Date __/__/____

Saathi

Tutorial - 5

Que 1

Sol:

BFS	DFS
• Stands for Breadth first search	Stands for Depth first Search.
• BFS used queue to find the shortest path	DFS uses stack to find shortest path
• BFS is better when target is closer to source	DFS is better when target is from source
• As BFS consider all neighbours so it is not suitable for decision tree used in puzzle games	DFS is more suitable for decision tree. As with one decision we need to traverse further to argument the decision. If we search the conclusion
• BFS is slower than DFS	DFS is faster than BFS.

Application of DFS

- Using DFS we can find path between two vertices.
- We can perform topological sorting which is used to scheduling jobs.
- We can use DFS to detect cycles
- Using DFS, we can find strongly connected components of a graph.

Application BFS:

- BFS may also used to detect cycles.
- Finding shortest path and minimal Spanning tree in unweighted graphs.
- In networking Finding a route for packet transmission.
- Finding a route throughth GPS navigation system

Date __/__/____

Saathi

Ques2

Sol:- Breadth First Search (BFS) uses Queue data Structure. In BFS you mark any node in the graph as source node and start traversing from it. BFS visited an adjacent unvisited node, marks it as done and insert it into Queue.

DFS uses stack data structure because DFS traverse a graph in a depth word motion and uses stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

Ques3

Sol:- Sparse Graph:
A graph in which the number of edges is much less than the possible number of edges.

Dense Graph:
A dense Graph is a graph in which the number of edges is close to the maximal no. of edges of edges.

→ If the graph is sparse, we should store it as list of edges.
Alternatively if a graph is dense, we should store it as a adjacency matrix.

Ques4

Ans→ DFS can be used to detect cycle in a Graph. DFS for a Connected Graph produces a tree. There back edge present in the graph. A back edge is an edge that is from a node to itself or one of its ancestor in the tree produced by DFS.

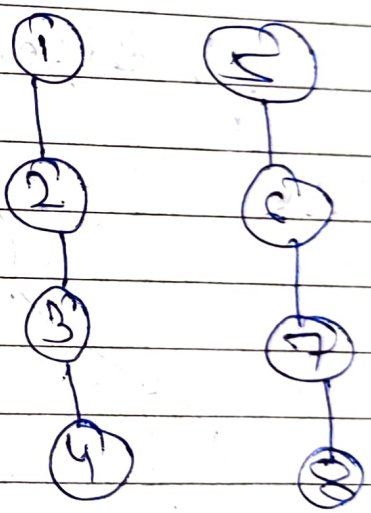
BFS can also be used to detect cycles. Just perform BFS while keeping a list of previous nodes to at each node visited or else construct a tree from the starting node. If I visit a node that is already marked by BFS, I found a cycle.

Ques5

Sol'n Disjoint Set Data Structure:

- It allows to find out whether the two element are in the same set or not efficiently.
- A disjoint set can be defined as the subsets when there is no common element between the two sets.

Eg. $S_1 = \{1, 2, 3, 4\}$
$S_2 = \{5, 6, 7, 8\}$



Operations performed:

(i) Find:

```
int find (int v)
{
    if (v == parent [v])
        return v;
    return parent [v] = find (parent [v]);
}
```

