Now,

~~Of~~ Always $O(n^2)$

Lec- 26.

Randomized Algorithm ⇒ Power lies a win

Big Question: Are things really random?

quick sort

↳ Also divide & conquer

f(input, random bits)

random variable

15, 12, 3, 4, 112, 7, 6, 52

3, 6, 7, 4, 12, 15, 112, & S2

pivot

correct posn

Base case: 1 item & return that.

Correctness is clear.

Que How much time it takes?

3, 4, 6, 7, 12, 15, 52, 112.

$T(n) = T(n-1) + O(n)$

whatever ~~left or~~ smaller left, larger right.

Worst case behaviour of quicksort : $O(n^2)$

Always this (Ask)

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

↓

$O(n \log n)$

complex Algo (constant is large)

Random Bits to the Rescue!

$X_1, X_2 \ldots X_n$

random permutation

↓

$Z_1, Z_2 \ldots Z_n$

& then choose first one

↓

Behaves like $O(n \log n)$ (more later).

first,

How to generate a random permutation?

$X_1, \ldots X_n$

↓

$n!$ options (each equally likely.)

$j \in \{1 \ldots n\}$ ← n is from 2^k (sample with replacement)

2^k ↓ (0 to $2^k - 1$)

k random bits

X_j,

If n not power of 2^k.

$2^k < n < 2^{k+1}$

$[0, 2^{k+1}-1]$

↑

some no.s larger than n here {wasteful}

↓

just toss again if get such no.

$[0, 2^k - 1] \xrightarrow{\text{Add 1}} [1, 2^k]$

$n > 2^k \rightarrow > 50\%$ times we get a good (not wasteful) sample

Toss with prob = p of seeing a head

How many coin tosses do we need to see the first head.

[Geometric RV]

Expectation $\frac{1}{p}$.

$\frac{1}{p}$ samples in expectation

∴ since $> 50\%$ roughly ≤ 2 tosses are needed in expectation to get a good no

$+p + (1-p)p + (1-p)^2 p \ldots$

$p \times \frac{1}{1-(1-p)}$

$p \times \frac{1}{p} = 1$ limit

How do we get second no. of permutation?

$x_1 \ldots x_{j-1} \; x_{j+1} \ldots x_n$

choose one number

↓

choose one index from here without replacement.

retain index as $j+1$

$\{1, \ldots j-1, j \ldots n-1\}$

$\{X_j, X_i \cdots \}$

↓ At end.

$\{z_1, z_2 \cdots z_n\}$

$\| \quad \|$

$X_j \quad X_i$

very first pivot $= z_1$

$= z_{i_j}$

$z_{i_1}, z_{i_2}, \cdots z_{i_j}, z_1, z_{i_{j+2}}$

$z_{i_1} \leq z_1$; $z_{i_{j+2}} \geq z_1$

z_{i_1} is the pivot for left part

we perform algo on random permutation of first element

$z_{i_{j+2}}$ is the pivot for the right part

pivot choosen is the first elt. of random permutation

First element of each side is selected.

$T(n)$: is a random variable

lucky → $O(n \log n)$

unlucky → $O(n^2)$

R.V. → first thing to see → expectation of R.V.

$E[T(n)] = ?$

$\triangleq$ average behaviour of random variable in the limit of inf.

$\lim_{n \to \infty} \frac{\text{no. of Heads}}{n}$

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

Instead of $O(n^2)$

few random permutations needed?

$$E[X_1 + X_2 + \dots + X_j] = E X_1 + E X_2 + \dots + E X_j$$

$T(n)$: # comparisons ← the only operation we are doing are comparisons.
↳ (complicated by itself)

$$Z_{ij} = \begin{cases} 1 & \text{if } i \text{ is ever compared with } j \text{ in the algo} \\ 0 & \text{o/w} \end{cases}$$

when are i & j not compared?
when i & j are on separate sides of pivot

simplifying Assumption no no.s are same

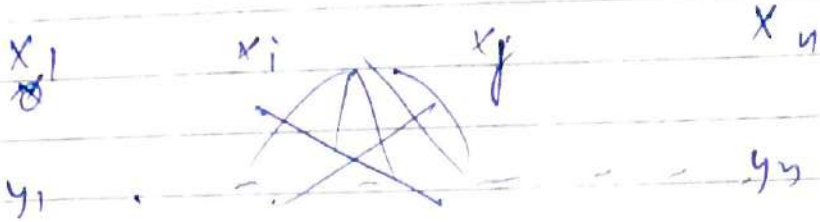
[Σx: what should we do o/w]

true sorting order: x_1 (smallest) ... x_n (largest)

Input order after random permutation. y_1 ... y_n

$x_i \quad \ldots \quad x_j$

$x_i < x_{i+1} \quad x_{j-1} < x_j$



what is prob. that x_i & x_j will ~~can~~ ever be compared during algo?

~~What~~

Case-1 Pivot is either left of x_i or right of x_j.

$\Downarrow$

x_i & x_j travel as gp.

can be compared down the line but not ~~exactly~~ to at this step.

Case-2 Pivot is b/w x_i & x_j

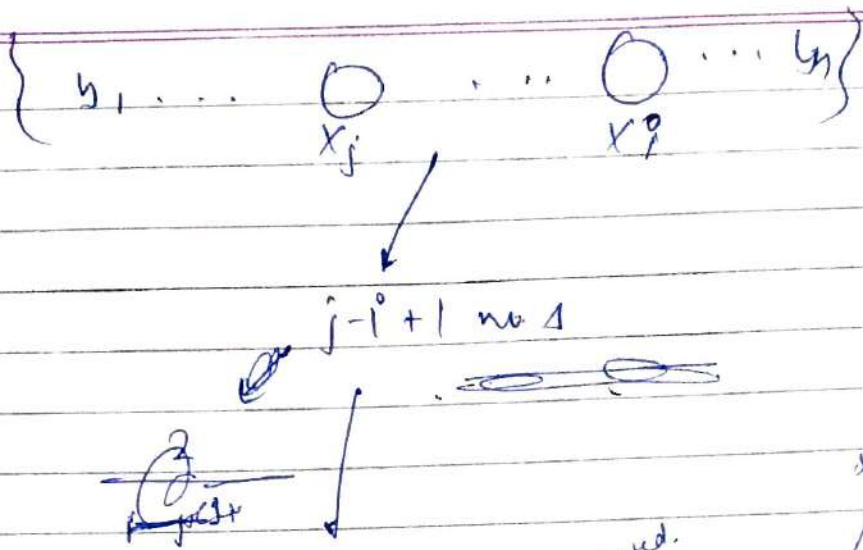
$\Rightarrow$ In future also x_i & x_j won't be compared

Case-3 (Only case when x_i & x_j will be ~~can~~ compared)

happens iff

"x_i ~~& x_j~~ x_j is chosen to be the first pivot within the subgp $(x_i, x_{i+1} \ldots x_{j-1}, x_j)$

Prob. x_i or x_j is chosen to be the pivot.



Total runtime $\Rightarrow \sum_{i=1}^{n} \sum_{j=i+1}^{n} z_{ij}$ $\quad i < j$

$$E[Z] = E\left[\sum_{i=1}^{n} \sum_{j=i+1}^{n} z_{ij}\right]$$

$$= \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[z_{ij}]$$

$$= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \qquad = O(n \log n)$$

$j - i := t$

$$= \sum_{i=1}^{n} \sum_{t=1}^{n-i} \frac{2}{t+1}$$

$j - i = t$, $r = t + 1$.

$$= \sum_{i=1}^{n} \sum_{r=2}^{n-i-1} \frac{2}{r} \leq 2 \sum_{i=1}^{n}$$

$$\leq \underbrace{2 \sum_{i=1}^{n} \sum_{t+1=1}^{n} \frac{1}{t+1}}_{\text{H.P.} \Rightarrow \underbrace{\log n}_{\text{roughly}} + \ldots} + \text{extra terms.}$$

$\leq 2 (n \log n)$

16/10/24 Lec-27

Randomized Algo continued

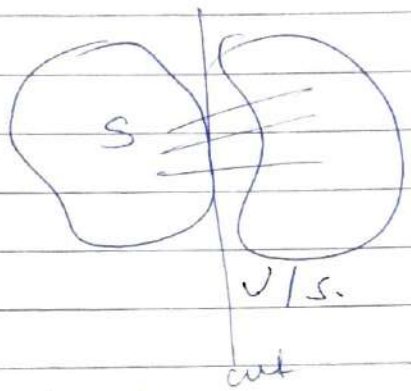
nlogn is much closer to n than n^2

Date : Page :

cstdlib → quicksort (theoretically mergesort is better but quicksort is practical)
library

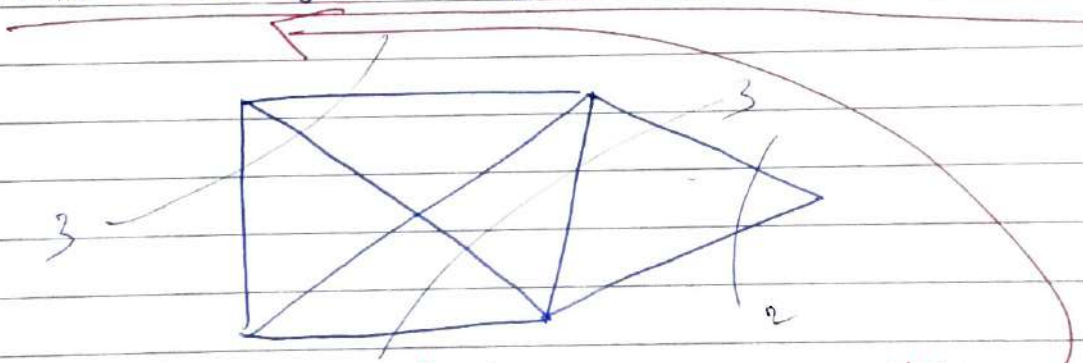
Today: Min-Cut (Undirected graphs)

find S : such that no. of edge crossings are as small as possible.



① Reliable Network

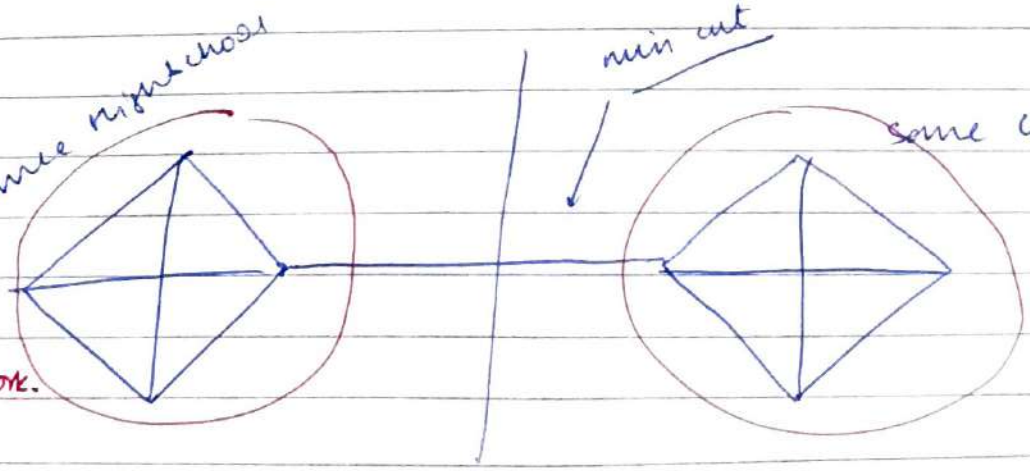
Min no. of links to take out to make disconnected



Network designer needs to maximize this so that attacker cannot harm easily

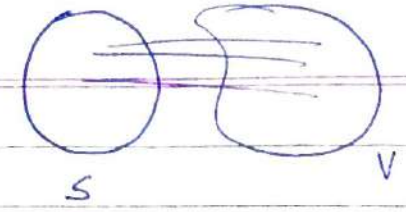
② Clustering

- FB
- community detection in social network.



Def^n [Min Cut]:

$S \subseteq V$



$$\underset{S \subseteq V}{\text{argmin}} \; |E \cap (S \times (V \setminus S))|$$

Set of edges that cross w S = cut(S)

check

Symmetry

How many cuts are possible? [For each S we get a cut]

$2^{n-2} = 2^{\Theta(n)}$

$\rightarrow 2$ — empty & full cut don't matter.

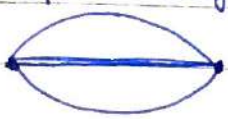
Algo: Go over each cut & see which has min no. of edges.

$\hookrightarrow 2^{\Theta(n)}$

$\therefore$ not going to scale for large n.

Enter Randomized Algo

Parallel Edges: Multiple Edges btwn 2 endpoints



- no self loops.
  ↳ diagonal entries 0

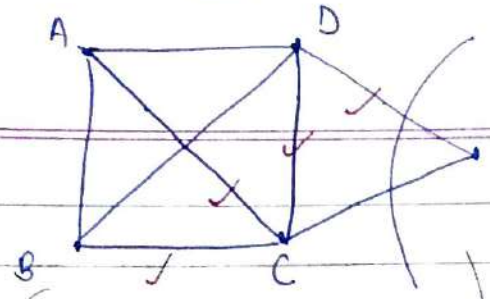
[check]

	1	2
1	0	3
2	3	0

count of edges from 1→2.

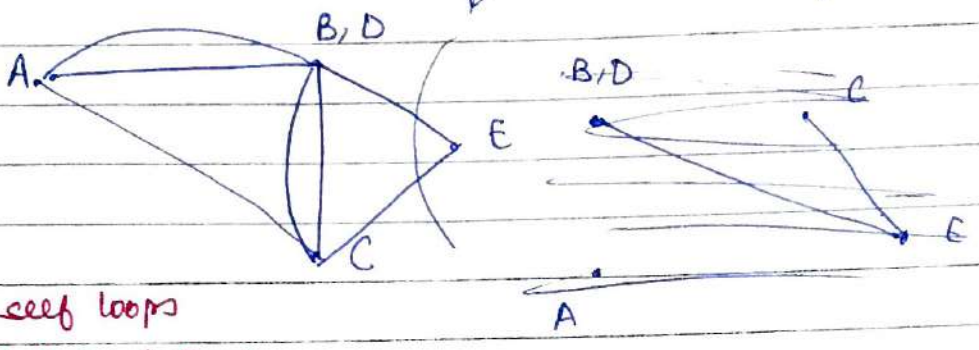
count of edges from 2 → 1

Edge Relaxations



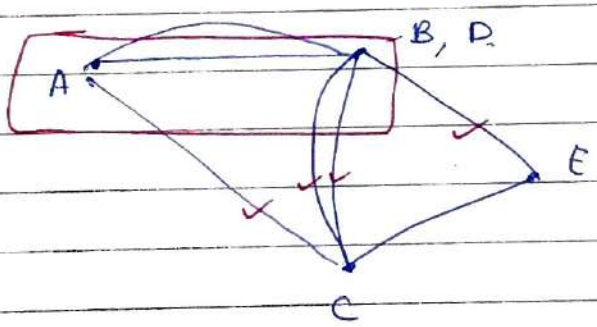
G

(B, D) edge relaxed



no self loops
but can introduce parallel edges.

How is relaxation related to min cut?



Obs: Any cut in the ~~contract~~ relaxed graph is also a cut before relaxation in the original graph. (but not the other way around)

All cuts with B on one side & A on another will count BA as an edge ~~in cut~~ crossing cut.

but if BA is relaxed then in relaxed one BA will not be part of cut.

- Relax operation cannot reduce the min cut size!

Relaxation is naturally related to the problem

as we keep on ~~gluing~~ gluing until we reach S.

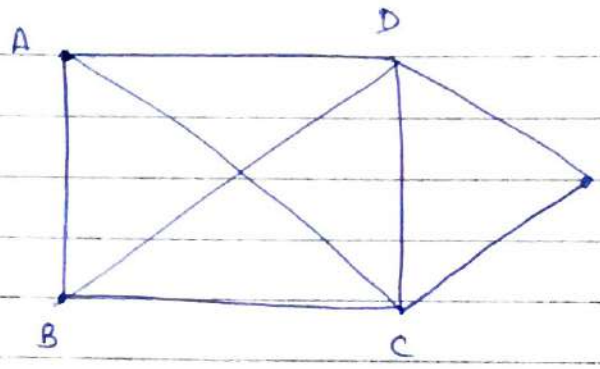
Algorithm

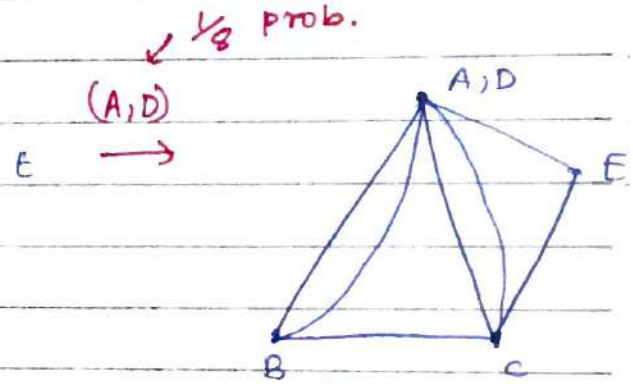
(1) Pick one edge (X, Y) at random.

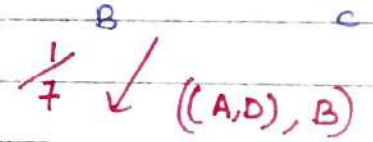
(2) Relax that edge.

(3) go back to (1) until only 2 vertices left

iterate (n-2) time

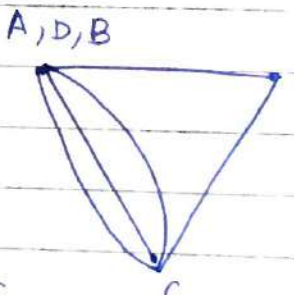


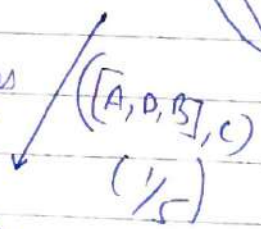


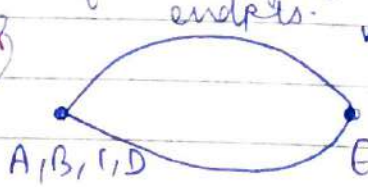


Donot relax any neighbour of E

First make those edges which ~~have no chance of being~~ won't be ~~doubled. relaxed~~ ~~are not involved in~~ out of joined nodes. in older graph.





