$$\left\{ G = (V, E) \; ; \; \begin{array}{l} |V| = n \\ |E| = e \end{array} \right\}$$

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

→ The Time Complexity of BF Algo. for a Complete Graph having 'n' vertices is $\underline{O(n^3)}$.

- Complete Graph ⇒ $e = O(n^2)$.

Derive a DP based recurrence using P of opt. repr. solution to the problem.

GRAPH TECHNIQUEES :-

Traversal :- Visiting all the nodes of the tree/graph in a specified order and processing the info. only once.

Graph Traversals.

- → Depth First Search (DFS)
  - Undirected
    - Conn.
    - Discon.
  - Directed
    - DAG
- → Breadth First Search (BFS)
  - FIFO BFS
  - LIFO BFS ..

I. DFS in undirected Graphs :-

a) Connected Graphs :- Any 2 vertices connected via edge/ nodes.

- 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). [Stored in stack].

→ Dead Node :- Node which has been fully explored.

"In BFS — stored in queue."

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;

" (d/f times are +ve Integers).

(x) d/f

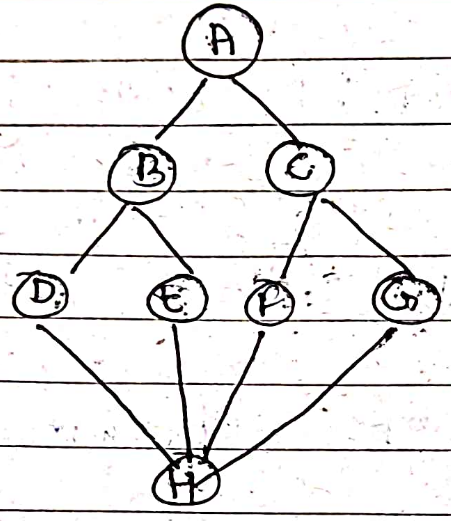
└ Representation

DFS in undirected Connected Graph.

→ One major difference between graph and tree traversal is :-

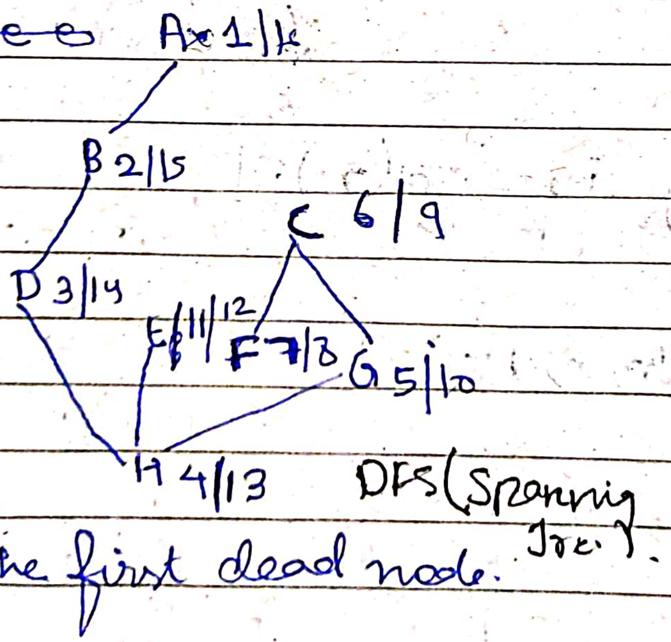
- Tree traversal ~~PA~~ is always unique.

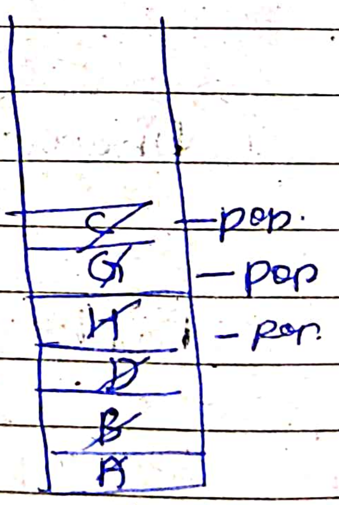
- Graph traversal ~~PA~~ can not be unique, it can be start from any vertex.



