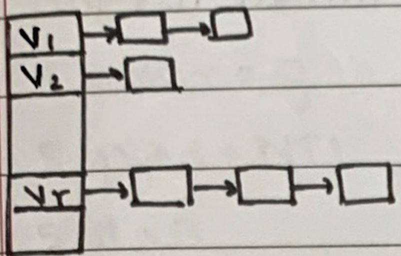
★ Graph

→ Adjacency matrix : $O(V^2)$ ★ Used for dense graphs

→ Adjacency list : $O(|V|+|E|)$ ★ Used for sparse graphs



Linked lists

→ insert new vertex at the start of the linked list

→ Incidence matrix : $O(|V||E|)$ ★ Used by mathematicians

e_1 e_2 . . . . e_m

~~v_1~~

~~v_2~~ +1

⋮ −1

v_n

★ edge between

v_i & v_j : $v_i \rightarrow v_j$

+1 −1

→ 3 rows are linearly dependent

then ∃ a cycle between v_i, v_j, v_k

→ Rank of matrix

Those edges ∃ Spanning Tree

Date
Page

★ ADT of graph

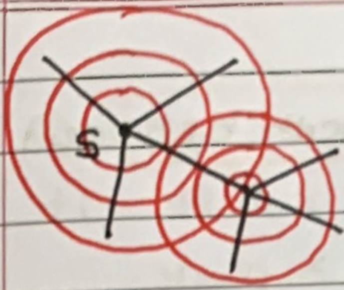
1. Create ( )
2. insertedge(vi,vj) : O(V) (# placed first in linked list)
3. deleteedge (vi,vj) : O(VE)
4. deletevertex : O(V)
5. insert vertex : O(V)
6. adjacent (v)
7. BFT(S)
8. DFT(S)

★ BFT(s):

∀u:

1. u.color
2. u.pred / u.π : Predecessor
3. u.d : Discoverytime

★ color	:	criteria
1. w	:	unexplored
2. g	:	traversed but unfinished
3. b	:	finished

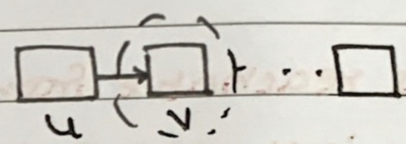


★ Wavefront model

→ Algo:

s.color = g
s.pred = NIL
s.d = 0
enqueue s to Q
While !Q.empty()
u ← dequeue Q
v.color = g
v.pred = u
v.d = u.d + 1
enqueue v into Q
(If v.color = black skip above steps)
u.color = b

Q: Head . . . . Tail



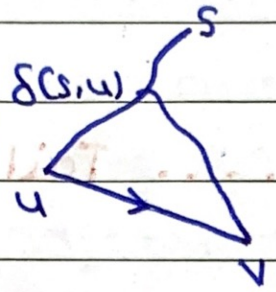
★ Space complexity
Input: O(V+E)
Workspace: O(V) # queue has atmax V elements
Output: O(V)

★ Note: Queue ⇔ node.color = grey

★ $\delta(v', v'')$: Length of shortest path between v' and v''

① 

$\delta(s,v) \le \delta(s,u) + 1$



$\delta(s,v)$ is shortest

$\therefore \delta(s,v) \le \delta(s,u) + 1$

★ u is reachable from s

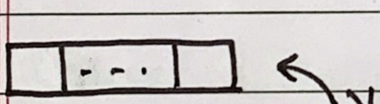
★ If u is unreachable from s $\delta(s,u) = \infty$

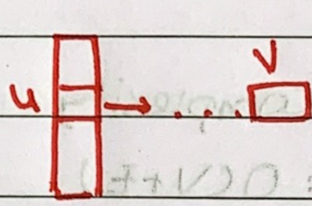
② $\forall_v \; v.d \ge \delta(s,v)$

Induction on no. of enqueue operations

Just before enqueue $s.d = 0 \ge 0$ (Basis)

Induction Step





(u : already in the queue)