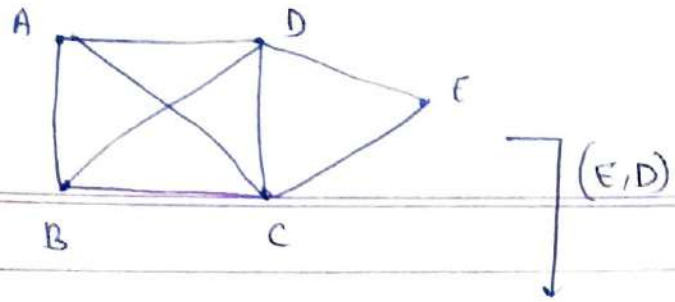


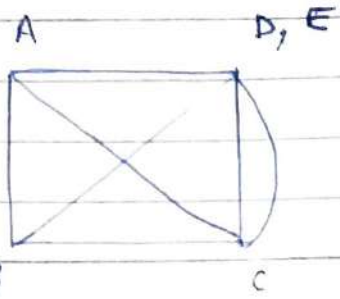
do not have vertices of relaxed edge as endpts.

Successful run! Since cuts have backward compatibility

i.e, cut is related is cut in prev.

⇒ final cut also present in original.





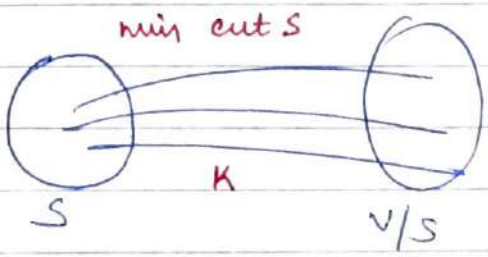
~~Claim:~~

Claim: The cut returned by the algo is the min cut with at least some p probability where $p > 0$

Pf: Consider any mincut set cut (S)
set of edges that crosses to V/S.

~~We'll~~ We'll show that no edge in cut (S) is picked by the algo with prob. $\geq p$

First step 0 (n vertices)
Let K be the size of min cut
$\Rightarrow$ degree $\geq K$ for each vertex



*How many edges are there in total in graph?

~~dense → high degree~~

min-cut size ⇒ K ∴ degree of each vertex ≥ K

(o/w S could be that node)

∴ # edges ≥ $\frac{n \cdot K}{2}$

Bad case: choose these K edges

Prob: (any edge in cut(S) is not relaxed) ≤ $\frac{K}{\binom{nK}{2}}$ = $\frac{2}{n}$

second step (n-1 vertices)

relaxed graph: (n-1) vertices

By prev. obs relaxed graph also has min cut size ≥ K

prob(any edge ~~prob~~ in cut(S) is ~~not~~ relaxed in 2nd step | ~~eq 1st step~~ | none relaxed in step 1) ≤ $\frac{K}{\frac{(n-1)K}{2}}$ = $\frac{2}{n-1}$

This also holds for any ith step

↳ n-i+1 vertices.

prob(any edge in cut(S) is relaxed in ith step | ~~not relaxed~~ none relaxed in step i-1) ≤ $\frac{2}{n-i+1}$ $n-2 \geq 1 \geq 1$

E_i ← the event that no edges in cut(s) is relaxed at step i

$$Pr[E_1] \geq \left(1 - \frac{2}{n}\right)$$

F_i ← $E_1 \wedge E_2 \wedge \ldots \wedge E_i$

the event that no edges in cut(s) is relaxed upto step i

$$Pr[F_{n-2}] = Pr[E_{n-2} \wedge F_{n-3}] \quad \text{Bayes Rule}$$
$$= Pr[E_{n-2} \mid F_{n-3}] \cdot Pr[F_{n-3}]$$

↓ Play the same game

$$= Pr[E_{n-2} \mid F_{n-3}]\ Pr[E_{n-3} \mid F_{n-4}] \ldots Pr[F_1]$$
$$Pr''[E_1]$$

$$\Rightarrow Pr[E_j \mid F_{j-1}] \geq \left(1 - \frac{2}{n-j+1}\right)$$

not independant
∵ cut edges were relaxed then our calcn would be wrong.

$$Pr \geq \left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{2}{n-j+1}\right) \ldots \underbrace{\left(1 - \frac{2}{3}\right)}_{\frac{1}{3}}$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right) \ldots \left(\frac{1}{3}\right)$$

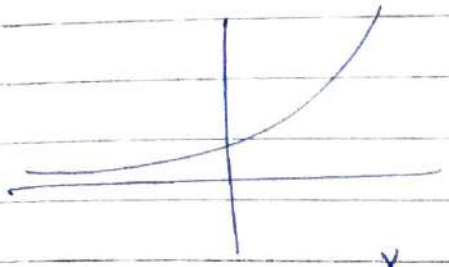
$$\geq \frac{2}{n(n-1)}$$ ← cut(s) is still preserved at end of algo with prob at least this

not too bad!
Inverse poly!

we'll improve the success prob by repeating this algo t times and take the min.

$$\Pr(\text{All of them go bad}) = \left(1 - \frac{2}{(n)(n-1)}\right)^t \leq \text{small}$$

$$\leq e^{-\frac{2}{n \cdot n-1} t} \leq \delta$$



$e^x \geq 1+x$

$$t \geq \ln \frac{1}{\delta} \frac{(n)(n-1)}{2}$$

For 99.9% → $\delta = 0.001$
then get t

$$30 \frac{(n)(n-1)}{2} = 15 n^2$$

$$t = O(n^2)$$

ef we need to implement relax^n.

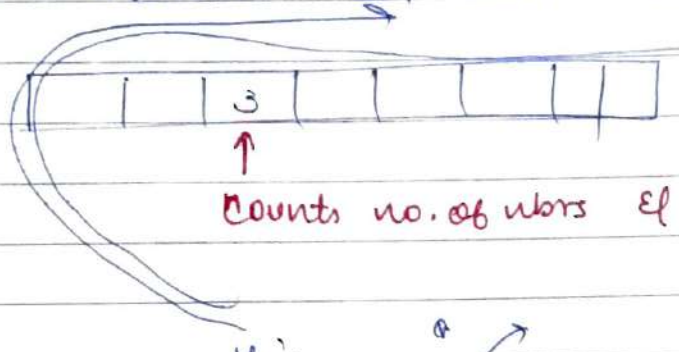
lec-28

- Revise pseudo code of algo for coding Test.

o/p

- one sided errors : always more than min size.
  - ↳ can improve accuracy by repeating

- we'll maintain adj. matrix & deg. array.



"check"

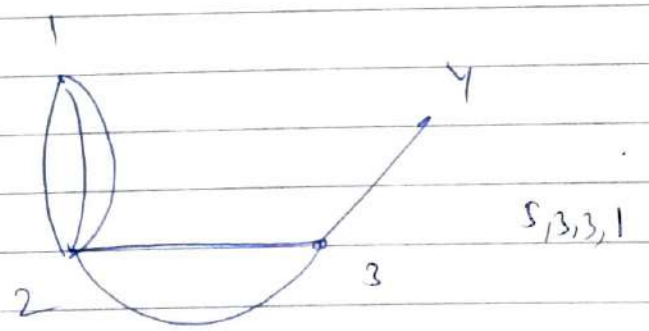
counts no. of nbrs & includes parallel edges too

this

We need a good data structure we already know algo

- How do we select an edge at random?

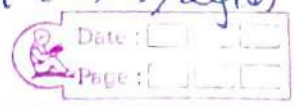
- $\frac{15}{\left(\sum_{v} \deg_v\right)}$ 1, 2, 3, 4, . . , 15, 7, 9



selected, $2 \rightarrow \frac{5}{12}$ Select 1 with 3/5

3 with 2/5

- Select vertex with prob. prop to $\deg(v)$
- select nbr w with prob prop to $wt(u,v)/\deg(v)$



$$\frac{2}{\sum \deg(v)} = \frac{2}{2 \cdot m} = \frac{1}{m}$$

(2 ← each unselected edge)

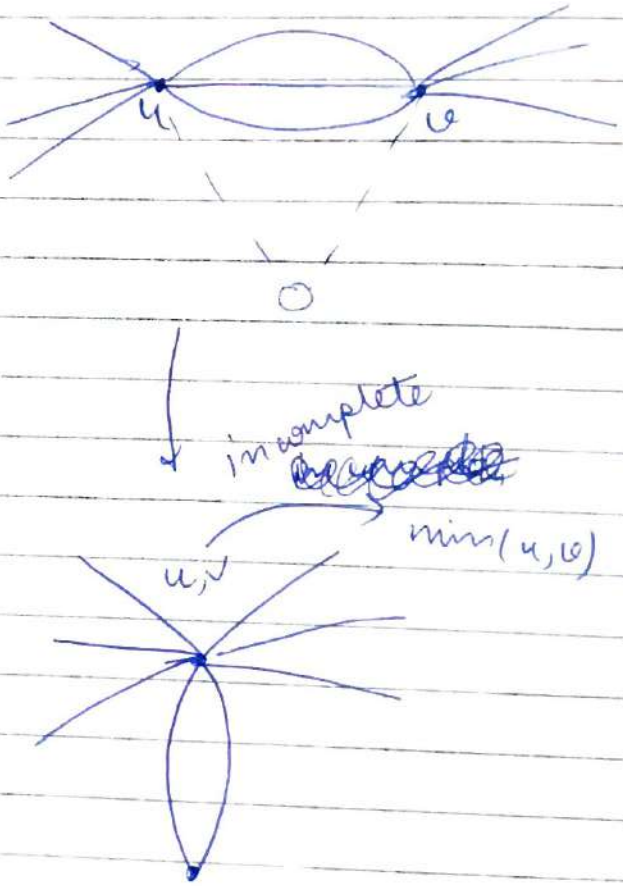
$\Rightarrow$ each edge is selected with unifm prob

How to perfm $\text{relax}(u,v)$?

w.log $u = \min(u,v)$

$\deg(u) \leftarrow \deg(u) + \deg(v) - 2\,wt(u,v)$

$\deg(v) \leftarrow 0$



for every $w \in V$

$wt(u,w) \leftarrow wt(u,w) + wt(v,w)$

$wt(w,u) \leftarrow wt(u,w)$

$wt(v,w) \leftarrow 0$

$wt(w,v) \leftarrow 0$

T.C.

n = # of vertices

$O(n)$ update — $O(n)$ # iterations — $O(n^2 \log 1/\delta) = O(n^4 \log 1/\delta)$ repeating

Correctness: $(1-\delta)$ prob. You'll return the min cut correctly

David K enger.

In randomized algo, we ask another question

Q4: Can we avoid randomness?

Maxflow min-cut

Randomized Algo

Las-Vegas Algos
T.C. is random variable
& O/P is always correct
Ex: Quick Sort

Monte-Carlo Algos
T.C. is same but O/P may be wrong (it is in RV)
Ex: Min-Cut

Another Monte Carlo Algo

Polynomial Identity Testing

$$P(x_1, x_2, x_3) = -2x_3^2(x_1+x_2)(x_1-x_2) - x_3^2 + (x_1^2 + (1+x_2)(1-x_2))x_3^2 \equiv 0$$

Input s.t.
- evaluation is easy
- exp^n is diff.

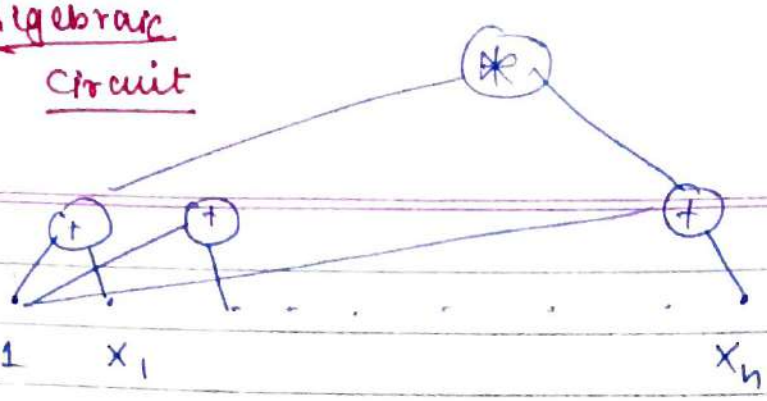
$(1+x_1)(1+x_2) \ldots\ldots (1+x_n)$

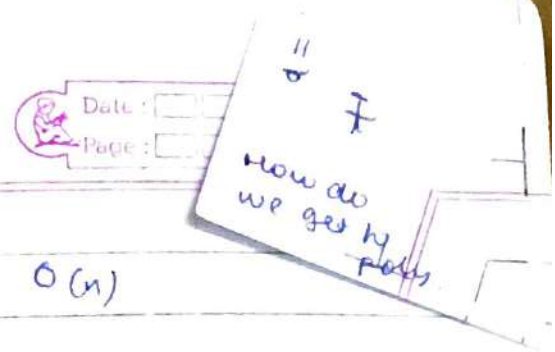
2^n terms in this exp^n → $O(2^n)$ to write exp^n

$O(2^n)$ to multiply 2.

~~Can't multiply~~

Algebraic Circuit





Deg(P) : max deg. of any monomial in poly

Ex.

$P(x) = x^3 + 3x^2 + 4x + 5$

$\deg(x_1^{d_1} x_2^{d_2} \dots x_n^{d_n}) = \leq d_i$

(monomial)

check

Assumption : $P(x_1, x_2 \dots x_n)$ is of degree $\leq d$.

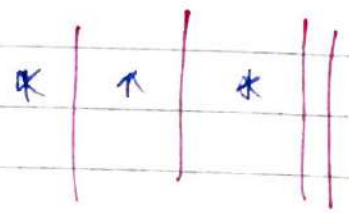
~~Cancellation~~

Trivial Algo: Just do cancellation!

≡ How many monomials can P have

$x_1^{d_1} \dots x_n^{d_n} : d_1 + d_2 \dots d_n \leq d$

Combinatorics

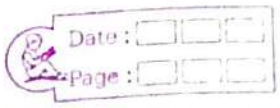


$d+n-1$ many monomials $\approx \geq n^d$.

C_d

$q = n/2$

the t.c. can be as large as $n\Theta(n)$



For univariate → just evaluate on $d+1$ pts.

multivariate → we need ind^n of randomization also

Procedure PIT

check

1. Choose a set S of inputs large enough
2. Choose a random input from S: $(r_1 \dots r_n)$
3. Evaluate $P(r_1 \dots r_n)$
   - if 0: yes
   - if $\neq 0$: $P \not\equiv 0$

can repeat like karger (from 2?)

Obs: If $P \equiv 0$ then the algo is always correct.

Remark: $P(r_1, \dots r_n) = 0$ even if $P \not\equiv 0$ when $(r_1 \dots, r_n)$ is a root of this polynomial P.

We'll show if $|S|$ is large prob. of above event happening is small

claim let $P \neq 0$ then

$$Pr[P(r_1 \ldots r_n) = 0] \leq \frac{\deg(P)}{|S|}$$

Pf: Induction on n, keeping deg. fixed.

Base ($n=1$)

# of roots P can hv $\leq \deg(P)$

$\therefore$ prob that we select a root $\leq \frac{\deg(P)}{|S|}$

In case of multivariate polynomial: there can be only many roots even when deg fixed.

I.H. This is true for $(n-1)$ variate poly.

I.S. Fact also holds for n variate $P(X_1, X_2) = X_1 - X_2$

$P(X_1, \ldots X_n) \longrightarrow$ wlog, let X_1 be a variable (α, α) that contributes to a non-zero monomials

$= X_1^{<K} \cdot Q(X_2 \ldots X_n) + R(X_1, X_2 \ldots, X_n)$

K: Biggest pwr of X_1

$X_1^{<K}$

for finite fields.

$\downarrow$

infinite soln $< |S|$.

$(r_1, r_2, \ldots, r_n) \longrightarrow (r_2, \ldots r_n)$

ignore, also a random input from S

$P(x_1, r_2, \ldots r_n) = x_1^K Q(r_2 \ldots r_n) + R(x_1, r_2 \ldots r_n)$

Q is a poly : $(n-1)$ variables.

Great for efficient fields

$Q \neq 0 \therefore X_1$ contributes.

Date: Page:

$$\Rightarrow \Pr\underbrace{[Q(r_2 \ldots r_n) = 0]}_{\text{Event } E} \leq \frac{\deg Q}{|S|}$$

We'll think of choosing $n-1$ elts. from S instead of n)

Condition on the event that $\bar{E}$ happens.

$P(r_1, r_2 \ldots r_n)$ becomes univariate poly of degree $k \geq 1$ in X_1.

Now plug back r_1

$$\Pr\underbrace{[P(r_1, r_2 \ldots r_n) = 0]}_{F \mid \bar{E}} \leq \frac{k}{|S|} \leq \frac{\deg(P) - \deg(Q)}{|S|}$$

Now its a matter of counting

$$k + \deg(Q) \leq \deg(P)$$

$$k \leq \deg(P) - \deg(Q)$$

L-29

$P(X_1, \ldots X_n) = X_1^k Q(X_2, \ldots, X_n) + R(X_1 \ldots X_n)$

$X_1 < k$

$(r_1, r_2 \ldots r_n) \longrightarrow (-, r_2 \ldots r_n)$

E $Q(r_2 \ldots r_n) = 0 \Rightarrow \leq \frac{\deg(Q)}{|S|}$

$F|\bar{E}$: $r_1^k Q(r_2 \ldots r_n) + R(r_2 \ldots ?)$ mult.

E + F: Univariate = 0 $\leq \frac{\deg(P) - \deg(Q)}{|S|}$

Good Event: does not evaluate to 0

$\bar{F} \wedge \bar{E} \Rightarrow P(r_1 \ldots r_n) \neq 0$

F : univariate evaluates to 0

$P(r_1 \ldots r_n) = 0 \Rightarrow F \vee E$

$A \Rightarrow B$
$Pr[B] \geq Pr[A]$

$Pr[P(r_1 \ldots r_n) = 0] \leq Pr[F \vee E]$

$1 - Pr[\bar{F} \wedge \bar{E}]$] Bayes.

