NAME→ Aman Dwivedi
SECTION→ CST
ROLL No→ 07

Date ________
Page ________

DAA
Assignment → 3

1- What is difference between DFS and BFs? Write application of both the algoritms.

→

	DFS	BFS
1-	It stands for Depth First Search.	It stands for Breath first Search.
2-	It use stack to find shortest path.	It uses Queue to find the shortest path.
3-	It is better when target is far from source.	It is better when target is closer to source.
4-	It is faster than BFS.	It is slower than DFS.
5-	Complexity: O(V+E)	Complexity O(V+E).

Applications of DFS:

* If we perform DFS on unweighed graph, then it will create minimum spanning tree for all pair shortest path tree.

* We can detect cycles in a graph using DFS. If we get one back-edge during BFS, then there must be one cycle.

* Using DFS, we can find path between two given vertices u and v.

Application of BFS :

* In peer-to-peer network like-bit torrent, BFS is used to find all neighbour nodes.

* Using GPS navigation system BFS is used to find neighbouring places.

* In networking, when we want to broadcast some packets, we use the BFS ~~ag~~ algoritm.

* BFS used in Ford-Fulkerson algorithm to find maximum flow in a network.

Q- What Data Structure are used to implement BFS and DFS, why?

→ The Data Structure used in BFS is a Queue and a graph. The algorithm makes sure that every node is visited not more than once.
It is used Queue because to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

The data Structure used by DFS is Stack date Structure.
It uses stack data Structure because for keep tracking on the current node it require last in first out approach which can be implemented by stack, after it reaches the depth of a node then all the nodes will be popped out of stack. Next it searches for adjacent node which are not visited yet.

3- What do you mean by Sparse and dense graph? Which representation of graph is better for sparse and dense graph?

→ Sparse graph is a graph in which the number of edges is close to the minimal number of edges. Sparse graph can be disconnected graph. As the name indicates sparse graph are sparsely connected.

Dense graph is a graph in which the number of edges is close to the maximal number of edges, or other ~~o~~ words if every pair of vertices is connected by on edge.

- If graph is sparse, we should store it as a list of edges.

- If the graph is dense, we should store it as an ~~af~~ adjacency matrix.

4- How can you detect a cycle in a graph using BFS and DFS?

→ Steps to detect ~~g~~ a cycle in a graph using BFS:

1- Compute in-degree for each of the 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- Remove a vertex from the queue. and increament count of visited nodes by 1.

4- Repeat step 3 untill queue is empty.

5- If count of visited nodes is not equal to the number of nodes in the graph has cycle, otherwise not.

Steps to detect a cycle in a graph using DFS:

1- Take the number of edges and vertices as user input.

2- Initialize the current index visited, and recursion stack using recursion.

3- Mark the index in the recursion stack and the current node as visited.

4- Make ~~the index in the recursion stack and the current~~ recursive calls for all the unvisited adjacent vertices for the current node, and if any of these recursive function return true, return true as final answer.

5- If any adjacent vertices are already visited in the recursion stack, mark the answer as true.

6- Return false if none of the above steps return false.

5- What do you mean by disjoint set data structure? Explain 3 operation along with examples, which can be performed on disjoint sets.

→ It is also known as union-find data structure and merge-find set. It is a data structure that contains a collection of disjoint or non-disjoint sets. It also allows to find out whether the two elements are in the same set or not efficiently.

Three operations performed on disjoint sets:

1- Making new sets:

```
function MakeSet(x) is
    if x is not already in the forest then
        x.~~print~~parent := x
        x.size := 1
        x.rank := 0
    end if
end function.
```

