ASSIGNMENT:- 05

Ans 1.)

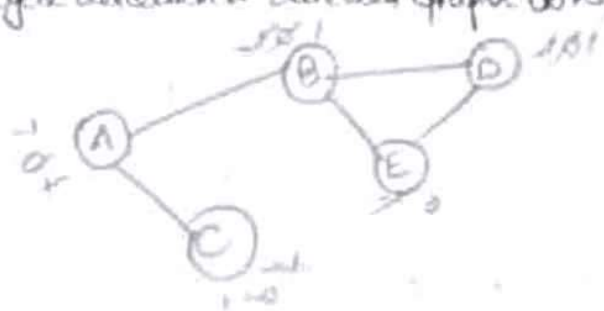
BFS	DFS
• uses queue data structure.	• uses stack data structure.
• Stands for Breadth First Search	• stands for Depth First Search.
• can be used to find single source shortest path in an unweighted graph and we reach a vertex with min. no. of edges from a source vertex.	• we might traverse through more edges to reach a destination vertex from source.
• Siblings are visited before children.	• Children are visited before siblings.
Applications:-	Applications:-
• Shortest path & minimum spanning tree in unweighted graph.	• Detecting cycle in graph.
• Peer to peer networks.	• Path finding.
• Social Networking websites.	• Topographical Sorting.
• GPS Navigation systems.	• Solving puzzles with only 1 solution.

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

In DFS; stack data structure is used as it is beneficial for backtracking. For DFS, we retrieve it from root to the farthest node as much as possible; giving it a LIFO like approach.

Ans 3.) Dense Graph is a graph in which 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 directed Graph (BFS)



-1 - unvisited.
0 - into queue
1 - traversed.

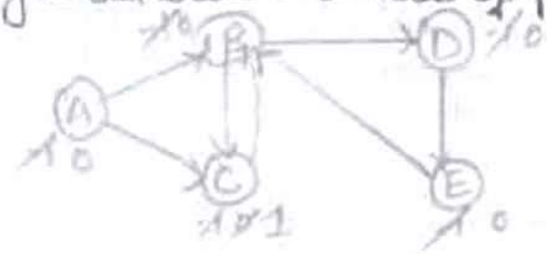
Queue: | A | B | C | D | E |

Visited set:- | A | B | C | D |

When D checks its adjacent vertices it finds E with 0.

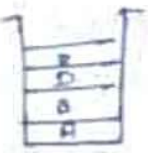
If any vertex finds the adjacent vertex with flag 0, then it contains cycle.

Cycle detection in directed graph (DFS)



-1 - unvisited.
0 - visited & stack push.
1 - visited & popped.

Stack:-



visited:-
ABCDE
→ D → D → E → B

Here E find B with 0.
⇒ the a cycle.

Parent Map

Vertex	Parent
A	—
B	A
C	B
D	B
E	D

Ans 5.) The DISJOINT SETS data structure is also known as Union-find data structure and merge-find set. It is a data structure that contains a collection of disjoint or non-overlapping sets.

The DISJOINT SETS means when the set of partitioned into the disjoint subsets, various operation are performed on it.

In this case, we can add new sets, merge them & also find the representative member of a set. It also allows to find out whether the two elements are at same set or not effectively

Operations on Disjoint sets :-

(1.) Union:-

(a) If sets S1 & S2 are 2 disjoint sets, then union S1 & S2 is a set of all

(b) If the sets should be disjoint S1 ∪ S2 replaced S1 & S2 which no longer exists

(c) It is achieved by simply making one of the trees as a subtree of other, that is; to set parent field of one of the roots of the trees to other root.

Merge with 1

(2) Find:-
Given an element of; to find the set containing it.

Eg:- find(x) ⇒ S1
find(4) ⇒ S2

(3) Make Set(x):-
create a set containing x
makeset(1) = {1}

Ques 6)

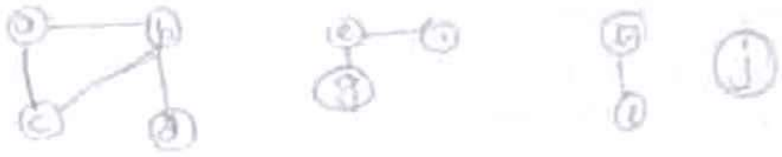
BFS:- child G D F H C E A B Path:-
Parent - G G G H C E A G → H → C → E → A → B

DFS:-

Node visited	Stack
G	G
D	F
H	C
F	E
C	A
E	B
A	
B	

Path
G → F → C → E → A → B.

Ans 7)



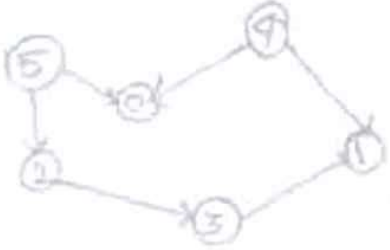
$V = \{a, b, c, d, e, f, g, h, i, j, l\}$

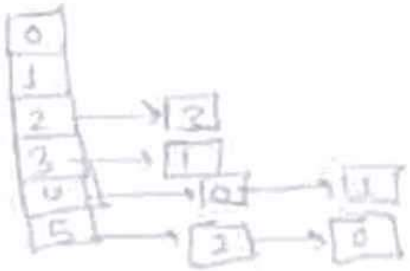
$E = \{(a,b), (a,c), (b,c), (b,d), (e,i), (e,g), (h,l), (j)\}$

Vertex / Edge ↓	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}	{l}
							{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}		{a}				{g}	{h}	{i}	{j}	{l}
(e,i)	⋮										
(e,g)											
(h,l)											
(j)	{a,b,c,d}		{e,g,i}				{h,l}		{j}		

Finally - {a,b,c,d}, {e,i,g}, {h,l} & {j}

Ans 8)





Algo:-

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

(2) Go to node 1, again it has no outgoing edges so push it in stack & mark as visited.

(3) Go to node 2, process all adjacent and mark it visited.

(4) Node 3 is visited, so continue with next node.

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

Output:-
5 4 2 3 1 0.

Ans 9.) Heap is generally preferred for priority queue implementation because heaps provide better performance compared to array & linked list.

Algorithms when priority queue is used:-

(1) Dijkstra's Algo :-
shortest path algorithm ; when the graph is stored in the form of adjacency list or matrix; priority queue can be used to extract minimum effectively when implementing it.

(2) Prim's Algo :-
To store keys of Nodes & extract minimum key node at every step.

Ans 10.)

Min Heap	Max Heap
• For every pair of parent & child node, parent node always has lower value than descendent child node.	• For every pair of the parent and descendant child node, the parent has higher value than child node.
• Value of nodes increases as we traverse from root to leaf node.	• Value of nodes decreases as we traverse from root to leaf node.
• Root node has the lowest value.	• Root node has the largest value.