$1 - Pr[\bar{E}]\, Pr[\bar{F}|\bar{E}]$

$\geq$ $\geq$

$Pr[E] \leq \frac{\deg(Q)}{|S|}$ $Pr[\bar{F}|\bar{E}] \geq 1 - \left(\frac{\deg P - \deg Q}{|S|}\right)$

$Pr[\bar{E}] \geq 1 - \frac{\deg(Q)}{|S|}$

$\therefore Pr[P(r_1 \ldots r_n) = 0] \leq \frac{\deg(P)}{|S|} - (\quad)$

$\leq \frac{\deg(P)}{|S|}$

The induction step holds!

Graph Theory Algo

We have one sided error

∵ $P \equiv 0 \Rightarrow$ never mistake

o/w $\Rightarrow$ commit a mistake w.p. $\leq \frac{\deg(P)}{|S|}$

repeat r times:

$$e^x \geq 1 + x$$

$$\left(1 - \frac{\deg(P)}{|S|}\right)^r \leq e^{-\frac{\deg(P) r}{|S|}} \leq \delta$$

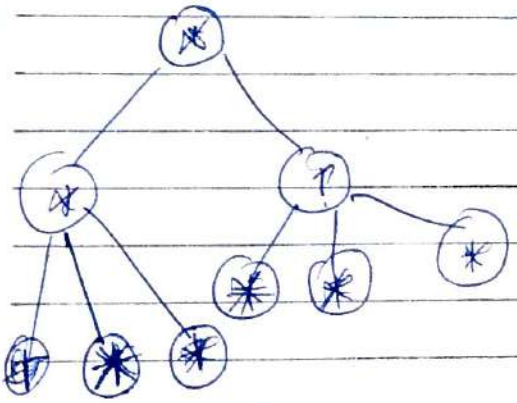
$$r \geq \frac{|S|}{\deg(P)} \left(\ln \frac{1}{\delta}\right)$$

∴ T.C. $O\left(\ln \frac{1}{\delta} \frac{|S|}{\deg(P)}\right)$ many evaluations

Better than 2^n many monomials to evaluate!

$O(n)$

$|S| \geq 2\deg(P)$



$\geq 1-\delta$ if $P \neq 0$

always if $P \equiv 0$

Randomized Algo

No deterministic algo for PIT known!

Randomized Data structures

Hashing

- insert
- search
- delete

$O(\log n)$ n = size of database.

AVL Tree

Randomized

insert	$O(1)$
search	$O(1)$
delete	$O(1)$

expected T.C.

Las Vegas

Cyber Security:

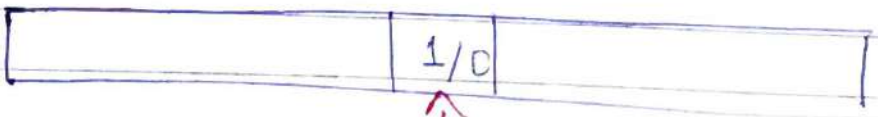
blacklist of IP addresses: malicious addresses

∴ insert, delete, search are imp.

Naive

1) ↳ ~~list of~~ Linked list (search & delete are $O(n)$ ↓ problem)

2) entry of each IP address (32 bit address ⇓ 2^{32} size array)

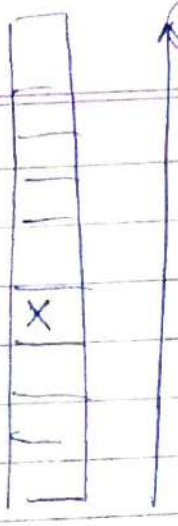


entry for each IP address.

(huge wastage of space)

h(x) → index of another array blacklist

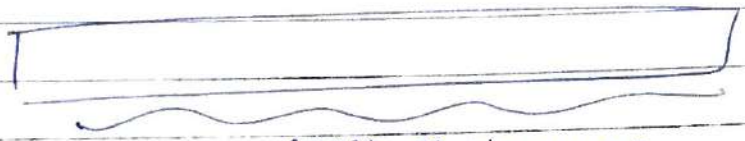
↓ ip address



weak

Hope : 1000 IP ~~addresses~~ addresses are bad.

Ideally



array of size 1000

Check

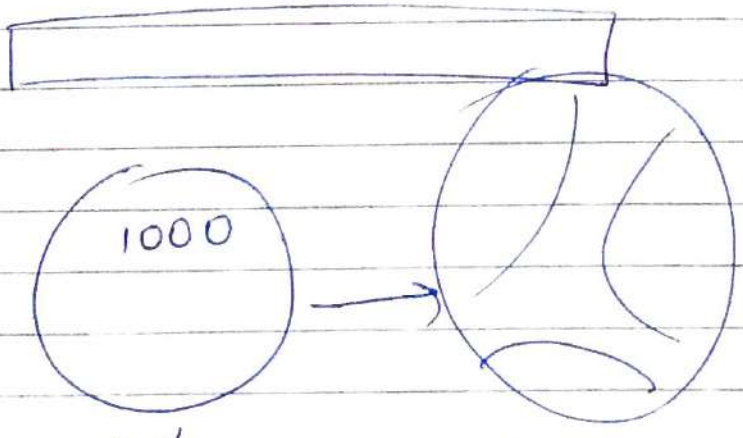
1000 → 1000 mapping

↓ No

~~Space~~

We don't ~~want~~ know when IP address will come.

Soln? →



We have to use a deterministic f^n : h : search.

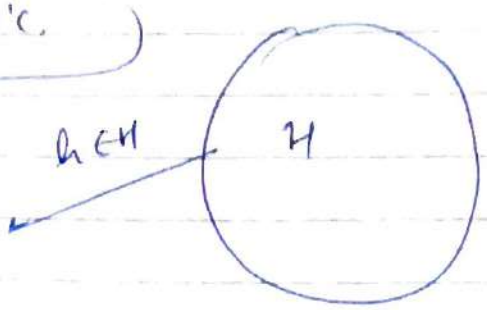
we have a large class of functions. H

~~but a large no. of~~ of

choose one

f^n randomly.

h is deterministic.



If hash f^n is deterministic ⇒ any IP address may come

⇓

~~many~~ Hash collisions designed by adversaries.

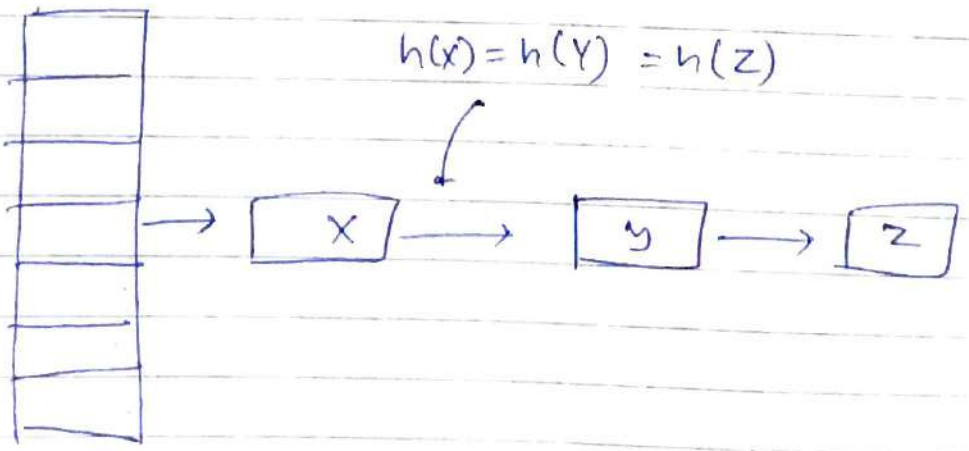
$2^{32} \longrightarrow 1000$ .. ~~collision~~

$2^{32}/1000$?

Collision

$X \neq Y \xrightarrow{\text{but}} h(x) = h(y)$

Adj. list Kind of



Properties of Hash functions

(1) minimise collisions

↳ o/w worst case $O(n)$ search ∈p delete ∈p insert → worse than AVL

(2) easy to compute

$h: U \longrightarrow \{1, \ldots, M\}$

huge size; $|M|$ is small

∈p h has above properties

Gold standard for hashing

Universal Family of Hash fⁿ (Universal Hashing)

$\mathcal{H}$: be a big class of hash functions, each hash function mapping $U \longrightarrow [M]$

Defⁿ [Univ. Family]: A family (i.e., set) of hash functions $\mathcal{H}$ is called universal if $\forall\, x, y \in U$

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq 1/M$$

$\max \leq 1/M$

Why $1/M$?

(If random fⁿ then prob. x ∈l y map to same)

$\frac{1}{n} \cdot \frac{1}{M}$

	a	b
h_1	0	0
h_2	0	1

always collide → $Pr = 1$

not collide → $Pr = 0$

$M = 2 \Rightarrow 1/2$

	a	b
h_1	0	1
h_2	1	0

	a	b	c
h_1	0	0	1
h_2	1	1	0
h_3	1	0	1

$h : U \to \{0,1\}$

$\{a,b,c\}$

Once we choose h that h is fixed for all IP addresses

(i) $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$ ∴ uniw

(ii) $\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 0 \leq \frac{1}{2}$

(iii) $x = a, y = b$

$\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 0 \not\leq \frac{1}{2}$

	a	b
h_1	0	0
h_2	1	0
h_3	0	1

$\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 0 \leq \frac{1}{2}$

	a	b
h_1	0	0
h_2	1	1

$\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 \not\leq \frac{1}{2}$

Why are univ. families interesting?

claim: Let $x_1, x_2 \ldots x_n$ be any seq of items.

(think of n inserts back to back)

If we choose $h \in H$ randomly where H is universal.

↳ $U \to [M]$

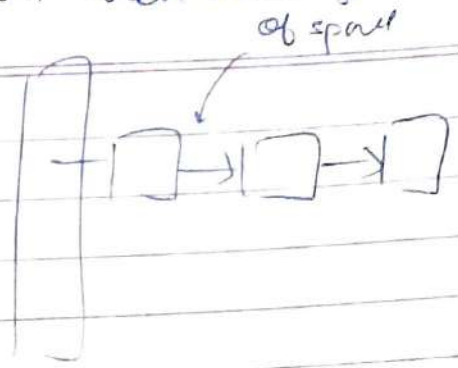
$$\mathbb{E}[\#\text{ collision}] < \frac{n}{M}$$

$\mathbb{E}$: w.r.t random choice of f^n.

If we know no. of requests is n then we can choose M s.t. Expected no. of collisions < 1

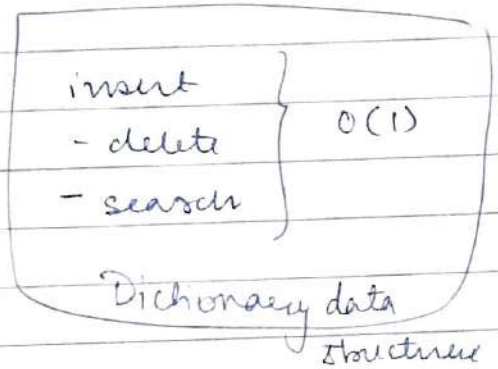
Lec - 30

- if lots of collision then wastage of space



- we want O(1) collisions. ⇒ $\mathcal{H}$: family of hash functions

-

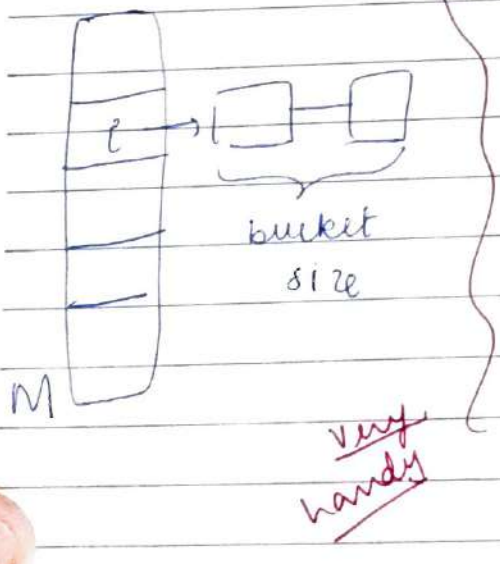


- claim: Let $\mathcal{H}$ be a universal family that maps from $U \to [M]$.
Let $x_1, \ldots x_N$ be any set of N items.
If we choose any $h \in \mathcal{H}$ randomly, then the bucket size ~~is not too big~~ for any $i \in [M]$ in expectation is at most N/M

$$\mathbb{E}\left[\#\text{ collisions at any particular } i\right] \leq N/M$$

vary bucket size acc. to N.

to get req. no. of collision



very handy

Sample ~~for~~ Space for bucket size is $0 \cdots M$.

Pf:

Fix i arbitrarily

$C_{ij} \leftarrow 1$ if $h(x_i) = h(x_j)$

$\leftarrow 0$ o/w

$j \in [M]$ // $\{1, \ldots M\} \setminus \{i\}$

Bucket size at $i = \sum_{\substack{j=1 \\ j \neq i}}^{M} C_{ij}$

$\mathbb{E}$(Bucket size at i) $= \mathbb{E}\left[\sum_{j \neq i} C_{ij}\right]$

$= \sum_{j \neq i} \mathbb{E}[C_{ij}]$

$= \sum_{j \neq i} \Pr[C_{ij} = 1]$

$= \sum_{j \neq i} \Pr_{h \in \mathcal{H}}[h(x_i) = h(x_j)]$

randomness is due to $\mathcal{H}$ & randomly chose.

$\leq \sum_{j \neq i} 1/M$ [since $\mathcal{H}$ is univ]

$= \frac{N-1}{M} < N/M$

Corollary Let $O_1, O_2, \ldots O_N$ be a seq^n of N insert/delete/search operation

where each O_i uses item x_i.

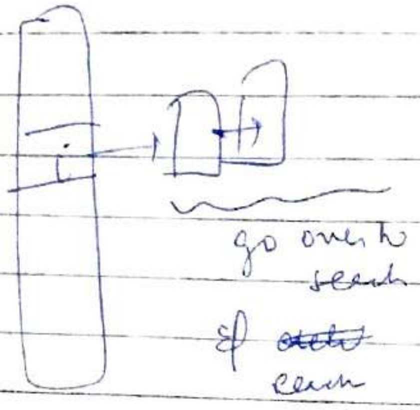
Then total cost for each operation is $O(N + N^2/M)$

Pf: Let t_i be the time taken for O_i. then total time $T = \sum_{i=1}^{N} t_i$

$$E[T] = E\left[\sum_{i=1}^{N} t_i\right] = \sum_{i=1}^{N} E\, t_i$$

$$= \sum_{i=1}^{N} E\left[(\text{bucket size at } i) + c\right]$$

(for performing the operation)



go over to search ~~of~~ each key e this

$$= O(N) + $$

in worst case is all operations are insert $\downarrow$ N

$$\leq O(N) + \sum_{i=1}^{N} i/M$$

$$= O(N) + O(N^2/M)$$

What is the choice of M?

since ~~of~~ it is at least $O(N)$ time complexity
$M = \Theta(N)$

$O(N + N^2/M) = O(N)$ for N operations

$O(1)$ for 1 operation. Amortized now?

Assumption: Computing Hash fn takes constant time!

(64 bit architecture?)

can we get universal family? Yes

construction — | Assumption U is $\mathbb{Z}$

$h : U \longrightarrow [M]$

convert to Number

10 character IITK ID

Whoa.

$z \in U$ & z is a perfect power of 2. z

$|U| = 2^u \equiv$ U can be encoded by U bits

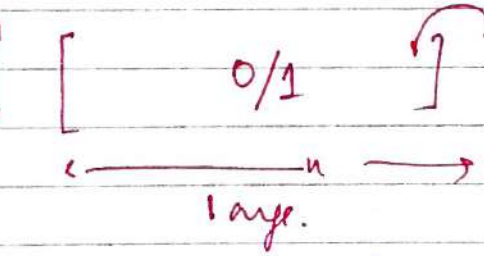
$M = 2^m \equiv$ M can be encoded by m bits.

little bits

$h : 2^u \longrightarrow 2^m$

H is defined as:-

Fix a matrix A of dimension: $m \times u$.

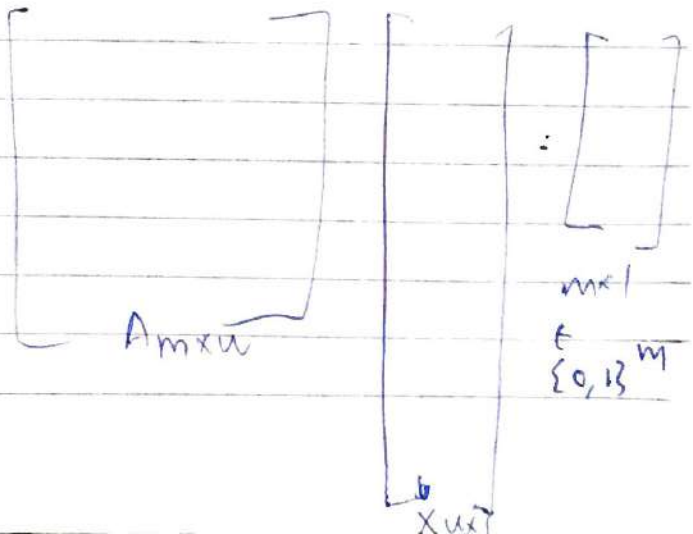


2^{mu} many matrices.

Each A matrix is a diff f"

$h_A(x) \in \{0,1\}^u$

u bit strings



multiplications & add^n
are mod 2.

1, 0 : 1+1 = 0

Date : Page :

For each A, A & h_A defines a mapping from $\{0,1\}^u$ to $\{0,1\}^m$

for each choice of A ⇒ diff^n mapping.

$\mathcal{H}$ consists of h_A's for all possible A's.

$$|\mathcal{H}| = 2^{mu}$$

Claim: $\mathcal{H}$ is universal

Example: 2 bit strings is range & domain

$U = \{00, 01, 10, 11\}$ $u = 2$

$m = 1$

Range = {0,1}

dimension 1×2

[0 0] [0 1] [1 0] [1 1]

4 choices of A

$h_{00}, h_{01}, h_{10}, h_{11}$

	00	01	10	11
h_{00}	0	0	0	0
h_{01}	0	1	0	1
h_{10}	0	0	1	1
h_{11}	0	1	1	0

collision

collision

$[1\ 0]\begin{bmatrix}1\\0\end{bmatrix}$

$= (1+0) \bmod 2 = 1$

Very hash f^n is univ.