A; B; D; H; G; C; F; E

→ ~~We have a~~



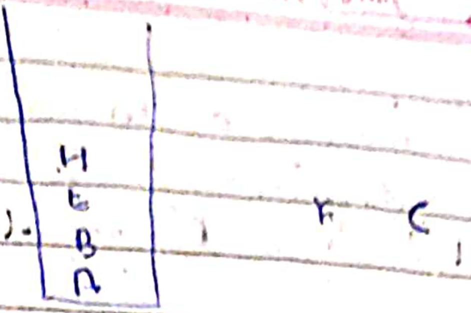


Stack contains live node.

∴ F is the first dead node.

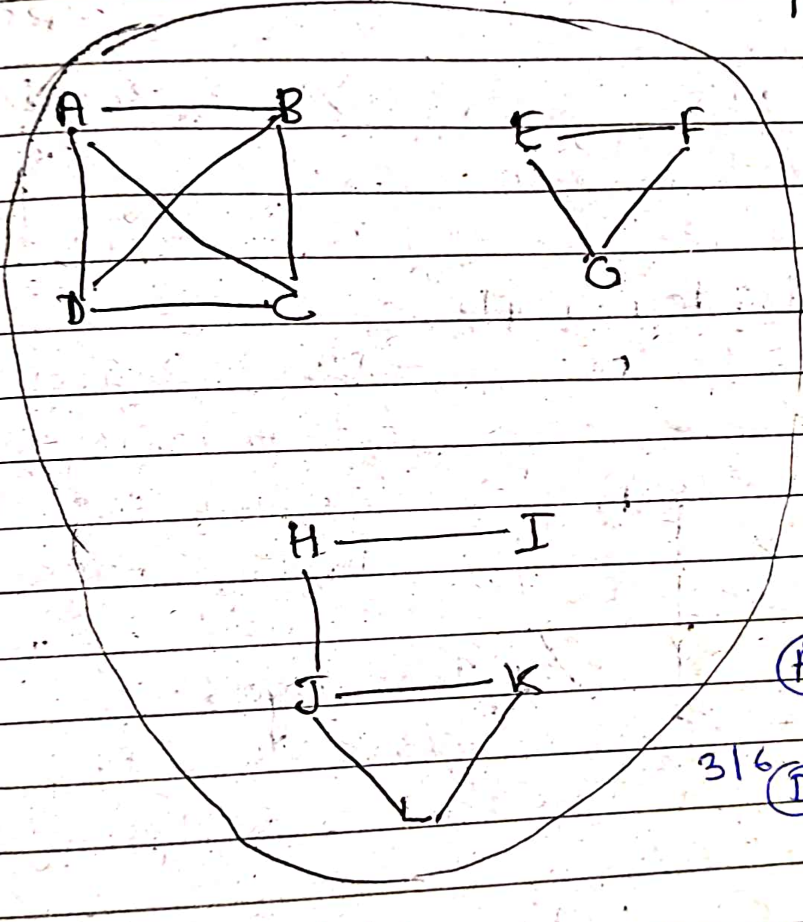
Here the condition for DFS is ~~the~~ for the connected graph ~~the~~ I comes to halt, Stack should be empty.