2- **Finding Set representatives:**

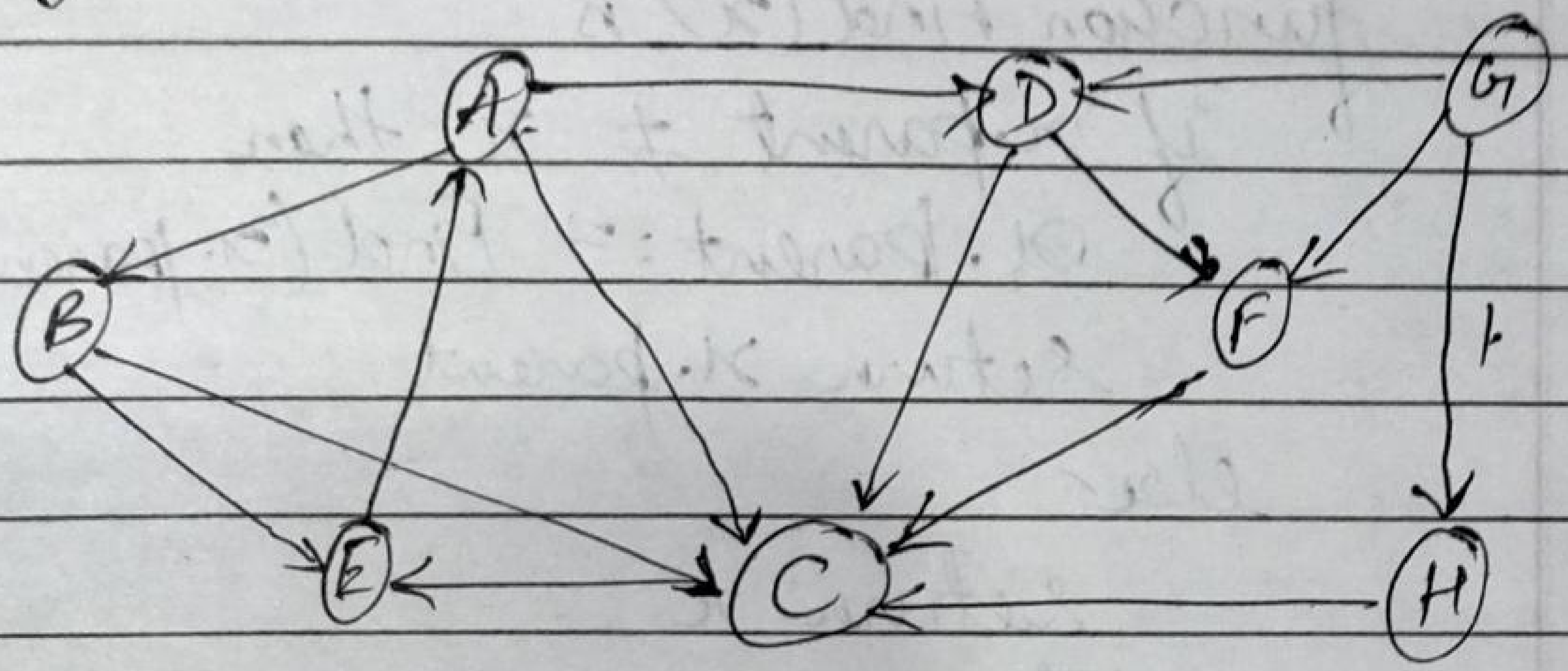
```
function Find(x) is
    if x.parent ≠ x then
        x.parent := Find(x.parent)
        return x.parent
    else
        return x
    end if
end function
```

3- **Merge two Sets:**

```
function union(x,y) is
  x := Find(x)
  y = Find(y)
  if x = y then
     return
  end if
    if x.size < y.size then
      (x,y) = (y,x)
    end if
    y.parent = x
    x.size = x.size + y.size.
  end function.
```

6- Run BFS and DFS on graph shown on sight side.

→



BFS:

Node	B	E	C	A	D	F
Parent		B	B	E	A	D

Unviorted nodes : G and H

Path : B → E → A → D → F

DFS:

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

Path : B → C → E → A → D → F

7- Find the no. of connected components and vertices in each compoment using disjoint set data structure:

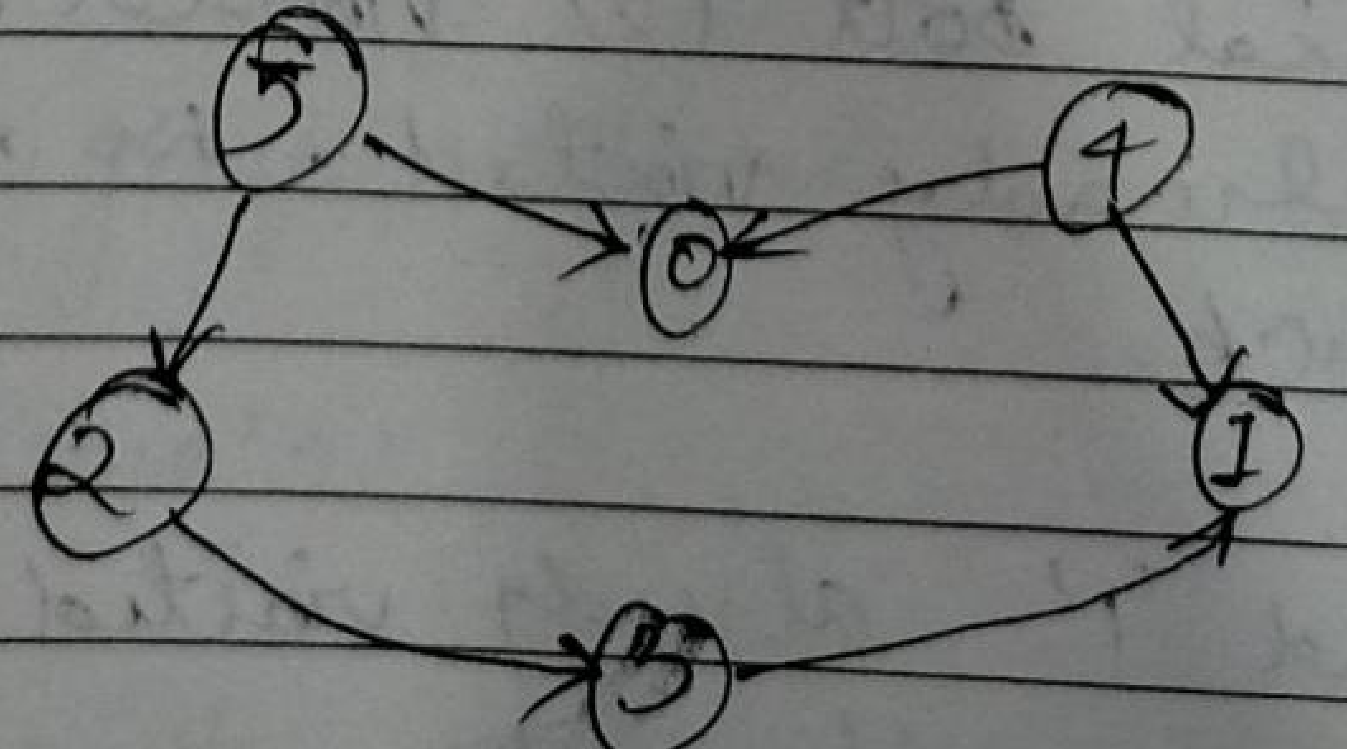
→ V = {a} {b} {c} {d} {e} {f} {g} {h} {i} {j}

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

no. of connected components = 3.

8- Apply Topology Sorting and DFS on graph having vertices 0 to 5.



Adjacent list : 0
1
2 → 3
3 → 1
4 → 0,1
5 → 2,0

visited :

false	false	false	false	false	false
0	1	2	3	4	5

stack (empty).

1- Topological sort (0) visited [0]: true.
list is empty no more recursion call.
stack [0]

2- Topological sort (1), visited [1) = true.
list is empty no more recursion call.
stack [0 | 1]

3- Topolgical sort (2), visited (2) = true.
Topological sort (3) visited (3) = true.
'1' is already visited. no more recursion call stack.

4- '0' and '1' already visited no more recursion call stack.
[0 | 1 | 3 | 2 | 4]

Date ____________

Page ____________

5- Topological sort (5) visited(5).= true.
[0 | 1 | 3 | 2 | 4 | 5

6- Print all elements from top to Bottom.
5, 4, 2, 3, 1, 0.

9- Heap data structure can be used to implement priority queue? Name few graph algorithms where you need to use priority queue and why?

→ Yes, we can use heaps to impleament the priority Queue. It will take O(log N). time to insert and ~~doubt~~ delet each element in the priority queue. Heaps are great for impleamenting a priority queue because of the largest and smallest element at the root of the ~~tree~~ for a max-heap and a min-heap respectively.

Few graph algorithm:

1- Dijkstra's.

2- Prims, etc.

10 - Difference between Max and Min heap?

→	Max heap	Min heap
1-	In this, the key is present at the root node must be greater than or equal to among the keys present at all its children.	In this, ~~Min heap~~. the key is present at the root node & must be less than or equal to among the key present at all ~~of~~ its children.
2-	In this, ; the maximum key element present at the root.	the minimum key element present at the root.
3-	It uses the descending priority.	It uses ascending priority.