Verify : This is universal

4 C 2 pairs , Pr (collision) $= 1/2$

---

claim: The hash f^n is universal

Pf: Fix x, y arbitrarily

$$\Pr_{h \in H}[h(x) = h(y)]$$

$$Ax = Ay \mod 2.$$

$(\ )_{m\times 1} \longrightarrow (\ )_{m\times 1}$

Since $x \neq y$, $z := x - y \neq 0$ — $z_i \neq 0$ for some i.

$$= \Pr_{A \in \{0,1\}^{m\times n}}[Az = 0 \mod 2]$$

$$\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}$$

$$= \Pr_{A \in \{0,1\}^{m\times n}}[Az \equiv 0 \mod 2]$$

m rows $\begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$ $\begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{pmatrix}$ $\begin{pmatrix} z \\ \vdots \\ z \end{pmatrix}$ $u \times 1$

$A_i \in \{0,1\}^u$

$= \begin{pmatrix} A_1 \cdot z \\ A_2 \cdot z \\ \vdots \\ A_i z \cdot z \\ \vdots \\ Az = z \end{pmatrix}$

since choosen randomly

$\Downarrow$

$A_i z \cdots = 0$ ~~independent~~

$\rightarrow$ independent.

check

$$= \prod_{j=1}^{m} \left[\Pr_{A_i \in \{0,1\}^u} \left[A_j \cdot z = 0 \mod 2 \right] \right]$$

$\uparrow$ for any particular i

~~Any particular~~

$A_{i_1} z_1 + \cdots A_{i_u} z_u = 0 \mod 2$

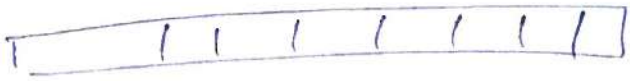
$x \neq y$ $\exists$ some j s.t. $z_j \neq 0$

$A_{ij} z_j \equiv A_{ij} z_j + \cdots$ ~~A_{ij}~~ $A_{i(j-1)} \cdot z_{j-1}$

$+ A_{i(j+1)} z_{j+1}$

$\rightsquigarrow \cdots$ ~~$A_{i_1} z_1 \mod 2$~~

$A_{i_u} z_u \mod 2$



Suppose sampling z_j at end.

↓

At this pt only varying is A_{ij}

z_j being 0 → 50%. Check

∴ this eqⁿ holds ~~for~~ with prop = $1/2$

$$= \frac{1}{2^m} = \frac{1}{M} \square$$

∴ this family is ~~uniform~~ universal.

construction -2 u → bit strings.

( ) ( ) ( ) ( )

$u = 32$ (IP address)

↓

4·8

~~8 bit chunk~~

$a_1, a_2, a_3, a_4 \in$ ~~256~~ $[0, 255]$

$V \to \{0, 1, \ldots K-1\}^l$ $K = 256$

$l = 4$

$M \leftarrow$ prime ~~no.~~ no. $> K$ $l = 9$ so

$U \to \{0, 1, \ldots, k-1\}^l$

$k = 256$

$l = 4.$

$M \leftarrow$ prime no. which is $\geq k$.

Fact: check for every $X > 1$ $\exists$ a prime b/w x & $2x$

H defn as follows:

fix 4 no. $c_1, c_2, c_3, c_4 \in [0, M-1]$

$h(x)$

↓

already split into 4.

(each part is b/w 0 to $1KM$

fix 4 no.s $c_1, c_2, c_3, c_4 \in [0, M-1]$

$h(x) = c_1 x_1 + c_2 x_3 \ldots c_4 x_4 \mod M$

↓

x_1, x_2, x_3, x_4

~~claim~~ : each $0 \leq x_i \leq k-1 = \{0, \ldots M\}$

~~claim~~ H : class of all the f^n s.t. c_1, c_2, c_3, c_4 varies

M^4

~~H.O~~

claim Above $\mathcal{H}$ is a universal family

$\mathcal{H}$: class of all h as c_1, c_2, c_3, c_4 → M^4

Date :
Page :

l chunks → ( )

let's prove $l = 4$
fix $x \neq y$ ~~a~~ arbitrarily.

$$\Pr_{h \in \mathcal{H}} \left[h(x) = h(y) \right] \quad x_1 - x_4, \; y_1 .. y_4$$

$$\Pr_{h \in \mathcal{H}} \left[c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4 = c_1 y_1 + c_2 y_2 .. c_4 y_4 \mod 2 \right]$$

$$= \Pr_{[c_1, c_2, c_3, c_4]} \left[c_1 (x_1 - y_1) \; ... \; c_4 (x_4 - y_4) = 0 \mod 2 \right]$$

check

$x \neq y$ let $x_3 \neq y_3$

~~Pr(c~~ $\Pr_{(c_1, .. c_4)}$ $c_3 (x_3 - y_3)$

Ex:

$\sum_{i=1}^{N} i/M$ vs $\sum_{i=1}^{N} N/M$

why is it reasonable to assume that computing hash f^n takes constant time.

Why ~~co~~ ~~collision~~ collisions were imp

Lec - 3| $M \leftarrow$ prime $\quad 2^k \leq M < 2^{k+1}$.

U: (universe)

$$\Pr_{h \in H}\left[h(x_1 \ldots x_4) = h(y_1 \ldots y_4)\right]$$

$\exists i : x_i \neq y_i \quad ; i = 3$

$$= \Pr_{h \in H}\left[c_1(x_1 - y_1) + \ldots + c_4(x_4 - y_4) \equiv 0 \pmod M\right]$$

don't care abt d sum d

$$= \Pr\Big[\underbrace{c_3(x_3 - y_3)}_{\neq 0} \equiv \underbrace{c_1(y_1 - x_1) + c_2(y_2 - x_2) + c_4(y_4 - x_4) \bmod M}_{d}\Big]$$

$\{0, \ldots M-1\}$ $\qquad d \in \{0, M-1\}$

c_3 is unique : $d \cdot (x_3 - y_3)^{-1} \bmod M$

$\leq \frac{1}{M}$ $\quad \because$ ~~it is only~~ etc.

we have M choices & have to choose 1

Fact : $a \in \{1, \ldots p-1\}$ where p is prime.

$\exists$ a unique b s.t. $a \cdot b \bmod p = 1$

b is c/n inverse of a.

Now we will see another family of hash^{n} : Perfect hash functions

Perfect Hash f^n

Perfect ≡ no collisions

Pigeonhole + diff^v size (set change) ⇒ collision

Idea: Assumption (Stray): The set we want to show is not changing $(x_1, \dots, x_N)$ is known from before

$h : U \to [M]$

large small

- no collision
- fast computation of $h \to O(1)$

Claim:

Let $\mathcal{H}$ be a universal family that maps U to $[c \cdot N^2]$ (mapping to larger set)

Let $h \in \mathcal{H}$ be random.

Let $x_1, \dots, x_N$ be a fixed set of items

the expected no. of ~~collisions~~ < 1/2

worst case → (max^m) bucket size i'll ~~bus~~ collision

Pf: Indicator RV (indicates event)

$^nC_2$ many events $c_{ij} = 1$ if $h(x_i) = h(x_j)$

$= 0$ o/w.

↓

$i < j$

Total # of collisions = $\sum_{i=1}^{N} \sum_{j=i+1}^{N} C_{ij}$

Linearity of expectation.

$$E[\quad] = \sum_{i=1}^{N} \sum_{j=i+1}^{N} E[C_{ij}]$$

$$= \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Pr_{h \in H}[C_{ij} = 1] \quad (h(x_i) = h(x_j))$$

$$= \sum_{i=1}^{N} \sum_{j=i+1}^{N} 1/N^2$$

$$= \frac{N(N-1)}{2 \cdot N^2} < 1/2$$

Corollary: ∃ some $h \in H$ which has no collisions.

Suppose ∀ $h \in H$ ~~there are collisions.~~

Contradiction

then Expectation is 1.

Suppose not then ∀ $h \in H$ there are collision

?

In fact we can show stronger stuff

If we pick h randomly then such h with 1/2 prob.

→ go to Universal family of ~~no~~ pick random h s.t. such h.

what do we get?

$O(1)$ for the 3 operations

How do we find such $h \in \mathcal{H}$.

Iterate over all $\mathcal{H}$ until we find $h \in \mathcal{H}$ for which there are no collisions

Construction time: $O(|H|)$

Markov's Ineq

X be non neg. R.V.

$$\Pr[X \geq a] \leq \frac{E[X]}{a} \quad \text{for any } a > 0.$$

Example: Throwing a random Ludo die (fair)

$$\Pr[X \geq 7] \leq \frac{3.5}{7} = \frac{1}{2}.$$

$\Pr[X \geq 7] = 0$

Pf: Suppose not

$$\Pr[X \geq a] > \frac{E[X]}{a}$$

$$E[X] = \sum_{x \in \Omega} \Pr[X = x] \cdot x$$

$$= \sum_{x \geq a} \Pr[X = x] \cdot x + \sum_{x < a} \Pr[X = x] \cdot x$$

$$\geq a \sum_{x \geq a} \Pr[X = x] + \geq 0$$

$$\geq a \underbrace{\Pr[X = x]}_{x \geq a} \quad (\Rightarrow \Leftarrow)$$

$> E[X]/a$

If expectation is small then
R.V. takes small values.

Markov's Ineq

$\Pr_{h \in \mathcal{H}}$ [total # of collisions ≥ 1] $< 1/2$

$\Pr_{h \in \mathcal{H}}$ [total # of collisions < 1] $\geq 1/2$

$= 0$

(no collision)

[Even that total no. of collisions $= 0$ ⇒ is 50%.

- Repeat it r times

In expectation will have to repeat ≤ 2 times

Geometric R.V.

$O(\ln 1/\delta)$ repetitions the random $h \in \mathcal{H}$

until we have
0 collisions
w.p $\geq 1-\delta$ (for any δ) $\log \frac{1}{\delta} \Rightarrow 1-\delta$

$e^x \geq 1+x$ $\left(\frac{1}{2}\right)^x$ δ $k/2$ δ

e

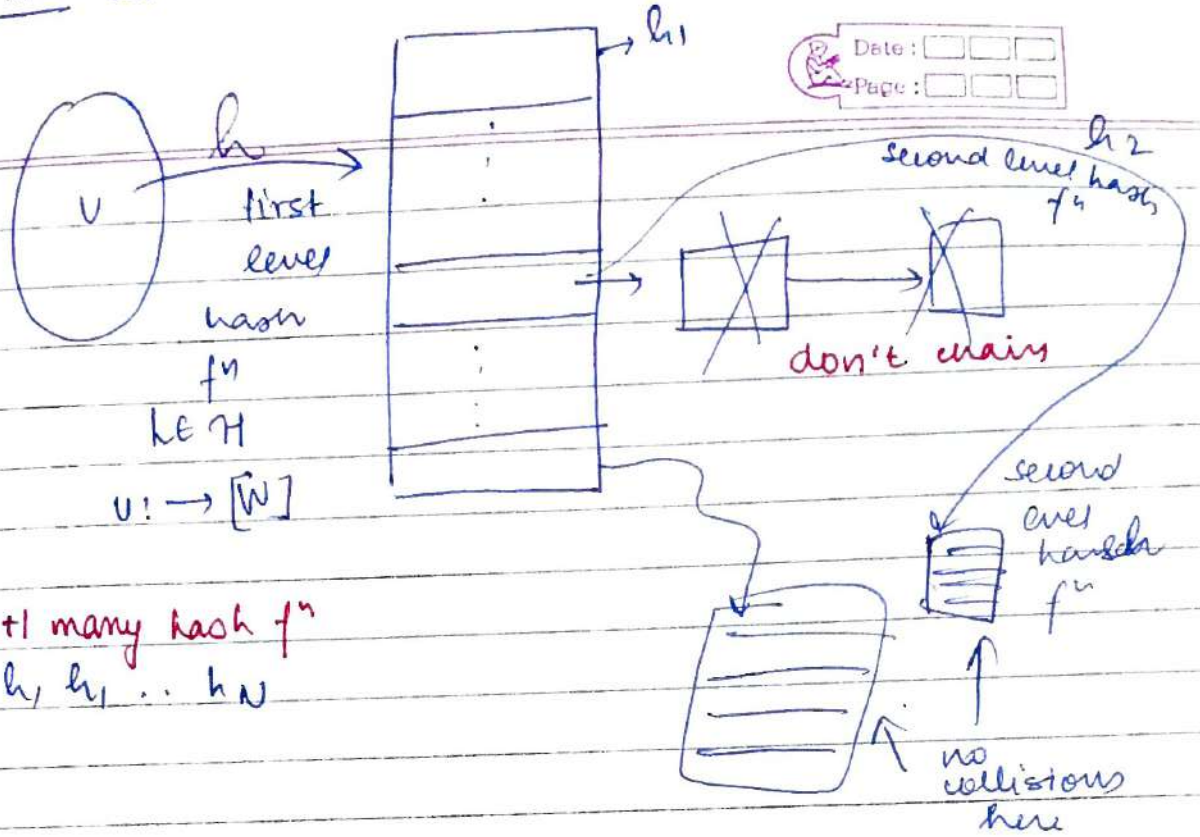
Issue with new hash fn ⇒ space usage is worst

$O(N^2)$

while in older one

$O(N)$

Fix: Use a two level hash fn.



N+1 many hash fn
$h_1, h_2 \ldots h_N$

~~h_1, h_2~~

$h_i : U \to [L_K^2]$ $\quad L_K$ = bucket size at k

Perfect !!!

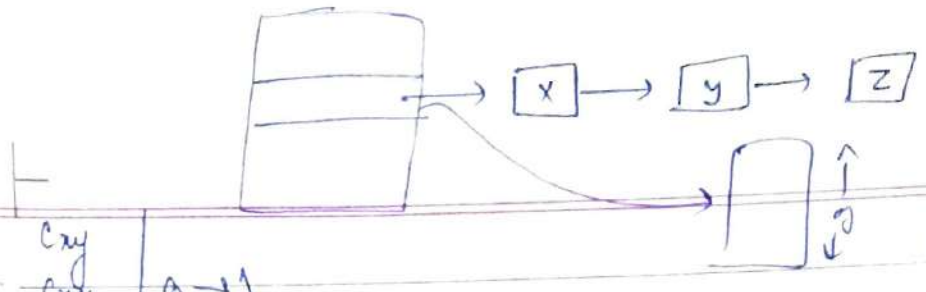
- first level: possible collisions
- second level: no collisions (use previous construction)

Claim: The space usage is O(N) in expectations.

Pf O(N) ← first level.

space usage at second level : $\sum_{K=1}^{N} L_K^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}$

stuck.



$C_{ij} = 1$ if $g_k(x_i) = h_k(x_j)$

C_{xy}
C_{xx}
C_{xz} $\quad q \to 1$

$$\mathbb{E}\left[\sum_{k=1}^{N} h_k^2\right] = \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{E}[C_{ij}]$$

$$= O(N) + \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbb{E}[C_{ij}]$$

defined w.r.t h (first hash fn)

This expectation is defined w.r.t $\mathbb{E}$ h

$$= O(N) + \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{N}$$

worst case.

$\Pr[h(x_i) = h(x_j)] \leq 1/M$ (universal)

$= O(N) \leq cN.$

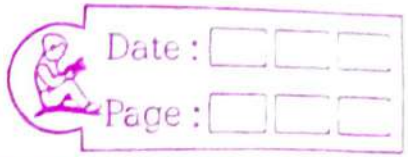
$$\Pr\left[\sum_{k=1}^{N} h_k^2 > 2 \cdot cN\right] \leq 1/2$$

$\log \frac{1}{\delta}$ trials we'll succeed.

Repeat until $\left(\sum_{k=1}^{N} h_k^2 \leq cN\right)$ for the first time.

$O(\log 1/\delta)$ repetitions ensure the above happen with prob $\geq (1-\delta)$

- worst case space usage is $O(N)$
- worst case T.C is $O(1)$ per operation.



$h : 2^u \rightarrow 2^m$ $\quad 2^{um}$ ways

m^4 (prime no. construction)

check ??? $Pr(x_i)$

Lec-32

6.6.
DP
Exercise

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

bbbbac → ((((bb)b)b)a)c)

Split based on the final

$(s_1\ s_2 \dots s_j)\ (s_{j+1} \dots s_n) = a$

$a = a \cdot c = b \cdot c = c \cdot a$

$\forall\ i, j$: $s_i \dots s_j$, $a/b/c$ (S) Yes/No

$\forall\ a, b, c$. DP[i, j, s]

DP[1, n, a] = Yes iff { DP[1, j, a] = Yes $\wedge$ DP[j+1, n, c]

DP[1, j, a]: b, c

DP[j+1, n, c]: Yes, c

vary j & do OR

Size of DP table $n \times n \times 3$

Per iteration → $\theta(n)$ per entry

$O(n^3)$

6.7

What is a good parametrization?

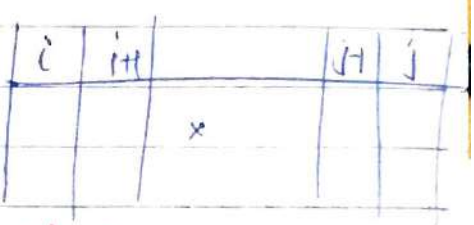
$X[1 \ldots n]$

$X[i \ldots j]$ ← longest palindromic subseq

call it $DP[i,j]$

final answer: $DP[1,n]$

$$DP[i,j] = \begin{cases} DP[i+1, j-1] \\ \max \begin{cases} DP[i+1, j] \\ X[i] = X[j] \rightarrow DP[i+1, j-1] + 2 \\ X[i] \neq X[j] \rightarrow DP[i+1, j-1] \end{cases} \end{cases}$$



i included, j not included

i not included, j included

both included.

both excluded.

$O(n^2)$ ← DP table

Filling each takes $O(1)$

- In exam clearly writes parametrization, Final ans, base case, recursive relation, time complexity.

6.10 - DP[i,k] = k heads in the coin $p_1, \dots, p_i$

- DP[n,k] ← final answer

- DP[j,t] ← p_j DP[j-1, t-1] + (1-p_j) DP[j-1, t]

Size of DP Table. cell

$O(n \times k)$ Filling takes const. time.

6.16 v_1 $g_1 \dots g_n$ v_n : profit

goal: max: (total profit) − (total transportation cost)

2^n: ⇒ parametrize subset of garages.

$\left(\sum_{i=1}^{n} g_i\right) - (d_{01} + g_{12} + g_{23} \dots g_{n-1,n} + d_{no})$

simplicity: Assume no going from home or coming back.