Depending on above graph, the valid/invalid DFS:-

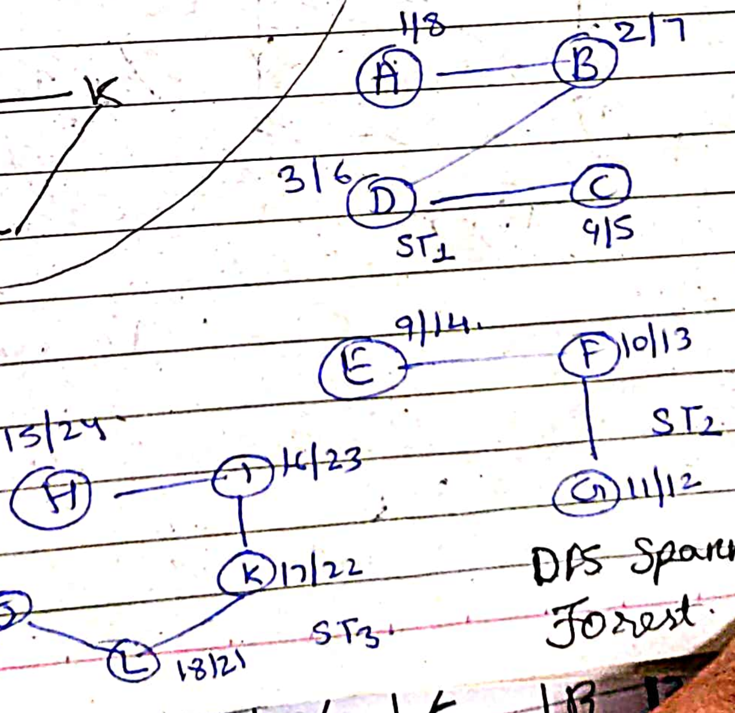


a) A, B, E, H, D, F, C, G ✓
b) A, C, F, E, H, G, B, D ✗
c) H, D, B, E, A, C, F, G ✓
d) H, F, C, B, A, D, F, G ✗
e) G, C, F, H, E, B, D, A. ✓

2) DFS in undirected Disconnected/Disjoint graphs:



: DFT
↓
Depth First traversal.
(3 connected components).
↓
Maximal connected subgraph.

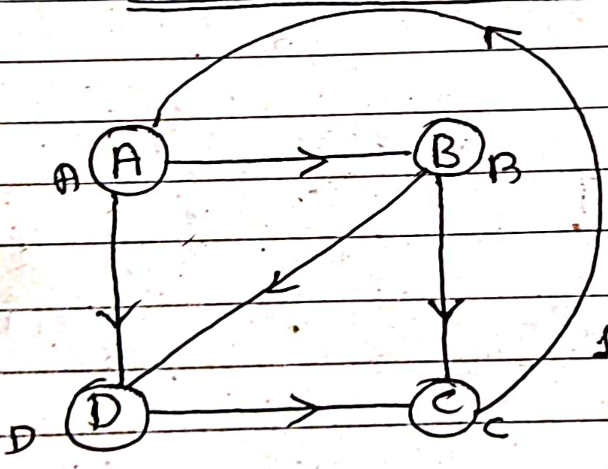


DFS Spanning Forest.

→ Consider a und. 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>	: Discn
3	<8,10>	<18,22>	<3,15>	<6,12>	: Dis <RSP><Q>
4.	<18,22>	<12,15>	<8,10>	<25,30>	Discon <R><S><P><Q>

* DFS in Directed Graphs :-



G(V,E).

⇒ DFS when carried out on a Directed, leads to the following types of edges;

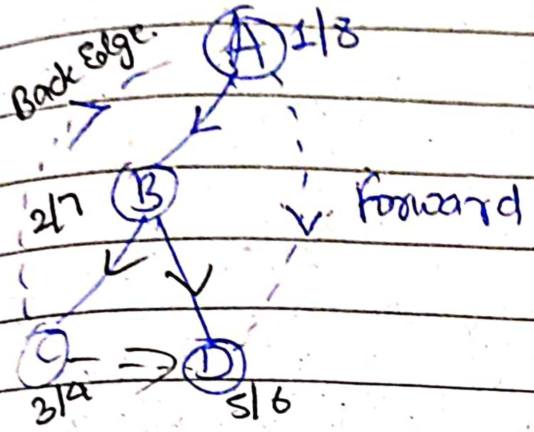
1) Tree Edge :- are part of DFS sp. Tree|forest.

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 Dex.

DFS Sp. tree.



1) Tree edges

<AB><BC><BD>

2) Forward Edge:

<AD>.

