DAA

Tutorial-5

Answer-1) Using BFS, we can find the minimum no. of nodes b/w a source node and destination node, while using DFS, we can find if a path exists b/w two nodes.

Applications:-

BFS - To detect cycles in a graph, min-distance, comparison, gps navigator.

DFS - To detect and compare multiple paths, detect cycle in a graph.

Answer-2) DFS: In DFS, we have to ~~search~~ traverse a whole branch of tree then you can traverse the adjacent nodes. So for keep tracking on the current node it requires last in first out approach which can be implemented by stack, after it reaches the depth of a node then all the nodes will be popped out of stack.

BFS: BFS does the search for nodes level by level i.e. it searches the nodes with their distance from the root. BFS require to visit the child nodes in order their parents were discovered. Whenever we visit a node, we insert all the nodes into our data structure. If you use a queue data structure, it is guaranteed that, you pop the nodes in order their parents were discovered.

Answer-3) As the name Indicate sparse graphs are sparsely connected. Usually the number of edges is in O(n) where n is the number of vertices. Therefore adjacency lists are preferred since they require constant space for every edge

Dense graphs are closely connected. Here number of edges is usually O(n²). Therefore adjacency matrix is preferred.

Answer-4) Cycle detection in BFS -

1) Compute in-degree for each vertex present in the graph and intialize the count of visited nodes as 0.

2) Pick all the vertices with in-degree as 0 and add them into a queue.

3) Remove a vertex from the queue and then

(i) Increment count of visited nodes by 1.

(ii) Decrease in-degree by 1 for all its neighboring nodes

(iii) If in-degree of a neighboring nodes is reduced to zero, then add it to queue.

4) Repeat 3 until queue is empty.

5) If count of visited nodes is not equal to the number of nodes in graph has cycle, otherwise not.

Cycle detection in DFS -

1) Mark the source_node as visited and mark as in-path node

2) Check if adjacent node has been marked as in-path node if yes then cycle found.

~~3) Repeat 2 for all adjacent nodes of source node~~

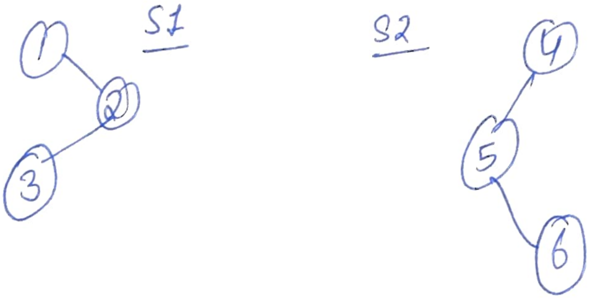
3) If adjacent node has not been visited then call the recursion by adjacent node.

4) Repeat 2, 3 until all the adjacent nodes of source node gets covered.

5) Now we are backtracking, unmark the source node in inpath node.

Answer-5) Disjoint Set Data Structure:- It is a DS that is used in various aspects of cycle detection. This is literally grouping of two or more disjoint sets.

eg:-



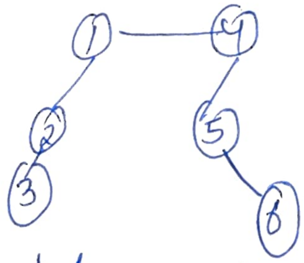
$S_1 = \{1, 2, 3\}$

$S_2 = \{4, 5, 6\}$

Operations →

1) Union → Merge two sets when edge is added.

$S1 \cup S_2 = S3$ →



2) Find( ) tells which element belongs to which set.

Find (1) = S1 / Find (4) = S2

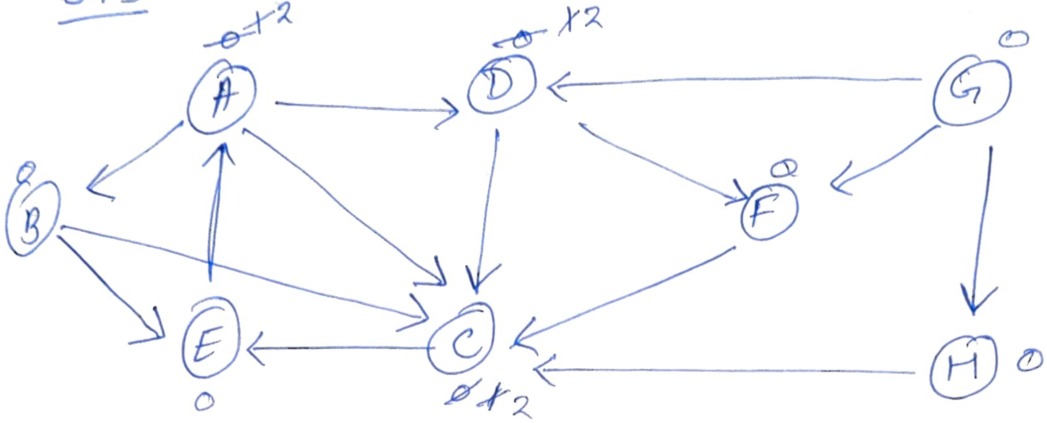
3) Intersection - output another set as common elements