$v.d = u.d + 1$

$\ge \delta(s,u) + 1$

$\ge \delta(s,v)$ (# ①)

$\therefore v.d \ge \delta(s,v)$

③ Q: $\boxed{v_1 \; \ldots \; v_i \; v_{i+1} \; \ldots \; v_r}$ Queue

$\forall i \quad v_i.d \leq v_{i+1}.d$

$v_r.d \leq v_1.d + 1$

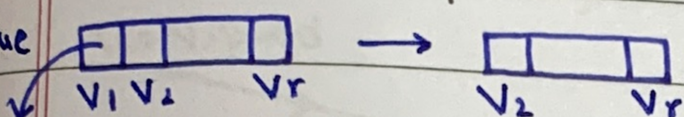
→ Induction on no. of queue operations

→ Basis $\boxed{S}$

$v_r = v_1 \Rightarrow v_r.d \leq v_1.d + 1$

→ Induction Step

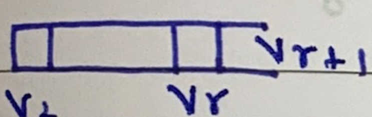
Dequeue



$v_1.d \leq v_2.d$ (# out of queue)

$v_r.d \leq v_1.d + 1 \leq v_2.d + 1$

Enqueue



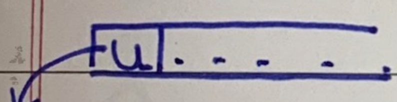
$u.d + 1 = v_{r+1}.d$ (# Flashback)

$v_1.d + 1 \geq v_{r+1}.d$

Flashback:

$v_r.d \leq u.d + 1$ (# Hypo)

$= v_{r+1}.d$



→ When queue is empty prop are trivially true.

(5) At the end $\forall_v \; v.d = \delta(s,v)$

→ If v is not reachable from s

$v.d = \infty \;,\; \delta(s,v) = \infty$

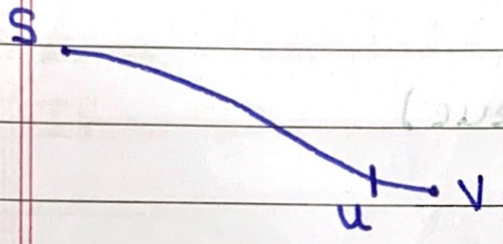
→ If v is reachable from s

Let $v.d > \delta(s,v)$

v is reachable from s

among all such vertices consider

$v \ni \delta(s,v)$ is min



$v.d > \delta(s,v)$

upto u shortest distance

minimality of v

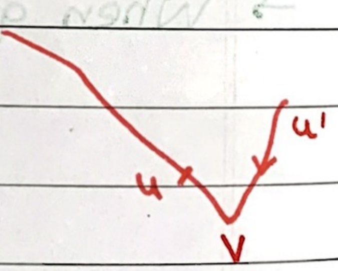
$\delta(s,u) = s.d$

$v.d > \delta(s,v) = \delta(s,u) + 1 = u.d + 1$

$\boxed{v.d > u.d + 1}$

★ v. color

w	$v.d = u.d + 1$
b	$v.d \leq u.d$
g	$v.d \leq u'.d + 1 \leq u.d + 1$



★ Before exploring u we exploring u'

$\therefore u'.d \leq u.d$

→ Hence no such v exists.

★ Colloraray:

Any v is reacable from s then

$v.d < \infty$

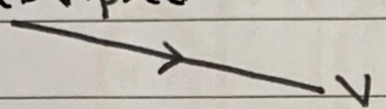
Proof:

Let $v.d > \infty \Rightarrow \delta(s,v) > \infty$

But $\delta(s,v) < \infty$ (#)

