front heuristic (at time t, when a_{ti} is accessed move it to the front.)

Thm: for any sequence $a_{t1} \dots a_{tN}$

total cost (A) $\le$ 2 total cost (OPT for σ)

A_i: cost increased by the alg at time i

1999 O_i: " " " opt " " - $\frac{1}{2}$(# inversions b/w σ_i & σ_{i-1})

q_i: potential at time i

23 THURSDAY DECEMBER

MESSAGES	PHONE CALLS

Amortized cost $A_i' = A_i + \Phi_i - \Phi_{i-1}$

we will show that $A_i' \le 4O_i$ for every step i

$\Rightarrow \sum A_i \le 4O_i$?

$A_1 + \cdots + A_N \le A_1' + \cdots A_N' \le 4(O_1 + \cdots O_N)$.

• potential is not just depend on A but also on O.

A: 1 7 5 6 3 4 10 — σ_i : ordering of element at time i

OPT: 10 4 3 6 5 7 1 — T_i : ordering of the element at time i

Given two permutation σ & τ of some elements $a_1, \ldots a_n$ the # inversion b/w σ & τ is defined as:

pairs (a_i, a_j) when appear in opposite order in 2 sequence.

σ — a_j — a_i —

τ — a_i — a_j —

1	2	3	4
4	1	3	2

$(1,4)\ (2,3)\ (3,4)$

1 2 3 4 } all inversions.
4 3 2 1

At time i element a is accessed.

potential.

A — b — c — a — ↓ (b,a) was inversion b4 but not now

O — c — a — b — ↑ (c,a) was not inversion but now ie- bcm one

A: $\leftarrow n_1 \rightarrow$ a, y x — Let I_1 be the element here which appear b4 a in the opt sequence as well

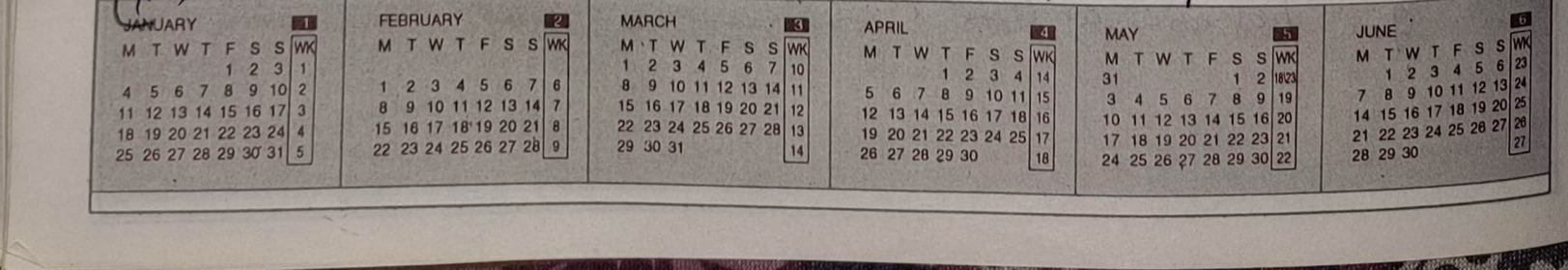
O: $\leftarrow m_1 \rightarrow$ a, y I_2 — element $\rightsquigarrow$ a which appear after in opt list

i) $|I_1| + |I_2| = n_1 - 1$

ii) $A_i = n_1 + 1$ (swap the pointer) a2 front

iii) $O_i = m_1 \geq I_1$

iv) change in potential $\Phi_i - \Phi_{i-1} = |I_1| - |I_2|$ → now inversion; → no longer inversion



MESSAGES | PHONE CALLS

FRIDAY DECEMBER 24

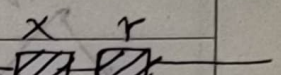
we have to show $A_i' = A_i + \phi_i - \phi_{i-1} \le 4O_i$

$(n_1 + 1) + |I_1| - |I_2|$

$= |I_1| + |I_2| + |I_1| - |I_2|$

$= 2|I_1| \le 2n_1 = 2O_i$

what if the opt swaps 2 elements.



2 cases. $0 + \Delta\phi \le 4+1$

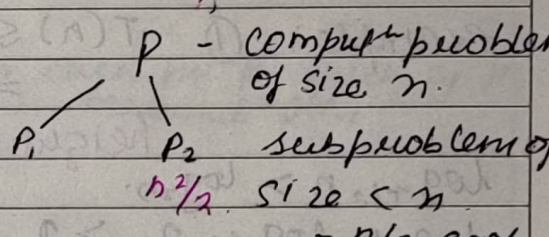
each time opt do swap, write the same. ⇒ if both of them ≤ 1 potential is increased by 1.

Class 15. 4/oct. Divide & Conquer.

solve subproblem & combine their solutn for P.

$< \frac{n^2}{2}$ times.

$\Rightarrow < \frac{n^2}{2} + \frac{n^2}{2} \le n^2$.



Merge sort / Quick sort

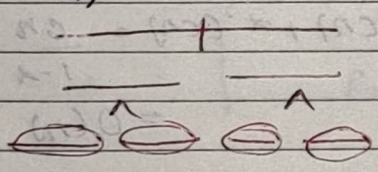
sort - n elements. {

i) Divide the i/p into 2 smaller i/ps.

ii) Recursively solve the smaller i/ps.

iii) "combine" the two soln. 1 5 6 9.

MS	QS
i) easy.	i) non-trivial.
ii) non-trivial	ii) easy.



5 1 6 9 10 2 12 3.

5 1(51) 6 9 69

[5] [1] [6] [9]

Merge Sort (A, n) {. if $n = 1$

$A_1 \leftarrow$ 1st half of A

$A_2 \leftarrow$ second half of A.

Merge sort $(A_1, n/2)$

Merge sort $(A_2, n/2)$

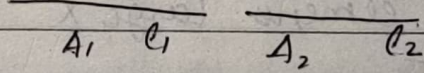
$A \leftarrow$ merge (A_1, A_2)

}

- merge procedure is cheap.

⇒ hence DC will work better.

$O(n \log n)$: think of DEC always



$\Rightarrow O(l_1 + l_2)$

1999

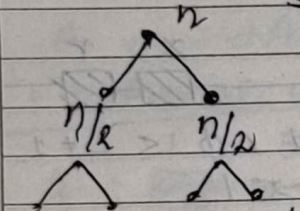
25 SATURDAY DECEMBER

MESSAGES	PHONE CALLS

Recurrence. $T(n)$: worst-case time to sort n elements.

$T(n) \leq T(n/2) + T(n/2) + cn$ → const.

