CS & IT ENGINEERING

Algorithm

Graph Algorithms

Lecture No.- 01

By- Dr. Khaleel Khan sir

Recap of Previous Lecture

Topic Dynamic Programming

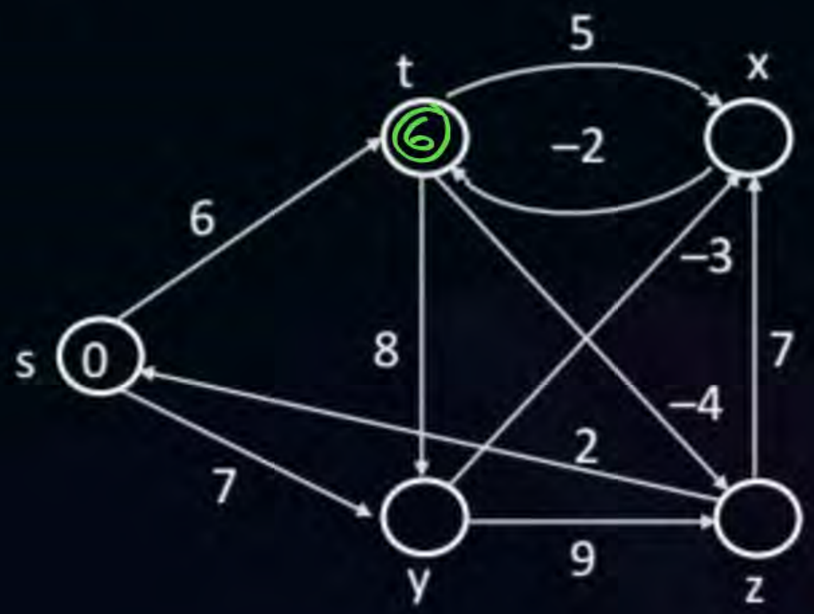
Topic

Topics to be Covered

Topic Depth First Search

Topic

Bellman Ford Algo



$s \to y \to x \to t$

$7 - 3 - 2 = 2$

Vertex Selected	d-values				
	s	t	x	y	z
{s}	−	6	4	7	2
		x			x
	−				

$s \to y \to x \to t - z$

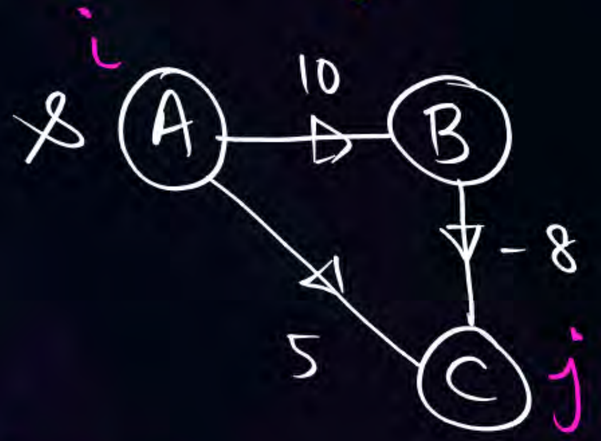
-2

Dijkstra's Algo.
- Failed to find Shortest Path for few vertices from Source (v_0

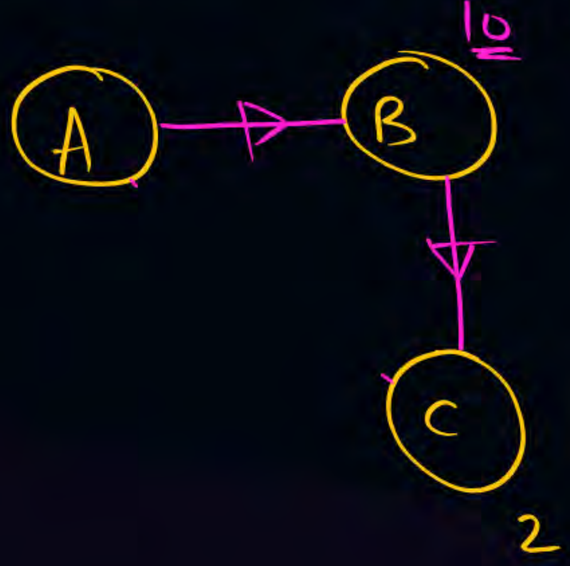
→ In Greedy Dijkstra's Algo ; The relaxation Process is Carried out w.r.to vertices ; The d-Value is once Selected Cannot be relaxed Further ;

