TUTORIAL→ 5

Ans→1

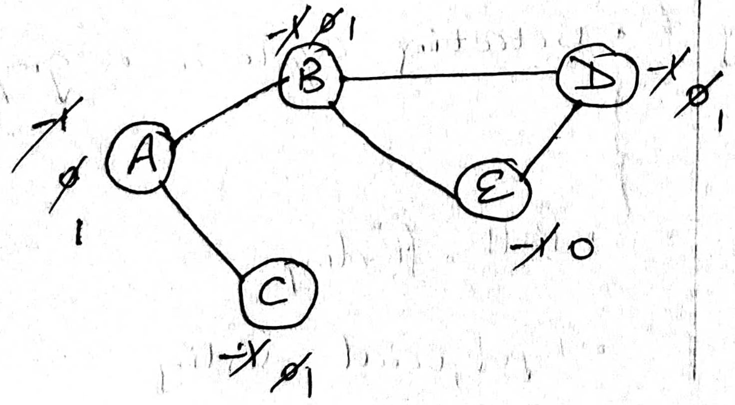
BFS	DFS
• uses queue DS	• uses stack DS
• Stands for Breadth First Search	• Stands for Depth First Search.
• Can be used to find Single source shortest path in an unweighted graph, & we reach a vertex with max. no. of edges from a source vertex.	• We might traverse through more edges to reach a destination vertex from a source.
• Siblings are visited before the children.	• Children are visited before the siblings.
Applications→	Applications→
• Shortest path & min spanning tree for unweighted graph.	• Detecting cycle in a graph.
• Peer to peer networks	• Path finding.
• Social Networking sites	• Topolgocial sorting

Ans→2 In BFS we use Queue DS as queue is used when things don't have to be proceesed immediately, but have to be proceesed in FIFO order like BFS.

In DFS stack is used as DFS uses backtracking. For DFS, we retrieve it from root to the farthest node as much as possible, this is the same idea as LIFO.

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 → traversed

ueue :

A	B	C	D	E

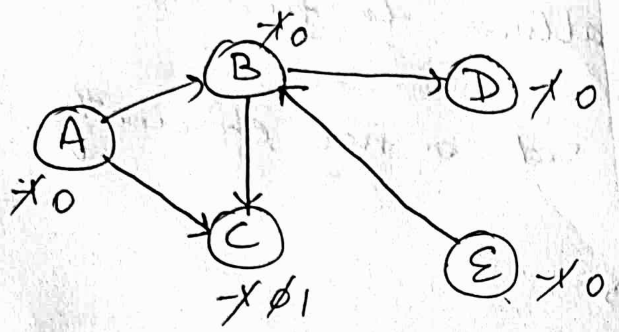
Visited set :

A	B	C	

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

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

Cycle Detection:



Stack :

E
D
B
A

Visited set : ABCDE

Parent Map :

Vertex	Parent
A	—
B	A
C	B
D	B
E	D

⇒ B → D → E → B

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

⇒ it contains a cycle.

Ans→6 The disjoint set ds. is also known as unio

find ds & merge-find set. It is a ds that contain

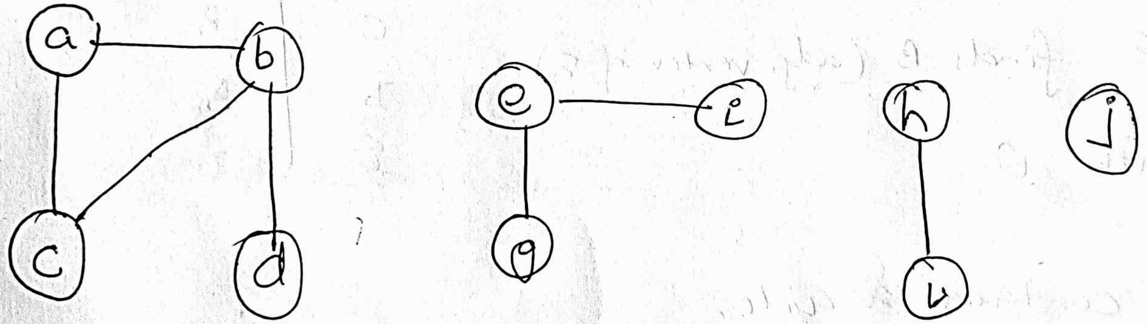
coll^{n} of disjoint or non-overlapping sets. The disjoint set means that when the set is partitioned into disjoint subsets, various option can be performed on it.

