Tutorial 5

Name - Amandeep Singh
Section - G
Roll No. - 53
Uni Roll No. - 2016623
Course - B.Tech CSE
Sem - 4

Ans 1

BFS	DFS
(i) uses queue data structure.	→ uses stack data structure
(ii) Stands for Breadth first search	→ Stands for Depth first search.
(iii) Can be used to find the single source shortest path in an unweighted graph.	→ we might transverse through more edges to reach a destination vertex from a source.
(iv) Siblings are visited before the children	→ children are visited before the Siblings.
Applications	**Applications**
→ Minimum spanning tree for an unweighted graph.	→ Detecting cycle in a graph.
→ peer to peer Networks	→ Path finding.
→ Social Networking websites	→ Topological Sorting
→ GPS Navigation systems	→ Solving puzzles with only one solution.

Ans 2 In BFS we use queue data structure as queue is used when things don't have to be processed immediatly but have to be processed in FIFO order like a BFS.

→ It is used in social networking sites and GPS Navigation Systems.

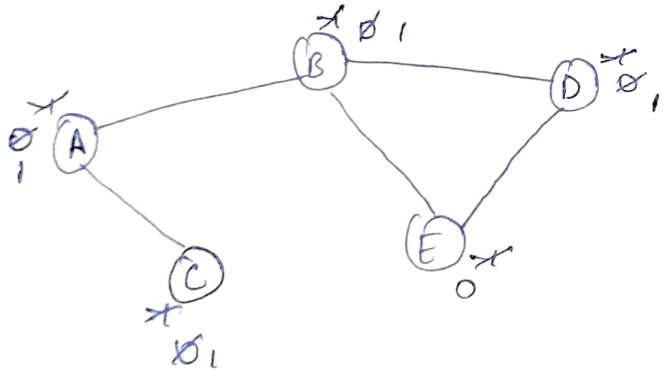
→ In DFS stack is used as DFS uses backtracking. (2)
For DFS, we retrieve it from root to the farthest node as much as possible, this is the same idea as LIFO (Last In First Out).
→ It is used in topological sortings.

Ans 3 → Dense graph is a graph in which the no. of edges is close to the maximal no. of edges

→ Sparse graph is a graph in which the no. of edges is close to the minimal no. of edges. It can be disconnected graph.

* Adjacency lists are preferred for sparse graph & adjacency matrix for dense graph.

Ans 4 Cycle detection in Undirected graph (BFS)



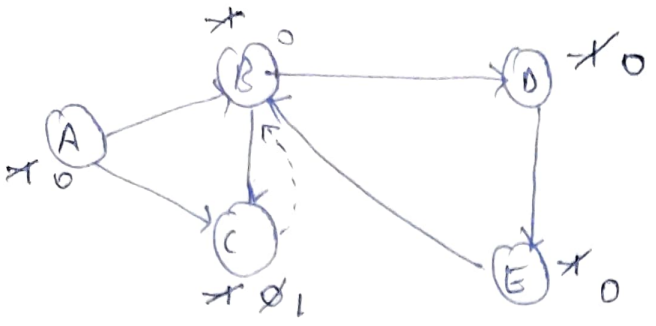
-1 → unvisited
0 → into the queue
1 → transversed

Queue → A B C D E

Visited set → A B C D

When D checks it adjacent vertices it finds E with 0
→ If any vertex finds the adjacent vertex with flag 0, then it contains cycle.

→ Cycle detection in Directed Graph (DFS)



Stack : E D B A

Visited set : A B C D E

Parent Map

vertex	Parent
A	–
B	A
C	B
D	B
E	D

⇒ B → D → E → B

Here E finds B (adjacent vertex of E) with 0.

⇒ It contains a cycle.

Ans 5 The disjoint set data structure is also known as union-find data structure & merge-find set. It is a data structure that contains a collision of disjoint or non-overlapping sets

→ The disjoint set means that when the set is partitioned into the disjoint subsets, various operations can be performed on it -

→ In this case, we can add new sets, we can merge the sets, & we can also find the representatives. members of a set.

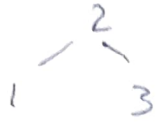
* Operations on disjoint Set :-

(1) Union → If S1 & S2 are two disjoint set, their uniform S1 ∪ S2 is a set of all elements x such that x is in either S1 or S2..