→ In Bellman Ford, the relaxation is Carried out w.r.to edges in multiple Iterations ;

↳ (n-1) Iterations



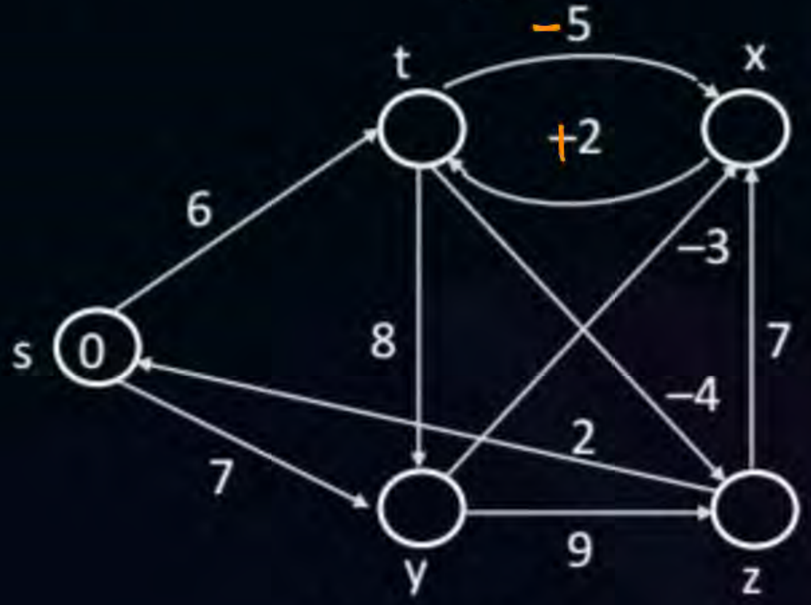
A-B-C



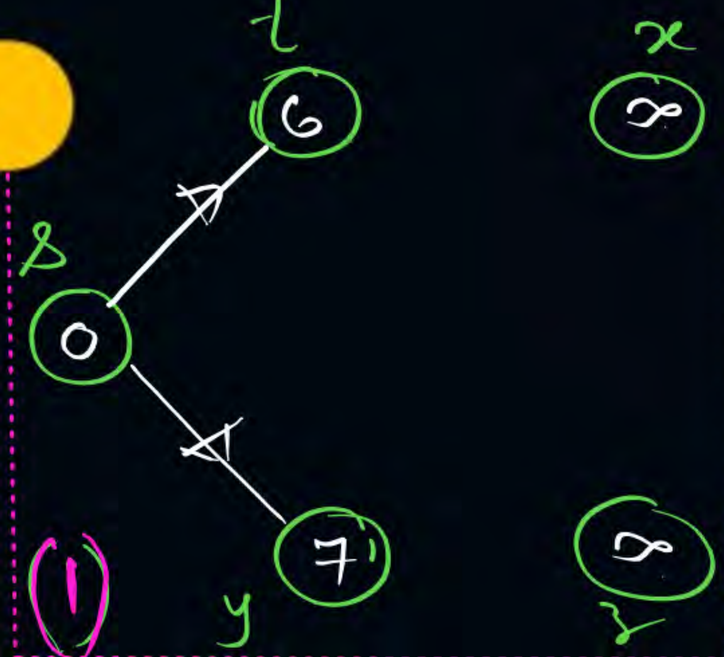
→ An unrestricted path b/w pair of vertices 'i' & 'j', in a graph, having n-vertices, cannot have more than (n-1) edges, without a loop/cycle;

