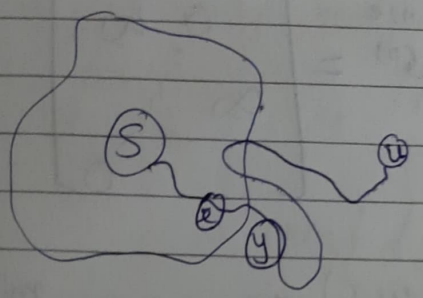
→ Induction → Basis → no. v. captured ∴ Inv 1 & 2 are satisfied

IS: u is the min extracted.
$\therefore u.d = \delta(s,u)$
- assume $u.d \neq \delta(s,u)$ SP from s to u
$\therefore$ must use uncaptured vertices.



$u.d > \delta(s,u) \geq \delta(s,y) = y.d$
$\therefore$ extract min should have given y instead of u.

→

	PQ	Binomial Heap	Fib heap
n Inserts in Q.		$O(n)$	$O(n)$
n Extract-mins		$O(n \log n)$	$O(n \log n)$
$O(m)$ Decrease-key		$O(m \log n)$	m
		$O((m+n)\log n)$	$O(m + n\log n)$

for less dense graph Binomial will use less tim

⇒ <u>All pairs Shortest Paths (APSP)</u> :-

$\forall (u,v) \in E^2$ find the SP from u to v.

- Matrix multiplication.

$w_{ij} = 0$, if $i=j$
$= $ weight of edge (i,j), if $(i,j) = E$
$= \infty$, otherwise.

$W = (w_{ij})$

$[W^{(1)} = W]$

→ define $W^{(m)}$ in general

s.t. $w_{ij}^{(m)}$ = the length of the SP from i to j using $\leq m$ edges.

- $W^{(0)}$: $w_{ij}^{(0)} = \begin{cases} 0 & \text{if } i=j \\ \infty & \text{otherwise} \end{cases}$

$$W^{(0)} = \begin{bmatrix} 0 & & & \\ & 0 & \infty & \\ & \infty & \ddots & \\ & & & 0 \end{bmatrix}$$

- $W^{(m-1)}$:

$(w_{i1}^{(m-1)} + w_{1j}) \min (w_{i2}^{(m-1)} + w_{2j}) \min \dots \min (w_{in}^{(m-1)} + w_{nj})$

$= W^{(m)}$

$w_{ij}^{(m-1)} + w_{jj} = w_{ij}^{(m-1)}$

$$\min_{k=1}^{n} (w_{ik}^{(m-1)} + w_{kj})$$

$\therefore W^{(m)} = W^{(m-1)} \times W$

	Std mM	here
	+	min
	*	+
	0	∞
	1	0

→ W^0

$W^{(1)} = W$

$W^{(2)} = W^{(1)} \times W = W^2$

$W^{(3)} = W^{(2)} \times W = W^3$

:

$W^{(k)} = W^k$

$W^{(n-1)} \longleftarrow$ APSP

Each matrix mul^n $O(n^3)$

$\therefore$ all mul. can be done in $O(n^3 \log n)$ time.

⇒ Floyd Warshall Algorithm (DP) $O(n^3)$

$d_{ij}^{(k)}$ = the length of SP from i to j using only $\{1, \ldots k\}$ as intermediate vertices.

$d_{ij}^{(0)}$ = only empty set available, no intermediates.

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k=0 \\ \min\{ \underset{①}{d_{ij}^{(k-1)}}, \underset{②}{d_{ik}^{(k-1)} + d_{kj}^{(k-1)}} \} \end{cases}$$

→ π_{ij} = the predecessor of j in the shortest path from i to j.

$\pi_{ij}^{(k)}$ = " using $\{1, \ldots k\}$ as intermediate.

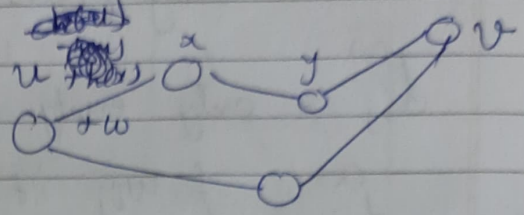
$$\pi_{ij}^{(k)} = \begin{cases} \text{if } k=0, \ i=j \rightarrow \text{nil} \\ i \neq j \ \& \ (i,j) \in E \rightarrow i \\ (i,j) \notin E \rightarrow \text{nil} \\ ① \ \pi_{ij}^{(k-1)} \\ ② \ \pi_{kj}^{(k-1)} \end{cases}$$

