Longest ...

Greedy Algorithms

- Make a locally optimal choice in hope of getting a globally optimal soln.
- $O(nW)$ — pseudo-polytime algo

Activity Selection:

- n activities — exclusive use of common resource

$S = \{a_1, \ldots, a_n\}$

$a_i = [s_i, f_i)$ $\quad f_i < f_j$ for $i < j$

Goal:- Select the largest possible set of non-overlapping activities.

i	1	2	3	4	5	6	7	8	9
s_i	1	2	4	1	5	8	9	11	13
f_i	3	5	7	8	9	10	11	14	16

$a_1\ a_5\ a_6\ a_8$ $\quad a_2\ a_5\ a_7\ a_9$ are all answers

$a_2\ a_5\ a_7\ a_8$

04/02/2023

- Greedy choice Property :- A.T Greedy choice is part of optimal sol^n.

$S_{ij} = \{a_k / f_i \le s_k < f_k \le s_j\}$

Thm :- Let $S_{ij} \neq \phi$, Let a_m be the activity in S_{ij} with the earliest finish time.

$$f_m = \min\{f_k / a_k \in S_{ij}\}$$

1) a_m is used in some max-size of mutually compatible activities of S_{ij}.

2) $S_{im} = \phi$, So choosing a_m leaves S_{mj} as only non-empty subproblem.

i	1	2	3
v_i	60	100	120
w_i	10	20	30
$\frac{v_i}{w_i}$	6	5	4

$W = 50$

The optimal sol^n is not greedy -

22/2/23

Greedy algorithms :-

1) GCP :- Greedy choice is part of an optimal sol^n
2) Optimal sub structure

Cont^n :-

$a_0 = (-\infty, 0)$

$a_{n+1} = (\infty, \infty+1)$

Pf of Thm :-

2) $S_{im} = \phi$

Let $S_{im} \neq \phi$, $a_k \in S_{im}$ but $f_k < f_m$ (contradiction)

1) If a_m is part of $A_{ij} \in S_{ij}$, If $f_k < f_m$

$A'_{ij} = A_{ij} \setminus \{a_k\} \cup \{a_m\}$

04/02/2023 17:38

Huffman Coding

	a	b	c	d	e	f
Freq	45	13	12	16	9	5
fixed length	000	001	010	011	100	101
Variable length	0	101	100	111	1101	1100

Encoding :-

$\Gamma = \{a, b, c, d\}$

$c_1 \{a=00, b=01, c=10, d=11\}$

~~bad~~ ~~0100111~~

$c_2 = \{a=0, b=110, c=10, d=111\}$

$c_3 = \{a=1, b=110, c=10, d=111\}$

0100 11 — bad w.r.t c_1

110 1111 — bad " c_2

110 1111 — bad, acad " c_3

Prefix-free code is uniquely decipherable (The code of one letter should not be the prefix of another)

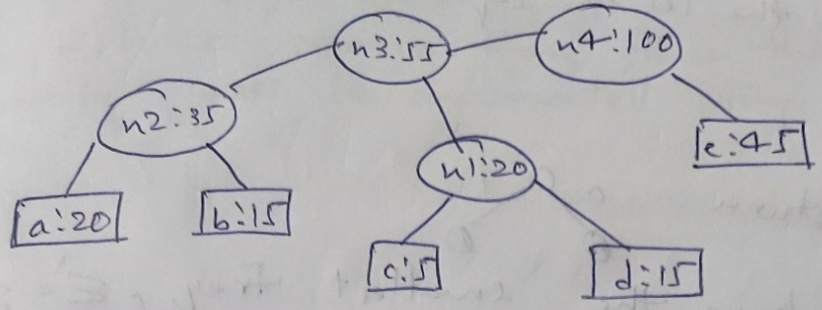
prefix code ≡ prefix-free code

Problem :- Given an alphabet $A = \{a_1, \cdots, a_n\}$ with freq $f(a_i)$. Find a binary prefix-code C for A that minimizes the no of bits.

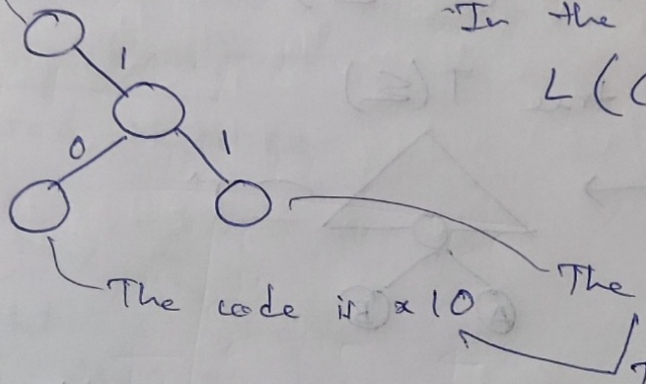
$$B(C) = \sum_{i=1}^{n} f(a_i) \underbrace{L(C(a_i))}_{\text{Length of code of } a_i}$$

04/02/2023 17:38

$A = \{a/20, b/15, c/5, d/15, e/45\}$



Why full binary :-



In the algo,
$L(C(a_i)) = d(a_i)$

There can be written as ∝1, ∝0 resp] so it must be full binary.

23/2/23

Lemma :- ∝ & y two letters with smallest frequencies. There is an optimal code tree in which these two letters are sibling leaves in the tree in the lowest level.

Pf:- "b & c" are siblings. $f(b) \le f(c)$

$f(x) \le f(y)$

$B(T) \le B(T')$

depth

$= B(T) - f(x)d(x) - f(b)d(b) + f(x)d(b) + f(b)d(x)$

$= B(T) - (f(b) - f(x))(d(b) - d(x))$

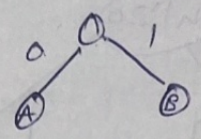
$\le B(T)$

Queue with 1 stack is possible.