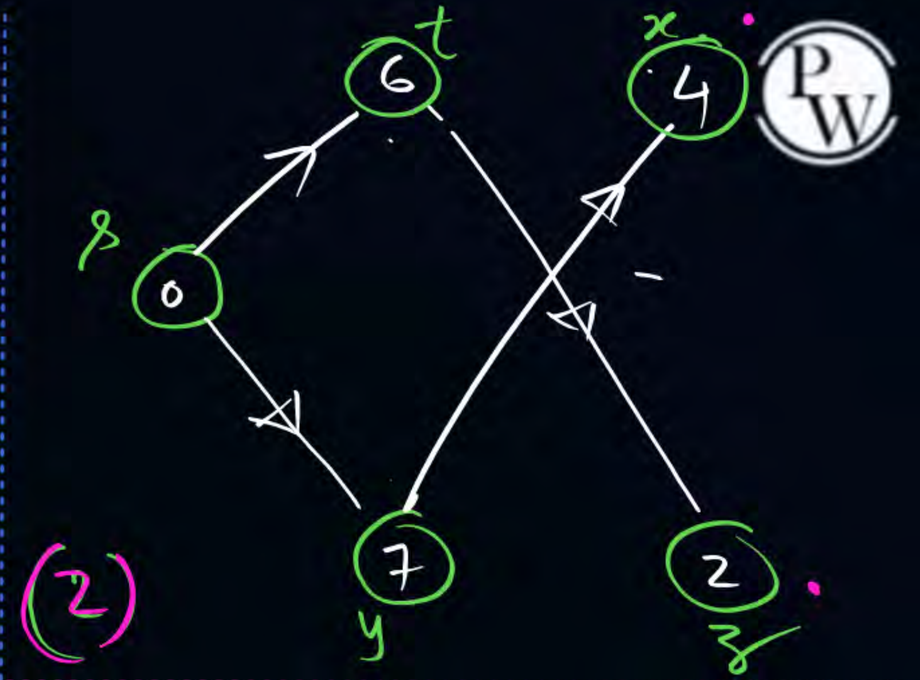
Bellman Ford Algo

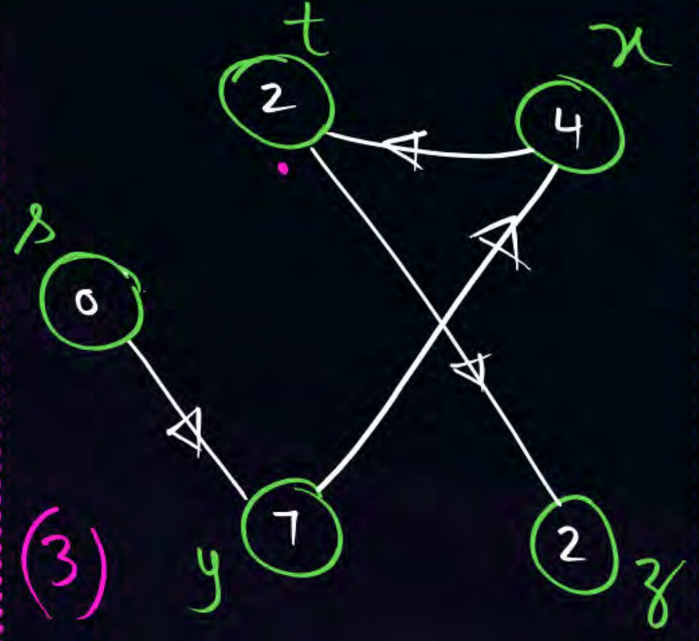
s: Src

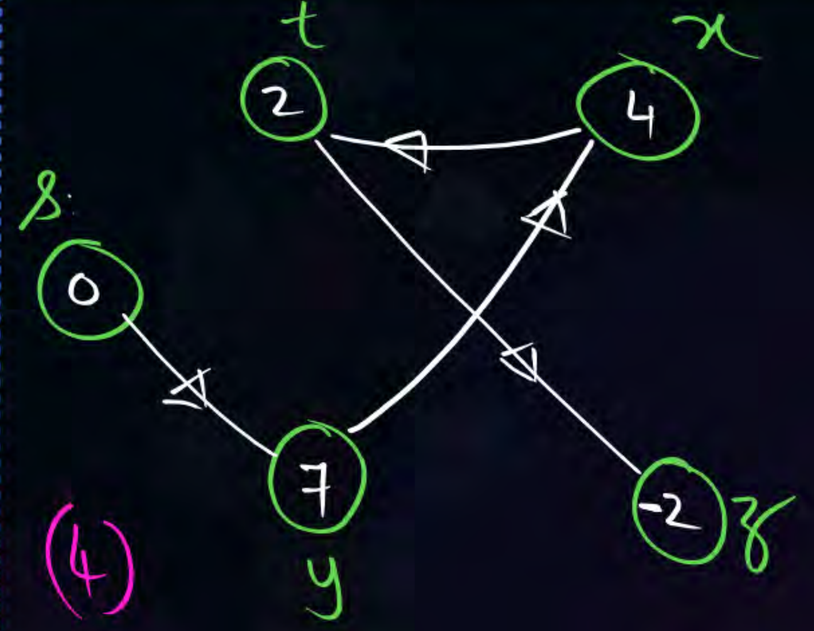


$$\begin{pmatrix} t & \xrightarrow{5} & x \\ :2 & & 4 \end{pmatrix}$$









Topic : Algorithms

$G=(V,E)$; $|V|=n$, $|E|=e$

Algorithm Bellman-Ford (G, w, s)