$\leq 2T(n/2) + O(n)$. sol^{n} is $O(n \log n)$



$+ cn$ (merge cost

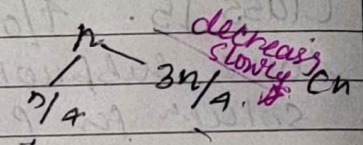
$+ c\frac{n}{2} + c\frac{n}{2} = cn$.

time $T(n) \leq cn$ height

$\leq cn(\log_2 n)$

example: ① $T(n) \leq T(n/4) + T(3n/4) + cn$

$= O(n \log_{4/3} n)$



height $\leq \log_{4/3} n$. $\left(\frac{3}{4}\right)^i n\ 1$ ← i

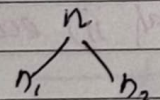
$\log_{4/3} n > \log_2 n$.

$\log_2 n \log_{4/3} 2 \geq 1$

const ⇒ going bigger.

② $T(n) \leq T(n_1) + T(n_2) + c_2$

$T(n) = O(n)$.



$+ cn$.

$+ cn_1 + cn_2 \leq \alpha(cn)$

$n_1 \quad n_2 \leq \alpha^2(cn)$

$n_2 + n_2 \leq \frac{9n}{10}$ (some chunk will be known array)

$n_1 + n_2 \leq \alpha n \quad \alpha < 1$

$cn + \alpha(cn) + \alpha^2(cn) = \frac{cn}{1-\alpha}$

$= O(n)$

$T(n) \leq T(n/2) + O(n)$

26 SUNDAY ③ binary search : $T(n) + T(n/2) + O(1)$

Quick Sort (A, n) {

$x \leftarrow A[1]$.

{ $A_1 \leftarrow$ of elements less than x

$A_2 \leftarrow$ all elements large x

o/p $A_1 \times A_2$

$A_1 \quad x \quad A_2$

$T(n) \leq T(n_1/2) + T(n_2) + cn$

worst Case 1 2 3 .. n. n

$n-1$

1999

MESSAGES | PHONE CALLS

MONDAY DECEMBER 27

Fix this? pick x "randomly" ⇒ expected running time ✱

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

1) Maximal point
i/p: set of n pts on the plane.

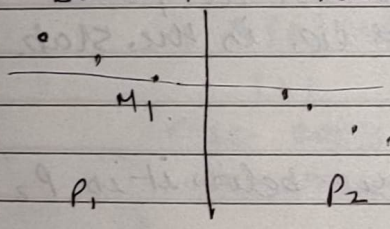
A point P is maximal x_p, y_p if there is no other point p' with $x_{p'} \geq x_p$ & $y_{p'} \geq y_p$

sol^n
Repeat all the maximal points
$O(n \log n)$ time ? — n^2 time. — for each pt p check if p is maximal $O(n)$

$T(n) = 2T(n/2) + O(n)$
$O(n \log n)$ time ?

$T(n) = 2T(n/2) + O(n)$ ←
- based on the middle x-coordinate
  → $O(n)$ ⇒ problem.

P1 P2
solve P_1 P_2 recursively.



combine answer ?
M_2: max pts in P_2
M_1: P_1
find out the highest y-coordinate q on M_2
M_1' ← set of pts in M_1 with y coord. higher than q.

Easier way out: finding middle
- sort the points on x-coordinate $n \log n$.
- for each recursive call
  - we will maintain the sorted pt

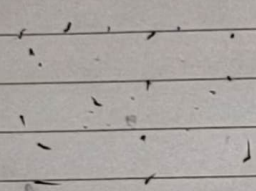
Maximal (P, n) { → points are in an array arranged be x-coord
P ← P[n/2]
P_1 ← left half of P
P_2 ← right half of P
M_1 ← maximal(P_1, n/2)
M_2 ← ——— (P_2, n/2)
M ← combine(M_1, M_2)
return M }

1999

28 TUESDAY DECEMBER

MESSAGES	PHONE CALLS

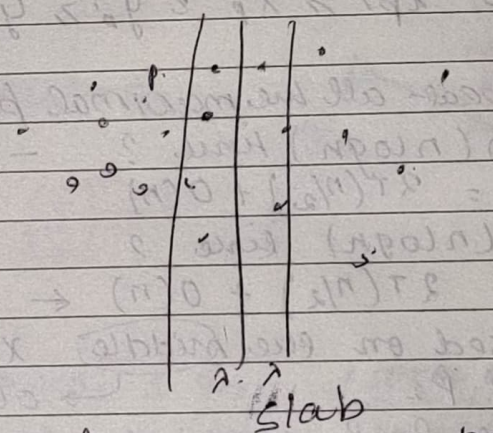
② Closest pair of points



n points in the plane.
Find the closest pair of points
n^2 time algo.
(x_1, y_1) (x_2, y_1). $O(n^2)$.
$(x_1 - x_2)^2 + (y_1 - y_2)^2$.

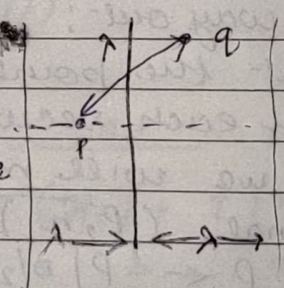
can we do $O(n \log n)$.
Divide into the two equal parts based on the x-coord
→ arrange acc to x.

```
Closest (P, n) {
    P1 ← left 1/2
    P2 ← right 1/2.
    (p, p') ← closest- (P1, n/2).
    (q, q') ← closest- (P2, n/2)

    λ1 = dist- (p, p') ≤ λ2 = dist- (q, q')
        o/p min (λ1, λ2)- ?!

    λ ← min (λ1, λ2).
    P1', P2' : points of P1, P2 lying in the
    slabs of width λ around x^
    for each p ∈ P1' {
        compare p with 4 points above it & 4 points below it in P2'
    }
    o/p the smallest dist found
}
```



only have to look at points which lie in the slab.

Class 16 : 7/Oct



Fix a point $p \in B_L$ consider all the points $q \in B_R$ dis $(p, q) \leq \lambda$ then there are atmost 8 choices for q infact ≤ 4 points above by ≤ 4 points below it- (why.