Edit distance on the set of sequences is a metric sp

13/3/23

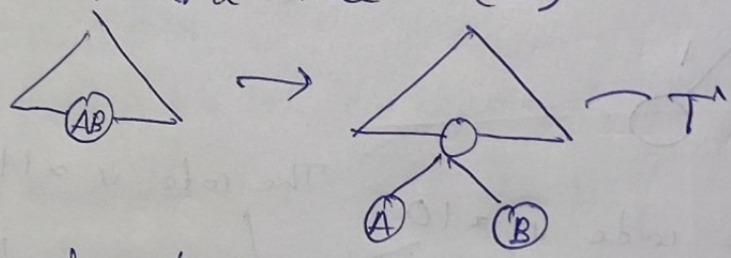
Algo

if $|\Sigma'| = 2$ return 

Let $a, b \in \Sigma$ have the smallest freq, $\Sigma' = \Sigma$ with a, b replaced by $\rho_{ab} = \rho_a + \rho_b$

Recursively compute T' (for Σ')

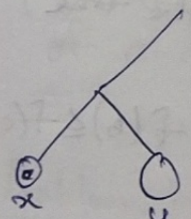
Extend T' to a tree $T(\Sigma)$



Proving correctness :-

Out of all trees with A & B as siblings, P.T T' is optimal.

Pf:- Suppose not.



$L(T^*) - L(T') \geq 0$

(T' is the tree obtained by $x \leftrightarrow a$, $y \leftrightarrow b$)

$L(T^*) - L(T') = (\rho_x - \rho_a) + (\rho_y - \rho_b)$

(x's depth in T^* − a's depth in T^*)

+ (y's depth in T^* − b's depth in T^*)

≥ 0

Running time :- $O(n \log n)$
(Priority queues) (n deletions each of log n time)

Is there a better running time than this?
Can priority queues be implemented using queues?

Scheduling problem :-
- one shared resource
- n jobs

w_i, l_i
weight, length

Completion time :-

$l_1 = 1$, $l_2 = 2$, $l_3 = 3$

$c_1 = 1$, $c_2 = 3$, $c_3 = 6$

I want to minimize the $\sum_{i=1}^{n} w_i c_i$

lengths are same : max weight
weights are same :- min length

Greedy algo :- Sort by $\frac{w_j}{l_j}$.

BFS, DFS

15/2/23

Graph Representation :-

Adj List:

Adj matrix:

BFS:- (A particular case of Dijkstra's algo)

i/p :- Graph $G=(V, E)$, $s \in V$
(directed/indirected)

o/p :- $d[v]$ = distance of v from s

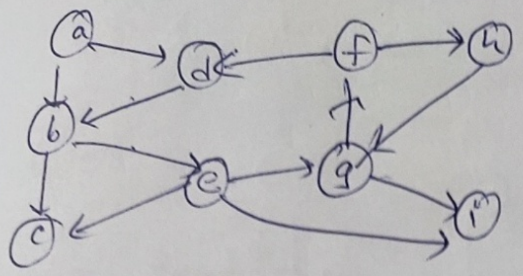
$\pi(v) = u \iff s \rightsquigarrow u \rightarrow v$ ('u' is the predecessor of v in the shortest path from s to v)

$\{(\pi(v), v) \mid v \in V \; v \neq s\}$ — BFS tree

$(d(v) = \infty \; \forall \; v \neq s, \; d(s) = 0)$

```
Q = {s}
while Q ≠ ϕ
do
    u ← dequeue(Q)
    for each v ∈ Adj[u]
    do
        if d[v] = ∞
        then
            ϕ(v) = u ←
            d(v) = d(u) + 1
            ENQUEUE(Q, v)
```

$\sum_{v \in V} d(v) = E$



Q: a b d c e
0 1 1 2 2

Proof:-

Invariant:- v comes after u in Q $\Rightarrow$ d[v] = d[u] or d[v] = d[u] + 1

2×2×2 — Rubik's cube diameter – 12

3×3×3 – 20

n×n×n $\Theta\left(\frac{n^2}{\log n}\right)$ – Erik demaine

DFS:-

I/p:- G = (V, E) directed / indirected, No source

O/p:- 2 timestamps on each vertex

d[v] = discovery time
f[v] = finishing time
} $\leq b^n$ | $\leq 2|V|$

$\pi(v)$

$d(v) < f(v)$

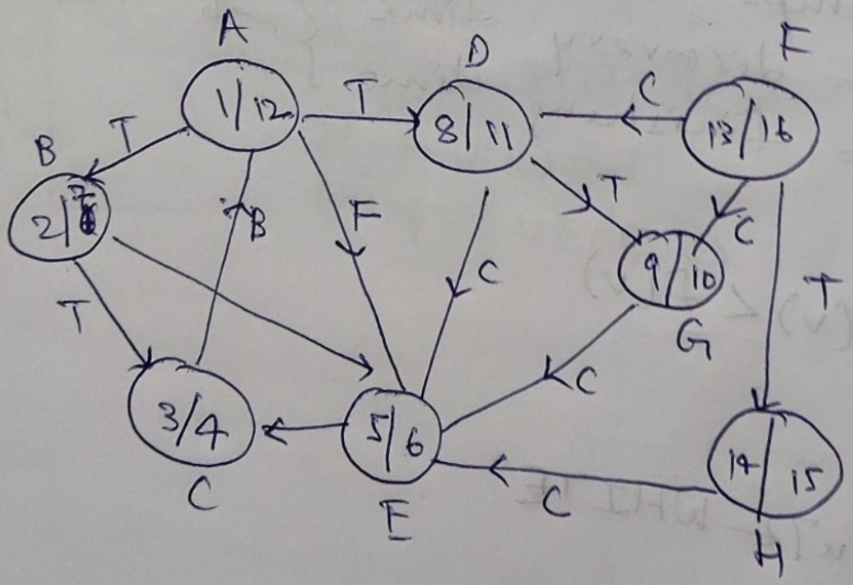
```
DFS(V,E)
for each u ∈ V
    do colour[u] ← WHITE
    time ← 0
for each u ∈ V
    do
        if colour[u] = WHITE
            then
                DFS_VISIT(u)
```