Fib Heap → Dijkstra's n times → $O(n^2 \log n + nm)$ $\leq O(n^3)$

BF → $O(n^2 m)$

⇒ Johnson's Algorithm:

each city v, h(v) allowance of v



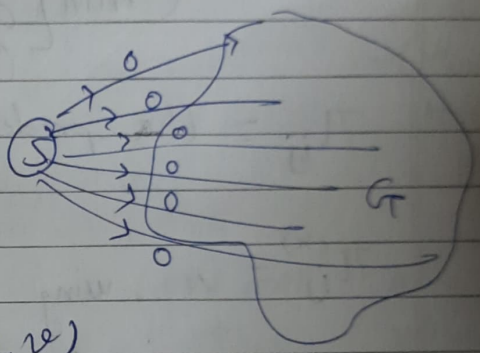
$u \to x$: $+h(u) - h(x) + w(u,x)$

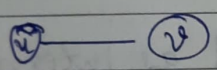
$x \to y$: $h(x) - h(y) + w(x,y)$

∴ total expandeture: $h(u)$ + path weight ϕ - $h(v)$

- Bellman Ford → check -ve cycle

$h(v) = \delta(s,v)$





$\delta(s,v) \leq \delta(s,u) + w(u,v)$

$0 \leq h(u) + w(u,v) - h(v)$

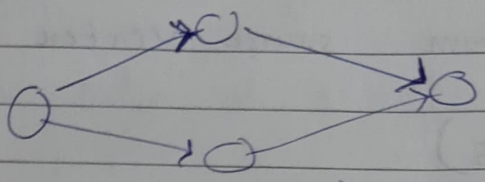
BF $O(nm)$

update. $O(m)$

n. Dijkstras $O(n^2 \log n + nm) \leq O(n^3)$

Floyd warshall

⇒ Directed Acyclic Graph:

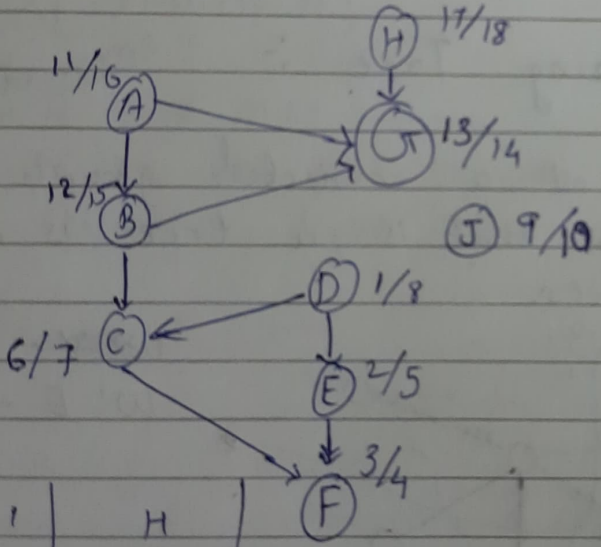


No directed cycle

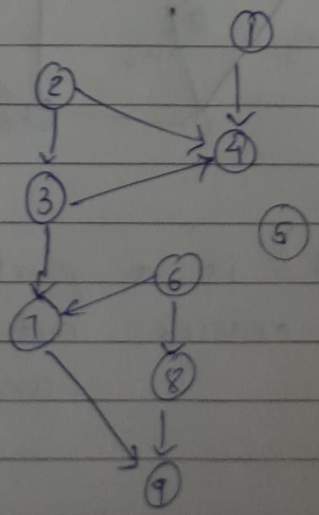
↳ not a tree.

- Topologically sorting of a DAG.

number vertices so that every edge is from a smaller to a larger no.



1	H
2	A
3	B
4	G
5	J
6	D
7	C
8	E
9	F



O(n+m)

Topo Sort (G)
- call DFS (G) → finish time
to compute v.f
for each v ∈ V
- stack the vertices in the finish order.
- return stack

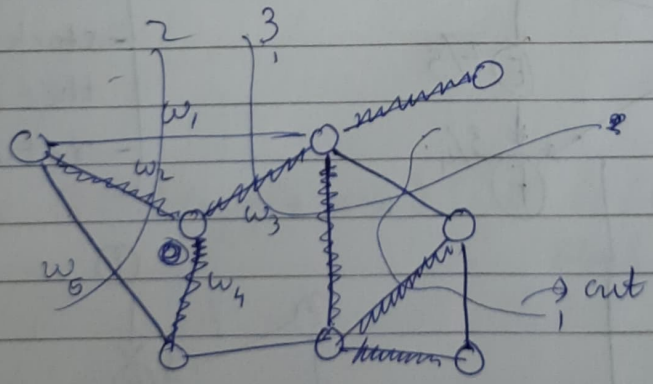
⇒ SSSP on DAG:-

Directed SPs from sourse vertex s.

```
Dag SPs (G, w, s)
[ TopoSort (G)                          O(n+m)
  Initialize (G, s);                    O(n)
  for each u in topological order       O(m)
    [ for each v ∈ Adj[u]
        Relax (u, v, w);
```

⇒ Minimum spanning Tree :-

spanning-tree of a connected graph G is a subgraph of G that is a tree is connected & has n vertices.



w: E → R

cut (S, V-S) is a partition of the graph.
(u, v) ∈ E crosses, cut (S, V-S) if its endpoints are on opposite sides.

a cut $(S, V-S)$ respects set of edges A if no edges of A crosses the cut.

a light edge of a cut is a minimum weight edge crossing it.