Parametrize the subset of garages

↳ 2^n many.

$g_1 \dots (g_i) \dots g_j \dots g_n$

↓

g_1 g_{i-1} g_{i+1} ... g_n Best tour on this

↓

first option of the best tour which is subset of this

check

cost of best tour in $g_1 \dots g_n$ where first garage is g_1

$\equiv$ first go from g_i to g_j

+

cost of best tour in $\{g_1 \dots g_n\} \setminus g_i$

Varying g_j now of take min

Now introduce concept of going back to house.

~~first~~ ..

$\equiv$ d_{0i} + (first go from g_i to g_j) +

cost of best tour in $\{g_1, \dots g_n\} \setminus g_i$ where first garage is g_i & last garage is g_k

+

d_{k0}

vary g_j now of take min

n options

$(n_0 \, 2^n \, on)$ ← DP table size.

(ASK)

$O(n^3 2^n)$

6.20

Parametrize the root node (∵ then left & right subtree makes sense)

$[p_1, \ldots, p_n, j]$

inorder is sorted

$[s, t, j]$ → cost of best BST on $p_s \ldots p_t$ where root is p_j (sorted so contiguous?)

$s \in \{1 \ldots n\}$

$t \in \{1 \ldots n\}$

$j \in \{1, \ldots n\}$

DP[s, j-1, k]

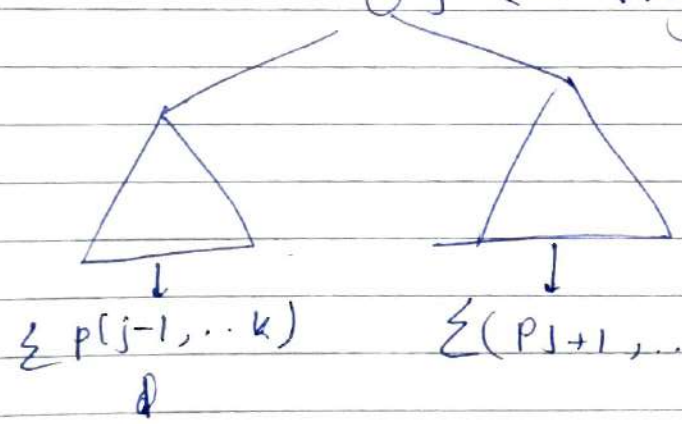
DP[j+1, t, w]

we ~~have~~ have to consider only one hop

$\exists j \leftarrow$ any of these ins

⇓

DP[s,t] : best BST only having $s \ldots t$



~~So~~.

Ex :- ~~Quick thing~~ # check a # sales # sorted so contiguous.

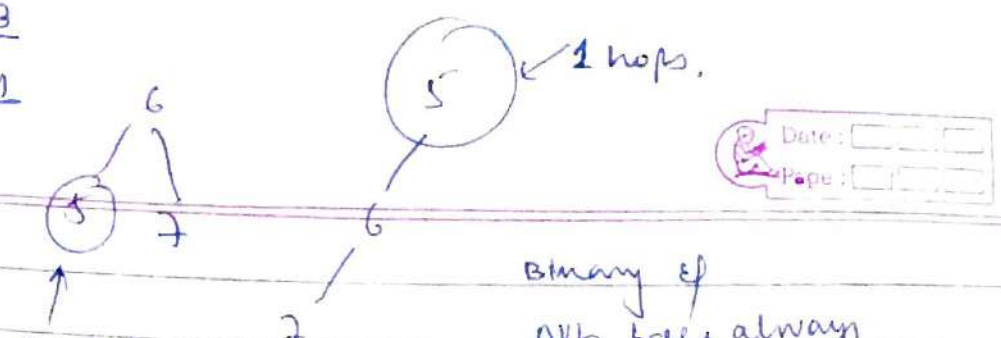
Rec-33

Corners

(8)

AVL

Trees



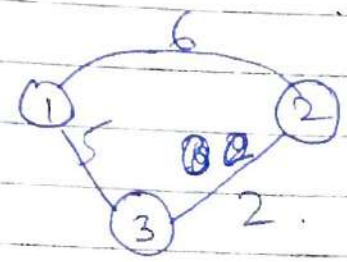
Binary of AVL trees always same hops to a no.

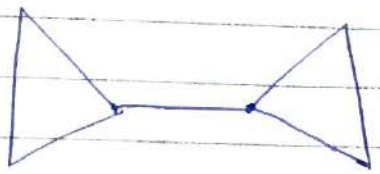
5.9 (j)

Dasgupta

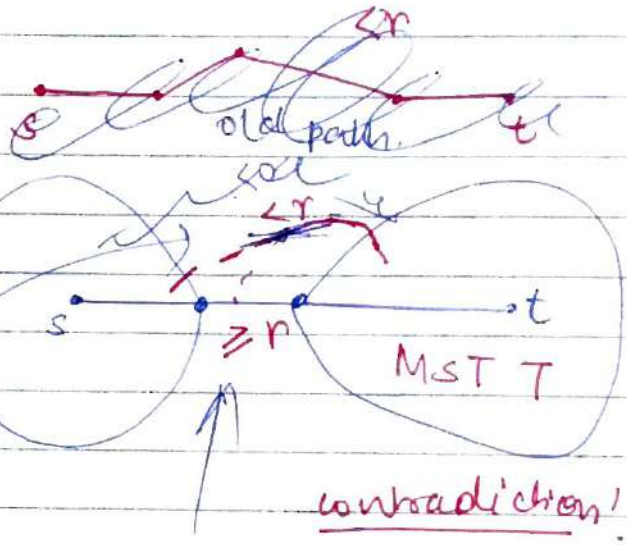
In MST any 2 nodes are connected by one of only one path

we use cut & paste argument





Pf by contradiction



contradiction!

Take unique path from s to x in MST & see first y edge

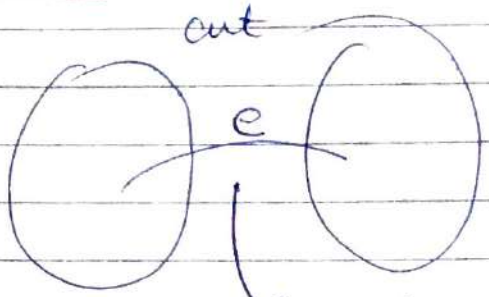
then we have the x path plus cut passing thru

5.23 $G = (V, E)$, $T = (V, E')$

$c : w(e) \longrightarrow \hat{w}(e)$

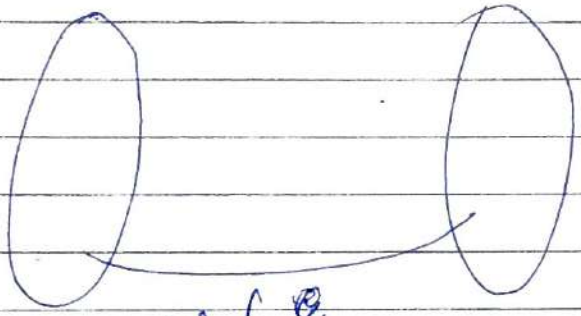
① $e \notin E'$, $\hat{w}(e) > w(e)$

no change



still min so no change

② $e \notin E'$, $\hat{w}(e) < w(e)$

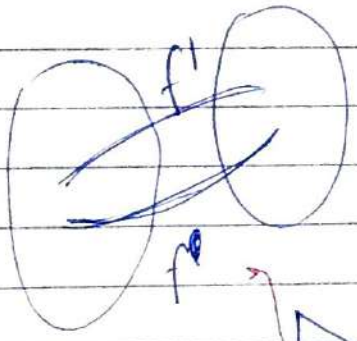


old f

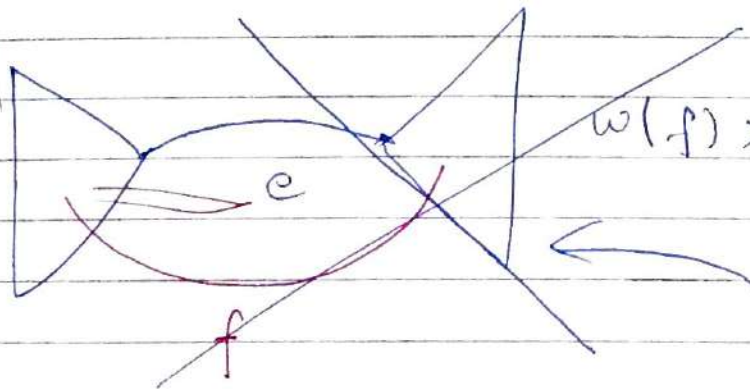
cut edge

$e \notin MST$

$\hat{w}(e) < w(e)$



f'



$w(f) > w(e)$

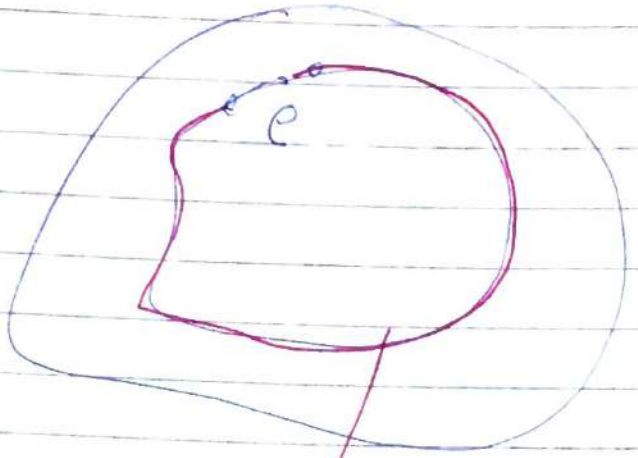
this is a special case

Earlier is a special case when e has part of MST

General case: you put edge e into MST

⇒ creates a cycle

⇒ just remove the max. edge of that cycle



How do we do this in linear time.

unique path b/w's end pts of e in MST.

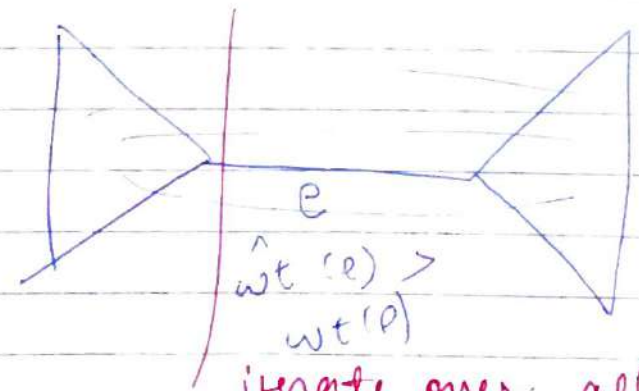
Traverse this

remove the max edge of this path.

∴ cycle length is $O(N-1)$ ∵ MST

(c) so still contains

(d) $e \in E'$, $\hat{w}(e) > w(e)$



iterate over all cut edges of

pick the min

How do we do this in linear time?

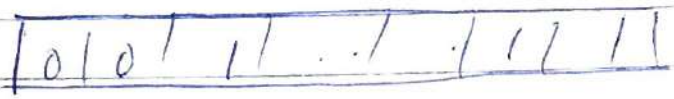
$(s,t) \leftarrow$ edge. (using adj list)

If we need to decide s & t on diff "sides"

Not DSO

↳ log n

Maintain binary array



on MST

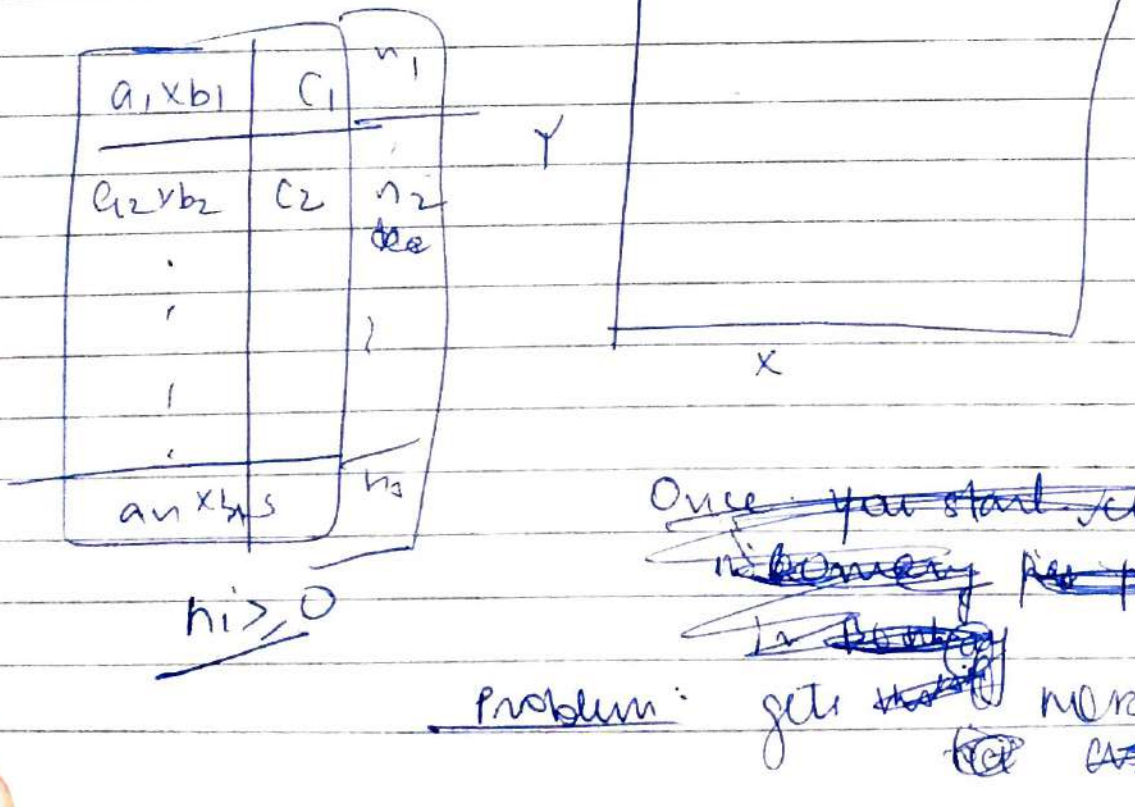
(i) First do DFS, with e not being considered.

then one component all 0
other all 1.

Then iterate over all edges

if find min cut edge.

6.14.



$n_i \geq 0$

Once you start selling

Problem: gets worse

Profit : $n_1 c_1 + \dots + n_2 c_2 + n_3 c_3$

Let us keep xy many variables

Date :
Page :

$P_1, \dots, P_{xy}$

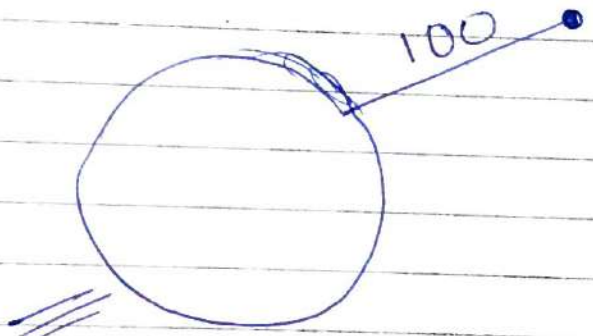
P_j denotes it ~~goes~~ goes to a pdt

(Sir sent soln to this prob
→ In mail
30/10/24

5.6- Suppose not then use pairing property

~~Soln~~

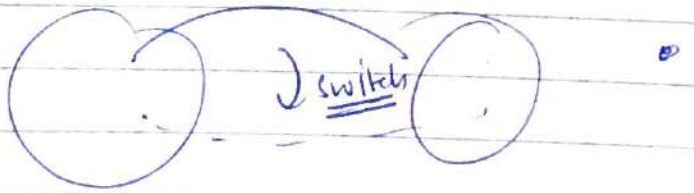
5.9 (a) False.



(b) remove heaviest edge, still connected ⇒ then MST

↑
Sol-1

Soln-2
Suppose



~~100 pairs~~

(e) Put this edge ⇒ creates a cycle
⇓
remove max wt edge

(f) Kruskal's algo will select this first if we know its correct.
(Prim's also ⇒ start with vertex of that edge)

In Greedy algo ✓ switching argument

(e) True (~~Kruskal~~ o/w we can switch)

Ex: 5.9 (b), soln-2 is out of past & right?

5.6 (b) $T_1 \cup \{e_3\}$ / set is ~~MST but~~ spanning Tree ~~but~~ ~~of~~ but wt of new tree is more or less.
⇒ There is a pairing. ⇒ MST not MST?

Pairing Property

~~Ex: Thm~~

Lec-

01/11/24 Ch 2 (2.32)

~~For~~ Naive: $O(N^2)$

For simplicity let's talk abt $\mathbb{R}$

$-\infty$ ——— ∞

$x_1, \ldots x_n \in \mathbb{R}$

- ~~sort~~ (WLOG) assume sorted ($O(n \log n)$)
- Only adjacent pts can be closest.
- $O(n)$ to return the output.

Now 2D

- can't ~~start~~ sort (on the surface) coz there are 2 ways to sort
- use Divide & Conquer:

(x_1, y_1) . . . . . (x_n, y_n)

(x_1, y_1) ⋮ (x_{ind}, y_{ind})

x_{ind+1}, y_{ind+1} ⋮ x_n, y_n

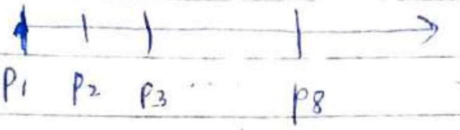
$d_L =$ ~~[illegible]~~ $d_R =$

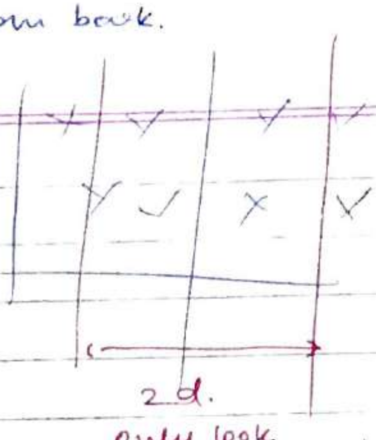
$d = \min(d_L, d_R)$

* How we are missing cross ~~Ones~~

splitting and all in $O(n)$ time required. ∴ (idea from recurrence rel^n)

Read algo from book.