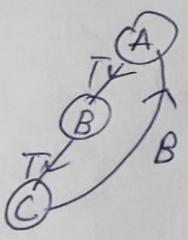
04/02/2023 17:38

```
DFS_VISIT(u)
    colour(u) ← GRAY
    time ← time + 1
     d[u] ← time
    for each v ∈ Adj[u]
        do
            if colour[v] = WHITE
            then
                DFS_VISIT(v)
        colour[u] ← BLACK
        time ← time + 1
         f[u] ← time
```

Edge classificat^{n}

1) Backedge :- Edge from descendant to ancestor
2) ~~Tree~~ :-
3) Forward :- " " ancestor " descendent





Complete the tree

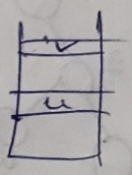
Parenthesis thm :-

For any vertices ~~(u,v)~~ u & v

$[d(u), f(u)]$, $[d(v), f(v)]$ are either disjoint or nested i.e either $d(u) < d(v) < f(v) < f(u)$ or $d(v) < d(u) < f(u) < f(v)$

Pf :- Argues using stack

$d(u) < d(v) < f(u) \Rightarrow f(v) < f(u)$



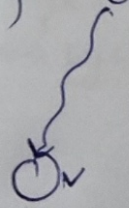
At the time $d(v)$,

u – GRAY and on stack

Corollary :- For $v \neq u$, v is a descendant of u in a DFS tree $\Leftrightarrow$ $d(u) < d(v) < f(v) < f(u)$

White path thm :- In a DFS forest of a graph G, vertex v is a descendant of vertex $u \Leftrightarrow$ at time $d[u]$ (just before coloring u to GRAY) there is a path from $u \rightsquigarrow v$ consists of only white colours.

Pf :- ($\Rightarrow$)



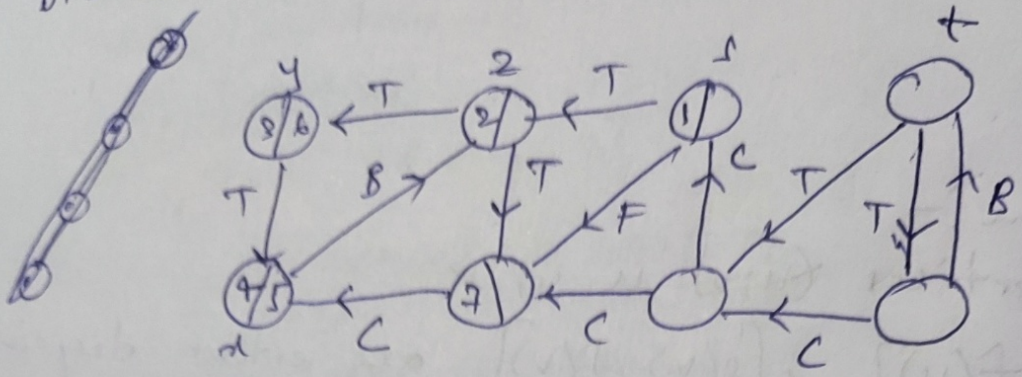
So there is a path from u to v in G.

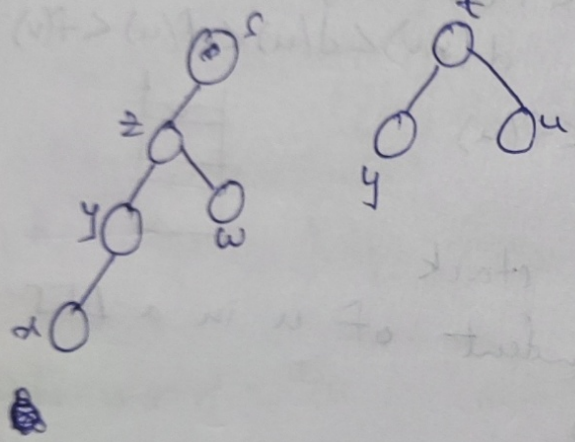
Every vertex in this path is also a descendant.

By prev corollary, all vertices are white.

17/3/23

DFS without stack (and more running time)





Thm:- In DFS of an undirected graph, we get only Tree and back edges. (No forward or cross edges).

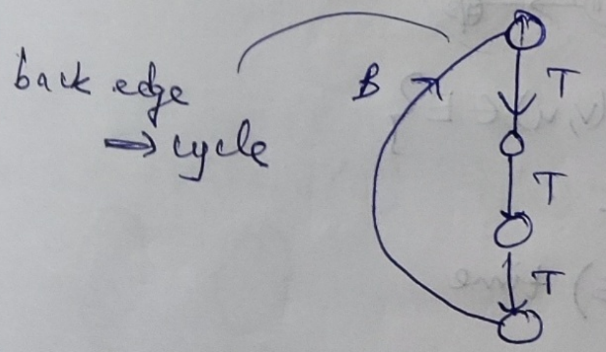
Applicat^{n}s:- Finding ~~a cyclic~~ g: whether a graph is cyclic/not?

Finding Bipartite graphs?

Top Sort:-

DAG ← Good models structures that have partial orders

Lemma:- A ~~DA~~ G ⟺ DFS of G yields no back edges.



TP:- cycle ⇒ back edge

$d[u], \exists u \xrightarrow{\text{white vertices}} v$ (white path thm)

v is a dependent of u in DFS

Top Sort of DAG :-

A linear ordering of vertices $\ni$ if $(u,v) \in E$, u appears somewhere before v.

