Name: Nikita kumari

Roll: 19

Section: CST SPL-1.

Date : / /
Page :

Tutorial- 45

Q1.

	DFS	BFS
→	Uses stack as a data structure for finding a path.	→ Uses queue data structure for finding the shortest path.
→	stands for depth first search.	→ stands for Breadth first search
→	There is concept of backtracking	→ There is no concept of backtracking.
→	Requires less memory.	→ Requires more memory.
→	Children are visited before the siblings.	→ Here, siblings are visited before the children.

Applications of BFS:

→ In GPS navigation system to find the neighbouring places.

→ Search engines crawlers are used BFS to build index.

→ Used to broadcast some packets in networking.

→ Used in Ford-Fulkerson algorithm to find maximum flow in a network.

Applications of DFS.

→ Performing DFS on unweighted graph, it will create minimum spanning tree for all pair shortest path tree.

→ We can detect cycles in a graph.

→ Topological sort can be done using DFS.

→ Using DFS, we can find strongly connected components of a graph.

Q2. DFS uses stack as a data structure because, to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

BFS uses Queue as a data structure because, BFS searches for nodes levelwise, i.e., it searches the nodes w.r.t their distance from the root (or source).

Q3. Sparse:- A graph is said to be sparse if, the number of edges is much less than the possible number of edges.

Dense:- A graph is said to be dense if, the number of edges is close to the maximal number of edges or if every pair of vertices is connected by one edge.

For sparse graph, Adjacency list is preferrable representation of graph and it is also the most widely used. whereas, In dense graph, the Adjacency Matrix representation is used.

Q4. By following the below steps, we can find a cycle in BFS :-

1. compute number of incoming edges for each vertex present in the graph and Initialize the count of visited nodes as 0.

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

3. Perform Dequeue, then :-
   - Increment count of visited nodes by 1.
   - Decrease in degree by 1
   - If in-degree of a neighbouring nodes is reduced to zero, enqueue.

4. Repeat step 3 untill the queue is empty.
5. If count of visited nodes is not equal to the number of nodes in the graph

has cycle, otherwise not.

Q5. Disjoint set :- partitioning the individuals into different sets according to the groups in which they fall.

→ Disjoint data structure maintains collection of disjoint sets and each set is represented by its representative which is one of its members.

* Three operations :-

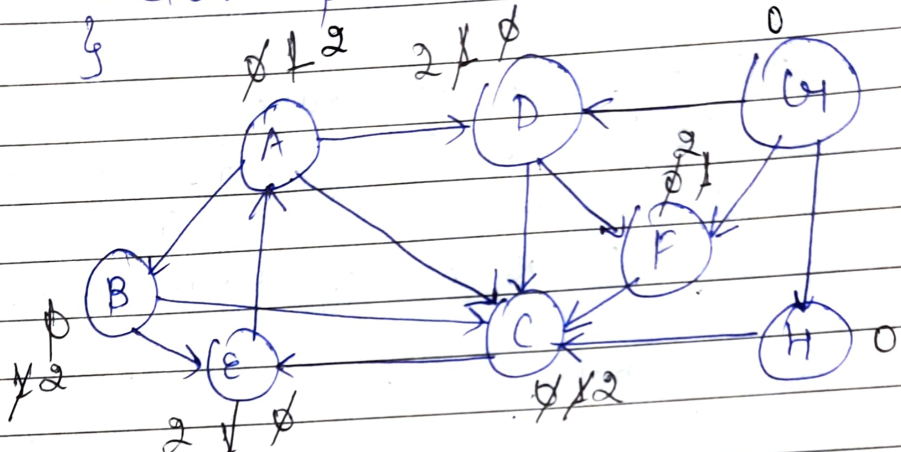
1. find()
2. Union ()
3. Union by Rank.

Example

```
    void union (int x; int y) {
    int xset = find (x);
    int yset = find (y);
     if( xset == yset)
         return;
   if (rank[xset] < rank[yset]) {
        parent[xset] = yset; }
  else if ( rank[xset] > rank[yset])
        parent[yset] = xset;
  else {
    parent[yset] = xset;
      rank[xset] = rank [xset] +1;
    }
  }
```

```
Int find ( Int x) {
  If ( parent [x] ! = x) {
     parent [x] = find ( parent [x]));
  }
 return parent [x];
}
```

Q6.



