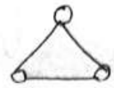
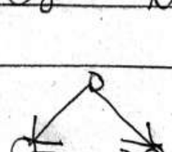


Tree	Graph
i) There is a unique node called root in trees.	i) all nodes called as root in graph.
ii) There will not be any cycle.	ii) A cycle can be formed.
iii) all trees are graphs.	iii) all graphs aren't trees.
iv) Less complexity compared to graphs.	iv) high complexity than trees due to loops.
v) Hierarchical	v) Network.

*) <u>Undirected Graph:</u>

A graph with only undirected edges.

*) <u>Directed Graph:</u> A graph with only directed edges is said to be ______.



3

Date

Page

*) Spanning Tree:

$|E|C_{|V|-1}$ — no of cycles.
Complete Graph n^{n-2}

→ connected

→ undirected graph

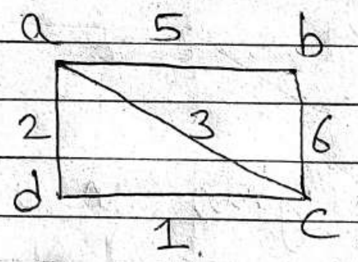
→ composed of all the vertices & some of edges.

⇒ Requirements for spanning tree:

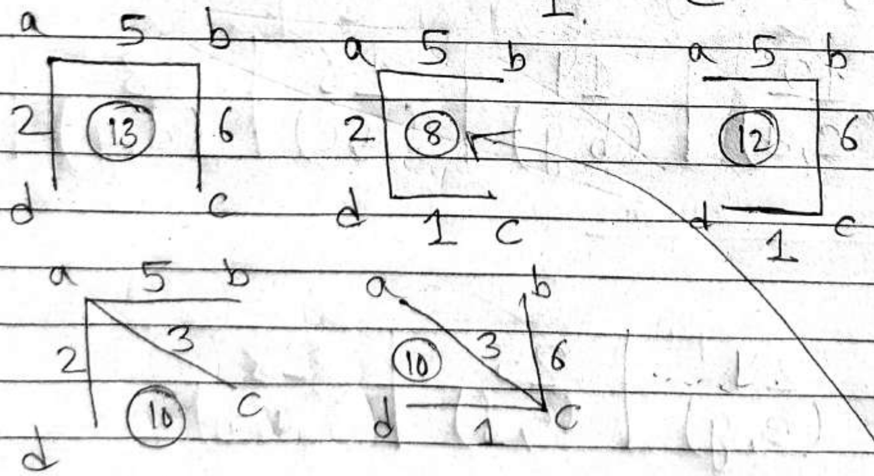
→ For n vertices, to construct the spanning tree, we require n vertices & $(n-1)$ edges

→ No loop or cycle exist.

Example:



$n = 4$
$n - 1 = 3$



~~${}^5C_3 - 2$~~
~~$20 - 2$~~
~~$= 20$~~

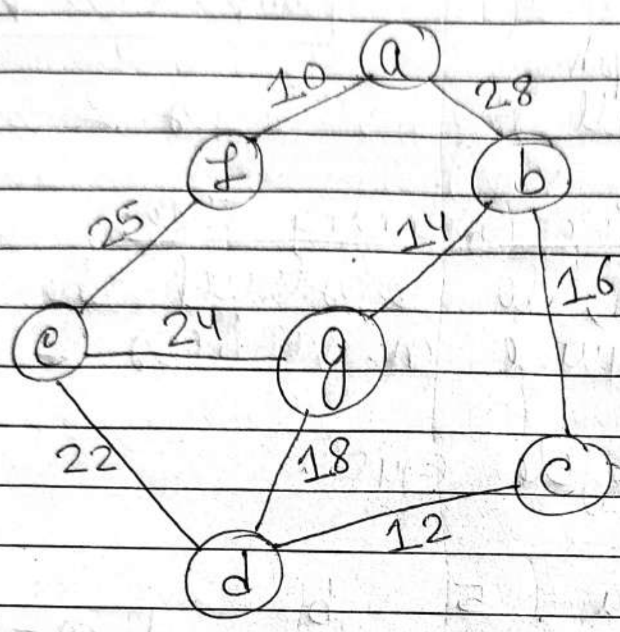
${}^6C_3 - 4$
120

4^2
$= 16$

Minimum Cost Spanning Tree = 8

④ Finding minimum cost spanning tree algorithms:

① **Kruskal's Algorithm:**



Sol^n^ →

Tabulating weights in ascending order:

✓	✓	✓	✓	×
10	12	14	16	18
(a, f)	(c, d)	(b, g)	(b, c)	(d, g)

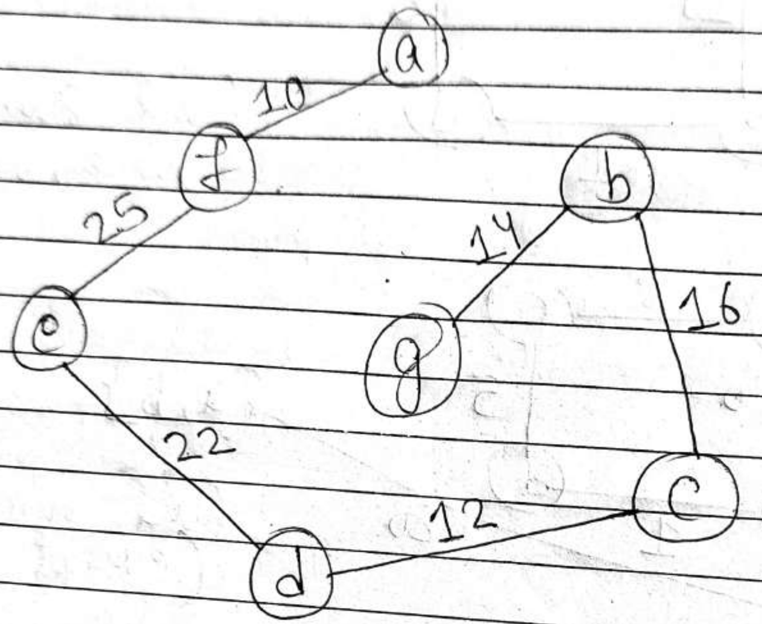
✓	×	✓	×
22	24	25	28
(e, d)	(e, g)	(e, f)	(a, b)

5

Now,

No. of vertices in spanning tree = $n = 7$

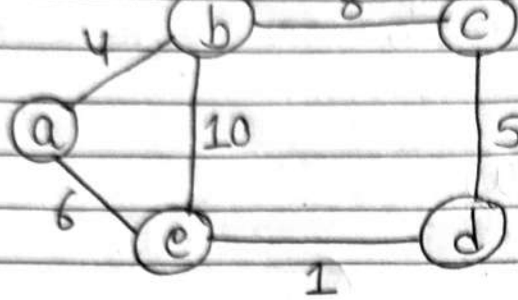
No. of edges " " " $= n-1 = 7-1 = 6$



Minimum Cost $= 10 + 25 + 22 + 12 + 16 + 14$

$= 99$

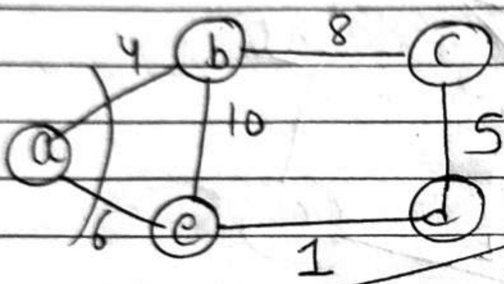
2) Prims Algorithm: (cut method)



No. of edges $= n-1 = 5-1 = 4$

(cut from any node, let a)

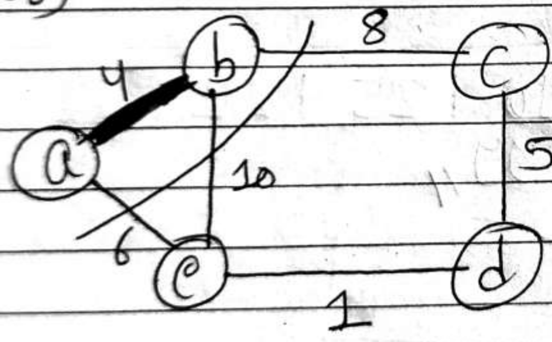
i)