Output vertices in order of decreasing finish time $\Theta(V+E)$

TOP_SORT (V, E)

DFS(V, E) to compute $f(v)$ $\forall v \in V$.

Correctness :- If $(u,v) \in E$, then $f(v) < f(u)$ u-GRAY

Argue with color of v

u: GRAY

v - GRAY (u,v) - back edge

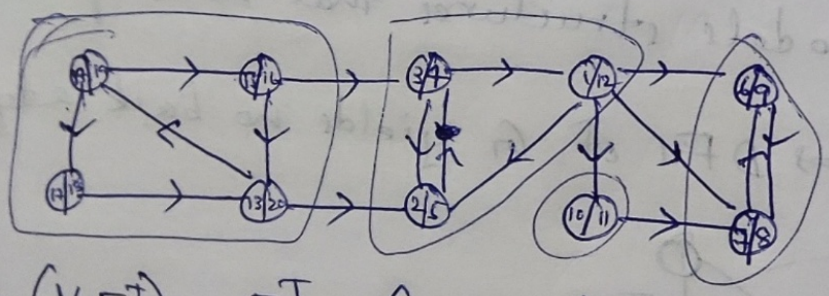
v - White from parenthesis thm, $f(v) < f(u)$

v - Black

20/3/23

Strongly connected components :- (SCC)

G is a maximal set of vertices $C \subseteq V$ $\ni$ $\forall u, v \in C$ $\exists$ $u \rightsquigarrow v$ & $v \rightsquigarrow u$



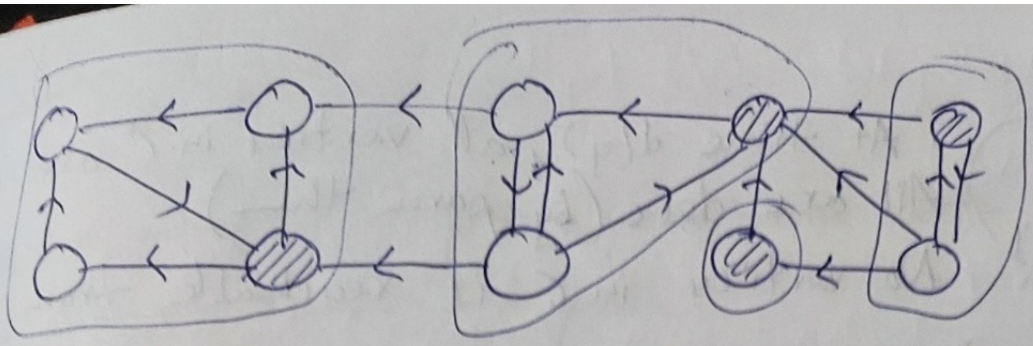
$G^T = (V, E^T)$ $E^T = \{(u,v) \mid (v,u) \in E\}$

G & G^T have same SCC.

KOSARAJU'S ALGO :- $\Theta(V+E)$ time

- DFS(G)
- Find G^T
- DFS(G^T) consider vertices in order of decreasing $f[u]$
- Output DFS tree.

04/02/2023 17:3



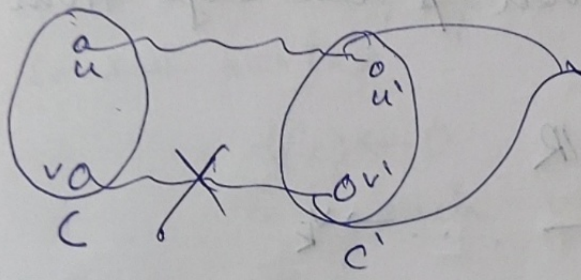
Component Graph :-

$G^{scc} = (V^{scc}, E^{scc})$

One vertex for each SCC

if ∃ an edge from the corresponding SCCs in G.

Lemma :- G^{scc} is a DAG.

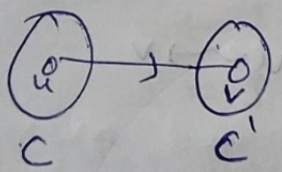


Only one of these path exists. If both of them exist, it becomes cyclic.

$d(U) = \min_{u \in U} \{d(u)\}$ Earliest discovery time

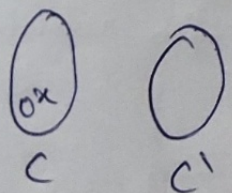
$f(U) = \max_{u \in U} \{f(u)\}$ Latest " "

Lemma :- C & C' distinct SCCs in $G = (V, E)$



~~$d(C) < d(C')$~~

Case 1 :- $d(C) < d(C')$ $f(C) > f(C')$

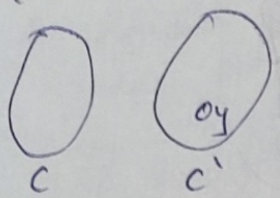


At time $d(x)$, all vertices in C & C' are white.

All vertices of C & C' are descendants (parenthesis thm)

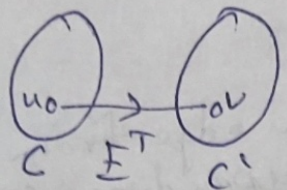
$$f(x) = f(C) > f(C')$$

$d(C) > d(C')$



At time $d(y)$, all vertices in C' are ...
All are desc (by parenthesis)
No vertices in C is reachable from ...

Cor:- C & C' distinct SCCs in $G_1 = (V, E)$



$f(C) < f(C')$

Cor:- C & C' are distinct SCCs in $G_1 = (V, E)$ &
Suppose $f(C) > f(C')$. Then $\nexists$ an edge from C' to ...