★ u is v.pred only when we find v in white color

u=v.pred



$v.d = u.d + 1$

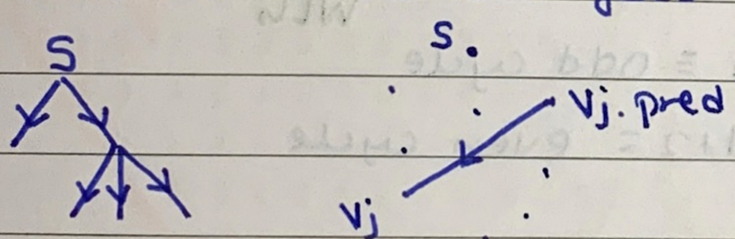
$v.d = \delta(s,v)$

$u.d = \delta(s,u)$

$\delta(s,v) = \delta(s,u) + 1$

★ Outarborescence: (Predecessor graph's special case)

rooted at s all arc goes away from s



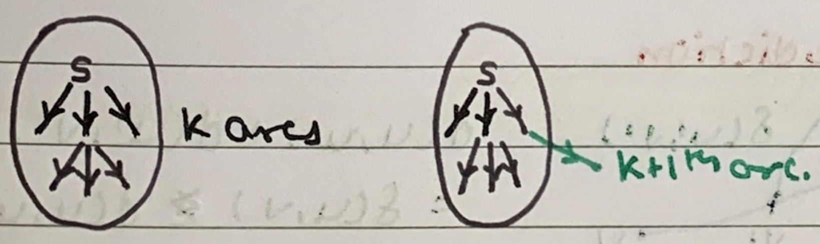
★ Breadth First tree

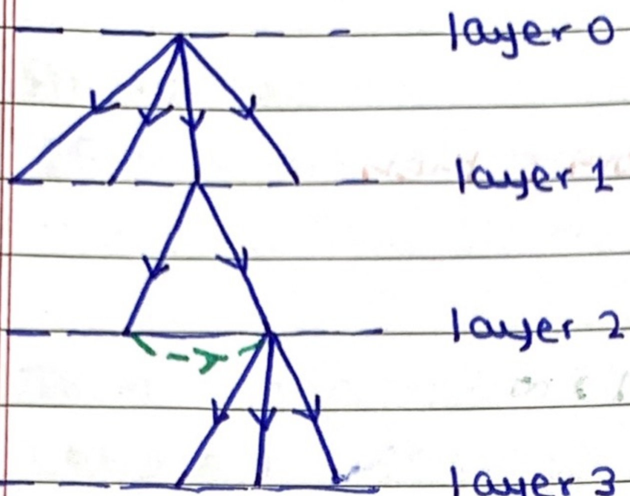
★ Shortest path tree

rooted at s

Basis: 0 arcs

Induction Step: k arcs

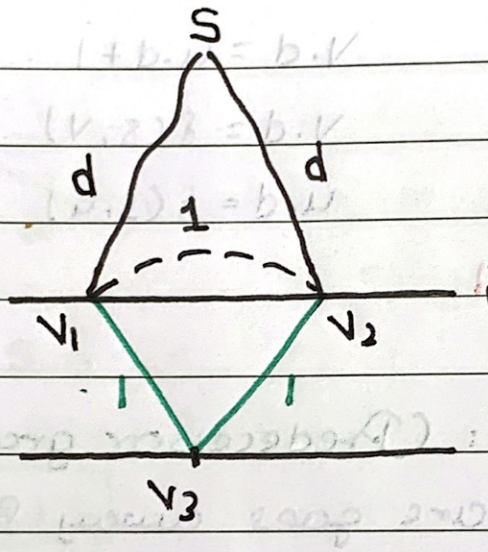




★ Every NON arborescence arc between
① layer i & layer i+1
② layer i & layer i

★ G may contain non-arborescence arcs.