1. Initialize-Single-Source(G,s) ⟶ n
2. for i ← 1 to $|V[G]| - 1$ — n
3. do for each edge (u, v) ∈ E[G] — e } $n \cdot e$
4. do Relax(u, v, w)
5. for each edge (u, v) ∈ E[G]
6. do if d[u] > d[v] + w(u, v)
7. then return FALSE — e
8. return TRUE

Test to detect -ve wt. cycle

FALSE ⟶ -ve wt. cycle

TRUE ⟶ (No -ve wt. cycle)

Time : $n + n \cdot e + e \Rightarrow O(n \cdot e)$

PYQ

The Time Complexity of BF Algo. for a Complete Graph having 'n' vertices is $O(n^3)$ ✓

Complete Graph ⇒ $e = O(n^2)$

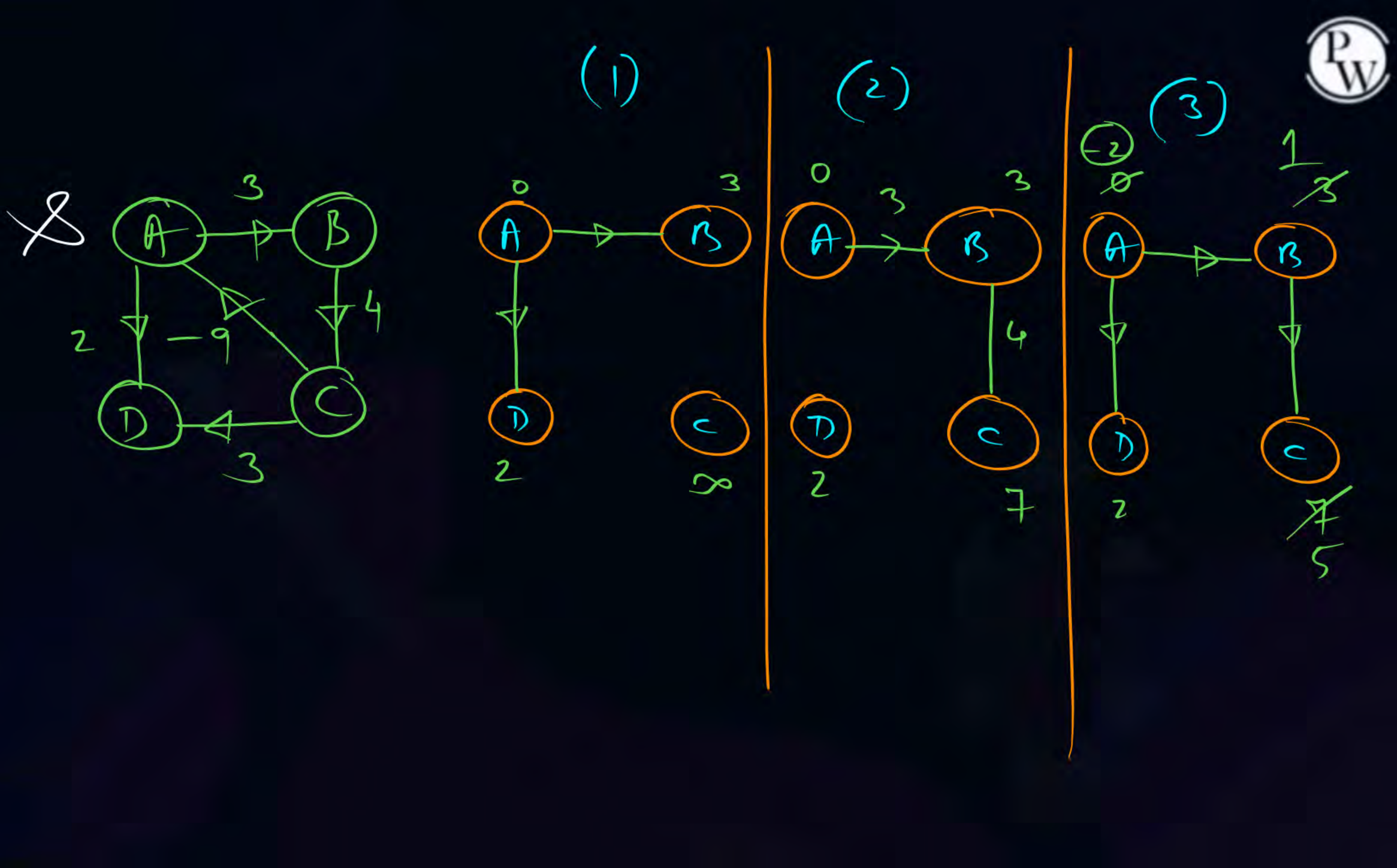
Topic : Algorithms

```
Initialize-Single-Source (G, s)
1     for each vertex v ∈ V[G]
2           do d[u] ← ∞
3              π[v] ← NIL
4     d[s] ← 0
Relax (u, , w)
1     if d[v] > d[u] + w(u, v)
2           then d[v] ← d[m] + w(w, u)
3                π[u] ← u
```