Path in Graph:-

$G_1 = (V, E)$ $\quad w : E \to \mathbb{R}$

$p = v_1 \to v_2 \to v_3 \to \dots \to v_k$

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

Shortest path:- $u \rightsquigarrow v$ is a path of min weight from u to v.

$$\delta(u, v) = \min \{ w(p) / p : u \rightsquigarrow v \}$$

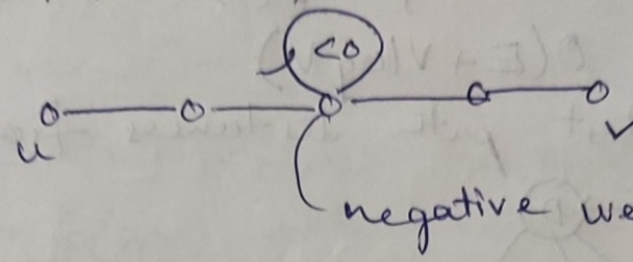
$$\delta(u, v) = +\infty \quad \nexists \; u \rightsquigarrow v$$

22/3/23

$\delta(u,v) = \min\{w(p) / p \text{ is a path from } u \text{ to } v\}$

Opt sub structure :- sub path

Thm :- $\forall\ u, v, x \in V,\ \delta(u,v) \leq \delta(u,x) + \delta(x,v)$



negative weights are dealt in Bellman Ford

Greedy :- S $\quad V \setminus S$: Q - priority queue

$s \in V$ source vertex

```
d(s) ← 0
for each v ∈ V \ S
do
    d(v) ← ∞
S ← ϕ
Q ← V
while Q ≠ ϕ                                   } V' times
do
    u ← Extract_min(Q)  ← log V
    S = S ∪ {u}
    for each v ∈ adj(u)                       } deg(v)
    do
        if d(v) > d(u) + w(u,v)               } relaxation
        then
            d(v) = d(u) + w(u,v)
                ↳ decrease key
                  log V
```

$O(V \log V + E \log V) = O(E \log V)$

~~Array~~ :- $O(V^2)$ — not told

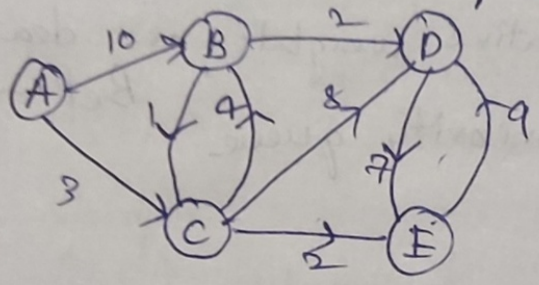
Fibonacci heap :-

Extract_min $O(\log V)$ amortized

decrease key $O(1)$ amortized

$O(E + V \log V)$

Ex :- Find the shortest path distances from A.



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

Lemma :- $d(s) \leftarrow 0$ & $d(v) \leftarrow \infty \quad \forall \; v \in V - \{s\}$

$d(v) \geq \delta(s, v) \quad \forall \; v \in V$

This invariant is maintained every any seq of relaxation steps.

Pf :- Suppose v be the first vertex for which $d(v) < \delta(s, v)$

u - vertex that caused $d(v)$ to change

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

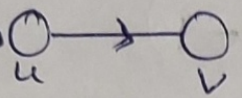
$d(v) < \delta(s,v)$

$\leq \delta(s,u) + \delta(u,v)$ (Δ ineq)

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

$\leq d(u) + w(u,v)$ (v is the first violation)

$\Rightarrow\Leftarrow$

Lemma:- 

Then if $d(u) = \delta(s,u)$ & (u,v) is reduced, then $d(v) = \delta(s,v)$ after relaxat^n.

Pf:- $\delta(s,v) = \delta(s,u) + w(u,v)$

$d(v) > \delta(s,v)$ before relaxat^n

$d(v) > d(u) + w(u,v)$ Test succeeds

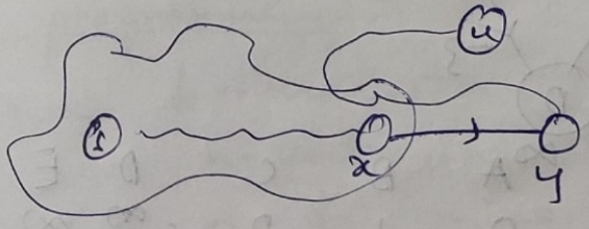
$d(v) > \delta(s,v) = \delta(s,u) + w(u,v)$

Algo sets $d(v) = d(u) + w(u,v) = \delta(s,v)$

Thm:- Dijkstra's terminates with $d(v) = \delta(s,v)$ $\forall\ v \in V$.

TP:- $d(v) = \delta(s,v)$ $\forall\ v \in V$ when v is added to S.

Suppose u is the 1st vertex added to S for which $d(u) > \delta(s,u)$



S: just before adding "u"

$d(x) = \delta(s,x)$

$\downarrow$ relax(x, y)

$d(y) = \delta(s,y)$

$\leq \delta(s,u) < d(u)$

$\Rightarrow\Leftarrow$ choice of "u".

29/3/23

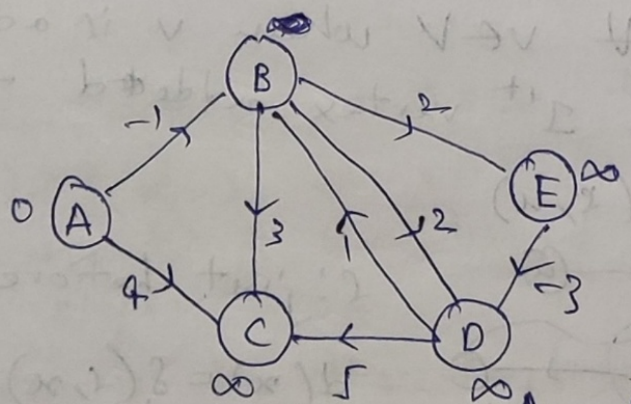
Bellmann Ford Algo:-

Find all SP length from source "s" to all (or) determines that a -ve weight cycle exists.

```
d[s] ← 0
d[v] ← ∞ for all v ∈ V−{s}
for i ← 1 to |V|−1
do
    for each edge (u,v) ∈ E
    do if d(v) > d(u) + w(u,v) then
          d(v) ← d(u) + w(u,v)
for each edge (u,v) ∈ E
do if d(v) > d(u) + w(u,v)
   then
        report_ve weighted cycle exists.
```

$d(v) = \delta(s,v)$ if no -ve weighted cycle.