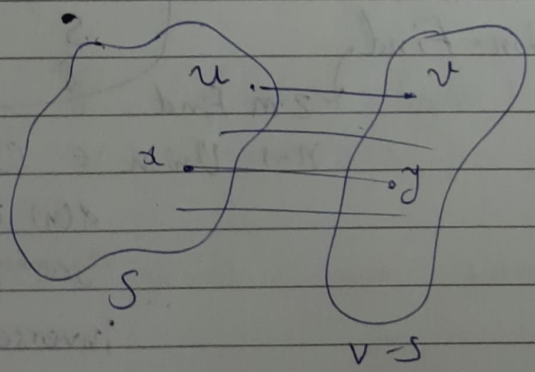
Framework - MST (G, w)

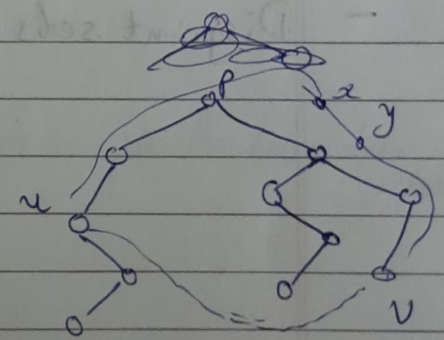
```
A = ∅
while A is not a ST yet
    find a safe edge (u,v) for A
    A = A ∪ {(u,v)}
return A;
```

A some subset of E
$A \to\to MST.T$
(u,v) is safe for A
if $A \cup \{(u,v)\} \to\to MST$

Theorem:

$A \to\to MST\ T$
$(S, V-S)$ that respects A
(u,v) is light edge crossing $(S, V-S)$ $\}\to$ (u,v) is safe for A.





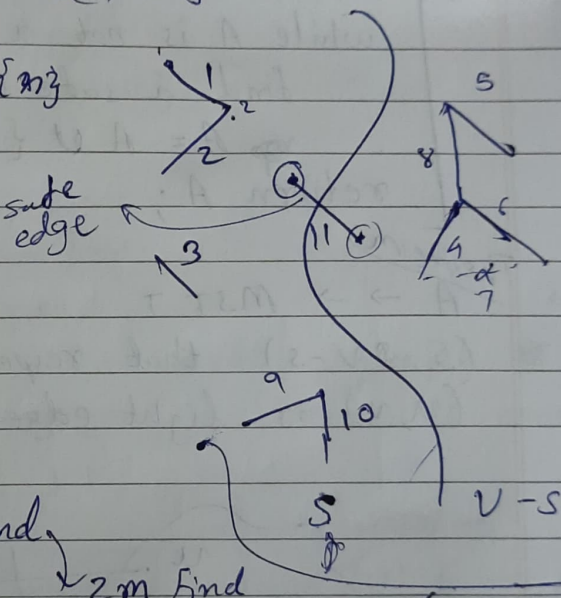
if $A \to\to T$
$A \to\to \{T - \{(x,y)\}\} \cup \{(u,v)\}$ also a MST.

→ Kruskal (G, w)

- Sort the edges on weights — $O(m \log m)$
- consider the edges in increasing order of weights. , $A = \emptyset$
- while (there is edge (u,v) left)
  - if (u,v) forms a cycle with A
    - then discard (u,v)
  - else $A = A \cup \{(u,v)\}$

initially $\{1\}\{2\}\{3\}\ldots\{n\}$

- lightest edge $(1,2)$
  $\{1,2\}\{3\}\ldots\{n\}$
- 2^{nd} lightest edge $(2,3)$
  $\{1,2,3\}\{4\}\ldots\{n\}$
  $x_1 \quad x_2 \quad x_k$



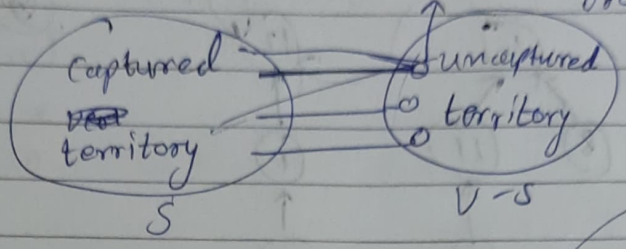
- Disjoint sets - Union-Find
  - $2m$ Find
  - $n-1$ Union. $\leftarrow O((m+n)\cdot\alpha(n))$

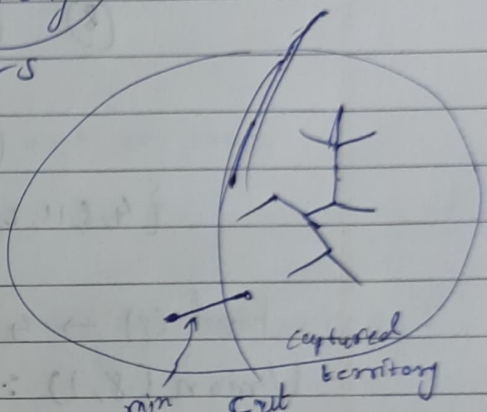
$\alpha(n)$ is extremely slow growing f^n.

inverse ackermann f^n

→ **Prim's Algorithm :-**

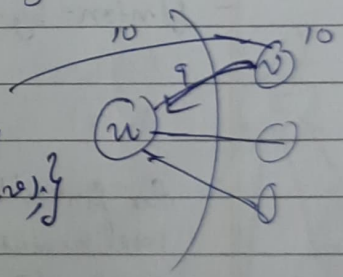
the weight of lightest edge from s to v. it.