(If we use circle to prove max value will be 6 instead of 8

1999

JANUARY							
M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY							
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH							
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL							
M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY							
M	T	W	T	F	S	S	WK
31				1	2	3	18 19
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE							
M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES	PHONE CALLS

WEDNESDAY 29 DECEMBER

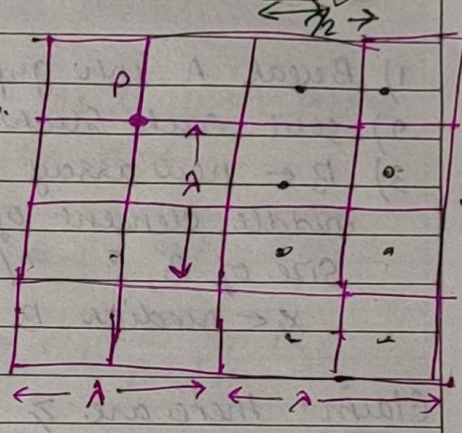
(2) P is sorted by x-coordinate. we also need to maintain the points sorted by y-coordinate.

closest-(P, P', n) {

sorted by x sorted by y coord

$P_1 \leftarrow P[1] \ldots P[n/2]$, $P_1' \leftarrow$ use P' to obtain

$P_2 \leftarrow P[n/2+1] \ldots P[n]$. P_1' is $O(n)$ time

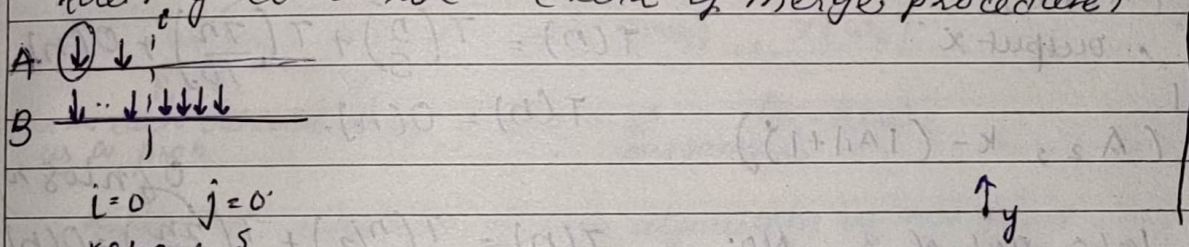


P': the points sorted by y coordinate

x^*: median x coord.

P_1': Scan P': (x, y) & $x \leq x^*$ add it to P_1'

Similarly we can obtain the points B_L, B_R obtained according to their y-coordinate. (kind of merge procedure)



i=0 j=0

repeat {

if A[i] > B[j], copy B[j] to C

j++

else copy A[i] to C.

increment i

}

sort $O(n \log n)$ X.

(3) A: find the median of A, $O(n)$ time.

A, k: find the k^{th} smallest number $O(n)$ (selection)

$T(n) = T(n_1) + T(n_2) + O(n)$ where $n_1 + n_2 \leq cn$ where $c < 1$.

select-(A, k) {

select-(A', k')

A' will be smaller array than A

1993

JULY							7
M	T	W	T	F	S	S	WK
		1	2	3	4		27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST							8
M	T	W	T	F	S	S	WK
30	31					1	31/36
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER							9
M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER							10
M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER							11
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

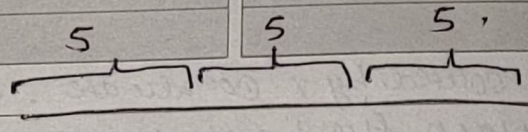
DECEMBER							12
M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

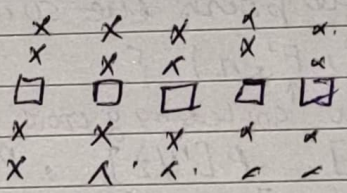
30 THURSDAY DECEMBER

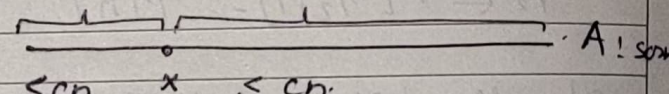
MESSAGES

PHONE CALLS

1) Break A into grp of 5 element $O(n)$.
2) sort each such group - $O(n)$ time
3) B ← new array formed by taking middle element of
size of B ? $n/5$
x ← median B

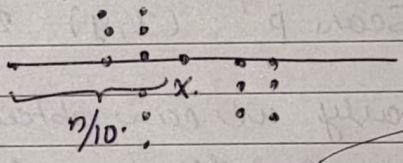






$T(n) = T(cn) + O(n)$.

Claim: there are $\geq 3n/10$ in A which are less than x
there are $\geq \frac{3n}{10}$ in a which are more than x.



if $k < |A_1| - 1$ then
    select (A_1, k).
if $k = |A_1| - 1$, output x
if $k > |A_1| - 1$
    select $(A_2, k - (|A_1| + 1))$.
}.

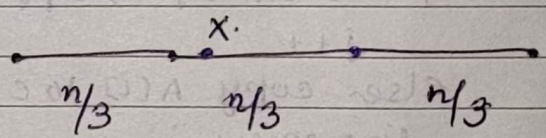
$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$$

$$T(n) = O(n).$$

Could we take grp of 3 No.

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

$O(n \log n)$



Practical version
1. Pick a number uar in A
2. follow (*)

Q. Suppose we have an exp where the prob of success is p. How many times do we have to repeat it till we get success?

X : random variable

$P_r[X = 1]$ $P_r[X = i]$

$$E(X) = \sum_{i=0}^{\infty} i \cdot P_r[X = i].$$

expected value of X

$= 1/p$

$P_r[X = i] = (1-p)^{i-1} p$.

$P_r[H] = 1/2$ 1
$P_r[T] = 1/2$ 0

$n/2$

tossing

1999

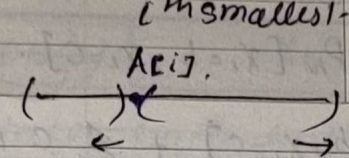
MESSAGES

PHONE CALLS

FRIDAY 31 DECEMBER

happen i/p.

$E[T_n]$ = ?



If we happen to pick the i^{th} smallest number as x then our recursive call would have size either $i-1$ or $n-i$

$\frac{1}{n}$

$Y_n = E[T_n]$.

$Y_n = \begin{cases} \max(Y_{i-1}, Y_{n-i}) & 1/n \\ Y_{\max(i-1, n-i)} & \text{w.p. } 1/n. \end{cases}$

$Y_n = \sum_{i=1}^{n} \frac{1}{n} \cdot [Y_i \text{ or } Y_{n-i}]$. $\sum_{i=1}^{n/2} \frac{1}{n}[Y_{n-i}] + \sum_{i=n/2+1}^{n} \frac{1}{n} Y_i$

Class 17

11/10/21. Selection problem A, k k^{th} smallest-

i) Deterministic : fairly complex algo

ii) Randomize alg : use randomness eg. toss of coin.

rand() H T

0 ½ 1

0 a_1 a_2 .. a_n 1

pick an element from.

Random Variable X : set of outcome $\rightarrow R$.

$\underbrace{\omega_1, \omega_2, .. \omega_k}_{p_1 \; p_2 \; p_k} \rightarrow X(\omega_1), X(\omega_2) .. X(\omega_k)$

with each outcome there is a prob.