Time :- $O(VE)$

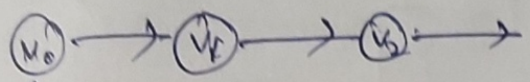


	A	B	C	D	E
End of 1st pass	0	−1	2	∞	∞
2nd pass	0	−1	2 (−1+3)	−2	1
3rd pass	0	−1	2	−2	1

Thm:- If G contains no -ve weight cycle, then $d(v) = \delta(s,v)$ $\forall$ reachable $v \in V$

Pf:- $v \in V$

Consider a SP from s to v



$$\delta(s,v_i) = \delta(s,v_{i-1}) + w(v_{i-1},v_i)$$
$$d(v_0) = 0 = \delta(s,v_0)$$
$$d(v_0) \geq \delta(s,v) \text{ - invariant}$$

After pass 1, $d(v_1) = \delta(s,v_1)$

2, $d(v_2) = \delta(s,v_2)$

⋮

k, $d(v_k) = \delta(s,v_k)$

∴ there are no -ve wt cycle, p is simple path

longest simple path $\leq |V|-1$ edges

Cor:- If a value $d(v)$ fails to converge after $|V|-1$ passes, $\exists$ a -ve wt cycle in G reachable from "s".

Linear programming :-

$A_{m \times n}$ b - m-vector c - n-vector

Find an n-vector x that maximizes $c^T x$ subject to $Ax \leq b$ or determine no such sol^n exists.

1) Simplex method - exp time
2) Interior pt method, poly time

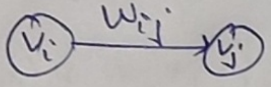
Feasibility problem:- No optimizat^n ~~problem~~ criteria
Find x s.t $Ax \leq b$.

Solving a sys of difference constraints
L-P where each row of A contains only a one & one -1 & the rest 0's.

Eg:-
$$\left.\begin{aligned} x_1 - x_2 &\leq 3 \\ x_2 - x_3 &\leq -2 \\ x_1 - x_3 &\leq 2 \end{aligned}\right\} \quad x_j - x_i \leq w_{ij}$$

$x_1 = 3, x_2 = 0, x_3 = 2$

Constraint graph:-

$x_j - x_i \leq w_{ij} \Rightarrow$ 

$A_{|E| \times |V|}$

Thm:- If the constraint graph contains a -ve wted cycle, then the system is unsatisfiable.

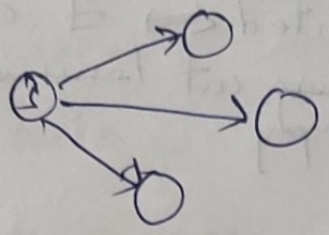
Pf:- $v_1 \to v_2 \to v_3 \to \cdots \to v_k \to v_1$

$x_2 - x_1 \leq w_{12}$
$x_3 - x_2 \leq w_{23}$
$\vdots$
$x_k - x_{k-1} \leq w_{k-1}$
$x_1 - x_k \leq w_{k,1}$

$0 \leq$ weight of cycle
$\Rightarrow \Leftarrow$
So no values for the x_i can satisfy the constraints

Thm:- Suppose no -ve wt cycle exists in the constraint graph. Then, the constraints are satisfiable.

Pf:- Add a new vertex s to V

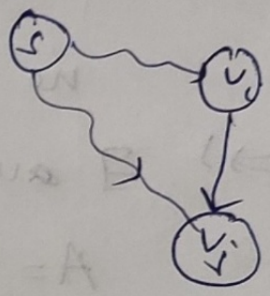


no -ve wt cycle
$\Rightarrow$ S.P exists

Claim:- $x_i = \delta(s, v_i)$ solves the constraints.

$$x_j - x_i \leq w_{ij}$$

$$\delta(s, v_i) + w_{ij} \geq \delta(s, v_j)$$



MST (Min wted Spanning Tree)

i/p:- undirected graph $G = (V, E)$ wts (can be -ve) for edges

o/p:- min wted spanning tree

$$(V(G), E(T)) \quad E(T) \leq E(G)$$

27/3/23

MST:- indirected $G = (V, E)$, c_e – cost of edge
(connected, distinct)

Prim & Kruskal

- $X = \{s\}$ (arbitrarily)
- $T = \phi$
- while $X \neq V$

  $e = (u, v)$ be the cheapest edge with $u \in X$ & $v \notin X$

  Add e to T

  Add v to X

  $O(EV)$ implement^n

T spans X:- Invariance

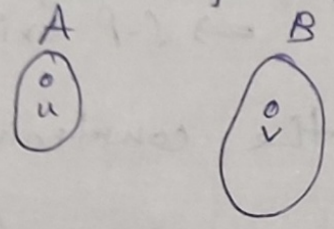
Claim 1:- T^* – spanning tree

cut:- is a partition of V into 2-non empty sets

(A, B) $2^n - 1$

Lemma:- A graph is not connected $\Leftrightarrow$ $\exists$ a cut (A, B) with "no" crossing edges (~~Data~~ empty cut lemma)

Pf:- ($\Leftarrow$)



no $u \rightsquigarrow v$ path $\Rightarrow$ G is not connected

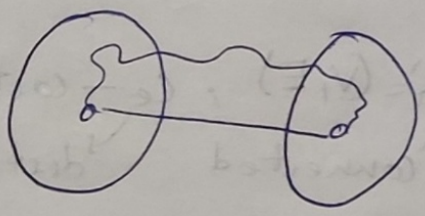
($\Rightarrow$) $\exists$ u, v $\ni$ there is no u–v path

$A = \{v / v$ is reachable from $u\}$

B – remaining vertices

(A, B) is a cut with no crossing edges.