Derive a DP based recurrence using P of Opt. Repr. Soln to the Problem

→ If the graph has no -ve wt cycle, reachable from Source, then Shortest Paths are always determinable after completing (n-1) It's of Relaxation.

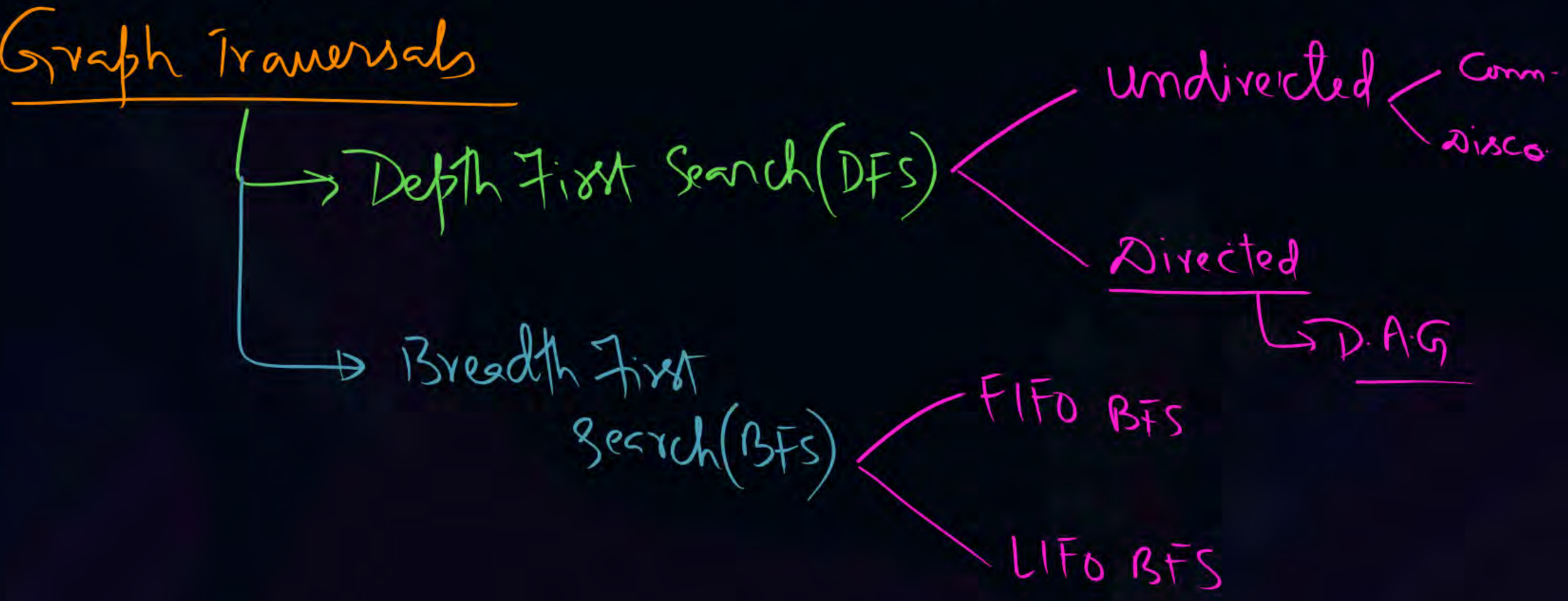
(& additional Relaxations will not change the d-values)