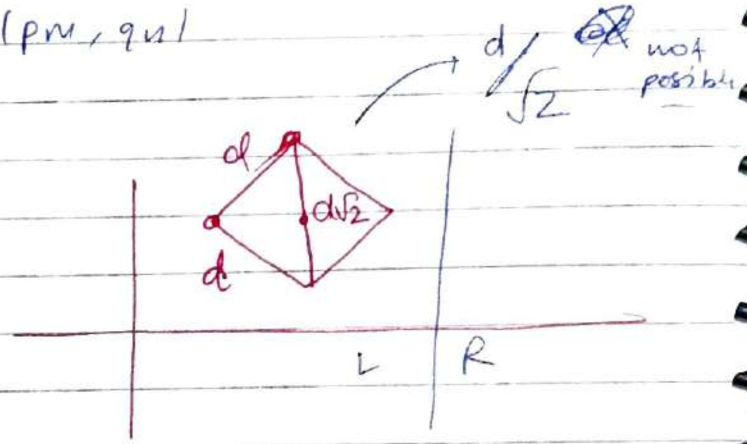


d(p1, p2)
d(p2, p3) } 7 distances
⋮ find min
d(p1, p8)

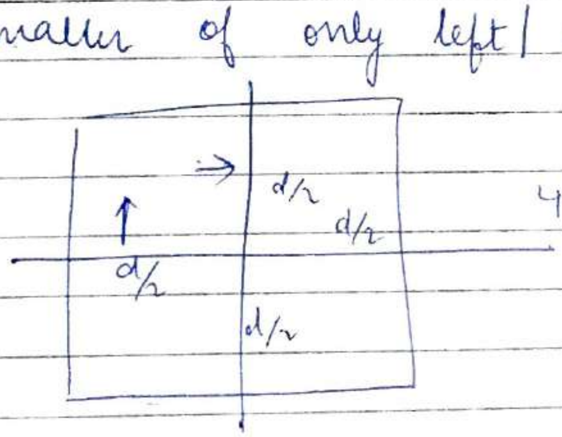
arg min = d(pM, qM)

7n.

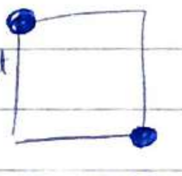
8 pt in sorted list will be beaten by one of these 7.



d = smaller of only left / only right.



4 box & 5 to place } Pigeonhole
↓
one contains 2
↳ farthest



↳ but still d/√2

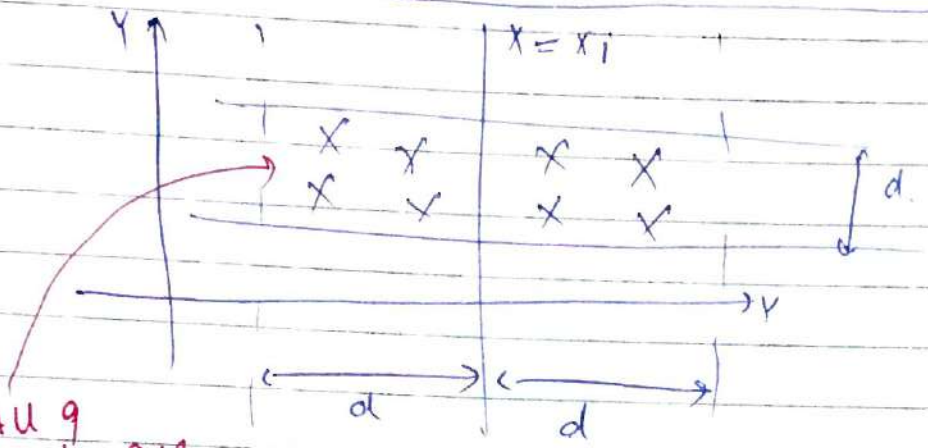
(b) For contradiction,
missed a crosspair s.t. dist < dist (pM, qM)
pj is not part of this immediate 7 pts to pi

only chance of making mistake $d(p_i, p_j) < d$. ~~due to sorting~~ ∵ sorted $|y_i - y_j| < d \Rightarrow |y_i - y_{i+7}| < d$ & so on.
~~$d(p_i, p_{i+7}) < d$~~

Revision sheet
→ Lectures
→ summaries
- BFS

→ remember to swap in heapify

→ Ask Sir abt the DFS algo
→ Get PS im plementation checked
~~- RP~~.



All 9 pts are part of 2d x d rectangle

(⇒ ⇐)

∴ one d x d would then contain at least 5 pts.

- Divide & Conquer
- DS.
- Graph
- DP, Greedy

(1.) Median Finding ⇒ O(n) & then split ⇒ O(n)
(2.) O(n) to remove x_i's farther from x_i than d
(3.) O(n log n) for sort (bottleneck)
(4.) ~~#~~ O(n)
~~(#)~~

$T(n) = 2T(n/2) + O(n \log n)$
$\Rightarrow T(n) = O(n \log^2 n)$

Can we do better? O(n log n)

Store 2 different orderings. Idea.

~~Sort the~~ ~~list~~ w.r.t input 2 list
- one w.r.t x (median easy)
- one w.r.t y

filter acc. to y.

In y sorted remove $x_j - x_i > d$.

4/11/24 lec

Up Nxt :-

insert, delete membership — hashing,

— heap

— bitmap,

- Red-Black Tree (like AVL)

- Greedy problem as heuristic

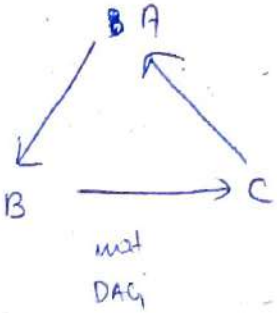
(NP $\overset{like}{\rightarrow}$ 2^n | → by Greedy as an attempt (approximate)).

- MST — Kruskal / Prim's / ~~Mst~~ Qui-z - II (last question)

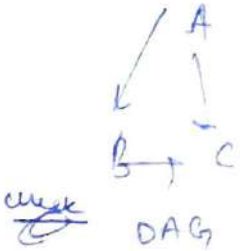
- Topological sort.

Directed Acyclic graphs : Directed graphs which don't have a cycle.

(DAG)



not DAG



cycle

DAG

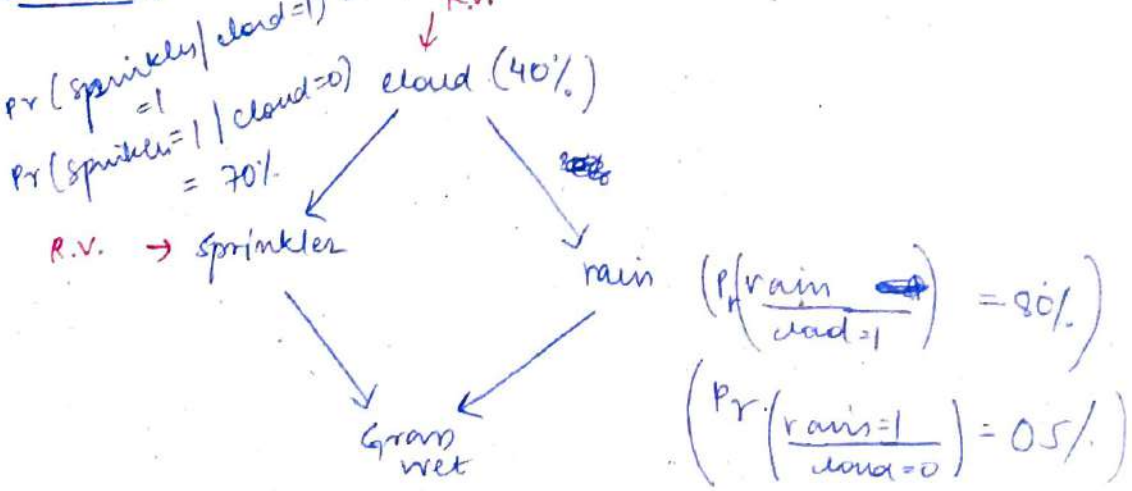
Example :-

Causality :

Pr(sprinkler=1 | cloud=1) = 20%

Pr(sprinkler=1 | cloud=0) = 70%

R.V.



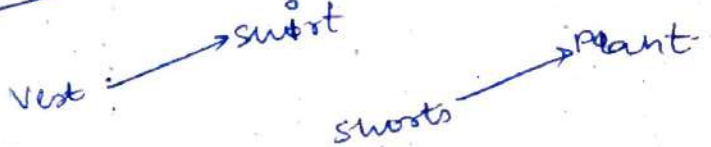
$\left(Pr\left(\frac{rain}{cloud=1}\right) = 80\%\right)$

$\left(Pr\left(\frac{rain=1}{cloud=0}\right) = 05\%\right)$

S	R	Pr(G)
0	0	1%
0	1	90%
1	0	90%
1	1	99%.

- No cycle

Example:

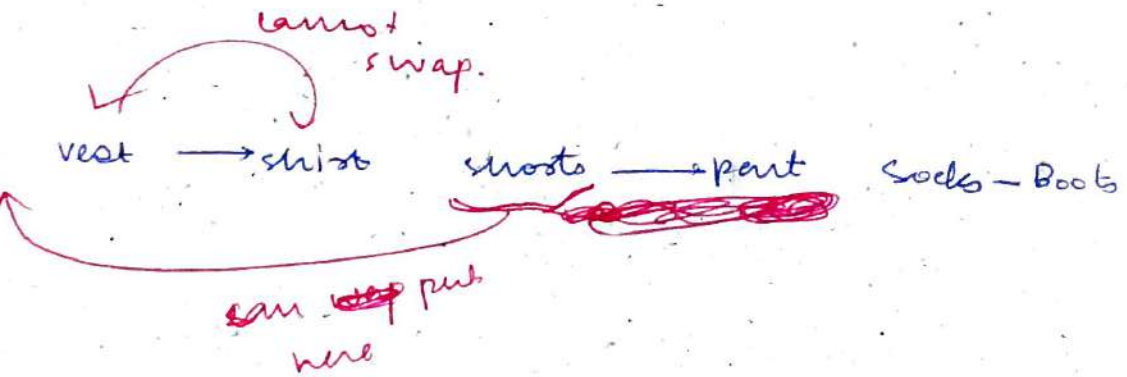


Socks → Boots.

DAG

Partial Order

Give an order where edge goes from LHS to RHS



Course A → Course B

↓

means pre requisite

A set of courses that needs to be covered

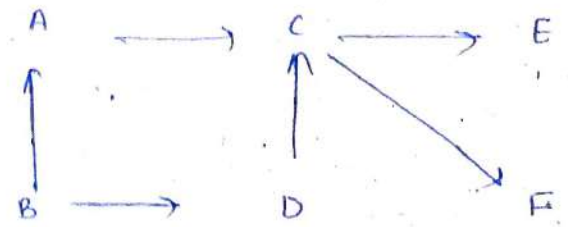
Can we do a valid ordering?

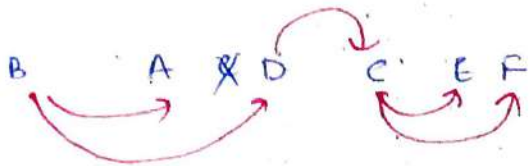
Yes!

How do we find some num! sort it always

Given a DAG we can always sort it s.t. edges go left to right

Topo sort

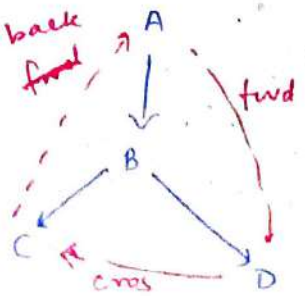




Edges always go from left to right.

What do we do?

DFS! How?



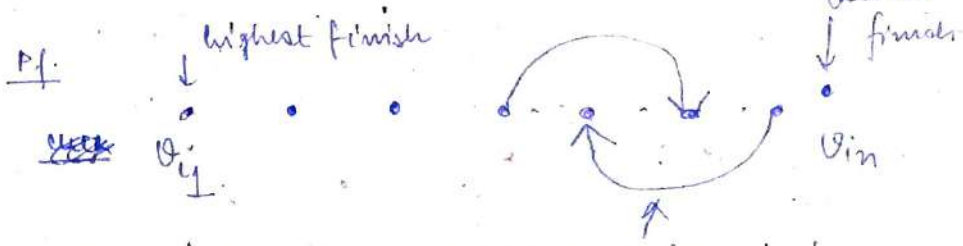
→ tree

How do we find source node?

↳ will finish last

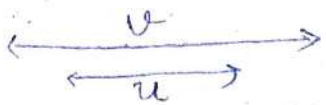
Fact: The vertex having highest finish time will always be a source node.

claim: sort acc. to finish time (high to lo) that will be a valid topological sort.

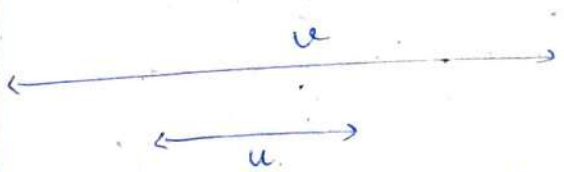


What's wrong if an edge goes from lo to high finish time?

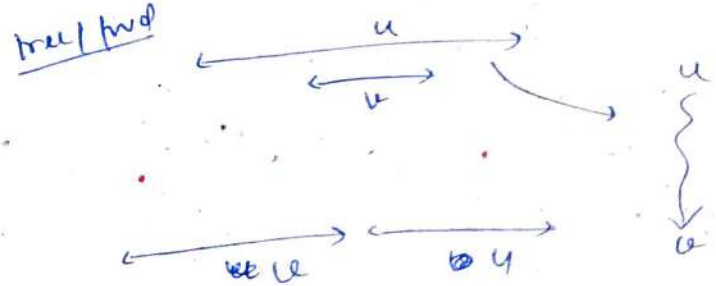
Pf by contradiction — we'll have cycle o/w.



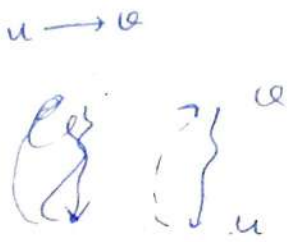
Only happens for back edge



only happens for back edge.



v's

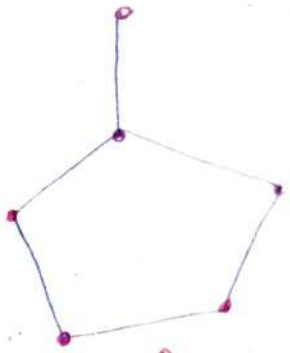


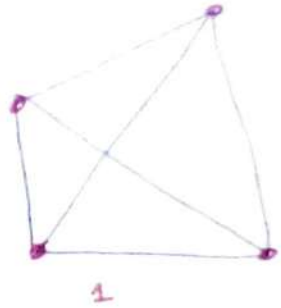
we know there are no cycles ⇒ no back edges.

So the above is a valid topo sort!

□

Now that we topo sort, we can understand more structures of DAGs

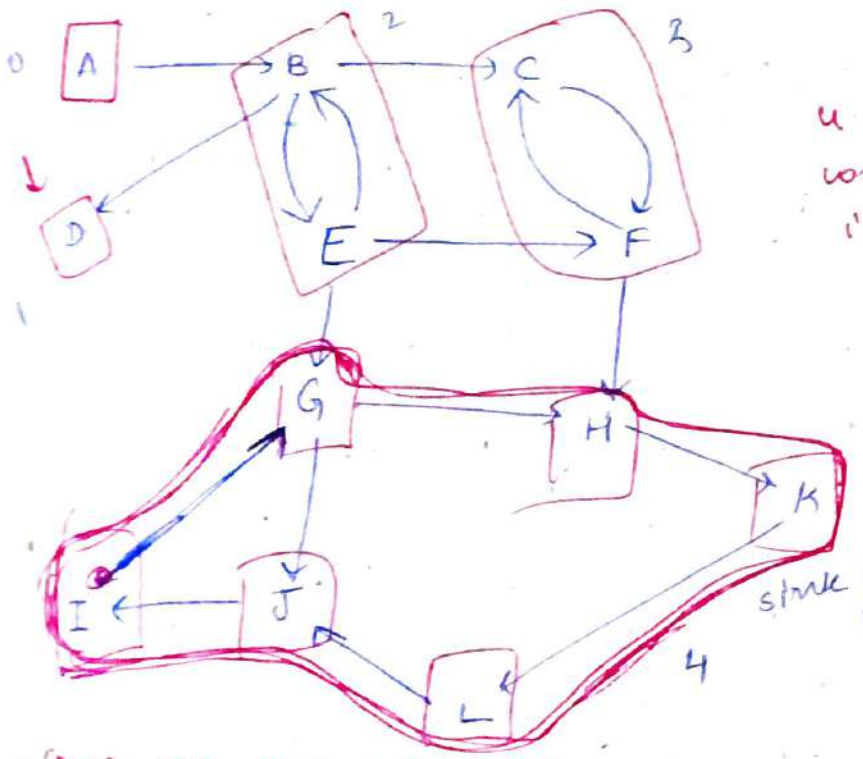




10 nodes

DFS: $O(|V|+|E|)$ Total # of components = 2

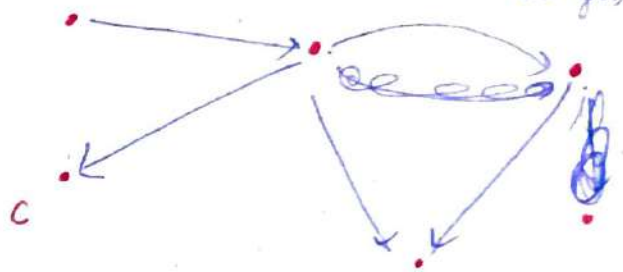
What is the meaning of components in a directed graph?



u & v are connected if we can go from u to v & from v to u using the edges.

- (B, D) are not connected.
- (B, E) are connected.

- Merge nodes which are in same connected component then the graph we get is a DAG. (remove multiple edges)



If there are cycles. In the final graph then we could get a larger component in original graph.

- How do we use ~~earlier~~ Topo sort to tag the original graph.

more meaningful thing is to discover single nodes

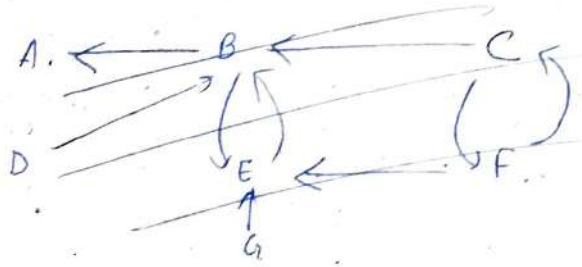
- If we get one node in SCC then apply DFS to visit all nodes of comp.

last start time / ~~node with~~ first finish time.

