Tutorial - 5

Ans 1.

BFS	DFS
(i) Uses queue data structure	(i) Uses stack data structure
(ii) Stands for Breadth First Search	(ii) Stands for Depth First Search.
(iii) 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	(iii) We might traverse through more edges to reach a destination vertex from a source.
(iv) Siblings are visited before the children	(iv) Children are visited before the siblings.
Applications :-	Applications :-
(1) Shortest path and Minimum Spanning Tree for unweighted graph	(1) Detecting cycle in graph
(2) Peer to Peer Networks	(2) Path finding.
	(3) Topological sorting.

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 in FIFO order like BFS.

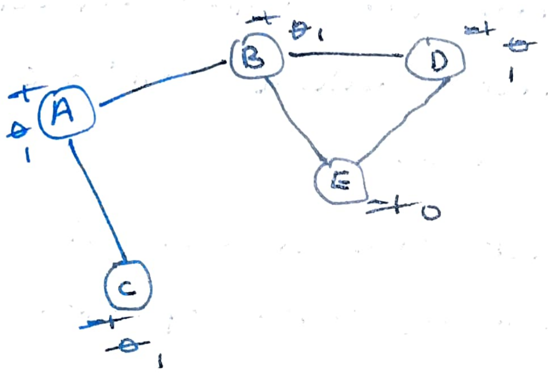
→ In DFS stack is used as DFS uses backtracking. For DFS, we retrieve it from root to the farthest node as much as possible. This is same idea as LIFO.

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

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

* Adjacency lists are preferred for sparse graph and Adjacency matrix for dense graph.

Ans 4. Cycle Detection in Undirected Graph (BFS)



-1 = unvisited

0 = into the queue.

1 = traversed.

Queue :-

A	B	C	D	E

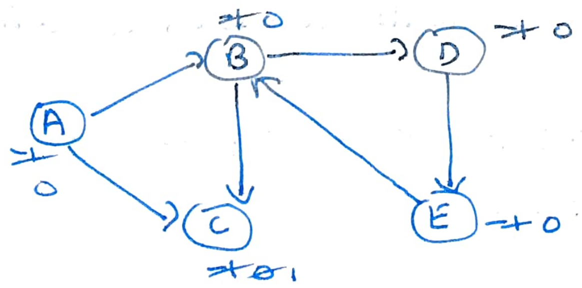
Visited set :-

A	B	C	D

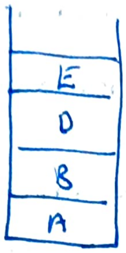
when D checks it adjocent vertices it finds E with 0.

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

* Cycle Detection in Directed Graph (DFS)



stack :



visited set :

ABCDE

Parent map.

Vertex	Parent
A	-
B	A
C	B
D	B
E	D

there 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 and merge-find set.

It is a data structure that contains a colln of disjoint or non-overlapping sets.

The disjoint set means that when the set is partitioned into the disjoint subsets, various operation can be performed on it.

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

* Operations on disjoint set.

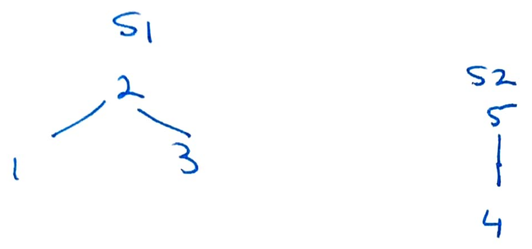
1. Union

a) If S1 and S2 are two disjoint sets, their union S1 ∪ S2 is a set of all elements x such that x is in either S1 or S2.

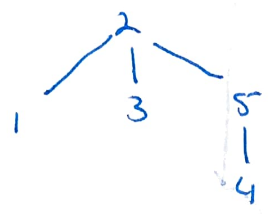
(b) As the sets should be disjoint S1 ∪ S2 replaces S1 & S2 which no longer exists.

c) Union is achieved by simply making one of the trees as a subtree of other i.e., to set parent field of one of the roots of the trees to other ~~r~~ root.

Ex.



S1 ∪ S2.