$\mathbb{E}X = \sum_{\text{outcomes } \omega} p_\omega \cdot X(\omega)$

Input I : execution of the algorithm is not fixed $\omega_1 .. \omega_k$

$\omega_1, .. \omega_k \rightarrow$ expectations of the alg depending on the outcome rand()

X : running time $X(\omega_i)$ = # steps in ω_i

$\mathbb{E}[X]$ ie $\mathbb{E}$ (running time) =

[eg (i) Toss n unbiased coins X_2 = # heads a

$\mathbb{E}[X] = \sum_{\text{outcomes } \omega} p_\omega X(\omega) = \sum_a \sum_{\omega: X(\omega)=a} p_\omega \overline{X(\omega)} = \sum_a a \sum_{\omega X[a]=a} p_\omega$

$= \sum_a a \, P_r[X=a]$

$\sum_{a=0}^{n} a \binom{n}{a} \frac{1}{2^n} = \frac{n}{2}$]

(i) $X = X_1 + X_2$ X, X_1, X_2 are three random variable

then $\mathbb{E}(X) = \mathbb{E}(X_1) + \mathbb{E}(X_2)$ Linearity of expectations, it does NOT impose any condition on X_1, X_2 not true.

$X_1 X_2 X$: take value is 1, 2 .. n $\mathbb{E}(X_1 X_2) = \mathbb{E}(X_1) * \mathbb{E}(X_2)$ X

Proof $\mathbb{E}[X] = \sum_a a \, P_r[X=a]$

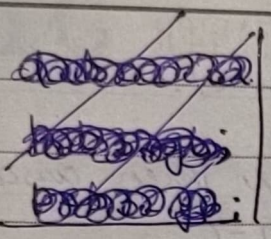
$= \sum_a a \, P_r[X_1 + X_2 = a]$ $X_1 = 1 \quad X_2 = a-1$ $X_1 = 2 \quad X_2 = a-2$

$= \sum_a a \sum_b P_r(X_1 = b, X_2 = a-b) \rightarrow \sum_a a \sum_{b,c} P_r[X_1 = b, X_2 = c]$

$b + c = a$

1999

Name & Address | Office | Residence



$\sum_{b=0}^{n}\sum_{c=0}^{n}(b+c)\,P_r[X_1=b, X_2=c]$ — incorrect. $P[X_1=b]\cdot P[X_2=c]$

$\sum_{b\neq 0}^{n} b \sum_{c} P_r[X_1=b, X_2=c] + \sum_{b,c} c\, P_r[X_1=b, X_2=c]$

$P_r[X_1=b]$

$\mathbb{E}[X_1] + \mathbb{E}[X_2]$

② X: P_1, X_1 ; $1-P_1$, P_2, X_2

$\mathbb{E}X = P_1\mathbb{E}X_1 + P_2\mathbb{E}X_2$

Appl^n i) n unbias coins $\mathbb{E}[\#\text{heads}] = n/2 = ?$

Indicator Random variables: 0 or 1: $\mathbb{E}[Y] = 0 \times P_r[Y=0] + 1 \times P_r[Y=1]$

$X_i = \begin{cases} 1 & \text{if the } i\text{th coin toss is } H \\ 0 & \end{cases}$ $= P_r[Y=1]$

$\mathbb{E}[\#\text{heads}] = \mathbb{E}[X_1] + \mathbb{E}[X_2] + \cdots \mathbb{E}[X_n] = n/2$

ii) X: random variable suppose there is some other random variable Y

suppose $X = X_i$

throw a dice, if dice shows i, toss an unbiase coin i times.

X = # heads

$E[X] = \sum_{i} P_r[\text{dice} = i] \cdot \mathbb{E}[\#\text{heads when we throw } i \text{ coin}]$

$= \frac{1}{6}\left(\frac{1}{2} + \frac{2}{2} + \cdots \frac{6}{2}\right)$

$X = \begin{cases} X_1 & \text{when } Y=1 \\ X_2 & Y=2 \\ \vdots \\ X_n \end{cases}$ suppose Y is independent on the random variable $X_1 \cdots X_n$.

then $\mathbb{E}X = P_r[Y=1]\mathbb{E}X_1 + P_r[Y=2]\mathbb{E}[X_2] + \cdots P_r[Y=n]\mathbb{E}[X_n]$

let $Z_i = \begin{cases} 1 & \text{if } Y=i \\ 0 & \text{otherwise.} \end{cases}$ $Z_1 \cdots Z_n$ are also indepen

$X = X_1Z_1 + X_2Z_2 + \cdots X_nZ_n$

$EX = \mathbb{E}(X_1Z_1) + \mathbb{E}(X_2Z_2) \cdots \mathbb{E}[X_nZ_n]$

$= \mathbb{E}(Z_1)\mathbb{E}(X_1) + \cdots \mathbb{E}(Z_n)\mathbb{E}(X_n)$

$= P_r[Y=1]\mathbb{E}[X_1] + \cdots + P_r[Y=n]\mathbb{E}[X_n]$

$\frac{1}{n} \quad \alpha \quad \alpha \quad \alpha \quad \frac{1}{n} \quad \alpha \quad \frac{1}{n}$

Selection (A, k) {

- pick an element x uar from A
- split A into A_1 & A_2
- if $|A_1| = k-1$
- if $|A_1| \geq k$ Selection (A, k)
- if $|A_1| < k-1$ Selection $(A_2, k - |A_1| - 1)$

}

T_n: Expected running time on array of size n

Name & Address | Office | Residence

- ~~post numbers (DNS)~~
- ~~total number of table attribute from ER diagram~~
- ~~Transact^n & concurrency control problems~~
- ~~NTM ≅ DTM ??~~

X_n : expected running time on an array of size n

T_n : # steps on an array of length n.

$X_n = \mathbb{E}\, T_n$

$T_n = T_{i-1} + T_{n-i} + O(n)$ if the element x is the i^{th} smallest element — Assume sorted in ↑ ord

$T_n \le \max(T_{i-1}, T_{n-i}) + O(n)$ if x is the i^{th} smallest #

$T_n \le T_{\max(i-1, n-i)} + O(n)$ if x is the i^{th} smallest # in A.

$$\therefore \mathbb{E}\, T_n \le \sum_{i=1}^{n} \left[\mathbb{E}\, T_{\max(i-1, n-i)} + O(n) \right] \cdot \frac{1}{n}.$$

$$X_n \le \underbrace{\frac{1}{n} \sum_{i=1}^{n} X_{\max(i-1, n-i)}}_{} + O(n).$$

$$= \frac{1}{n} \left[X_{n-1} + X_{n-2} + \cdots \underbrace{X_{n/2} + X_{n/2} + X_{\frac{n}{2}+1} \cdots X_{n-1}}_{} \right] + O(n).$$

if 1st smallest chosen bigger array of size 9

$$X_n \le \frac{2}{n} \left[X_{n/2} + X_{\frac{n}{2}+1} + \cdots X_{n-1} \right] + O(n)$$

$X_9 + X_8 + X_7 + X_6 + X_5 + X_5 + X_6 + \cdots X_9$

$$\le \frac{2}{n} \left[\frac{n}{4} X_{3n/4} + \frac{n}{4} \cdot X_n \right] + O(n)$$

$X_n \le X(\alpha n) + O(n) \quad \alpha < 1.$

$$X_n \le \frac{X_{3n/4}}{2} + \frac{X_n}{2} + O(n) \Rightarrow X_n \le X_{3n/4} + O(n)$$

Soln $O(n)$.