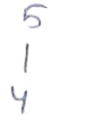
(2) Find → Given an element to find the set containing it.

eg →

S1



S2



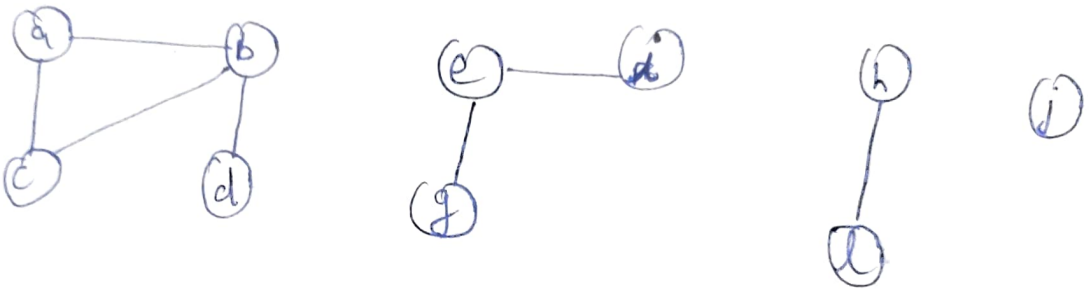
find (3) = S1

find (5) ⇒ S2

return in which set x belongs.

(3) Make-Set (x): Creates a set containing x.

Ans 7



$V = \{a, b, c, d, e, g, h, i, j, l\}$

$E = \{(a,b), (a,c), (b,c), (b,d), (e,i), (e,g), (h,l), (j)\}$

	{a}	{b}	{c}	{d}	{e}	{g}	{h}	{i}	{j}	{l}
(a,b)	{a,b}		{c}	{d}	{e}	{g}	{h}	{i}	{j}	{l}
(a,c)	{a,b,c}			{d}	{e}	{g}	{h}	{i}	{j}	{l}
(b,c)	{a,b,c}			{d}	{e}	{g}	{h}	{i}	{j}	{l}
(b,d)	{a,b,c,d}				{e}	{g}	{h}	{i}	{j}	{l}
(e,i)	{a,b,c,d}				{e,i}	{g}	{h}		{j}	{l}
(e,g)	{a,b,c,d}				{e,i,g}		{h}		{j}	{l}
(h,l)	{a,b,c,d}				{e,i,g}		{h,l}		{j}	
(j)	{a,b,c,d}				{e,i,g}		{h,l}		{j}	

we have,

{a, b, c, d}

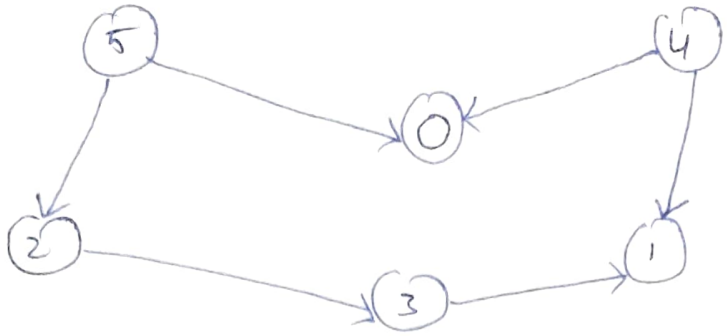
{e, i, g}

{h, l}

{j}

Ans

Ans 8



0

1

2 → 3

3 → 1

4 → 0 → 1

5 → 2 → 0

Algorithm :-

(1) Go to node 0, it has no outgoing edges so push node 0 into the stack & mark it visited.

(2) Go to node 1, again it has no outing edges, push node 1 into stack.

(3) Go to node 2, process all the adjacent nodes & mark 2 visited.

(4) Note 3 is already visited to continue with next node

(5) Go to node 4, all its adjacent nodes are already visited so, push node 4 into the stack & mark it visited

(6) Go to node 5, all its adjacent nodes are already visited so, push node 5 into the stack & mark visited.

5 4 2 3 1 0

Output

Ans 9 Heap is generally preffered for priority queue implementations because it helps provide better performance compared to array or linked list.

→ Algorithm where priority queue is used :-

(1) Dijikstra's shortest path Algo. :- when the graph is stored in the form of a adjancy list or matrix, priority queue can be used to extract minimum efficiency. when implementing Dijikstra's algorithm.

(2) Prim's algo. :- To store keys of nodes & another side extract the minimum key nodes at every step.

Ans 10

	Min Heap	Max Heap
(i)	for every pair of the parent & descendant child node, the parent node always has lower value than its child node	→ The parent node has greater value than descendant child node.
(ii)	The value of nodes inc. as we transverse from root to leaf node	→ The value of nodes decreases as we transverse from root to leaf node.
(iii)	Root node has the lowest value	→ The root node has the greatest value