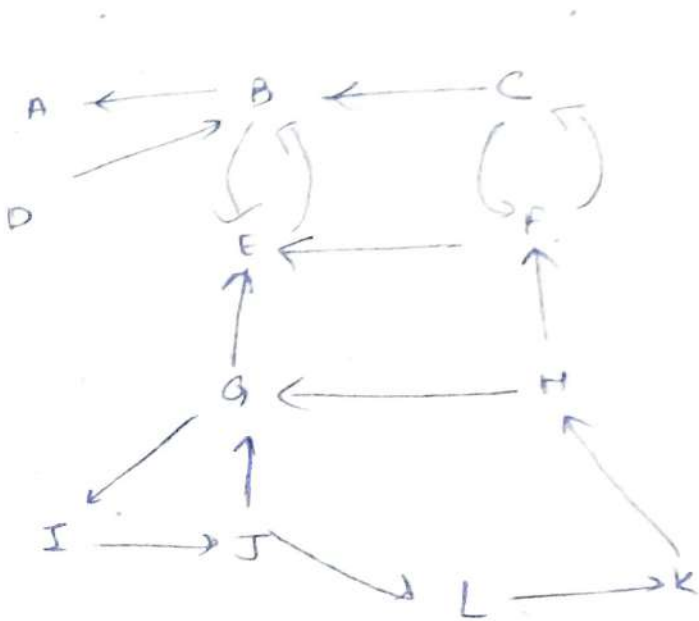
Fact: It is easy to find a source node using DFS.

Corollary: If we reverse the graph, the last finish time will be N

part of a SCC



reverse graph



(1) Reverse the graph G to get G^R

(2) Perform DFS on G^R & sort from high to low finish time.

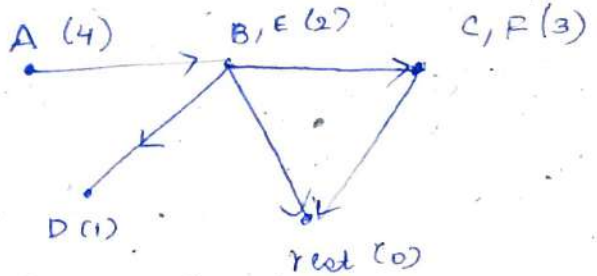
(to identify sinks ^ orally can also look at merged graph)

(3) Start DFS on G (with restarts) ** acc. to above sorting order.

(Remove sinks during your traversal)

6/11/24

Rec -



DAG

~~If we start DFS from rest (0)~~

~~From SCC~~

Procedure SCC

- perform DFS processing node ~~to smallest~~ in the G_R to get an ordering based on decreasing finish time.
- Process the vertices in the previous sorted order & perform DFS on G. (each restart of DFS increments another count which is the tag of the connected component)

Generalization of finishes last ~~part of~~ issue.

Fact: Let C_1 & C_2 be two strongly connected component s.t. ∃ a path from C_1 to C_2.

C_1 C_2

$u_1 \rightsquigarrow v_2$

Then

the highest finish time in C_1

$\not\exists$ v

the highest finish time in C_2

(no matter where you start DFS)

Pf:

2 cases

→ process C_2 earlier.

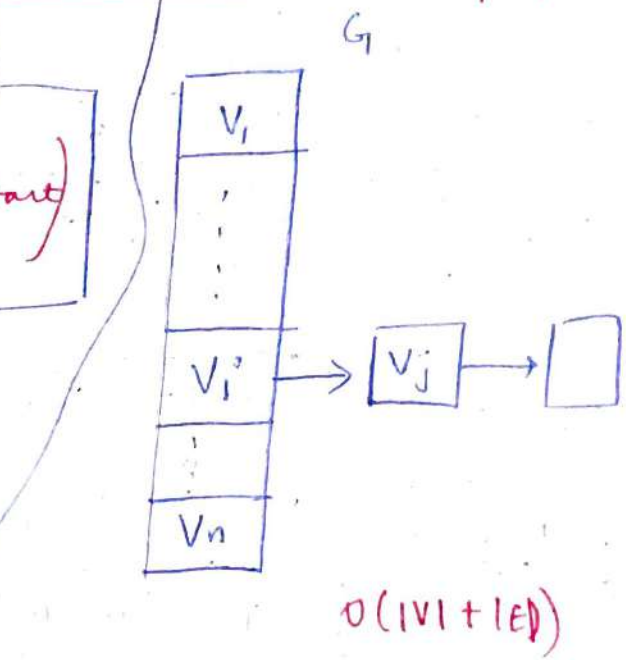
process C_1 earlier

Then correctness of algo is easy

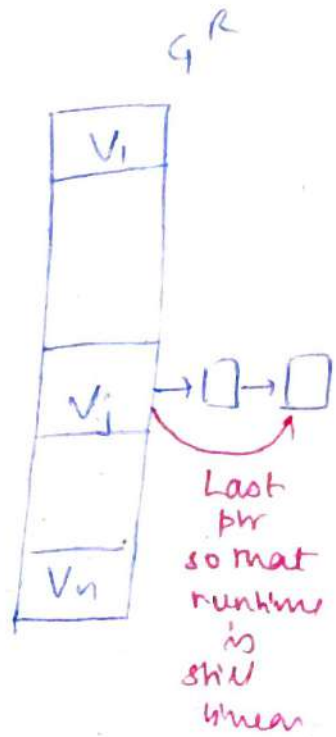
~~Once we have the~~

- Highest finish time will be one of the sinks.

How to reverse the graph



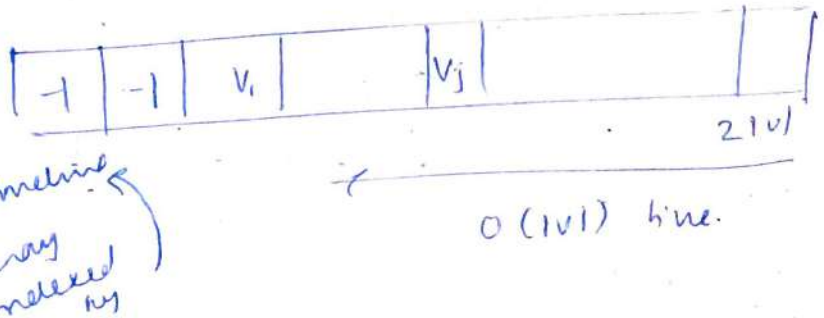
$O(|V|+|E|)$



TC → $O(|V|+|E|)$

Soohi

$O(|V| \log |V|) \longrightarrow O(|V|)$ how?



$O(|V| + |E|)$ total time!

Assignment

(can read: Das Gupta + its exercises).

Greedy ~~DFS~~ as "Heuristic"

- no pf of correctness ~~always correct~~ some
- based on intuition.
- work well in practice

Set cover problem (Extremely Practical)

$U : \{x_1, \ldots x_n\}$.

S { $S_1 \subseteq U$; ... ; $S_m \subseteq U$

Goal

Pick the smallest no. of subsets that covers U

$$\underset{T \subseteq S}{\operatorname{argmin}} |T|$$

$$\text{s.t.} \bigcup_{S_j \in T} S_j = U$$

U: {1, 2, 3, 4, 5, 6, 7, 8}

S_1: {1, 3, 5, 7}

S_2: {2, 4, 7}

S_3: {8, 5}

S_4: {3, 8}

S_5: {2, 4, 6, 8}

Ans: $S_1 \cup S_5 = U$

∴ any single set that covers U.

∴ answer is 2.

Try out all subsets 2^n times. (n is no. of sets given)

— no algorithm that can do this in poly time (n, m)

∈ NP hard

↓

easy to verify → $\leq k$. Decision Problem.

but difficult to solve.

Until all elts covered

- choose one having largest no of elt from Universe.
- remove elts from Univ which were in this subset.

Claim: The greedy rule returns a soln that is at most $\log n$ worse than the optimal soln.

Pf. Lets assume the best cover has size k.

$\Rightarrow$ First set must cover at least N/k elts.

Cool!

$\Rightarrow$ $n_t \leftarrow$ # leftover elts to cover after t iterations

$n_0 \leftarrow n$

Consider after $(t-1)$ iterations n_{t-1} elts remaining.

If k sets cover the whole n_{t-1}

greedy chosen set covers $> \frac{n_{t-1}}{k}$ new elts.

$$n_t \leq n_{t-1}\left(1 - \frac{1}{k}\right)$$

Suppose we take m steps of the greedy algo

$$n_m \leq n_0 \left(1 - \frac{1}{k}\right)^m$$

$$= n\left(1 - \frac{1}{k}\right)^m$$

$1+x \leq e^x$ $\forall x \in \mathbb{R}$ (= when $x = 0$)

$$n_m < n e^{-m/k}$$

We are done when

$$n_m < 1.$$

We want: $n e^{-m/k} < 1.$

$< \log n$

$$-m/k < \log(1/n)$$

$$m/k > \log(n)$$

$$m > k \ln(n)$$

We'll set $m = k \ln n.$

We can choose m

∴ at most our solⁿ goes as worse as ~~then~~

competitive ratio / approximation factor

Approximation Algo

$V = \{1, 2, \ldots, 6\}$

$S_1 = \{1, 2, 3\}$

$S_2 = \{4, 5, 6\}$

$S_3 = \{2, 3, 5, 6\}$

$S_1 = \{1, 6, 2, 0\}$

$S_2 = \{4, 5, 6, 7\}$

$S_3 = \{3, 4, 5, 6\}$

At least one might have to take all.

$S_4 : \{2, 3, 5, 6\}$

$\{S_3, S_4, S_2\}$ ← greedy

$\{S_1, S_2\}$ ← optimal

Question: This ratio bound is best or can we improve the greedy analysis?

Ans: Not all cases where greedy will be $\geq$ bound factor worse.

(Infinite class)

Exercise!

Hint

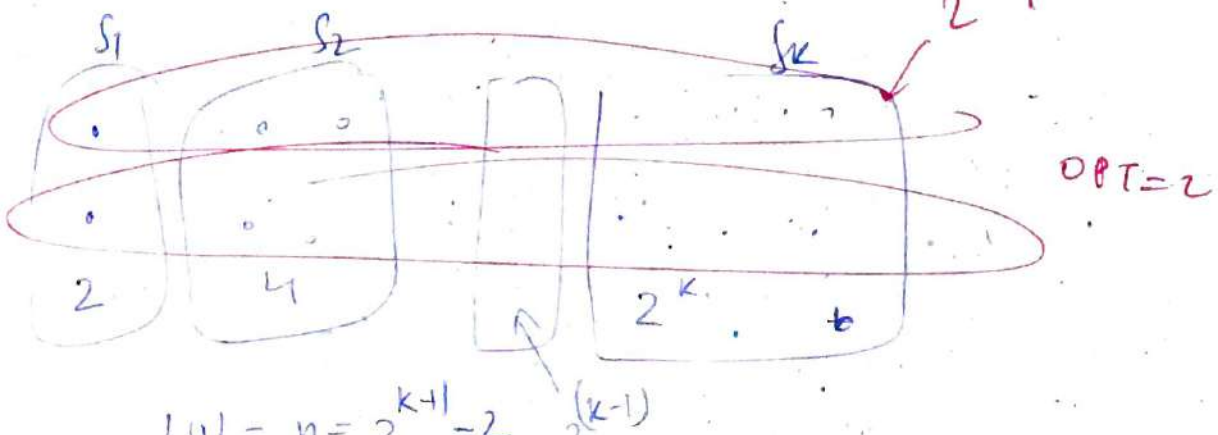
Why $n_t \leq n_{t-1}\left(1-\frac{1}{k}\right)$ ✓

Reasoning: greedy chosen set covers $\geq \frac{n_{t-1}}{k}$ new sets ✓

Lec – :

★ $K \ln n$
(natural log)

- Is the greedy analysis best?
- Is $2(\log n)$ approx inavoidable?



$|U| = n = 2^{K+1} - 2 \quad 2^{(K-1)}$

greedy : selects the 2^K sized set in the first step

S_{K-1} : 2^{K-1} additional

~~OPT size 2~~

S_{K+1} : $2^K - 1 - 2^{K-1} = 2^{K-1} - 1$

or S_{K+2}

[greedy $\Rightarrow K = \theta(\log_2 n) = \theta(\ln n)$
OPT $\Rightarrow 2$

So there is an infinite class of problems for which greedy fails.

Very strong -ve result

under ($P \neq NP$) / $\frac{1}{\log^{99} n}$. 99 . 999

No algo with $(1 - o(1)) \log n$ approx ratios possible in polynomial time

Poly (n, m) n^m;

Coun : "Approx algo"

T.C.

maintain an array for whether covered or not

0	1	1	. . .	1

U

T.C. : $O(mn\theta + k\log n)$

(mn: each iteration; k: at max these many iterations.)

Weighted set cover (A Generalization of prev one)

$U : \{x_1, \ldots x_n\}$

$S_1 \subseteq U : cost(S_1)$

⋮

$S_m \subseteq U : cost(S_m)$

} trying to cover all elt. with minimum total cost.

$cost() > 0$

Minimize : $\frac{cost(S_i)}{\#\text{ new elts } S_i \text{ is covering}}$ — minimize over S_i

· Claim This algo has approx ratio

$H_n = 1 + \frac{1}{2} + \frac{1}{3} \ldots \frac{1}{n} \approx \log n$.

Pseudo-code: $C \leftarrow \phi$

while $(C \neq U)$

- select S_i with min. $\frac{\text{cost}(S_i)}{|S_i \setminus C|}$
- $C \leftarrow C \cup S_i$

All the newly added items will get this charge.

Pf: charging argument.

Let $e_1, \ldots, e_n$ is the order of insertion of elts.

If ties select any permutation

We'll define the charge due to e_i

$\alpha_i \leftarrow$ charge of e_i

Total cost of algorithm $= \sum_{i=1}^{n} \alpha_i$

- We want to show $\alpha_i \leq ?$

consider the situation before e_i was inserted

at that pt $|C| \leq i-1$ ($\because$ e_i can be in any)

$\Rightarrow$ remaining elts # $\geq n-i+1$

e_i . . . . . e_n

Suppose the best soln is K. (Pick out the j sets that covers $e_i, \ldots e_n$ with cost $\leq K$)

min change e_1 ---- k_n

∴ the min change in the selected sets $\leq \frac{k}{n-i+1}$

$k_i \leq \frac{k}{(n-i+1)}$

Approx ratio is $\leq k H_n$

Q k is total sum BC we are overcounting.

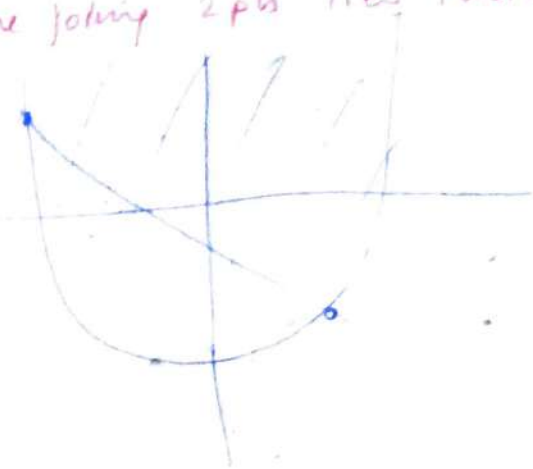
Convex Hull

(Computational Geometry)

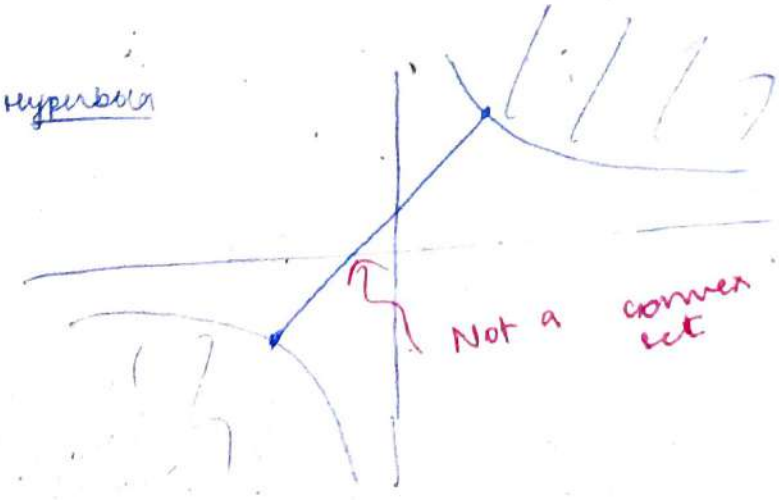
- How fast if correctly can we solve geometry problems.

- convex set - line joining 2 pts lies inside the set for any 2 pts

parabola $y \geq ax^2 + bx + c$

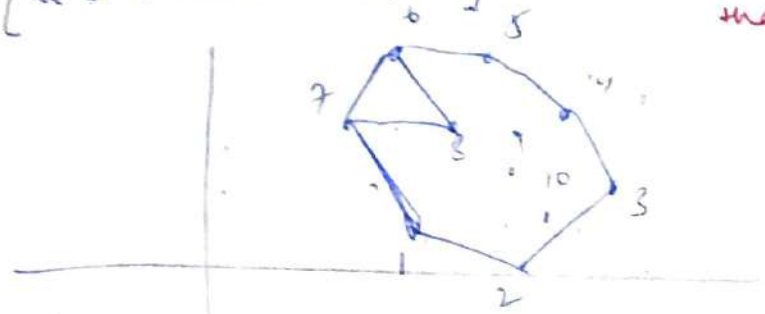


hyperbola



Def^n [Convex Hull] → smallest convex set (in terms of that includes area) all the pts.

[We'll restrict to $\mathbb{R}^2$]



Polygons are convex

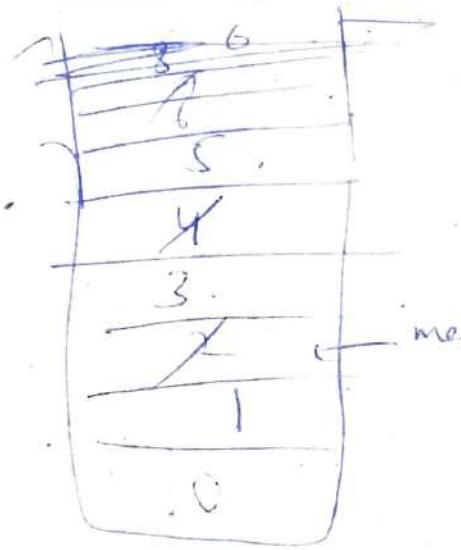
Take any line all pts have to lie on same side.