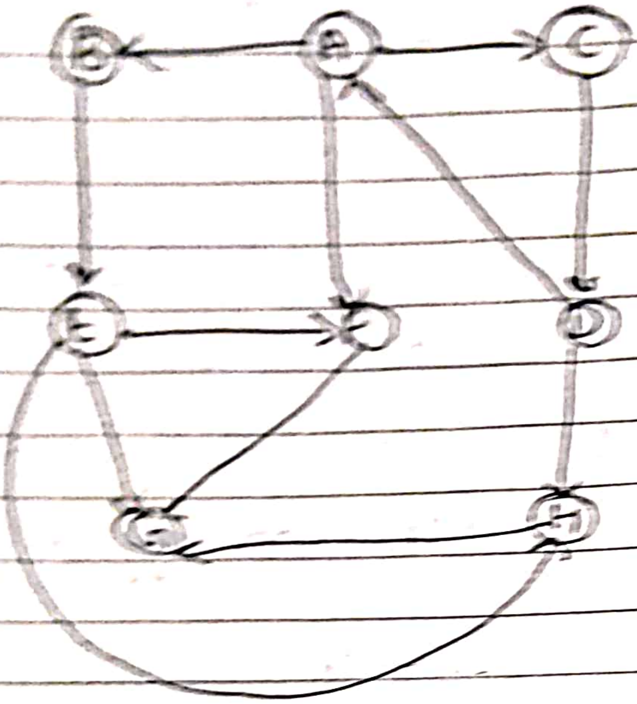
3) Back Edge:

<CA>

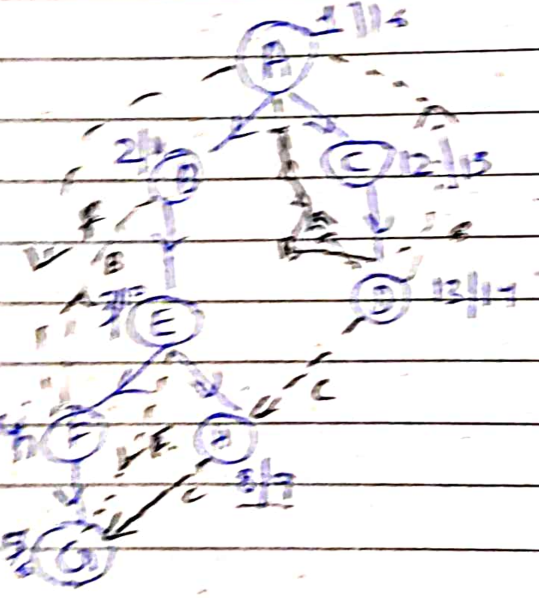
4) Cross:

<DC>

L2



→



∴ the question is how many back, cross and forward edges are there?