Fact 1:- Suppose $C \subseteq E$ has an edge crossing the cut (A, B), then so some other edges of C. (Cross edge lemma)



Fact 2:- If "e" is the only edge crossing some cut (A, B) then it is not in ~~any~~ any cycle.

1) Algo maintains invariant that T spans X. (by induction)

2) Cannot get stuck with $X \neq V$

$(X, V \setminus X)$ no crossing edges

$\Rightarrow\Leftarrow$ empty cut lemma

3) No cycles get created in T.

Lonely edge lemma:-

Claim :- Prim outputs a MST

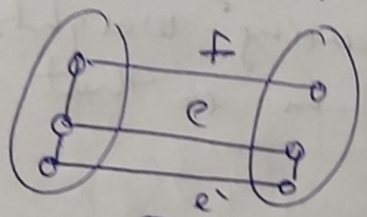
Cut property :- Consider an edge $e \in G$. Suppose there is a cut (A, B) such that "e" is the cheapest edge of G that crosses it.

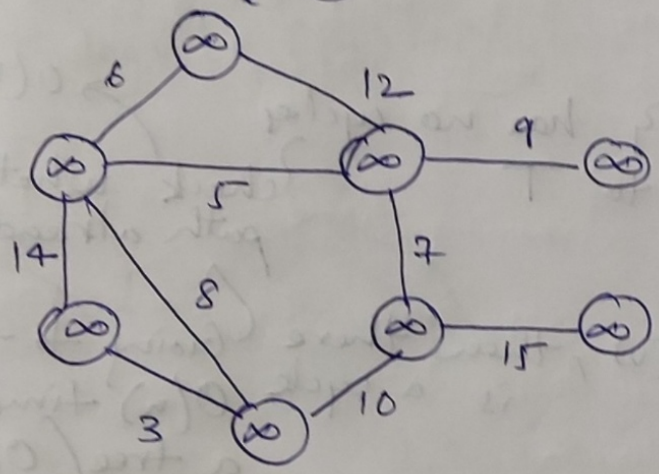
Then "e" belongs to "the" MST of G

⇓

Prim outputs the MST

Correctness :-






```
Q ← V
key[v] ← ∞   ∀ v ∈ V
key[s] ← 0   (arbitrary s ∈ V)
while Q ≠ ϕ
do
   u ← EXT_MIN(Q)
   for each v ∈ Adj(u)
      if v ∈ Q  &  w(u,v) < key(v)
      then
           key(v) ← w(u,v)
           π(v) ← u
```

$E \log V + V \log E$

30/8/23

T^* -

$X \neq V$? By empty cut lemma

No cycles :- Lonely edge lemma

~~Cut Property pf:- $e \in E(G)$~~

Kruskal's MST algo → $O(m \log n)$

- Sort edges in order of increasing weights
  {remaining edges $1, 2, \ldots, m$ so that $c_1 < c_2 \ldots <$

$T = \phi$

$O(m)$ { For $i = 1$ to m
  if $T \cup \{i\}$ has no cycles → $O(1)$ {check whether $\exists$ another path already
  - Add i to T

- Return T

If $Find(u) = Find(v)$, then there is a cycle (Going to take only $O(n)$ time as it is a tree ($O(V+E)$, n, $n-1$)

~~if Find(u)~~

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

↳ Overall time for leader pointer updates

$O(m)$ - randomized algo [Karger-Klein-Tarzan] (JACM 1995)

$O(m \alpha(n))$ - deterministic [Chazelle, JACM 2000]

↳ Inverse Ackermann f^n - grows slower than $\log^*$

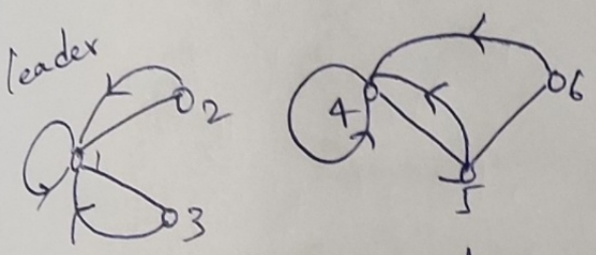
$\log^* n$ = # times you can apply $\log$ to "n" until result drops to 1.

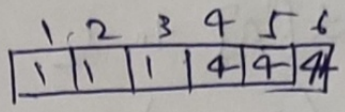
↳ inverse of tower ~~tower~~ f^n.

31/3/23

UNION - FIND :- Partitⁿ of a Set

Find(x) — O(1), Union(x,y) — O(log n)