$x_i \ge x_{i+1}$

(ii) Quicksort $(A_1, |A_1|)$, Quicksort $(A_2, |A_2|)$ ← x (A_1, A_2)

$T_n = T_{i-1} + T_{n-i} + O(n)$ wp $\frac{1}{n}$ (ie if x is the i^{th} smallest number)

$$X_n = \frac{1}{n} \sum_{i=1}^{n} \left[X_{n-i} + X_{i-1} \right] + O(n)$$

$$= \frac{2}{n} \left[X_1 + \cdots X_{n-1} \right] + O(n)$$

split this into 3 equal parts

$$\underbrace{X_1 + \cdots X_{n/3}}_{\le n/3 X_{n/3}} + \underbrace{X_{n/3+1} \cdots X_{2n/3}}_{\le n/3 X_{2n/3}} + \underbrace{X_{2n/3} + X_n}_{\le n/3 X_n}$$

$\frac{1}{4}$.

$\cdots$ 10

$X_0 + X_9$

$X_1 + X_8$

X_2

$\vdots$

X_5 X_{10}

Class 18 | 14/10/12

$$X_{n-i} = -c(n-1) + \frac{2}{n-1} \left[\underbrace{X_{i-1} \cdots X_{n-2}}_{} \right]$$

3/8 3/8

$$X_n = cn + \frac{2}{n} \left[X_{n-1} + \left(X_{n-1} - c(n-1) \right) \frac{(n+1)}{2} \right] \rightarrow c\,\frac{(n^2 - 2n + 1)}{n}$$

$$= cn + 2\frac{X_{n-1}}{n} + \left(1 - \frac{1}{n}\right) X_{n-1} - c\,\frac{(n-1)^2}{n}$$

$c(n - 2 + 1/n)$

$$\le \left(1 + \frac{1}{n}\right) X_{n-1} + 2c \qquad \frac{n+1}{n-i} X_{n-i+1} + 2c \left[\frac{n+1}{n-1} + \frac{n+1}{n-2} + \cdots + \frac{n+1}{n-i} \right]$$

$X_n \le \left(1 + \frac{1}{n}\right) X_{n-1} + 2c$ solve this (see lectures solve this there)

Name & Address | Office | Residence

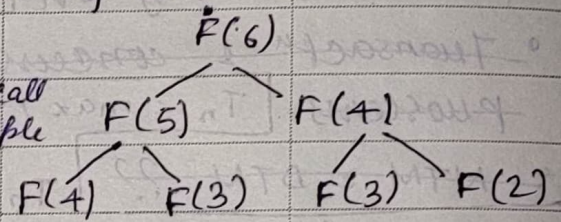
Dynamic Programming: same as divide & recursive fn call. except that we "store" some of the

(1) $F_0 = F_1 = 1$ $\quad F_n = F_{n-1} + F_{n-2}$ $\quad 1, 1, 2, 3, 5, 8, 13, 21, 34$

running time is exponential

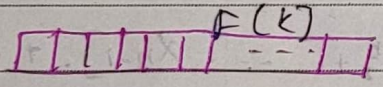
```
F(n){
  if n=0 or n=1
      return 1
  else
      return F(n-1)+F(n-2)
}
```

the same recursive call is being made multiple time



(1) DIVIDE & CONQUER ALG: the running time is high bcz we are making the same recursive call again & Again

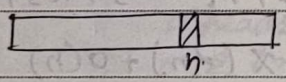
(2) what are all the possible recursive calls ?



$k \le n$ Store the values for this in a DP Table

$A[k] = F[k]$

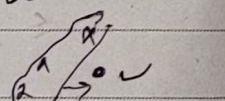
(3) Compute the entries of A in a manner such that when we want to compute entries in the order such that no recursive call is NEEDED



$A[n] = A[n-1] + A[n-2]$ if $n \ge 3$

$A[0] = A[1] = 1$ for $i = 3 \dots n$

$A[i] \leftarrow A[i-1] + A[i-2]$ save space $O(1)$

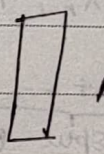


1, 2 precalculate

(2) knapsack Problem: items $1, \dots n$.

item i size s_i

profit p_i



Prob: find a ~~min~~ max profit subset of items which fit in the bag.

p_i / s_i

size 2 items

1	2	profit
100	100	

Divide & Conquer Alg.

```
int knapsack(B, 1,2, .. n) {
    if n=1
        return knapsack(B-s1, 2, .. n) + p1
        max { knapsack(B, 2 ... n)
}
```

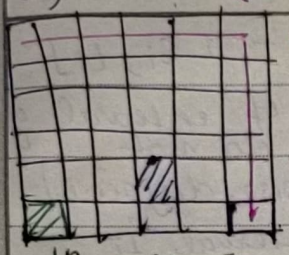
item 1 < take / not to take

if item 1 is selected

what are possible recursive call ? $(b, i \dots n)$ B_n

$0 \le b \le B$

→ i ↓↓↓ Name & Address | Office | Residence



A[b, i] will store the max profit that we can get when bag has size b and the items are i, --, n

knapsack (b, i---n) {

④ max { knapsack (b - s_i, i+1, --n) + P_i
knapsack (b, i+1, -- n)

$A[b,i] = max(A[b-s_i, i+1] + P_i, A[b, i+1])$ — only if $b \geq s_i$

base case: for i = n-1 to 1 — for b = 0 to B. — running time O(nB)

for b = 0 to B. — for i = n-1 downto 1 ④

④

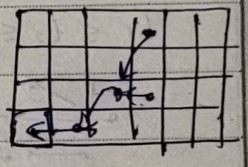
Save space only previous column is required.

Is polynomial time? $B = 10^{200}+1$ — 201 bits

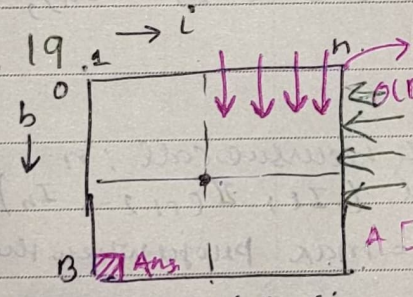
① Is this an efficient Algo? $O(nB)$: exponential time also ($n \log B$) needed.