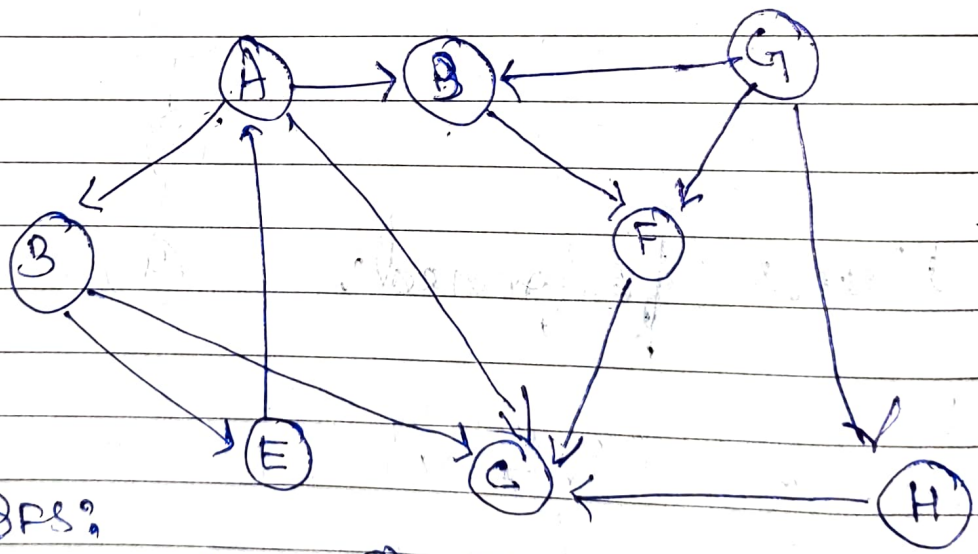
Date ___/___/___

Union:

```
Void union (int a, int b)
{
    a = find (a)
    b = find (b)
    if (a != b)
    {
        if (size[a] < size[b]
        { swap (a, b) }
        Parent [b] = a;
    } Size [a] += Size [b];
}
```

Ques 6

Soln



BFS:

Node :	B	E	C	A	D	F
parent :	-	B	B	E	A	D

Path : B → E → A → D → F

Date __/__/____

Saathi

DFS:

Node processed	B	B	C	E	A	D	F
Stack	B	CE	EE	AE	DE	FE	E

path: B → C → E → A → D → F

Ques:

Sol: V = {a} {b} {c} {d} {e} {F} {g} {h} {i} {j}

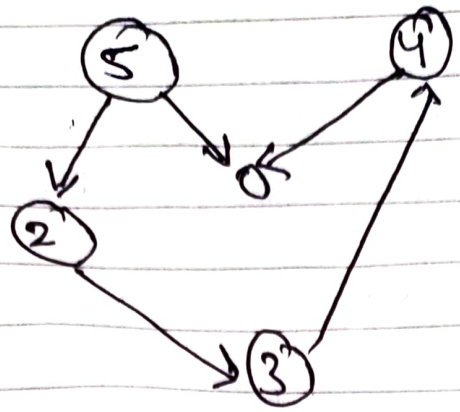
E = {a,b} {a,c} {b,c} {b,d} {e,f} {e,g} {h,i} {j}

(a,b)	{a,b} {c} {d} {e} {f} {g} {h} {i} {j}
(a,c)	{a,b,c} {d} {e} {f} {g} {h} {i} {j}
(b,c)	{a,b,c} {d} {e} {f} {g} {h} {i} {j}
(b,d)	{a,b,c,d} {e} {f} {g} {h} {i} {j}
(e,f)	{a,b,c,d} {e,f,g} {h} {i} {j}
(h,i)	{a,b,c,d} {e,f,g} {h,i} {j}

No. of Components connected = 3

Page No. ____

Ques8



Adjacency list

0 →
1 →
2 → 3
3 → 1
4 → 0, 1
5 → 2, 0

Visited:

0	1	2	3	4	5
False	False	False	False	False	F

Stack (empty)

Step 1: Topological Sort(0), visited[0] = true

Stack | 0 |

Step 2: Topological Sort(1), visited[1] = true

Stack | 0 | 1 |

Step 3: Topological Sort(2), visited[2] = true
↓
Topological Sort(3), visited[3] = true.

Stack | 0 | 1 | 3 | 2 |

Date __/__/____

Saathi

Step 4:

Stack | 0 | 1 | 3 | 2 | 4 |

Step 5:

Stack | 0 | 1 | 3 | 2 | 4 | 5 |

Step 6: Print all elements of stack from top to bottom.

→ 5, 4, 2, 3, 1, 0

Ques

Sol^n Algorithms that uses priority Queue:

i) Dijkstra's Algorithm -
When a graph is sorted in the form of list or matrix, priority queue can be used to extract minimum efficiency when implementig Dijsktra's Algorithm.

ii) Prim's Algorithm:
It is used to implement prims algorithm to store key of nodes to extract minimum key node at every step.

Page No.

Date __/__/____

iii) Data Compression:
It is used in Huffman's code which is used to compress data.

Ques 10

Sol →

Min Heap	Max Heap
• In min heap the key present at root node must be less than or equal to among the keys present at all its Childrens.	In max-heap the key present at root node must be greater or equal to the key present at all its childrens
• Uses the ascending priority.	Uses decending priority.
• The minimum key present at the root node.	The minimum key present at the root node.