A
B
C
D
3
2
−9
4
3
(1)
(2)
(3)
0
∞
7
5

Graph Techniques:

Traversal:

Traversal: visiting all the Nodes of the Tree/Graph in a specified order and processing the info only once;



1. DFS in Undirected Graphs :

a) Connected Graphs :

Terminology :

a) Status of a Node :

- **E-Node** : Exploring Node (Node which is currently being explored)
- **Live Node** : Node which is not fully explored (Live Nodes are Stored in Some D.S)
- **Dead Node** : Node which has been fully explored

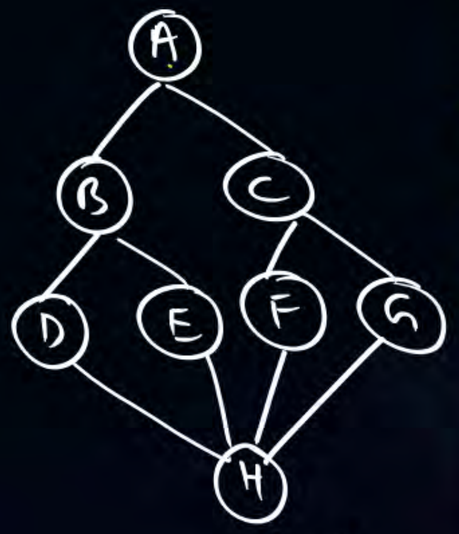
Timing - Values Associated with Nodes, during Traversal

(i) Discovery - Time : $d(x)$: The Time @ which the Node is visited for the First Time,

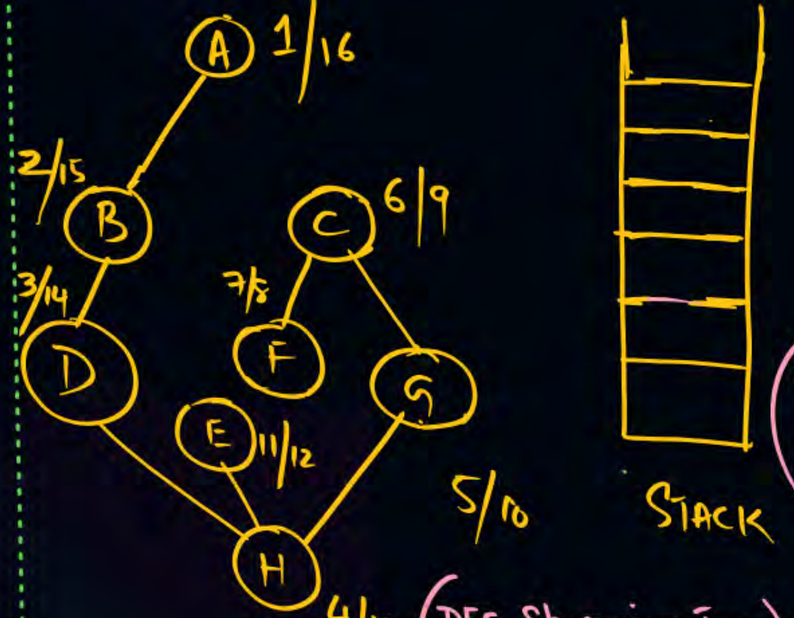
(ii) Finishing Time : $f(x)$: The Time @ which the Node becomes Dead Node,

(x) d/f (d/f Times are +ve Integers)

1) DFS in Undirected Connected Graph : A; B; D; H; G; C; F; E; ✓



$G = (V, E)$



(DFS Spanning Tree)

STACK

STACK Empty

Valid / Invalid (DFS)