② what if we also want to find out the actual set of item to pick

store from where your answer is coming



21/10/21 Class. 19.



save space: to compute we need only 1 column

knapsack.
O(B) space
O(Bn) time

A[i, b] = max profit $i, i+1, \ldots n$ using a knapsack of capacity B.

$A[i,b] = max(A[i+1, b], A[i+1, b-s_i] + P_i)$ ④ — $b \geq s_i$

* why it is an exponential time.

Aside. while (k < n)
if (k divides) o/p not prime
else k+1
output prime.

* not preferable.

1, 2, -- n.

s_i P_i

$\sum_{i=1}^{n} (\log s_i + \log P_i) + \log B$ — ≥ 1

② what if we want to find the actual soln (also store the soln at each table entry)

fill last col
for i = n-1 to 1
for b = 0 to B
④
trace back arrow.
slanting move: pick an item.

i) $A[i,b] = A[i+1, b]$

ii) $A[i,b] = A[i+1, b-s_i] + P_i$

i) O(B) space if we just want the max profit

ii) O(nB) space if we want the actual soln.

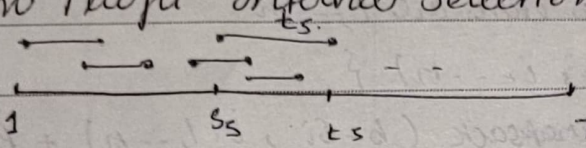
iii) O(nB) time

Q. can we get O(nB) time, O(B) space & get the actual soln?

YES (can)

Name & Address | Office | Residence

Maximum Profit Interval Selection :



$I_1 \dots I_n$ | $I_i\ [s_i, t_i]$

pick a subset of interval of max size which do not overlap (pairwise disjoint)

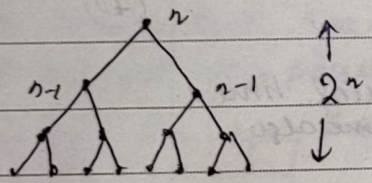
p_i : profit of interval I_i

Pick a subset of interval of max which do not overlap. (pairwise disjoint)

Select ($I_1, \dots I_n$) {

max (select ($I_2, \dots I_n$) P_1 + select (all interval among $I_2 \dots I_n$ which do not overlap with I_1))

}



Dynamic Program: what are all possible recursive call?

Select (any subset) ← 2^n

we need to combat that ⇒ sort in 1 order of t_i s_i

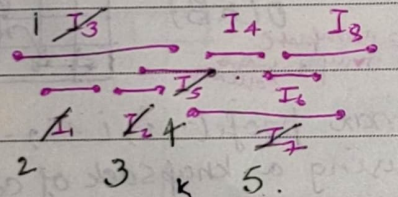
$I_1\ I_2\ I_3 \dots I_n$

$t_1 \le t_2 \le t_3 \le \dots \le t_n$

Select ($I_1, \dots I_n$) {

max (Select ($I_2, \dots I_n$), P_1 + select ($I_k \dots I_n$ (interval start after t_1) $s_i \ge t_1$

}

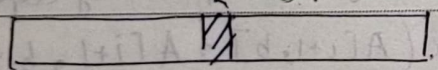


possible recursive call : n.

Select ($I_\ell, I_{\ell+1}, \dots I_n$) $A[n] = P_n$.

$A[k]$:= max profit when the interval are $I_k - I_n$

I_ℓ is the 1st interval starting after t_k.



after $k = n-1$ down to 1.

$A[k] = \max (A[k+1], A[\ell] + P_k)$

$O(n \log n)$; time complexity $O(n)$; space.

trace out actual soln ; keep a ptr where to get max^

- we have two halls instead of one hall!

Prob: give a dynamic program for this

find the max profit subset of interval such that at any time t no more than 2 such interval could

we need to make recursive call. more generic, after how much time you can take 2nd interval.

think about this

Name & Address | Office | Residence

③ you own two shop A, B $A \rightleftharpoons B$ cost C one of money.

A P_1^A P_2^A - - - P_n^A P_i^A: amount of money exceed if present at A at day i

B P_1^B P_2^B - - P_n^B P_i^B:

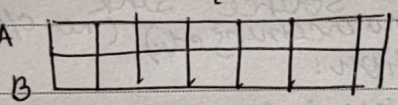
on day 1 A: A A B B B A A A // max [total profit - cost]

Max_Profit (1, 2, - - n) {, need to know when I am → more parameter

P_i^A + maxProfit (.2, -n)

⇒ max profit (A, i - - n) {.

P_i^A + max [maxprofit (A, i+1 . . n), maxprofit (B, i+1 . . n) - C]



T [A, i] ; max profit on day i (i, . . n if we start at A

T [B, i]

④ Edit-distance problem: edit - ① Add a character ② Delete a character

Given 2 strings $S_1, \ldots S_n$ & $t_1 \ldots t_m$ define EDIT DISTANCE b/w them is defined as the min # of edits in the first string to get to the second string.

A B A B A → A B A B → A A B → A C A B.

A C A B

other way of defining: we only insert characters in both the string so that they are identical.

A B C A B A | A B * A B A

A B C A B A | A * C A B *

start ⇒ edit-distance

Edit-Distance (A[1 . . i], B[1 . . j] {

min (Edit-Distance (A[1 . . i-1], B[1, . . j] + 1 OR

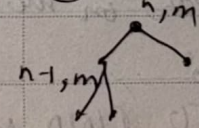
, [A[1 . . i-1], B[1 . . j-1] + 1 . .

, (A[1 . . i-1], B[1 . . j-1])

ⓐ A[n] +

ⓑ + B[m]

ⓒ A[n] == B[m]



n

exponential time: what are diff recursive call

A[1 . . i] B[1 . . j] running time = 2^n

⇒ nm (diff recursive call)

(270340)

Name & Address | Office | Residence

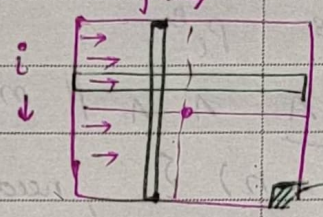
store all the recursive call $T(i,j)$ — new code ← highlighted