- Prim (G, w, r)

```
for u ∈ V { u.key = ∞, u.π = nil }
r.key = 0 ;
Q ← V ;
while ( Q ≠ ∅ )
    u = Extract-Min (Q)
    for each v ∈ Adj[u]
        [ if v ∈ Q and w(u,v) < v.key
            { v.π = u ; v.key = w(u,v); } ]
```

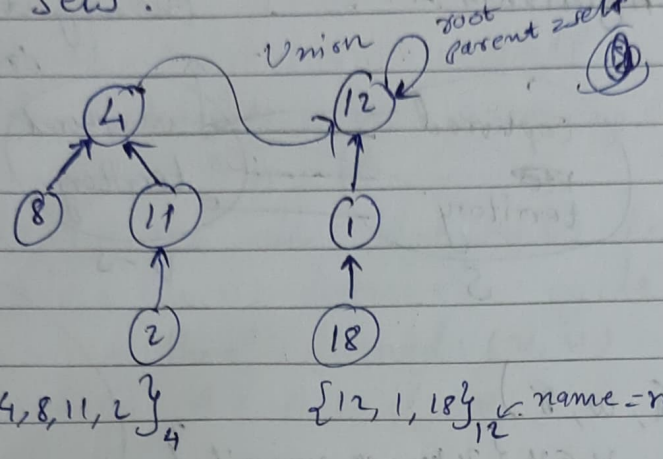


P : $O(m + n\log n)$

K : $O(m \log n)$ easier code & for lesser data better

for ~~D~~ $v \in Q$ use bitmap.

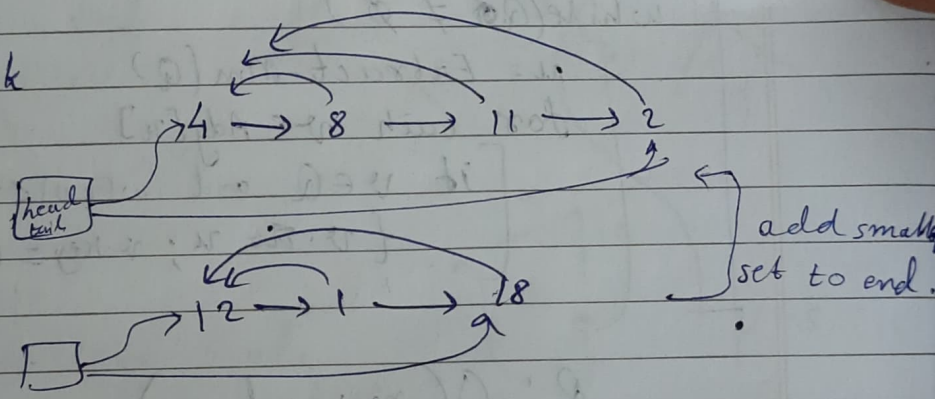
⇒ Disjoint sets :-



Find (8) → 4 , Find (1) → 12

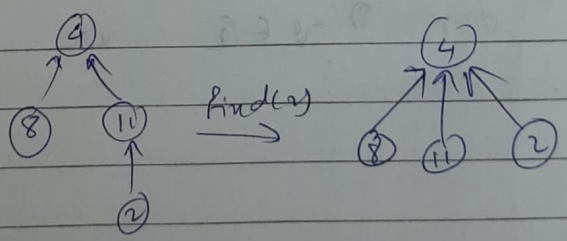
- Union (8, 1) :→ find(8) & (1) then join trees.

- Union-by-rank



$O(n \log n)$ for n unions.

- path compression



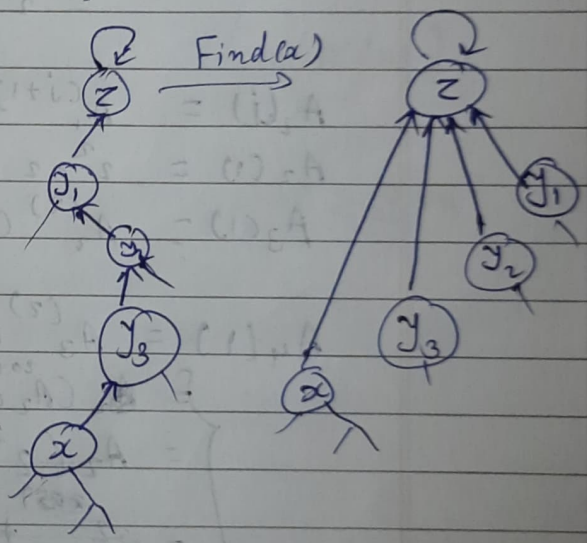
→ Disjoint set Union Find:

- MakeSet (x)
  - x.p = x
  - x.r = 0

- Union (x, y) → returns root of tree.
  - Link (Find (x), Find (y)) UwR.