a) A, B, E, H, D, F, C, G ✓

b) A, C, F, E, H, G, B, D ✗

c) H, D, B, E, A, C, F, G ✓

d) H, F, C, B, A, D, F, G ✗

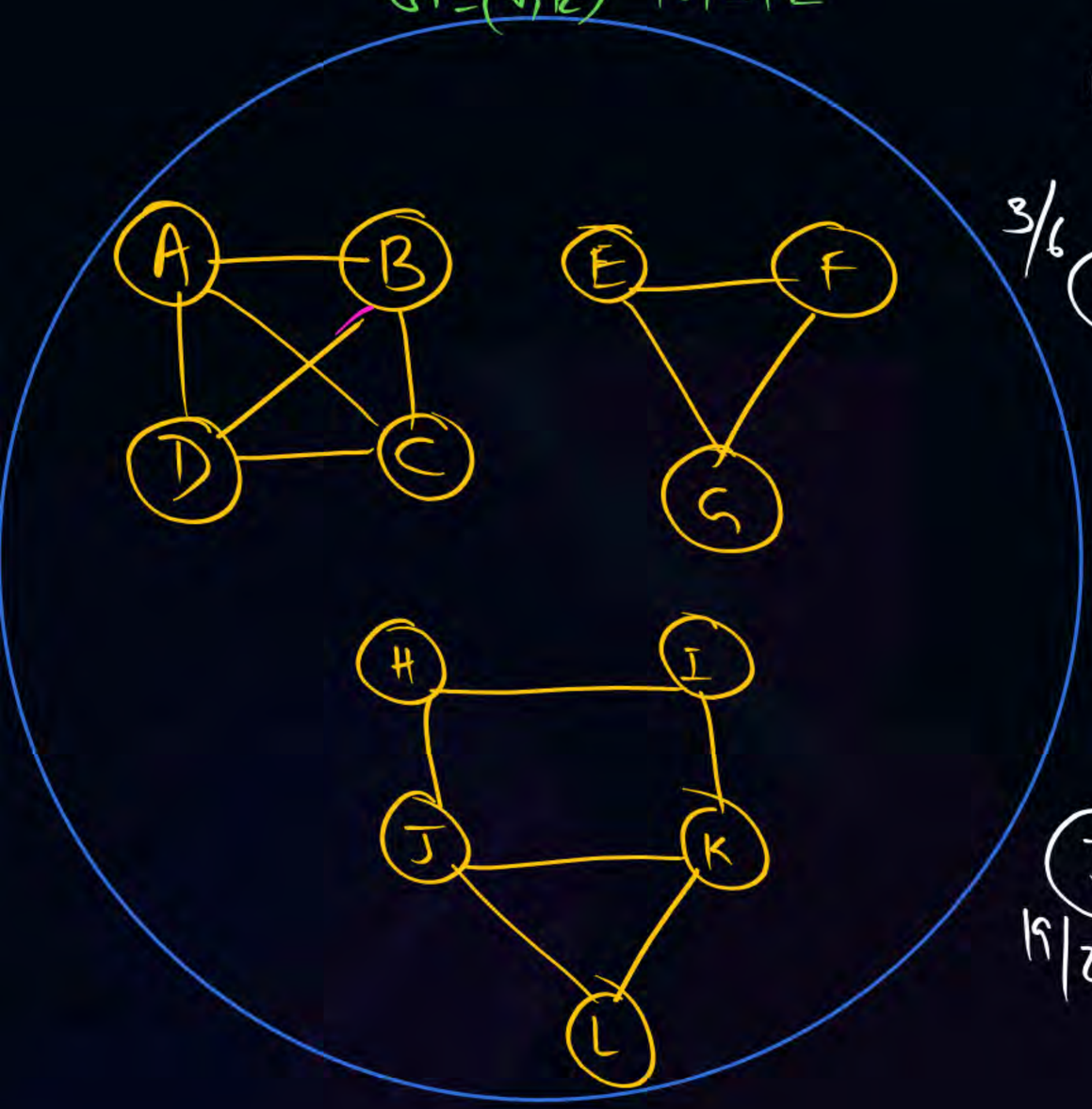
e) G, C, F, H, E, B, D, A ✓

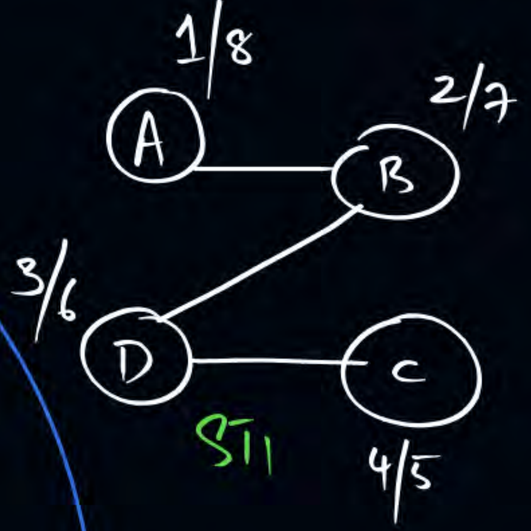
2) DFS in undirected Disconnected/Disjoint Graphs : [D.F.T]