In this case, we can add new sets, we can merge the sets, & we can also find the representative member of the sets. It also allows to find out whether the 2 elements are in same set or not efficiently.

Operations on disjoint set →

1) UNION 2) FIND 3) Make_Set (X)

Ans→7



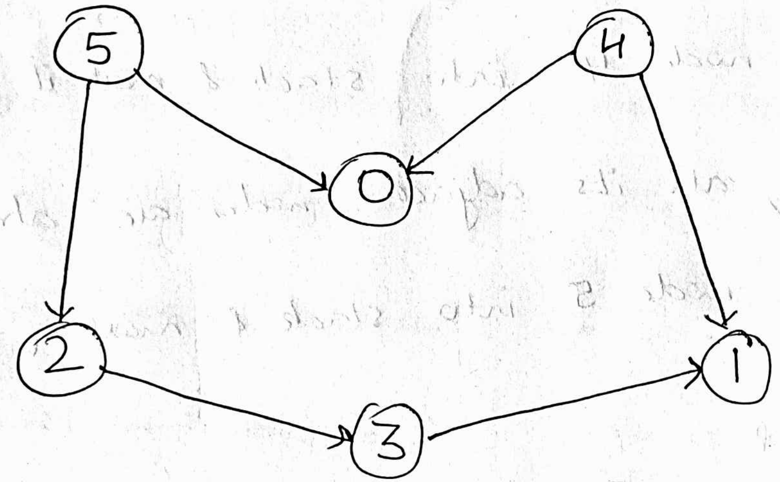
$V = \{a, b, c, d, e, ~~f~~, g, h, i, j, L\}$

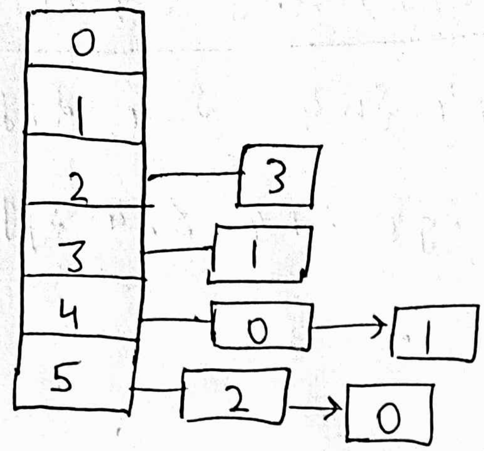
$E = \{(a,b), (a,c), (b,c), (b,d), (e,i), (e,g), (h,L), (j)\}$

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

We have : {a, b, c, d} {e, i, g} {h, l} {j}

Ans → 8





Algo:

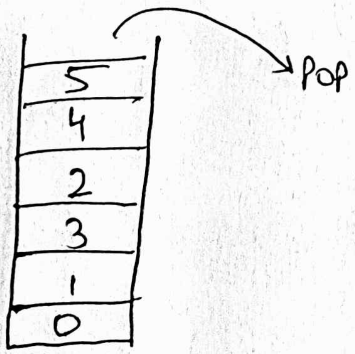
1) Go to node 0, it has no outgoing edges so push node 0 into the stack & mark it visited

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

3) Node 3 is already visited so continue with next node.

4) Go to node 4, all its adj. nodes are already visited so push node 4 into stack & mark it visited.

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



5 4 2 3 1 0

(output)

u→9

Heap is generally preferred for priority queue implementation ∵ heaps provide better performance compared to arrays or Linkedlist.

Algos where priority queue is used →

1) Dijkstra's shortest path algo
2) Prim's algorithm → to store keys of nodes & extract min key node at every step.

Ans → 10

Min Heap	Max Heap
• For every pair of the parent & descendent child node, the parent node always has lower value than des. child node.	• For every pair of pair & des. child node, the parent node has greater value than des. child node.
• The value of nodes inc. as we traverse from root to leaf node.	• The value of nodes decreases as we traverse from root to leaf node.
• Root node has Lowest value.	• The root node has the greatest value.