- Link (x, y) ⟶ if (x == y) return 0;
  - if (x.r > y.r)
    - y.p = x;
  - else
    - x.p = y;
    - if x.r == y.r;
      - y.r++;
  - return 1;

- Find (x)
  - if (x ≠ x.p)
    - x.p = Find (x.p);
  - ~~else return x;~~
  - return x.p;



Find with PC (path compression):

- UwR: $O(m \log n)$ # make sets n, Unions n-1, Finds
- Find with PC: $O\left(f \frac{\log n}{\log\lceil f/n \rceil}\right)$ $O(m \cdot \alpha(n))$

→ $A_k(j) = \begin{cases} j+1 & \text{if } k=0 \\ A_{k-1}^{(j+1)}(j) & \text{if } k \geq 1 \end{cases}$

- $A_0(j) = j+1$

$A_0(A_0(j)) = A_0(j+1) = j+2$

$A_0^{(3)}(j) = j+3$

$A_0^i(j) = j+i$

- $A_1(j) = A_0^{(j+1)}(j) = j+j+1 = 2j+1$

$A_1(A_1(j)) = A_1(2j+1) = 4j+3$

$A_1^{(3)}(j) = A_1(4j+3) = 8j+7$

$A_1^i(j) = 2^i j + 2^i - 1$

$= 2^i(j+1) - 1$

$A_2(j) = A_1^{(j+1)}(j) = 2^{j+1}(j+1) - 1$

$A_2(1) = 2^2 \cdot 2 - 1 = 7$

$A_3(1) = A_2^{(2)}(1) = A_2(A_2(1)) = A_2(7) = 2^8 \cdot 8 - 1$

$= 2047$

$A_4(1) = A_3^{(2)}(1) = A_3(A_3(1)) = A_3(2047)$

$= A_2^{2048}(2047)$

$= A_2(A_2^{2048}(2047))$ ~~$= A_2(2^{2048} \cdot 2048 - 1)$~~

~~$= A_2(2^{2059} - 1)$~~

~~$= 2^{2059} \cdot 2^{2059} - 1$~~

$= A_2^{2048}(2047)$

$>> A_2(2047)$

$= 2^{2046} \cdot 2048 - 1$

$> 2^{2048} = 16^{512} >> 10^{80}$

$$\alpha(n) = \min\{ i \mid A_i(1) \geq n \}$$

→ **Lemma 1:-** $\forall x \in V$, $x.r \leq x.p.r$

$x.r < x.p.r$ if $x \neq x.p$

initially $x.r = 0$.

and ↑ through time.

$x.p.r$ ↑ through time.

strictly inc.

Cor 2: On any node-root path ranks ↑

Lemma 3: $x.r < n-1$

Lemma 4:-

→ S a set of m' operations

n make sets $\quad O(m'\alpha(n))$

$\leq n-1$ unions

Finds

$m \leq 3m'$

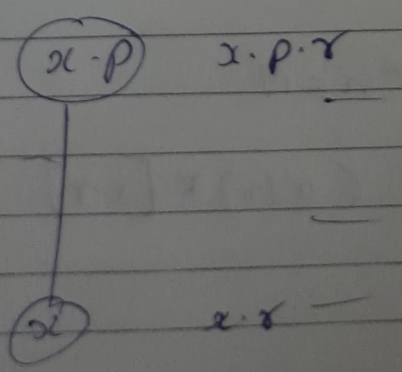
S of m operations — n make sets

$\leq n-1$ Link

Finds

$O(m\alpha(n))$

→ Level (x): $\max\{ k \mid A_k(x.r) \leq x.p.r \}$



~~$Iter(x) = \max\{ A_x^i \; x \cdot i$~~

→ $Iter(x) = \max\{ i / A^i_{level(x)}(x.r) \leq x.p.r \}$

→ $\phi_q(x) = \begin{cases} \alpha(n) * [x.r] & , \text{if } x \text{ is a root } \vee \; x.r = 0 \\ (\alpha(n) - level(x)) * [x.r] - Iter(x) & , \text{otherwise.} \end{cases}$

→ Lemma 5:

$0 \leq level(x) < \alpha(n)$

① $x.p.r \geq x.r + 1 = A_0(x.r)$

$\downarrow$

$level(x) \geq 0$

$A_{\alpha(n)}(x.r) \geq A_{\alpha(n)}(1) \geq n > x.p.r$

$\downarrow$

$level(x) < \alpha(n)$

→ Lemma 6: $1 \leq Iter(x) \leq x.r$

$A^{(1)}_{level(x)}(x.r) \leq x.p.r \rightarrow Iter(x) \geq 1$

$A^{(x.r+1)}_{level(x)}(x.r) = A_{level(x)+1}(x.r) > x.p.r$

→ Lemma 7: $\forall x \in V, \; \forall n \in N$

$0 \leq \phi_q(x) \leq \alpha(n) * [x.r]$

① $x.r = 0$ $\quad \phi_q(x) = 0$

② x is a root $\quad \phi_q(x) = \alpha(n) * [x.r]$

③ $x.r \neq 0$, x is non root

$\phi \uparrow$ — level & iter $\downarrow$

$$\phi_q(x) \geq (\alpha(n) - \alpha(n)-1) * [x.r] - x.r = 0.$$

$$\phi_q(x) \leq (\alpha(n) - 0) * [x.r] - 1 = \alpha(n) * [x.r] - 1$$

→ Lemma 8:- x is a non root.