★ layer i & i+1 (# discovery)



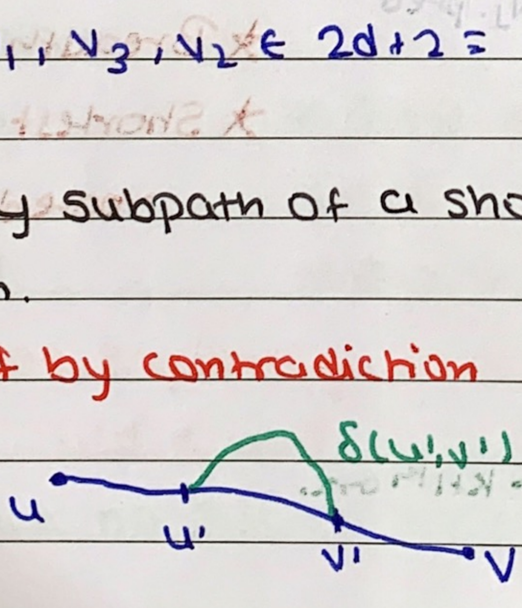
j = i+1 or i-1
↑
WLG

★ $v_1, v_2 \in 2d+1 \equiv$ odd cycle

★ $v_1, v_3, v_2 \in 2d+2 =$ even cycle

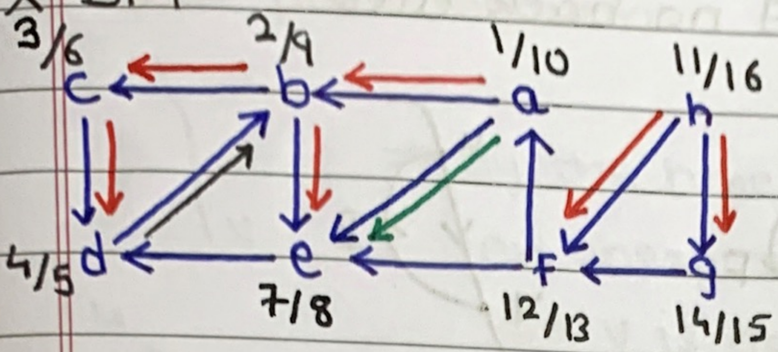
★ every subpath of a shortest path is a shortest path.

→ Proof by contradiction



$$d(u,u') + d(u',v') + d(v',v) = \delta(u,v) \geq d(u,u') + \delta(u',v') + \delta(v',v)$$

* DFT



→ : Tree arcs (T)

→ : Forward arcs (F)

→ : Back arcs (B)

remaining arcs : Cross arcs (C)

* ∀u

① u.color

② u.d : discovery time

③ u.f : finish time

④ u.pred

* n nodes max finish time is 2n

○→→→→→

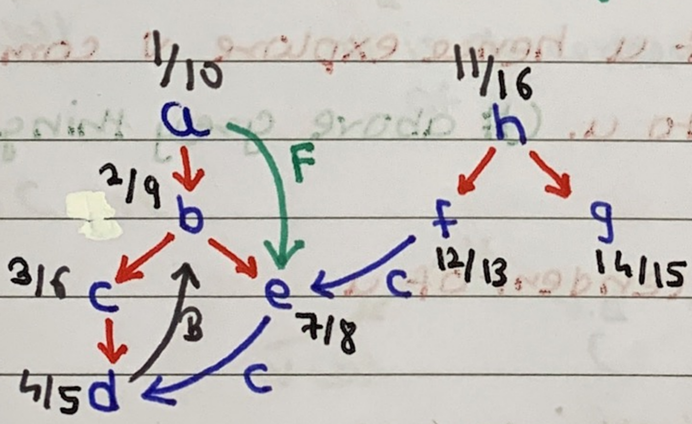
Θ(V+E) : Space Complexity

* Stack at max n elements

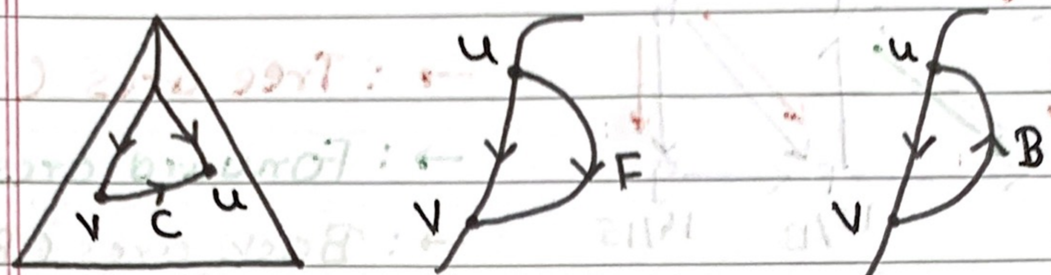
* System stack is used here

* We get a collection of outarborescence forest. (Branching)