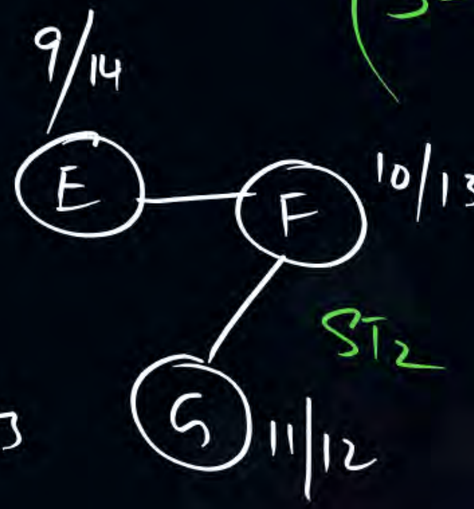
(Depth First Traversal)

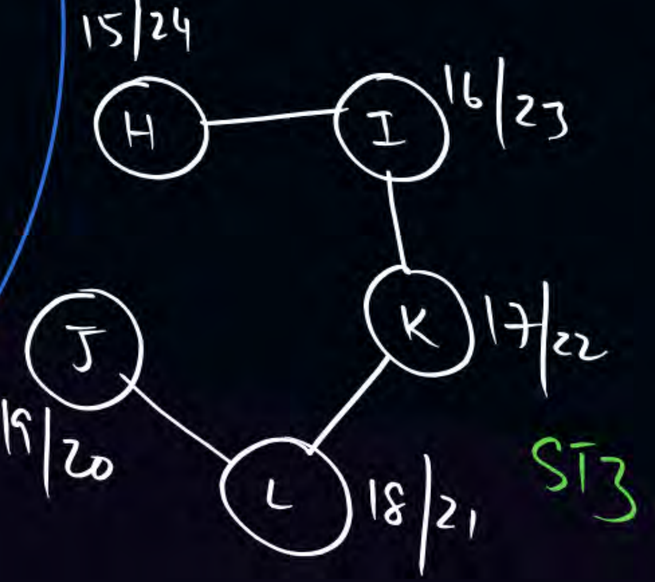
(3-Connected Components)

(Maximal Connected Subgraph)









D.F.S Spanning forest

Consider a undirected graph with 4 vertices <P, Q, R, S>
DFS is carried on it, generating the following d/f Times,

	P	Q	R	S	
1)	⟨3, 25⟩	⟨5, 18⟩	⟨8, 15⟩	⟨10, 12⟩	: Connected
2)	⟨12, 25⟩	⟨5, 10⟩	⟨6, 8⟩	⟨15, 20⟩	: Disconn
3)	⟨8, 10⟩	⟨18, 22⟩	⟨3, 15⟩	⟨6, 12⟩	: Disc. ⟨RSP⟩ ⟨Q⟩
4)	⟨18, 22⟩	⟨12, 15⟩	⟨8, 10⟩	⟨25, 30⟩	: Disconn. 4-Comp.
	(P)	(Q)	(R)	(S)	

Topic : Graph Techniques

DFS in Directed Graphs:

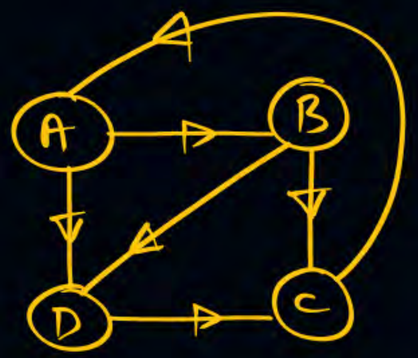
→ DFS when carried out on a Directed, Leads to the following Types of Edges;

1) Tree Edge: are part of DFS Sp. Tree/forest

2) Forward Edge: Leads from a Node to its non child descendent in the Sp. Tree;

3) Back Edge: Leads from a Node to its ancestor

4) Cross Edge: Leads to another Node which is neither anc nor desc;



$G=(V,E)$

DFS Sp. Tree

1) Tree Edges
$\langle AB \rangle \langle BC \rangle \langle DD \rangle$

2) Forward Edge:
$\langle AD \rangle$

3) Back Edge:
$\langle CA \rangle$

4) Cross:
$\langle DC \rangle$

Topic : Graph Techniques

THANK - YOU