Find, Link → $\phi_q(x) \uparrow^{\alpha}$ doesn't inc

if $(x.r \geq 1) \wedge$ (either Level(x) or Iter(x) changes)

$\phi(x) \downarrow$

Proof:

q^{th} operation = Find / Link. $x.r \neq n.\alpha(n)$

level(x) can only ↑ or remain same.

Iter(x)

↳ if level(x) same Iter(x)

↳ if level(x) ↑ → Iter(x) can ↓

→ Level(x), Iter(x) $\overset{same}{\leftrightarrow}$ $\phi(x)$ same $\leftrightarrow$

Level(x) $\leftrightarrow$ Iter(x) ↑ $\phi(x) \downarrow$

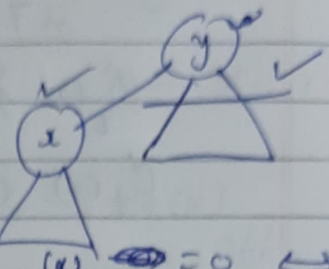
Level(x) ↑ Iter(x) $(x.r-1) \downarrow$ $\phi(x) \downarrow$

} $\phi(x) \uparrow^{\alpha}$

Lemma 9: MakeSet's Am.cost is $O(1)$.

Lemma 10: Link's Am cost is $O(\alpha(n))$.

proof: AC cost : $O(1)$



y's children : ϕ ↑ by lemma 8

x: $\phi_{i-1}(x) = \alpha(n) * [x.r]$

if $(x.r = 0)$ $\phi_i(x) = 0$ $\phi_{i-1}(x) = 0$ ↔

if $(x.r > 0)$ $\phi_i(x) \leq \alpha(n)[x.r] = \phi_{i-1}(x)$ ↓

y: $\phi_{i-1}(y) = \alpha(n) * [y.r]$

if $y.r$ ↔ , $\phi(y)$ ↔

if $y.r$ ↑ $\phi(y)$ ↑ by $\alpha(n)$

ϕ diff ≤ $\alpha(n)$

∴ Am cost = $O(\alpha(n))$

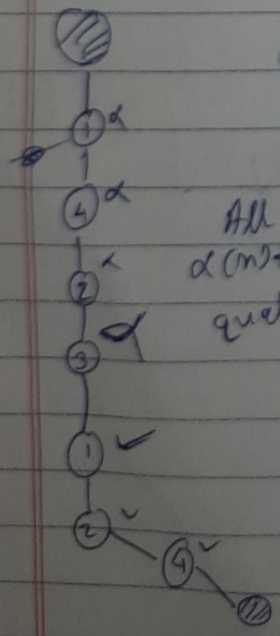
Lemma 11: Find's Am cost is $O(\alpha(n))$

Ac. cost : $O(s)$ length of findpath = s.

claim: $\max\{0, s-(\alpha(n)+2)\}$ nodes have ϕ ↓ by ≥ 1.

$0 \geq s - (\alpha(n)+2)$ | $s - s + \alpha(n) + 2$

$\alpha(n)+2 \geq s$ | $= O(\alpha(n))$



All but $\alpha(n)+2$ qualities

x is on the find path, $x.r > 0$

suppose a proper ancestor y of x has the same level as x.

level(x) = level(y) = k

$$x \cdot p \cdot \gamma \Rightarrow A_k^{(\text{Iter}(a))} (\alpha \cdot \gamma)$$

$$y \cdot p \cdot \gamma \Rightarrow A_k^{-} (y \cdot \gamma)$$

$$y \cdot \gamma \Rightarrow x \cdot p \cdot \gamma$$

$$x \cdot \rho \cdot r \geq A_k^{(\text{Iter}(a))} (o(\cdot r))$$

$$y \cdot \rho \cdot r \geq A_k^{-} (y \cdot r)$$

$$y \cdot r \geq x \cdot \rho \cdot r$$

→ Efficiency - Time complexity.