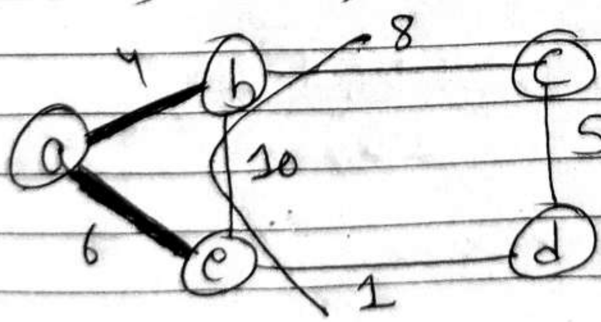
$(4<6)$ so $\{a, b\}$

(cut simultaneously (except visited)

so,



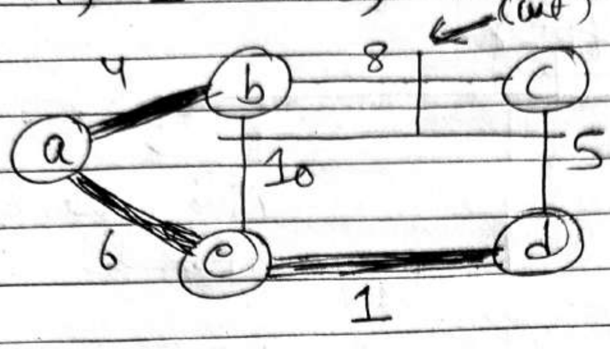
Since, $(6<8<10) \Rightarrow$ so, $\{a, b, e\}$



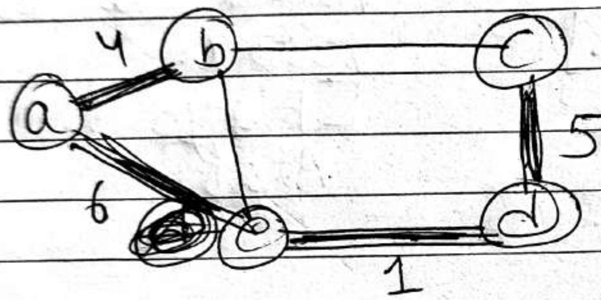
7

Since, $1<8<10$, so $\{a, b, e, d\}$

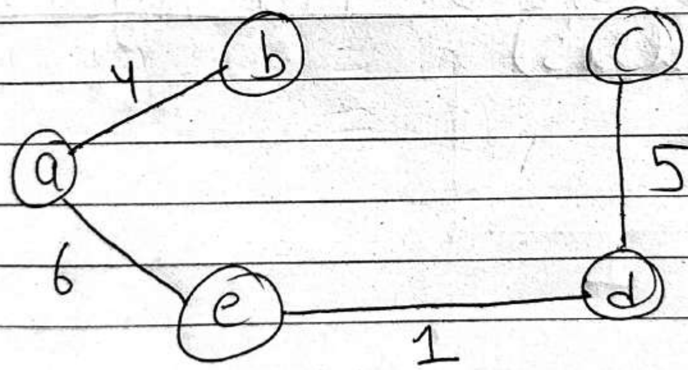
(cut)



Since, $5<8<10$, so, $\{a, b, e, d, c\}$



So,



Minimum Cost $= 4+6+1+5$
$= 16$

8 Shortest Path Algorithms:

Types

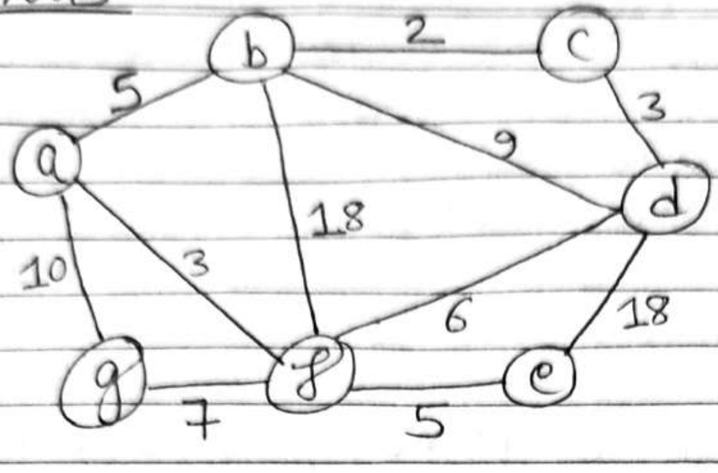
- Single Source Shortest Path
  → Cost of shortest path from source to destination [Dijkstra's Algo.]