→ All the arcs that belong to it (tree arcs:



★ G is acyclic iff ∃ no back arcs



No cycle — No cycle — ∃ cycle

★ A vertex v is a descendent of u iff v is discovered during the time in which u is grey.
(u.d < v.d)

★ Parenthesis Theorem:

u.d v.d v.f u.f w.d w.f
(( )) ( )

→ Put in sorted order:

u.d v.d v.f u.f
(( ))

→ v is descendent of u hence explore v completely before going back to u. (# above grey thing)

u.d u.f v.d v.f
( ) ( )

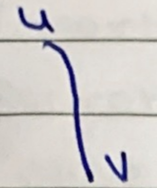
→ Then v is not descendent of u

★ Nesting of descendent intervals:

(( )) ⇒ v is descendent of u
u.d v.d v.f u.f

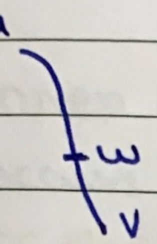
(# Parenthesis Thm)

★ White path theorem:



⇒ by the time u is discovered there exists an unexplored path from u to v

⇔ v is descendent of u



$w.f < u.f$ (w predecessor of v)

$u.d$ $w.d$ $w.f$ $u.f$

(( ))

$w.f < u.f$

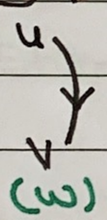
$\underline{u.d < v.d < w.f}$

v explored while w is grey

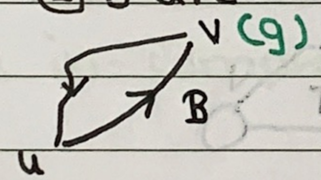
⇒ $u.d < v.d < w.f < u.f$

★ Categories:

① T arc

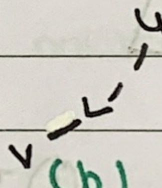


② B arc



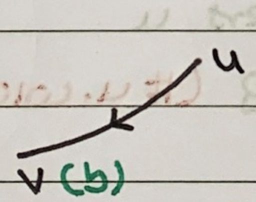
u is descendent of v

③ F arc



$v.d > u.d$

④ C arc



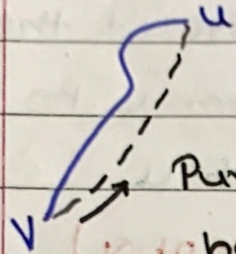
$v.d$ $v.f$ $u.d$ $u.f$

( ) ( )

$v.d < u.d$

★ Undirected graph

(T ✓ B ✓ F ✗ C ✗)

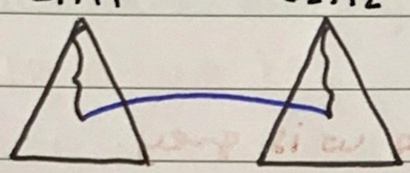


Pursue this edge in → direction

hence B ✓ & F ✗

★ Predecessor hence ∃ tree arcs

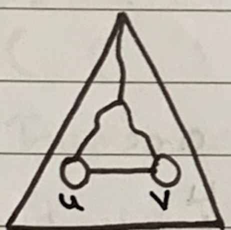
$d_1 | f_1$ $d_2 | f_2$



No cross arcs

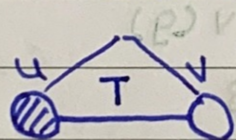
connected

$d_1 < d_2$



Let u be discovered first

∴ color of u is grey ⇒ $u.d < v.d$



① If u discovers v

→ Then uv ∈ T (# v.color = w)

② If v discovers u

→ Then vu ∈ B (# u.color = g)