Approximation, Randomized $(1+\epsilon)$

Optimal x ⟶ too hard

close to it ~

error bound

ρ - approximation algo

$$\max\left\{\frac{\vec{c}}{c^*}, \frac{c^*}{c}\right\} \leq \rho$$

- $O(n^k)$ prob. of fail - $\left(1 - \frac{1}{n^e}\right)^r$

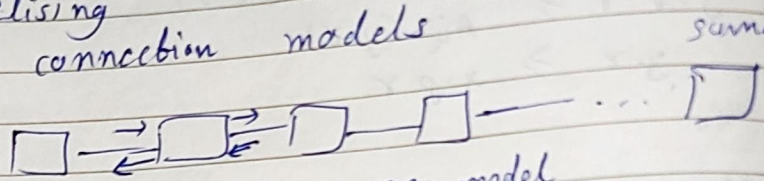
Randomized ⟨ Las vegas
Monte Carlo

- parallelising multiple instrs runs at a time

sequential algo 1 instr runs at a time

⟹ Parallelising

→ inter connection models



Linear array model

- Odd Even transposition sort

7 5 3 8 1 6 4 2

5 7 3 8 1 6 2 4

5 3 7 1 8 2 6 4

3 5 1 7 2 8 4 6

3 1 5 2 7 4 8 6

1 3 2 5 4 7 6 8

1 2 3 4 5 6 7 8

✓

- 0 1 0 0 1 1 0 0

rightmost '1' moves in 1st move : N-1 moves (reaches ~~right~~ pos.)

" in 2nd move : N-2 moves

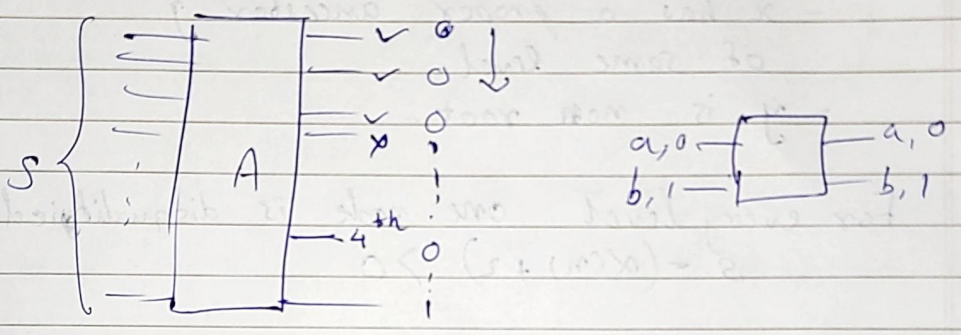
2nd rightmost 1 → adj to RM 1 → ~~move in~~ 1st odd, even

→ adj to 0 → ~~even~~ even, odd

every 1 will reach correct pos. in N-1 steps.

→ **0-1 principle :-**

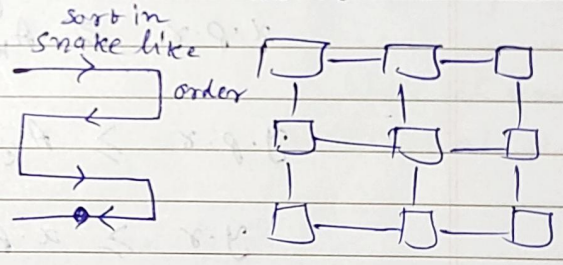
Any sorting algorithm that works on Binary ~~string~~ sequence, will work for any other sequence.



→ 2-D sorting in 2D mesh connection model.

$\sqrt{N} \times \sqrt{N}$ mesh

4	9	7	3
12	16	11	15
10	2	5	14
6	13	8	1



row

3	4	7	9
16	15	12	11
2	5	10	14
13	8	6	1

col →

2	4	6	1
3	5	7	9
13	8	10	11
16	15	12	14

↓ row

1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

←

1	2	4	3
8	7	5	6
9	10	11	12
16	15	14	13

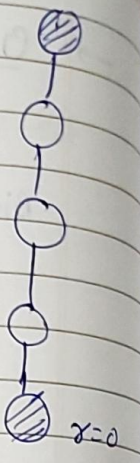
col ←

1	2	4	6
9	7	5	3
8	10	11	13
16	15	14	12

$O(\log N)$ phase of $O(\sqrt{N})$ steps

$P = N$, $T = \sqrt{N} \log N$, $pT = N^{3/2} \log N$ cost

⇒ Am cost of Find $O(\alpha(n))$.



S $O(s)$ actual cost

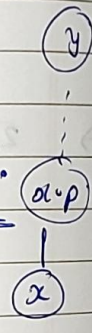
node x on the find path s.t.

non root $x.r > 0$

- x has a proper ancestor y of same level.
- y is non root.