- All pairs Shortest Path
  → Cost to shortest path from each vertex to every vertex. [Floyd Warshall's Algo.]

i) Single source Shortest Path Algorithm:
(Dijkstra's)

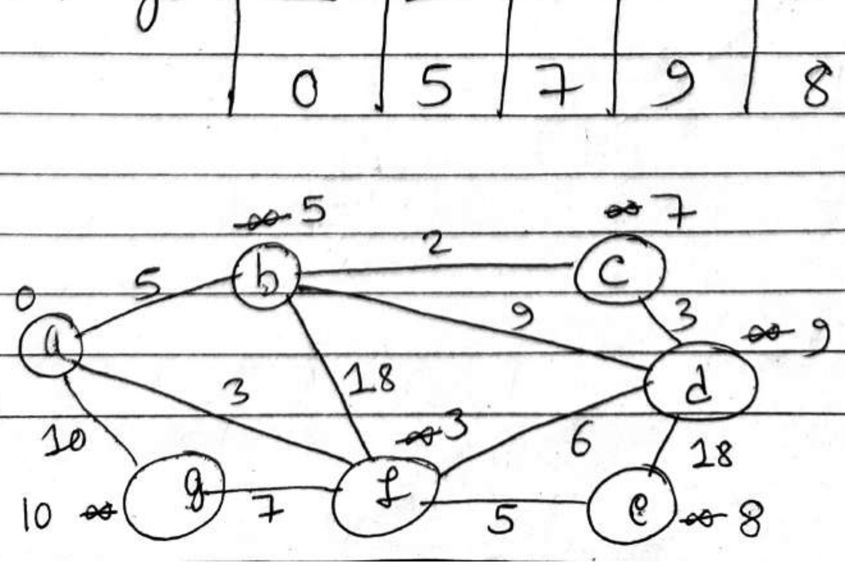
9

Dijkstra's



The calculation for shortest path is tabulated below:

	a	b	c	d	e	f	g
{a}	$\boxed{0}$	∞	∞	∞	∞	∞	∞
{a, f}	$\boxed{0}$	5	∞	∞	∞	$\boxed{3}$	10
{a, f, b}	$\boxed{0}$	$\boxed{5}$	∞	9	8	$\boxed{3}$	10
{a, f, b, c}	$\boxed{0}$	$\boxed{5}$	$\boxed{7}$	9	8	$\boxed{3}$	10
{a, f, b, c, e}	$\boxed{0}$	$\boxed{5}$	$\boxed{7}$	9	$\boxed{8}$	$\boxed{3}$	10
{a, f, b, c, e, d}	$\boxed{0}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{8}$	$\boxed{3}$	10
{a, f, b, c, e, d, g}	$\boxed{0}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{8}$	$\boxed{3}$	$\boxed{10}$
	0	5	7	9	8	3	10



Next page

Date ______
Page ______

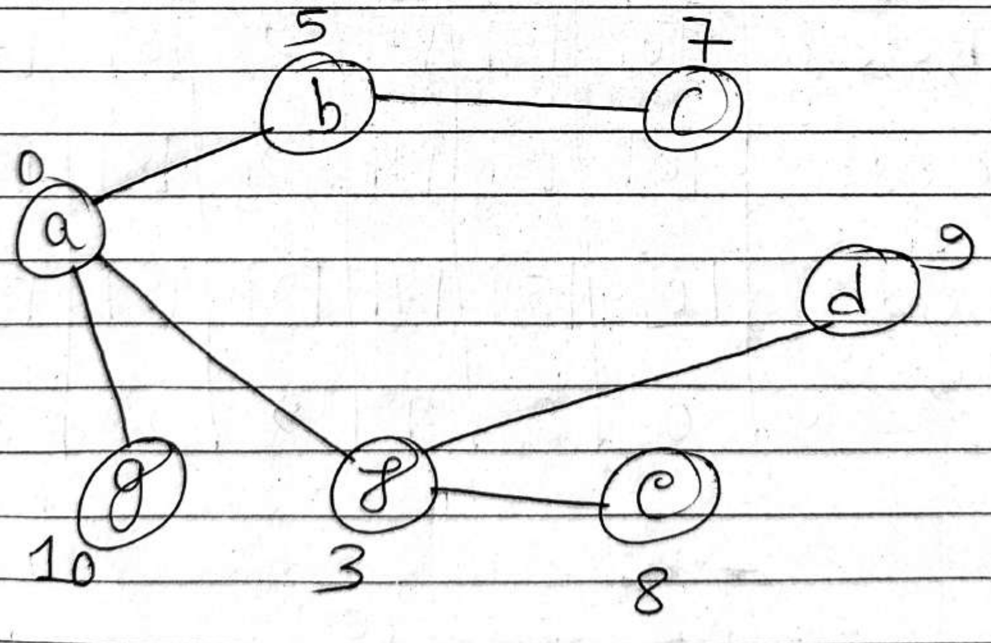
Advantages:

→ used in Google Maps.
→ telephone n/w makes use of it.

Disadvantages:

→ It conducts a blind scan which takes a lot of processing time.
→ It can't give proper result for the graphs having -ve weighed edges.

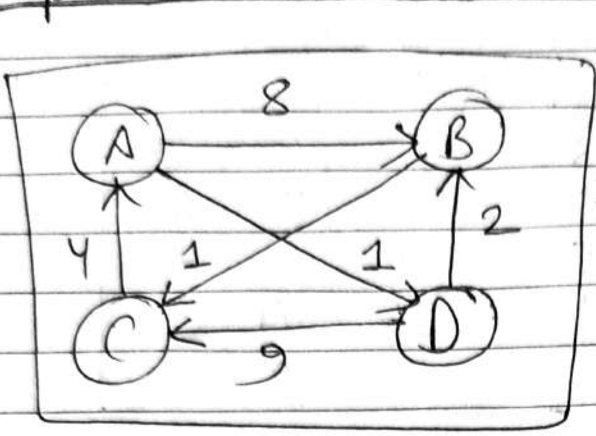
⇒ Finally we get the following Shortest Path Tree (SPT):



11

Date
Page

ii) All pairs shortest Path (Floyd Warshall Algo):



→ ~~Extra~~

Step 1: Initializing the distance matrix D^0.

$$D^0 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

(don't use indirect path)

Step 2: Using Node A as the intermediate node.
(using indirect path from/through A)

(row & column along A remains same)

$$D^A = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

Rough:

$B \to C = B \to A, A \to C$ (consider if possible)

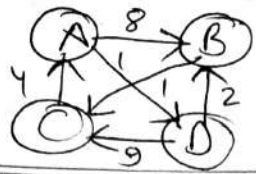
Since, $B \not\to A \Rightarrow \infty$ & $B \to C \Rightarrow 1$

$1 < \infty$, 1 is inserted

$C \to B \Rightarrow \infty$

$C \to A + A \to B = 12$

12



Step 3: Using Node B as intermediate node.
(using indirect path via A, B.)
(so row & column along B will be same as in D^A)

$$D^B = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & 9 & 1 \\ B & \infty & 0 & 1 & \infty \\ C & 4 & 12 & 0 & 5 \\ D & \infty & 2 & 3 & 0 \end{array}$$

(considering D^A)

$A \to C \Rightarrow \infty$

$A \to B + B \to C \Rightarrow 8 + 1 = 9$ (inserted)

$A \to D \Rightarrow 1$

$A \to B + B \to D \Rightarrow 8 + \infty \Rightarrow \infty$ ($1 < \infty$, 1 inserted)

$D \to C \Rightarrow 9$

$D \to B + B \to C \Rightarrow 2 + 1 = 3$ ($3 < 9$, 3 inserted)

Step 4: Using Node C as intermediate node.
(using indirect path via A, B & C.)
(so row & column along C will be same as in D^B)

$$D^C = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 8 & 9 & 1 \\ B & 5 & 0 & 1 & 6 \\ C & 4 & 12 & 0 & 5 \\ D & 7 & 2 & 3 & 0 \end{array}$$

$B \to A \Rightarrow \infty$, so, $B \to C + C \to A \Rightarrow 1 + 4 = 5$,

$D \to A \Rightarrow 9 + 4 = 13$ or $2 + 1 + 4 = 7$ ($7 < 13$, 7 is inserted)

$B \to D \Rightarrow B \to C + C \to A + A \to D \Rightarrow 1 + 4 + 1 = 6$

(13)

Date
Page

Step 5: Using Node D as intermediate node.
(using indirect path via A, B, C & D.)