Applying BFS :- (Assuming start node = A)

Node	A	B	C	D	E	F
parent	–	A	A	A	B	D

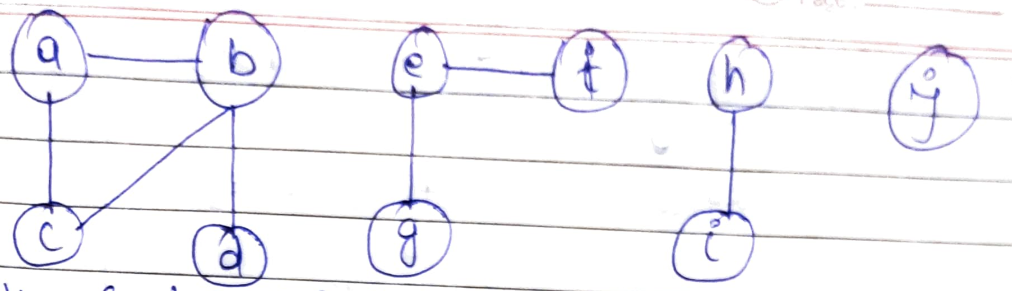
path → A → D → F

Note :- Here, G & H are not reachable node.

Applying DFS :-

	Node processed	Stack (←)
	A	A
	B	B C D
Possible path :	E	E C D
A → B → E → C → D → F	C	C D
	D	D
	F	F
		–

Q7.



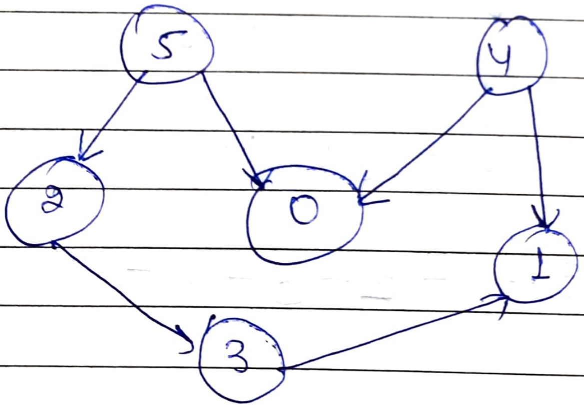
V = {a} {b} {c} {d} {e} {f} {g} {h} {i} {j}

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

Ans No. of connected components = 3.

Q8.



Applying topological sort.

1. Topological sort(0), [True]
↓
2. Topological Sort (1) [True]

3. Topological sort (2) [True]
↓
Topological sort (3) [True]
↓
Topological sort (1), visited [1]
= False

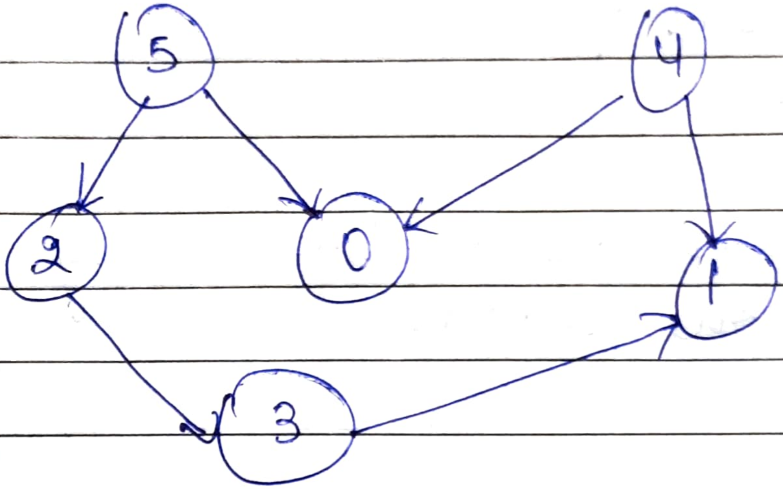
Stack
5.
4
2
3
1
0

4. Topological sort (4) [True]
↓
Topological sort (0), [False]
↓
Topological sort (1), [False]

5. Topological sort (5) [True]
↓
Topological sort (2) [False]
↓
Topological sort (0) [False]

Hence, 5 - 4 - 2 - 3 - 1 - 0 Ans.

* Applying DFS



Stack (←)

Node pr.	
5	5
2	2 0
3	3 0
1	1 0
0	0
	–

possible path: 5 - 2 - 3 - 1 - 0.

Q9. We can use heaps to implement the priority queue. It will take O(log N) time to insert and delete each element in the priority queue.

priority queues are used in many algorithms like Huffman codes, Prim's algorithm, Dijkstra's shortest path Algorithm, etc.

Because:- Priority queue is a special type of queue in which each element is associated with a priority value. And, elements are served on the basis of their priority.

Q10.

Min Heap	Max Heap.
→ In a min-heap, the minimum key element present at the root.	→ In a max-Heap, the maximum key element present at the root.
→ It uses the ascending priority.	→ A max heap uses descending priority.
→ The smallest element has priority.	→ In this, the largest element has priority.
→ The smallest element is the first to be popped from the heap.	→ The largest element is the first to be popped from the heap.