I/P: set of pts in $\mathbb{R}^2$

$(x_1, \ldots x_n) = P$

O/P: convex Hull of P (boundary pts in CW or ACW order)

0 1, 2, 5, 4, 7, 6, 7



makes ACW turn

3 horizontal

pop till ACW again

Assumptions (for now): no collinear pts of max n.

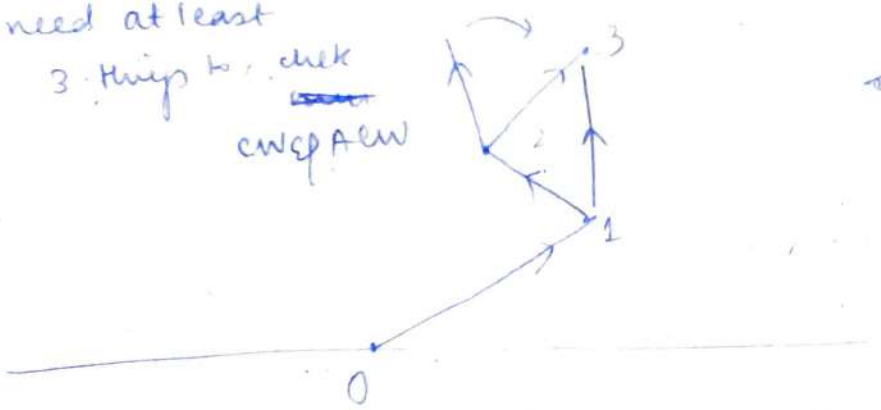
Algo - o start from min y coordinate.

of compute angles of sort. If 3 pts seen.

start comparing angles instead ~~of pts~~ see CW of ACW.

~~Invtie~~ - sort acc. to angle. ⟶ $n \log n$ time

- need at least 3 things to check CW & ACW



each pt is popped at most once ⟹ $O(n)$.

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

- full discussion abt the proof
- General set cover : K'sop analysis

11/11/24

Rec-

DFS ~~Toposort~~ DFS - Dasgupta.pdf

3.22

(i) source node

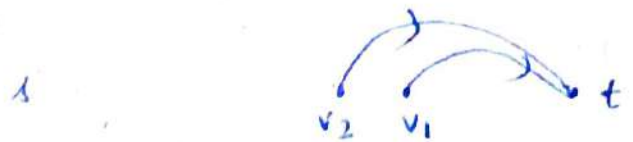
(ii) DAG (?)

3.23

How many paths from s to t.

Obs: s has to come before t in Toposort
o/w there are 0 paths

All nodes that go to t directly.



$[s:t] = [s:v_2] + [s:v_1]$

3.22

Soln:- Look at meta graph (with SCCs as merged) then do toposort & start from left & see if every node reachable.

Soln2:- DO DFS & take s to be node that finishes last. If every node reachable from this node then done.

DP[j] ← no. of paths from s to j

DP[t] ← DP[v_1] + DP[v_2]
where (v_1, t) (v_2, t) are every parents of t.

Base case: S + when j is before S in topo sort order

T.C. ⇒ linear

3.24. Directed path that visits every vertex once

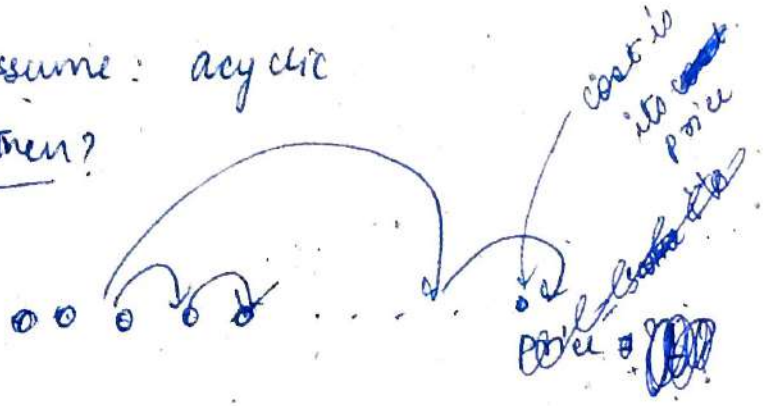
(27)

do toposort & check if there is edge btw every 2 consecutive nodes.

3.25

(a) Assume: acyclic

Then?



DP[ ] ← cost for node j

← min

s.t. (j, i) ∈ E

DP[i]

price of itself

build DP from right to left

TC: $\sum_{v \in V} \text{outdeg}(v) = O(E)$

(b) cycles

2 tier structure of directed graph.

↓

DAG ← zoom in SCC

For entire SCC ⇒ same cost ★

Combine (a) of (★) to get final algo.

To Toposort of ~~merged~~ SCC: Meta graph

& then from ~~left~~ right to left fill DP taking care that all nodes in merged SCC have same cost

Sir didn't say this I wrote it so ↴ get it checked.

3.8 Pouring water

Question Description.

Pour till source empty or destination full.

Can we have 2 units in 7 or 4?

Pourings → edges

Nodes ⟶ 3-Tuple (quantity state)

Total nodes → 11 × 8 × 5 (∵ everything happens in $\mathbb{Z}$)
0 to 10 0 to 7 0 to 4

(0,7,4) This source node has only 2 outgoing edges as no other states possible.

(4,7,0)

(7,0,4)

(9,2,0)

(8,2,1)

(4,2,4)

(9,0,2)

(8,1,2)

(2,7,2)

start DFS ep see if any of these reachable

Algo: start DFS from source node ep see if any of the reachable nodes has exactly 2 in 2nd or 3rd coordinate of tuple.

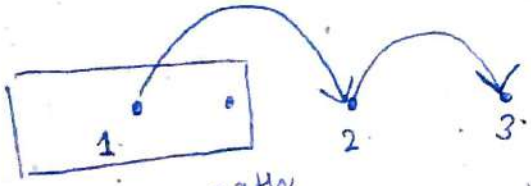
3.16 $G = (V, E)$

$\forall u, v \in V \quad (u, v) \in E \Rightarrow u$ is a prereq of v

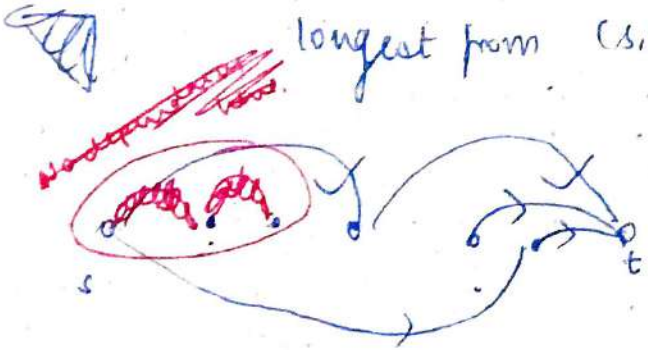
WT: # of sums

min.

Toposort:

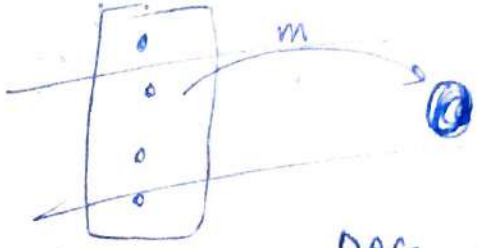


Longest path in this will work.

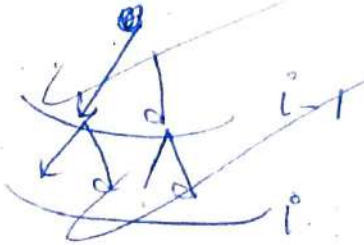


longest from (s, t) = max

other way (BFS starting from source)

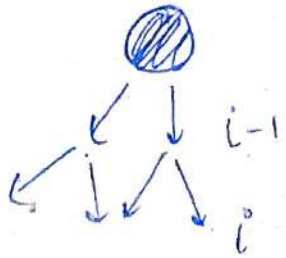


DAG → do BFS



Soln-2

Merge all source nodes into 1 & then do BFS.



& we started from all sources
Since BFS, hence we'll get shortest path, ✱

3.17 (Infinite paths)

(a) occurring only many times ∴ subset of SCC
∵ v_i, v_j occur infinitely many times in trace
⇒ there is a path from v_i to v_j & v_j to v_i ∴ subset of SCC.

(b) if doesn't start from s cond^n then we just need cycle.

Role of s is special.
↳ How?

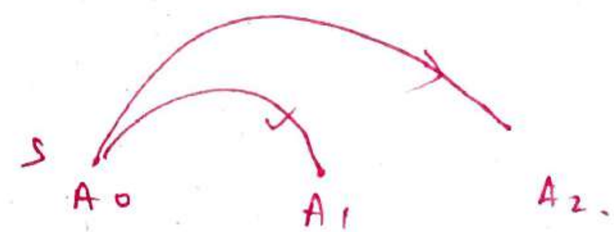
(i) If s is part of SCC then done

(ii) that — Now we need to check that s is connected to SCC with ≥ 2 nodes..

~~look at reachability from S.~~
Start DFS from s & check if able to reach an SCC with ≥ 2 nodes.

(c) Look at all SCC reachable from S.

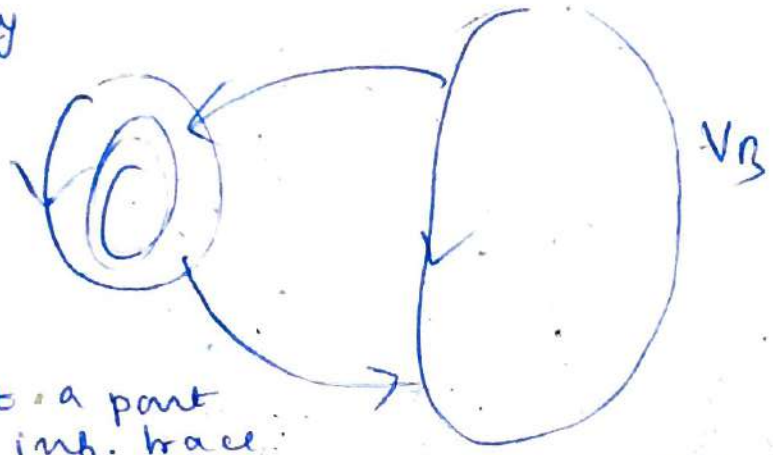
If any intersect with V_g then yes.



$|N_q \cap A_i| \geq 1$

~~$|\mathrm{Inf}(v) \cap A_{v_j}| \geq$~~

(d) Slightly tricky



↓

Like V_B could be part of SCC which is not a part of the inf. trace.

V_B → remove then use V_C

then ~~V_B~~ use fact that V_B can be there but only finitely many times.

Algo ? — Need to ask Sir.

13/11/24

Lec -

3.16 (Ask)

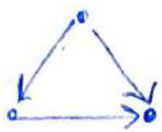
Remove source nodes

switching argument

Greedy & longest path go hand in hand.

ith step of greedy algo ≡ ith node of longest path.

BFS won't work



DP[i] ← longest path that ends at i

& B. Base case is source node.

3.17

(d) Remove V_B vertices & see paths.

How do we recover paths that visit V_B finitely (we only missed those that ~~were connected~~ ~~node btw~~ had ~~S btw~~ a V_B btw S & scc.)

Remove incoming edges from V_B & only keep outgoing edges.

These V_B's become sources & see if infinite trace from them.

Then look if g. goes to those V_B's.

Sol^n 2. Only keep V_B reachable from s (remove all incoming edges to V_B. & connect ~~V_B~~ s to V_B with an edge)

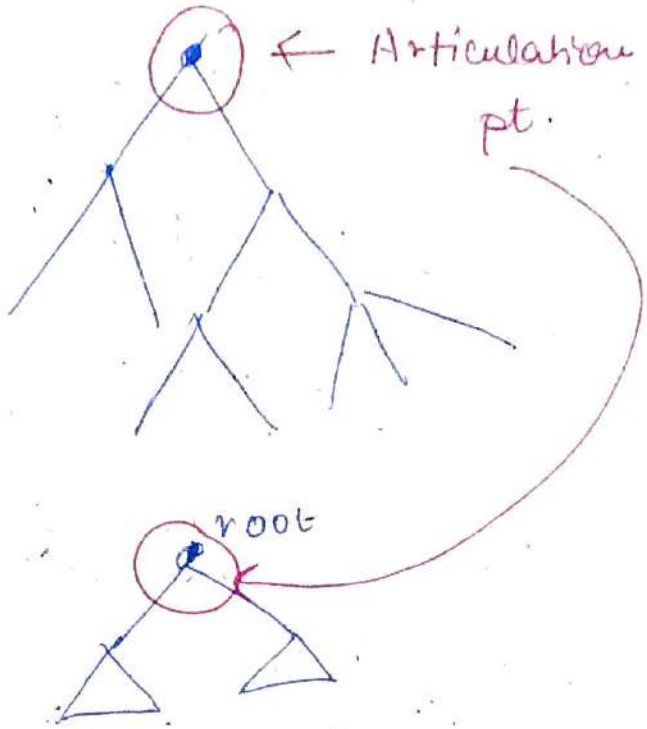
Lecture

(1)

Lec #2 Articulation Pt & Bridges

Assume : Graph is connected

Biconnected : each edge part of some cycle.



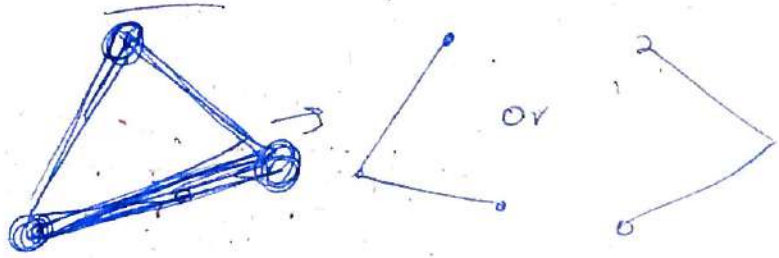
Articulation

Remove Hu pt & see if a graph becomes disconnected

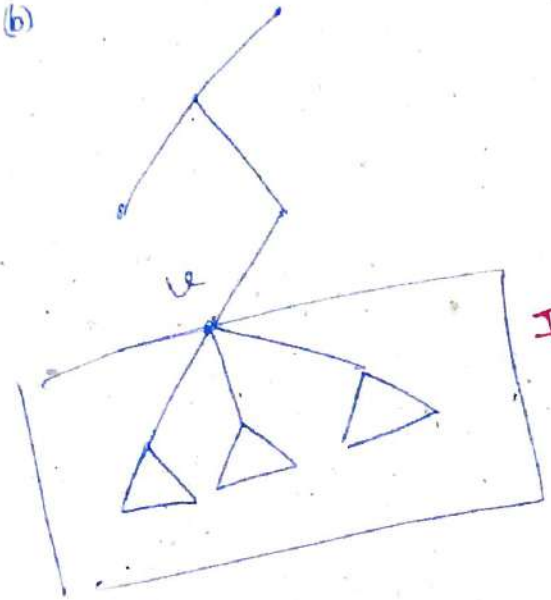
In undirected ~~edge~~ tree — fwd edge & cross edge are absent ??

(fwd edge ≈ back edge in undirected)

DFS



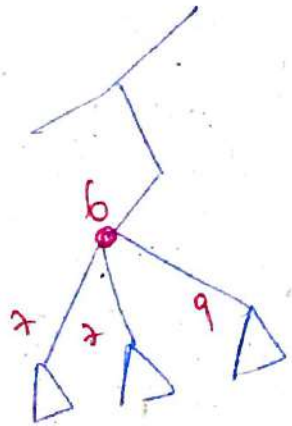
(b)



If there is no edge going from descendant/child of descendant. proper to ~~parent~~ ~~descendant~~ ascendant.

v.d ⇒ discovery time ≃ start time.

Be careful to check !!!



v.low = min { 6, 2, 2, 1, 5 }

~~How do we get~~

How do we get the array

~~# How do we get that array~~

(~~do recursion~~). (do recursion.

Post order traversal of DFS tree

(

∀ u ∈ V,

v.low ∈ V.S.

v.low ← level shortest ancestor from node.

child.low

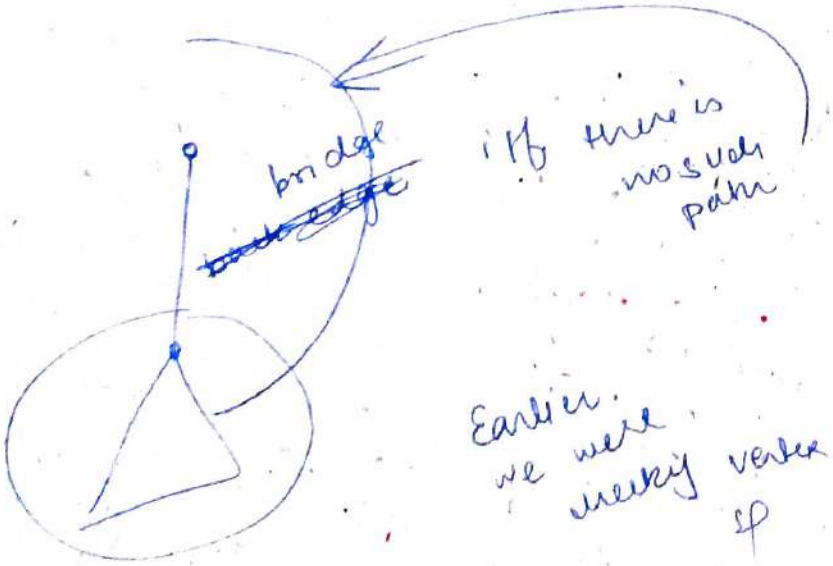
if ~~v.low~~ < v.s] ⇝ articulation pt

(e) Bridge

part of cycle ⟶ No other way if we remove edge.

back edges are all part of cycle.

Bridges ⊆ Tree edge.



iff there is no such path

Earlier we were merging vertex up now edge.

If we take out bridge edges then we are left with biconnected components

(∵ if not part of cycle ⇒ bridge
if taken out)

Biconnected component ⇒ DIY