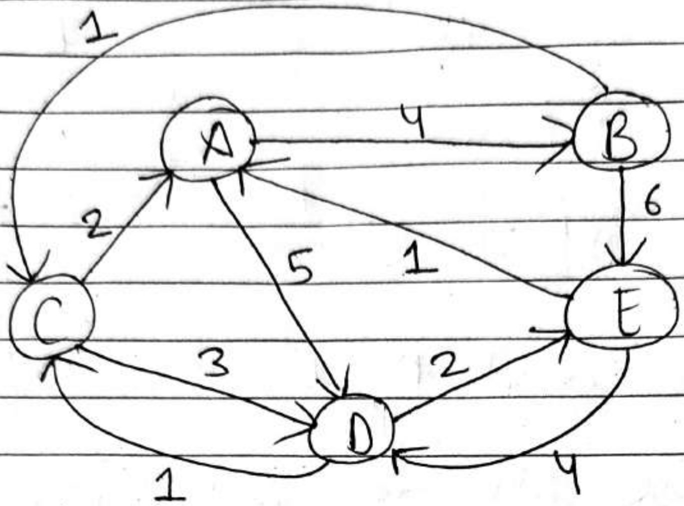
$$D^{D} = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 3 & 4 & 1 \\ B & 5 & 0 & 1 & 6 \\ C & 4 & 7 & 0 & 5 \\ D & 7 & 2 & 3 & 0 \end{array}$$

$A \to C \Rightarrow A \to D + D \to B + B \to C \Rightarrow 1+2+1 = 4$

⇒ Finally, since all the nodes have been treated as an intermediate node, we get the following all pairs shortest path in matrix form.

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 3 & 4 & 1 \\ B & 5 & 0 & 1 & 6 \\ C & 4 & 7 & 0 & 5 \\ D & 7 & 2 & 3 & 0 \end{array}$$ //

*) Floyd Warshall Algorithm (Example 2):



(Direct way only)

$$D^0 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 4 & \infty & 5 & \infty \\ \infty & 0 & 1 & \infty & 6 \\ 2 & \infty & 0 & 3 & \infty \\ \infty & \infty & 1 & 0 & 2 \\ 1 & \infty & \infty & 4 & 0 \end{bmatrix} \end{array}$$

$$D^A = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 4 & \infty & 5 & \infty \\ \infty & 0 & 1 & \infty & 6 \\ 2 & 6 & 0 & 3 & \infty \\ \infty & \infty & 1 & 0 & 2 \\ 1 & 5 & \infty & 4 & 0 \end{bmatrix} \end{array}$$

$$D^{B} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 4 & 5 & 5 & 10 \\ \infty & 0 & 1 & \infty & 6 \\ 2 & 6 & 0 & 3 & 12\,\cancel{\infty} \\ \infty & \infty & 1 & 0 & 2 \\ 1 & 5 & \cancel{\infty}\,6 & 4 & 0 \end{bmatrix} \end{array}$$

$$D^{C} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 4 & 5 & 5 & 10 \\ 3 & 0 & 1 & 4 & 6 \\ 2 & 6 & 0 & 3 & 12\,\cancel{\infty} \\ 3 & 7 & 1 & 0 & 2 \\ 1 & 5 & \cancel{\infty}\,6 & 4 & 0 \end{bmatrix} \end{array}$$