For every level one node is disqualified

$s - (\alpha(n) + 2) > 0$



$k = level(x) = level(y)$

$x.p.r \geq A_k^{(Iter(x))}(x.r)$ $(Iter(x) = i)$

$y.p.r \geq A_k(y.r)$ $(level(y))$

$y.r \geq x.p.r$

$y.p.r \geq A_k(y.r) \geq A_k(x.p.r) \geq A_k(A_k^{Iter(x)}(x.r))$

$= A_k^{i+1}(x.r)$

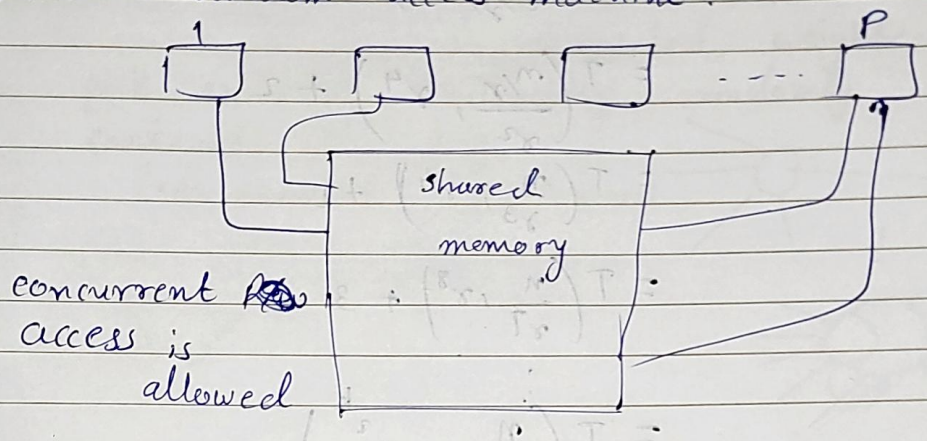
PC↓ $x.p = y.p$ so, $x.p.r = y.p.r$

$y.p.r$ has not ↓

$\therefore x.p.r \geq A_k^{(i+1)}(x.r)$

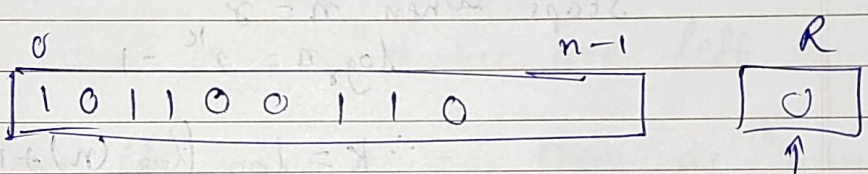
$\therefore \phi_{diff} = s - (\alpha(n)+2)$ $\therefore$ Am cost $= O(\alpha(n))$

→ Parallel random access machine :-



- Concurrent write allowed only if all write same value.

→



OR of A[0] ··· A[n-1]

initial write 0 by all p.

→ p = N-1 ;

P_i : if (A[i] == 1) R = 1 ;

→

	4	3	0	2
4	0	0	0	0
3	1	0	0	0
0	1	1	0	1
2	1	1	0	0

n

0 → max

1

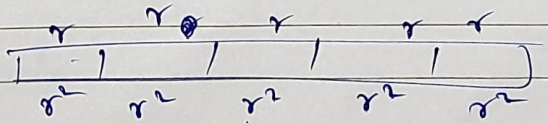
1

1

n^2 processors

O(1) max of n int.

- n·r processors



$\frac{n}{r} \times r^2 = nr$

max of each in O(1).

If n → n/r

$$T(n, r) = T\left(\frac{n}{r}, r^2\right) + 1$$

$$T(n,r) = T\left(\frac{n}{r}, r^2\right) + 1$$

$$= T\left(\frac{n/r}{r^2}, r^4\right) + 2$$

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

$$= T\left(\frac{n}{r^7}, r^8\right) + 3$$

$$= T\left(\frac{n}{r^{2^k-1}}, r^{2^k}\right) + k$$

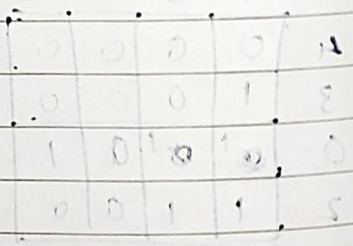
stops when $n = r^{2^k - 1}$

$$\log_r n = 2^k - 1$$

$$\therefore k = \log_2(\log_r(n) + 1)$$

if $r = 2$;

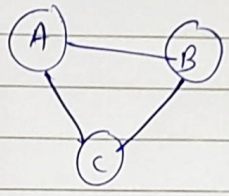
$O(\log \log n)$ time using $2n$ processors.

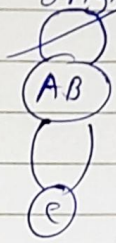


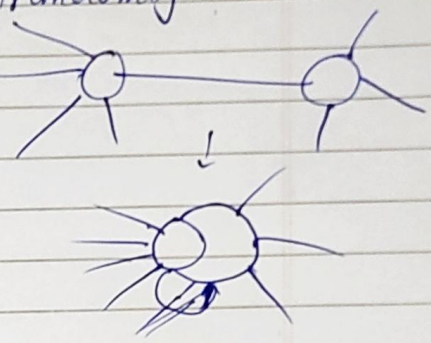
⇒ Min cut of a graph:

undirected, weighted

- pick an edge uniformly randomly
- contract this edge
  (edge remember original names)







- Keep multiedges, remove self loops

→ Repeat until only 2 vertices are left.

Declare the edges between them as the min cut.

- we can show that this is indeed a mincut is $\frac{2}{n^2}$
- prob. of failure = $\left(1 - \frac{2}{n^2}\right)$
- repeat by $\frac{n^2}{2}$: failure $\left(1 - \frac{2}{n^2}\right)^{\frac{n^2}{2}} < \frac{1}{e}$

Repeat n times

$$P(\text{failure}) < \frac{1}{e^n}$$

- $O(n^3)$
- monte carlo