② $T(i,j) = \min(T(i,j-1)+1,\ T(i-1,j)+1,\ T(i-1,j-1)]$ ↑ only if $A[i]=A[j]$

for $i = 1 \dots n$

for $j = 1 \dots n$

②

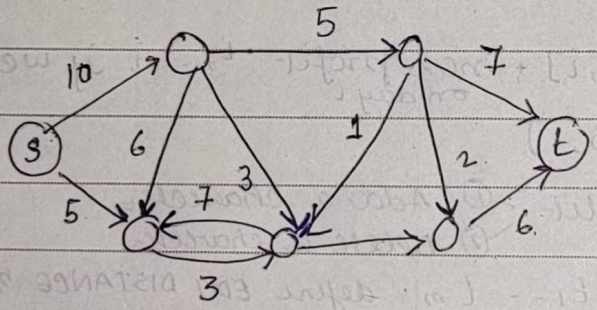


space could be modified into → putting star is 2^n obj

Maximum flow electric current-, water, traffic — greedy, D&C, DP

Problem: i/p G, e has a capacity $u_e > 0$, s t 2 vertices

source (no incoming edge) sink (no o/g edge)



Notation:

- electrical current: each edge is a wire u_e: max current that can flow on e.
- water pipes: each edge is water pipe u_e: rate at which water can flow on this pipe
- traffic: each edge is a road. u_e: how much traffic flows on each edge (rate of traffic) eg 10 trucks/hr

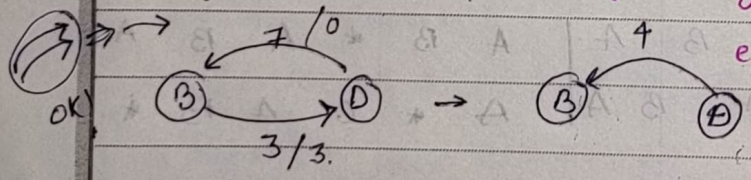
What do we want?

Def^n flow: A flow is specified a quantity f_e for each edge

i) $0 \le f_e \le u_e$ $\forall$ edge e

ii) flow conservation for every vertex v other than s or t:

$\sum_{e:\text{comes to } v} f_e = \sum_{e:\text{goes to } v} f_e$



No we don't have flow into they are not same.

Let f be flow.

flow going out of $s := \sum_{e \text{ leaving } s} f_e$

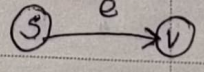
flow in coming to $t := \sum_{e \text{ enter } t} f_e$

Claim: flow coming into t = flow going out of s

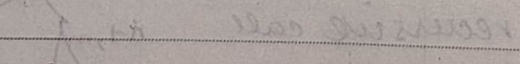
Pf: for every vertex $v \neq s,t$

$\sum_{e \in \delta^-(v)} f_e = \sum_{e \in \delta^+(v)} f_e$

Add all eq^n.



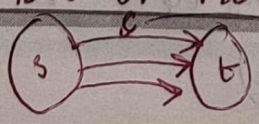
$\sum_{e \in \delta^+(s)} f_e = \sum_{e \in \delta^-(t)} f_e$



e appears on the RHS for v & on LHS of w

GOAL:

TO FIND MAX VALUE OF FLOW



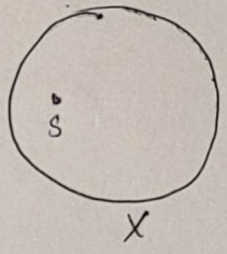
we can't send more than C flow

Def^ Cut is the set of vertices X st X contain S & does not contain t

we say that $e(u,v)$ leaves X if $u \in X$, $v \notin X$.

$e(u,v)$ enters X if $u \notin X$, $v \in X$

capacity $= \sum_{e \text{ leaving } X} u_e$



Claim max flow $\leq$ capacity (X) from s to t

Pf for every vertex $v \in X$, $v \neq s$.

$$\sum_{e \in \delta^-(v)} f_e = \sum_{e \in \delta^+(v)} f_e$$

Add all of the conditions.

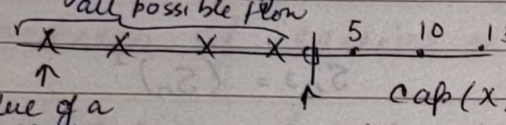
$$\sum_{e \in \delta^+(s)} f_e = \sum_{\substack{e \text{ going out} \\ \text{of } X}} f_e - \sum_{\substack{e \text{ coming into} \\ X}} f_e$$

17 SATURDAY JULY

MESSAGES	PHONE CALLS

Class 20 28/10.

every cut X gives us an upper bound on the max flow from s-t



suppose we want a flow f and an s-t cut X value of f = cap(X)
⇒ f is max flow & X is min cut

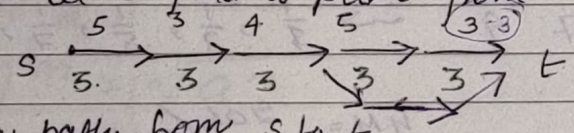
max flow ≤ min cap(X)
= X! X is an s-t cut-

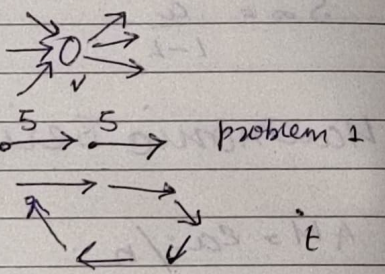
Thm: max flow from s-t = min capacity of a cut- //

Alg: start by building flow incrementally & stop when its value = capacity of a cut-

i) start w/ fe = 0 on all edges v

ii) let's find a path from s-t

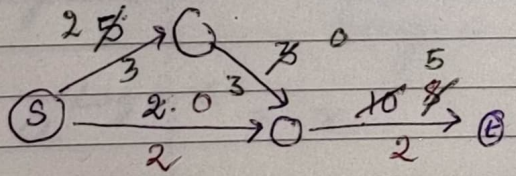




- find a path from s to t

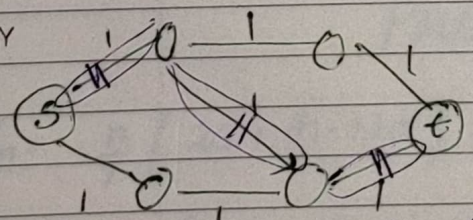
P: let C be the min capacity of an edge in P. Send C amount of flow on all edges in P.