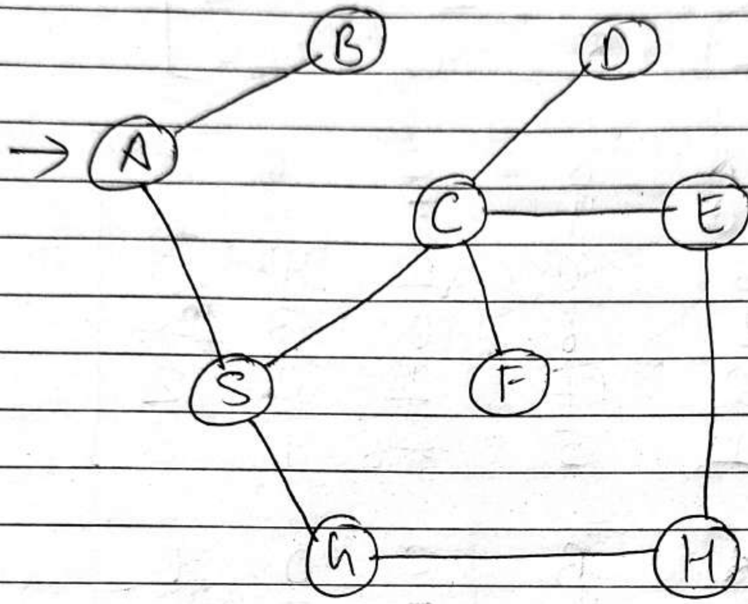
$$D^{D} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 4 & 5 & 5 & 7\,\cancel{10} \\ 3 & 0 & 1 & 4 & 6 \\ 2 & 6 & 0 & 3 & 5\,\cancel{12} \\ 3 & 7 & 1 & 0 & 2 \\ 1 & 5 & 5 & 4 & 0 \end{bmatrix} \end{array}$$

$$D^{E} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 4 & 5 & 5 & 7 \\ 3 & 0 & 1 & 4 & 6 \\ 2 & 6 & 0 & 3 & 5 \\ 3 & 7 & 1 & 0 & 2 \\ 1 & 5 & 5 & 4 & 0 \end{bmatrix} \end{array}$$

//

*) Graph Traversal:

i) Breadth First Search (BFS): [Queue]



Queue: ~~B~~, ~~S~~ ~~C~~, ~~G~~ , ~~D~~, ~~E~~, ~~F~~ , ~~H~~

O/p: A, B, S, C, G, D, E, F, H

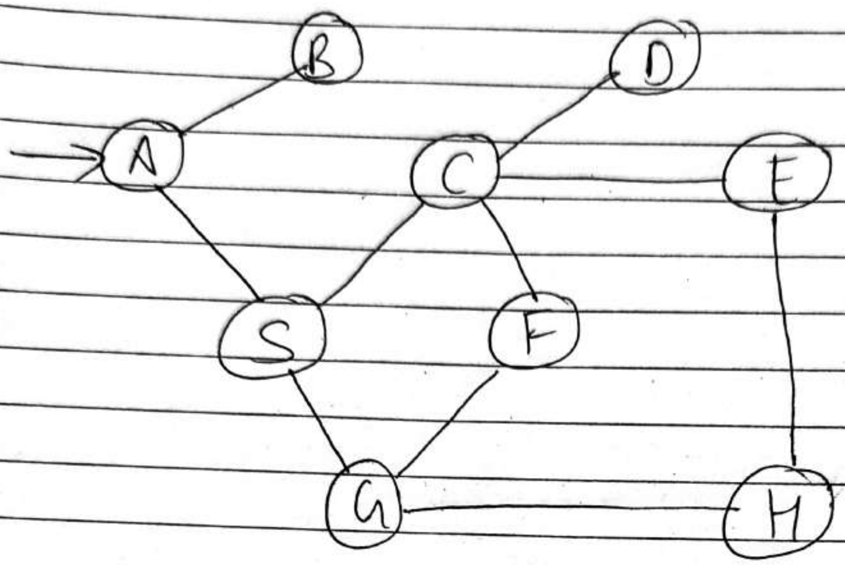
Steps:

i) Visit node
ii) Add all adjacent nodes to queue. (order doesn't matter.)
iii) Visit node in queue in FIFO order & add it to O/p.
iv) Add adjacent nodes of visited node in (iii) & repeat.

17

Date ______
Page ______

ii) Depth First Search (DFS):



Stack:

iii) ~~F~~
iv) ~~G~~
v) ~~H~~
vi) ~~E~~
ii) ~~D~~
vii) ~~C~~
viii) ~~S~~
i) ~~B~~
ix) ~~A~~

O/p: A B S C D E H G F

Steps:

i) Add visited node to stack.
ii) Add its one adjacent node (alphabetically order) to stack. (top of stack)
iii) Again visit the further nodes one by by & add into stack.
iv) Pop the node from stack if its all adjacent nodes have been visited.