$S1 \cap S2 = \{\phi\}$ $S4 \cap S5 = \{6\}$

$S4 = \{5, 6, 7, 8\}$ $S_5 = \{6, 9\}$

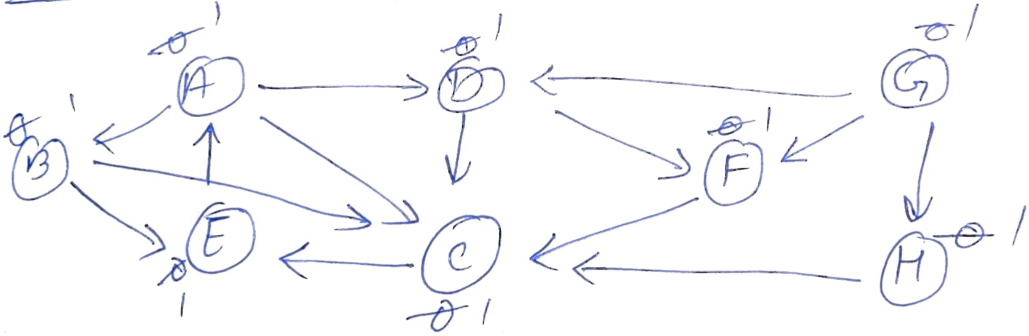
Answer-6

BFS →



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

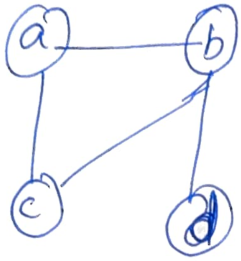
DFS →



node Processed	Stack
	G
G	DFH
D	CFH
C	EFH
E	AFH
A	BFH
B	FH
F	H
H	——

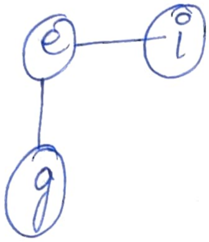
Answer-7 →

1



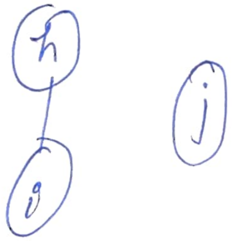
no.(v) = 4
no.(cc) = 1

2



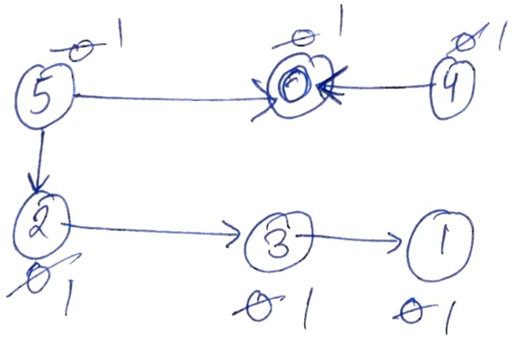
no.(v) = 3
no.(cc) = 1

3



no.(v) = 3
no.(cc) = 2

Answer-8) Topological Sorting



Adjacency List.

0 →
1 →
2 → 3
3 → 1
4 → 0, 1
5 → 2, 0

Stack

0	1	3	2	4	5

Topological = 542310

DFS Stack

4	0	1	3	2	5

Head → DFS → 5 → 2 → 3 → 1 → 0 → 4

Answer-9) Application of Priority Queue

1) Dijkstra's Algo ⇒ we need to use a priority queue here so that minimal edges can have higher priority.

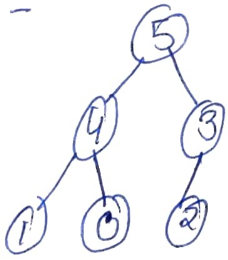
2) Load Balancing ⇒ Load balancing can be done from branches of higher priority to those of lower priority.

3) Interrupt Balancing ⇒ To provide proper numerical priority to more important interrupt.

4) Huffman Code ⇒ For data compression in Huffman code.

Answer-10) Max Heap → Here, value of parent is greater than the both children.

eg -



Min Heap → In this, value of parent is smaller than the both children.

eg -