- update the capacity of every edge to ue-fe. : remove edge if cap
- continue till no new path is found'



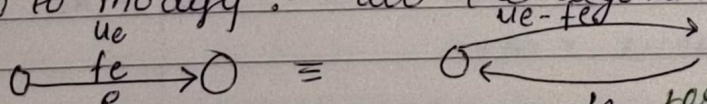
// is this going to find the max flow
NO! :(

18 SUNDAY



max flow possible = 2.
algorithm result = 1
using this edge was a "mistake."

how to modify :- all the algorithm to "undo" a mistake



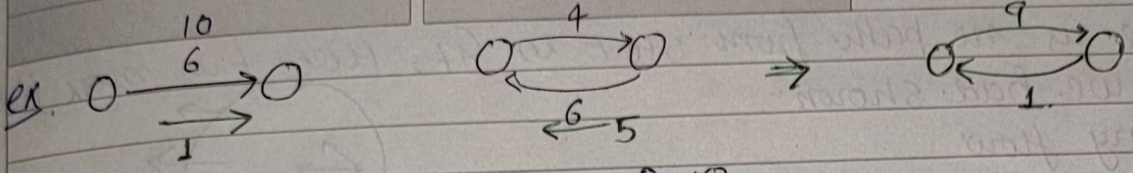
// same as updating capacity

1999

MESSAGES

PHONE CALLS

MONDAY JULY 19



Initialize $f = 0$ $G_f = G$

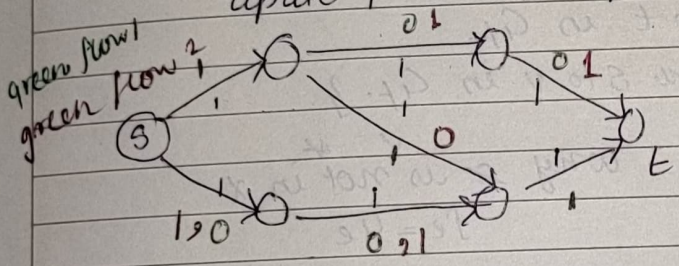
repeat:

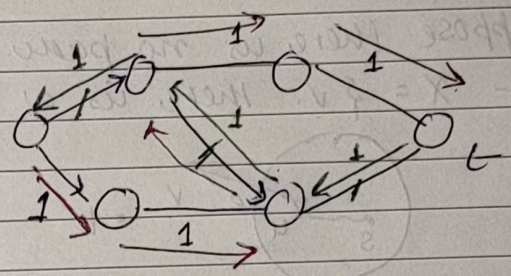
find a path P from s to t in G_f

let c be the smallest residual capacity of an edge in P

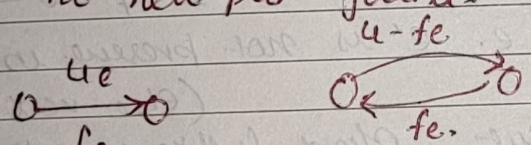
send c amount of flow along P

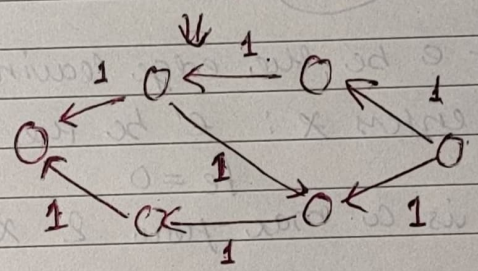
update f in G, update f.

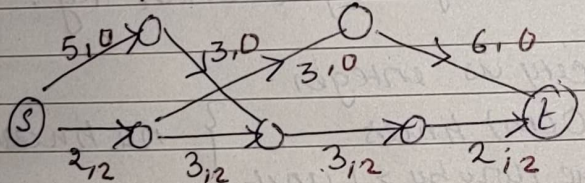


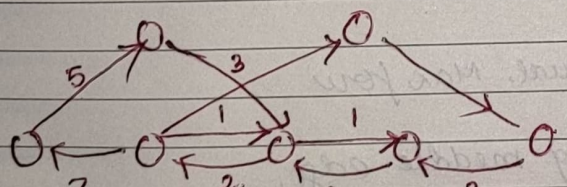


no new path found.









whenever we find a path from s-t in G_f we ↑ the flow from s-t

the process stops when there is no path from s-t in G_f

1999

JULY 7

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST 8

M	T	W	T	F	S	S	WK
30	31					1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER 9

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER 10

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER 11

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER 12

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

20 TUESDAY JULY

MESSAGES

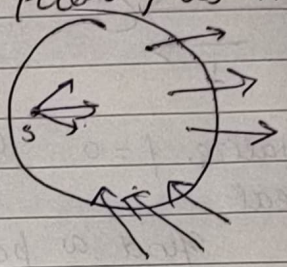
PHONE CALLS

Claim: if there is no path from s to t in G_f, then f is max flow

pf. last time we had shown:

If f is any flow

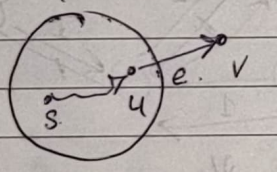
flow out of S = flow going out of X − flow coming into X



$$\text{value}(f) = \sum_{e \text{ leaves } X} f_e - \sum_{e \text{ enters } X} f_e$$

Suppose there is no path from s to t in G_f

Let $X = \{v : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$



$s \in X$

$t \notin X$.

Why e is not in X

$f_e = u_e$

i) Let e be the edge leaving X: e can't be present in G_f ($f_e = u_e$, $0 \longrightarrow 0$)

ii) e enters X: e' be the reversal of e. e' is not present in G_f (else $v \in X$)

$f_e = 0$

f is a max flow & X is min cut — Ford Fulkerson Alg.

Complexity: let's assume edge capacity is integers.

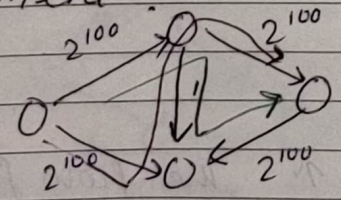
In each iteral → form G_f $O(m)$ times } nU times.

→ increase the flow by ≥ 1 unit } 2^{100} time.

$\Rightarrow O(mnU)$

Comment:

ii)



$2^{100}+1 \Rightarrow$ actual Max flow

just toggling middle edge.

Soln: - select shortest path in G_f (use BFS) = $O(mn^2)$ time

- find a path where the min residual cap is as large as possible

$O(mn^2 \log U)$ time

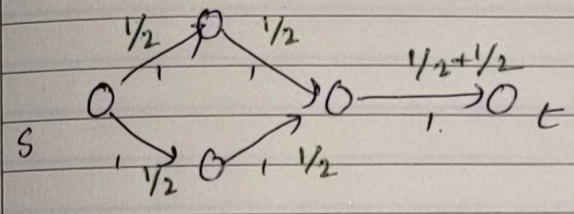
1999

MESSAGES	PHONE CALLS

WEDNESDAY
JULY 21

ii). If all capacities are integers, then the alg. always deals w/ integers ⇒ the max flow found by the alg. sends integral amount of flow on each edge. ↓

there is a max flow which only sends integer flow (integrality of max flow).



1999

JULY 7

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29

AUGUST 8

M	T	W	T	F	S	S	WK
30	31					1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33

SEPTEMBER 9

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38

OCTOBER 10

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43

NOVEMBER 11

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47

DECEMBER 12

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51