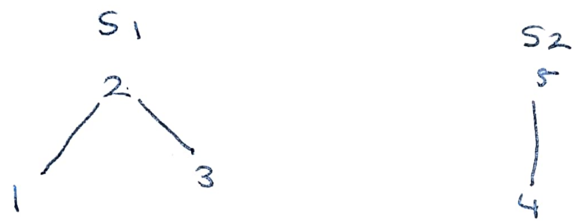


* Merge the sets containing X and containing Y into one.

2. Find

Given an element x, to find the set containing it

So,

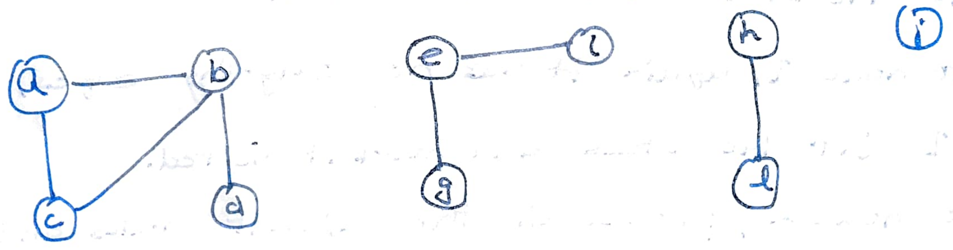


find(~~dy~~ 3) ⇒ S

return ~~k~~ in which set X belongs.

find (5) ⇒ S2.

Ans 7.



V = { a, b, c, d, e, g, h, i, j, ~~k~~ l }

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

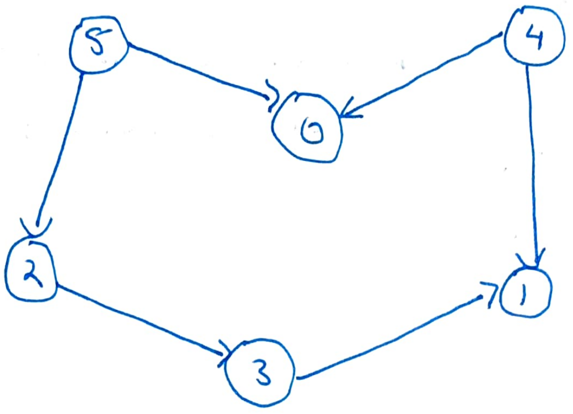
{a, b, c, d}

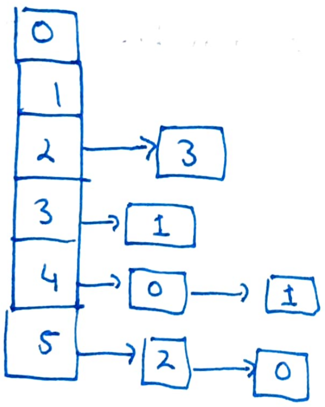
{e, i, g}

{h, l}

{j}

Ans 8.





Algo :-

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

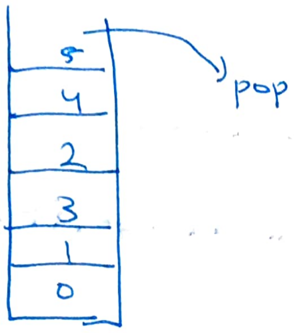
2. Go to node 1, again it has no outgoing edges, so push node 1 into the stack and mark it visited.

3. Go to node 2, process all the adjacent nodes and mark node 2 visited.

4. Node 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 and mark it visited.

=) 

5, 4, 2, 3, 1, 0
(output)

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

* Algorithms where priority queue is used]

1. Dijkstra's Shortest Path Algorithm :- When the graph is stored in the form of adjacency list or matrix, priority queue can be used to extract minimum efficiently when implementing Dijkstra algorithm.

2. Prim's algorithm :- To store keys of nodes and extract min. key node at every step.

Ans 10.

Min Heap	Max Heap
① Every path of the parent and descendent child node, the parent node always has lower value than descended child node.	(1) For every pair of parent & descendent child node, the parent node has greater value than descended child node.
③ Root node has the lowest value.	② The root node has the greatest value.