Name - Riya Chaudhary
Section - I
RollNo - 7

Tutorial - 5

1. DFS - DFS means depth First Search uses a stack to keep track of the next location to visit. It traverses according to tree depth.

Applications -

1. It is used in topological sorting, scheduling problems, cycle detection in graphs, & solving puzzles with only one solution such as a maze or a sudoku puzzle.

2. It ~~eth~~ helps in analyzing networks, if graph is bipartite.

BFS - BFS means Breadth First Search uses Queue data structure for finding the shortest path of unweighted graph.

Applications -

1. It is used to solve many problems in graph theory copying garbage collection, Cheney's algorithm.

2. It is used in Traversal mechanism on Tree.

2. BFS uses queue Data structure for finding the shortest path and DFS uses Stack data structure. Because in DFS, DFS algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration. and in BFS, queue is used because algorithm makes sure that every node is visited not more than once.

3. A dense graph in which the numberof edges is close to the maximal number of edges, means if every pair of vertices is connected by one edge.

Sparsh graph in which the number of edges is much less than the possible number of edges.
If the graph is sparse, we should store it as a list of edges.
If the graph is dense, we should store it as an adjacency matrix.

4. We do a BFS traversal of the graph. For every visited vertex 'v' if there is an adjacent 'u' such that u is already visited and u is not a parent of v, then there is a cycle in the graph.
If we don't find such an adjacent for any vertex. we say there is no cycle.
Using a Depth First Search (BFS) traversal algorithm we can detect cycles in a directed graph. If there is any self-loop in any node, it will be considered as a cycle, otherwise, when the child node has another edge to connect its parent, it will also a cycle.
We can use Dijkstra's shortest path algorithm, prim's algorithm, and heap Sort algorithm. So, Dijkstra's shortest path algorithm using priority queue. When graph is stored in the form of

5· The disjoint set can be defined as the subsets where there is no common element between the two sets Ex- S1= {1,2,3,4}
S2 = {5,6,7,8}.

There are 3 operations which is performed - making a new element set containing a new element.

Finding the representative of the set containing a given element.

Merging two sets.

10 individuals
a,b,c,d,e,f,g,h,i,j.

a ↔ b
b ↔ d
c ↔ f
c ↔ i
j ↔ e
g ↔ j

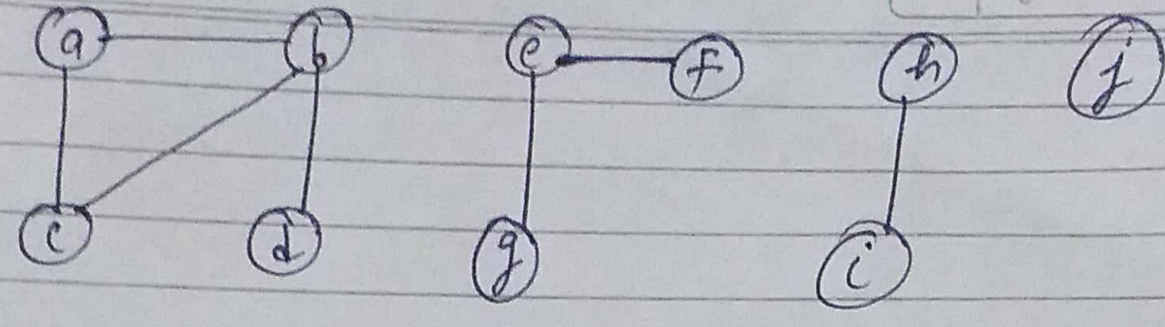
$G_1 = \{a,b,d\}$
$G_2 = \{c,f,i\}$
$G_3 = \{e,g,j\}$
$G_4 = \{h\}$

9. Heap data structure can be used to implement priority queue because it provides an efficient implementation of priority queues. It uses shift up operation. Swap the incorrectly placed node with its parent until the heap property is satisfied. For ex - As node 11 is less than node 32, So swap node 11 & node 32. Then, swap node 14 and node 32. At last, swap node 31 and node 32.

We can use Dijkstra's shortest path algorithm, prim's algorithm, and heap sort algorithm. So, Dijkstra's shortest path algorithm using priority queue. When graph is stored in the form of adjacency list. It can be used to extract minimum efficiently when implementig Dijkstra's algorithm.

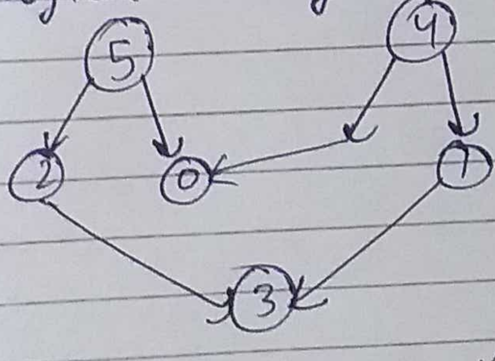
10.	Min heap	Max heap
1.	Key present at the root node must be less than or equal to keys present its children.	1. Key present at the root node must be greater than or equal to keys its children.
2.	The ascending priority	2. The descending priority.
3.	The smallest element is first to be popped from heap.	3. The largest element is first to be popped from heap.
4.	The smallest element has priority	4. The largest element has priority.
5.	The minimum key element present at the node root.	5. The maximum key element present at the root.

7·



Edge Processed	Collection of disjoint sets
initial	{a} {b}{c}{d}{e} {f}{g} {h} {i} {j}
(b,d)	{a}{b,d}{c}{e}{f}{g}{h}{i}{j}
(e,g)	{a}{b,d}{c}{eg} {h} {i}{j}{f}
(a,c)	{a,c}{b,d} {eg} {h}{i}{j}{f}
[h,i]	{a,c} {b,d}{e,g} {h,i}{j}{f}
(a,b)	{a,b,c,d} {e,g}{h,i} {j}{f}
(e,f)	{a,b,c,d} {e,f,g}{h,i} {j}
(b,c)	{a,b,c,d} {e,f,g}{h,i}{j}

8· Topological Sorting -



Topological Sorting 5 4 2 3 1 0
or 4 5 2 3 1 0