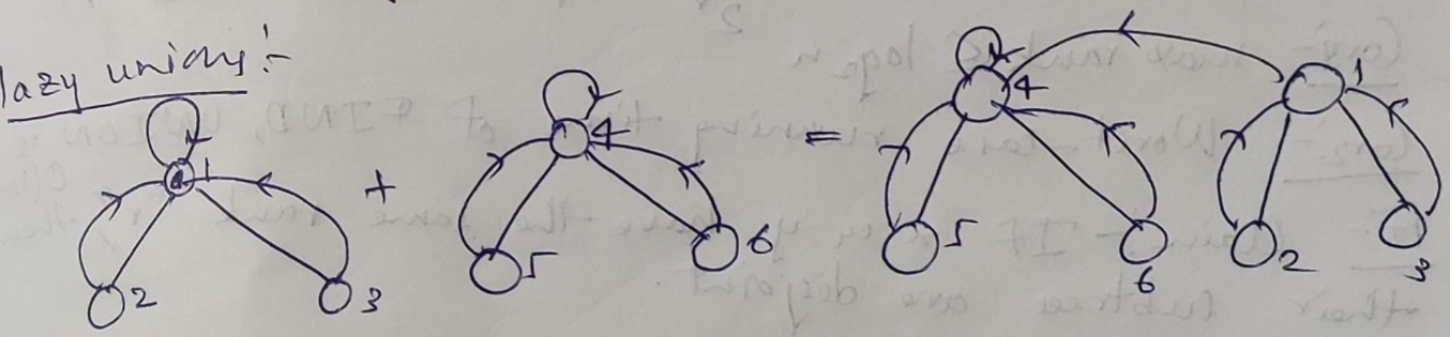


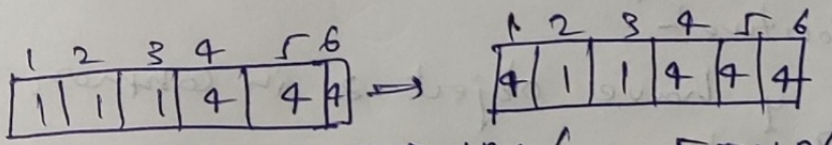
Union :- make every element in the smaller list to point the leader of the bigger list.

$O(n \log n)$ - for all unions

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

lazy union :-





Union reduces to 2 FINDS ($r_1 = FIND(x)$, $r_2 = FIND(y)$)

& O(1) time (link r_1 & r_2)

FIND - O(h), Union - O(h)

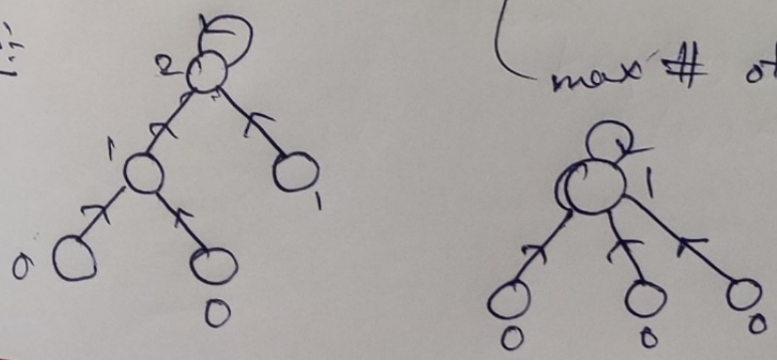
x is a root $\Leftrightarrow$ parent(x) = x

Union by Rank :-

For each $x \in X$, rank[x] = ~~height of x~~ 1 + max rank of x's children

(max # of hops from some leaf)

Ex :-



Union(x, y)

s_1 = FIND(x), s_2 = FIND(y)

if rank[s_1] > rank[s_2] then set parent[s_2] to s_1

else

parent[s_1] to s_2

ranks are same (say r) → r+1

ranks are diff → no change

non-root nodes → no change

Rank lemma :- Consider an arbitrary seq of UNION (+FIND) operations. For every $r \in \{0, 1, 2, \ldots\}$ there are atmost $\frac{n}{2^r}$ objects with rank "r".

Cor 1 :- max rank $\leq \log_2 n$

Cor 2 :- Worst-case running time of FIND, UNION is $O(\log n)$

pf :- Claim :- If x & y have the same rank "r", then their subtrees are disjoint.

Proving contrapositive :-

x & y have object z in common

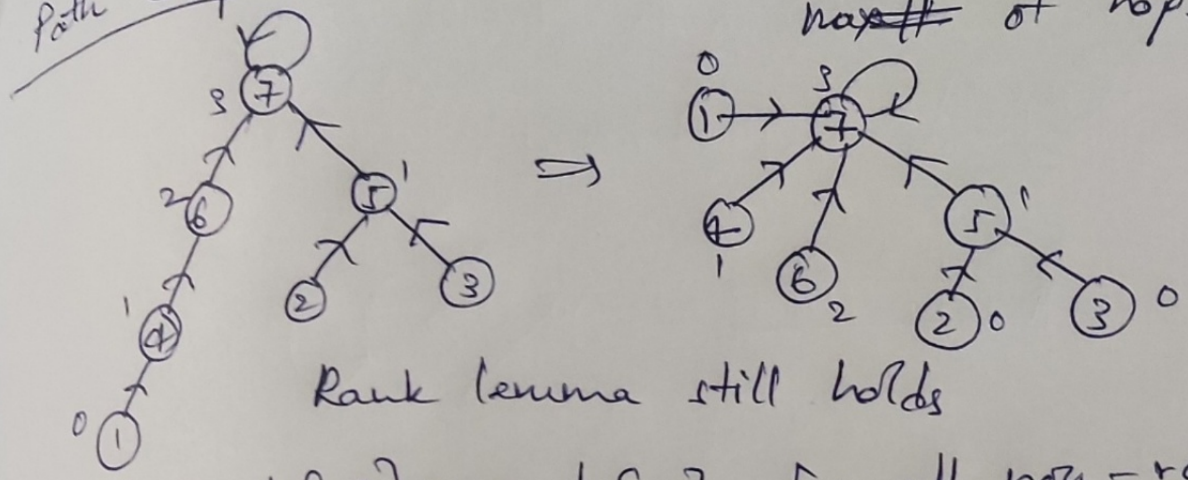
$\Rightarrow \exists$ path $z \rightarrow x$, $z \rightarrow y$

One of x & y is an ancestor of the other.

The ancestor has strictly larger rank.

Claim 2 :- The subtree of a rank-r object has size $\geq$

Path Compression :- (Rank becomes the upper bound on ~~max#~~ of hops)



Rank lemma still holds

rank[parent(x)] > rank[x] for all non-roots.

Thm :- With union by rank and path compression in UNION + FIND operations takes $O(m \log^* n)$ time

$\log^*(2^{65536}) = 5$

$2^{2^{2^{2^{2}}}} = 2^{2^{2^{4}}} = 2^{2^{16}}$

Hop craft
- Ullman