Back :- 2

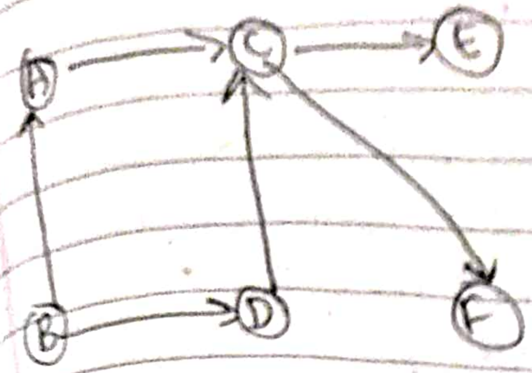
Cross :- 2

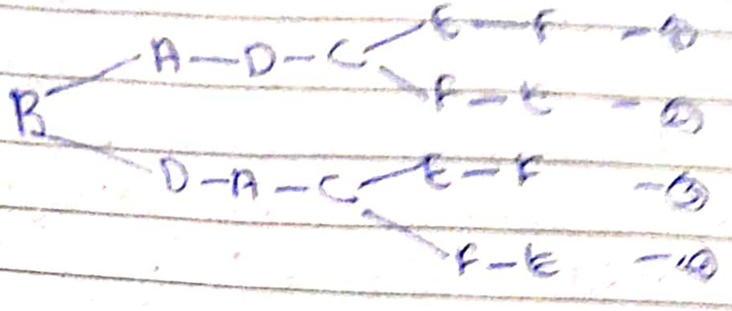
forward :- 2

DFS IN DAG { Directed Acyclic Graph }

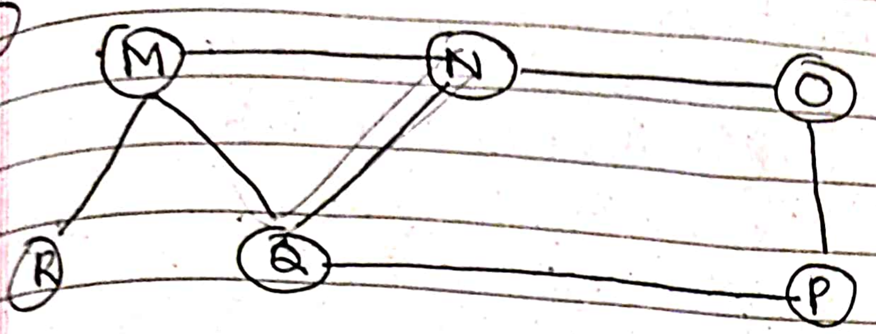
Topological sort:

<Linear order of the vertices, rep. the activities maintaining precedence.





2



BFS :-

M; R; N; Q; O; P

M

Live	~~R~~	~~N~~	~~Q~~	O	P
Parent	M	M	N	N	Q

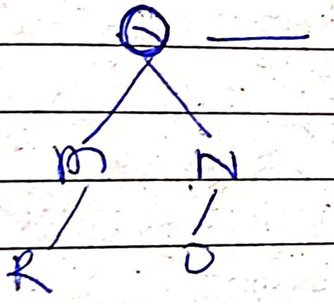
PDR

Q; M; N; R; O; P

Q; M; N

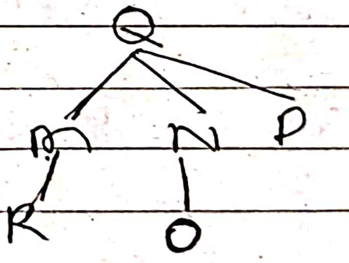
Li	~~M~~	~~N~~	~~R~~	O	P
P.	Q	Q	Q	N	O

~~M~~	~~N~~	R	O
Q	Q	M	N



Q	~~M~~	M	R	O
	Q	Q	M	

Q M; N;

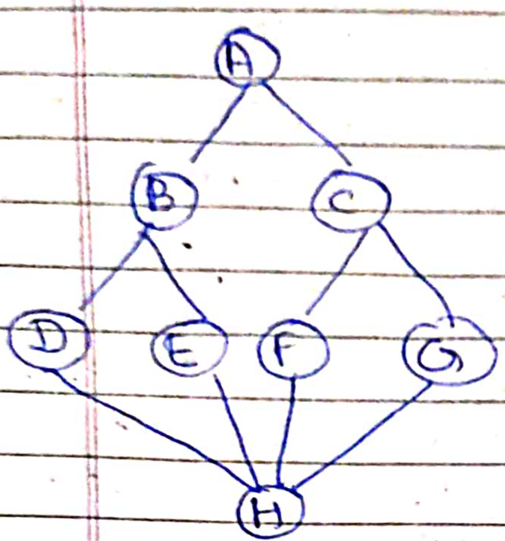


PRO

L	~~M~~	~~N~~	~~P~~	~~R~~	~~O~~
P	Q	Q	Q	M	N

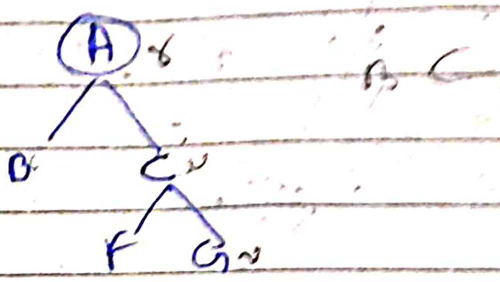
Q M N P R O

LIFO- BFS :-



~~A B C~~
A C G H E D F B

Now let's solve this using LIFO problem



Live	~~B~~	~~C~~	~~F~~	~~G~~	~~H~~	~~D~~	~~E~~
Parent.	A	A	C	C	C	H	H

∴ the sequence is A; C; G; H; E; D; F; B

⇒ this looks like DFS but it is not because after E D can not be searched properly.