Tutorial - 5

Q1) What is the difference between DfS and BfS? Write the applications of both the algorithms.

→

	BfS (Breath first Search)	DfS (Depth first Search)
①	BfS uses ~~Queque~~ Queue data structure for finishing the shortest path.	DfS uses stack data structure.
②	BfS can be used to find single source shortest path in an unweighted graph, because in BfS, we reach a vertex with minimum no. of edges from source.	In DfS, we might traverse through more edges to reach a destination vertex from a source.
③	Siblings are visited before the children.	Children are visited before the siblings.

Applications

• Shortest path & minimum spanning tree for unweighted graph.	Path finding
• Peer to Peer networks	Topological Sorting
• Cycle detection in undirected graph.	To test if a graph is bipartite

Q2) Which data structures are used to implement BfS and DfS? why?

→ • BfS does the search for nodes level-by-level, i.e., it searches the nodes w.r.t their distance from root. Here, siblings are visited before children. So, we use Queue as it is FIFO (first In first Out) data structure we visit the ~~Root~~ node which is discovered first the root.

- for DFS, we retrieve nodes from root to the farthest node as much as possible, same use as LIFO (last in first out). So, we use stack data structure; children are visited before the siblings.

Q3) What do you mean by sparse and dense graphs. Which representation of graph is better for sparse and dense graphs?

→ • A graph with relatively few edges in sparse, Sparse graph is a graph $G(V,E)$ in which $|E| = O(|V|)$ where
E = Edge & V = Vertex

- A graph with many edges is Dense. Dense graph is a graph $G(V,E)$ in which $|E| = O(|V^2|)$ where
E = Edge & V = Vertex.

- for Sparse graphs, Adjacency list can be used for representation.
- for Dense graphs, Adjacency Matrix can be used for representation.

Q4) How can you detect a cycle in a graph using BFS and DFS?

→ # Detecting a cycle in Directed graph using BFS.

① Compute in-degree (no. of incoming edges) for each of the vertex present in the graph & initialize the count of visited nodes as 0.

② Pick all the vertices with in-degree as zero & add them into a queue (Enqueue Operation).

③ Remove a vertex from the queue (Dequeue Operation) and then:
(a) Increment count of visited nodes by 1.
(b) Decrease in-degree by 1 for all its neighbouring nodes.
(c) If in-degree of a neighbouring node is reduces to zero,

then add it to the queue.

(4) Repeat Step 3 until the queue is empty.

(5) If count of visited nodes is not equal to the no. of nodes in the graph, the graph has cycle, otherwise not.

#Detecting a cycle in Directed graph using DFS.

(1) Create a graph using the given no. of edges & vertices.

(2) Create a recursive function that initializes the current index or vertex, visited & recursion stack.

(3) Mark the current node as visited and also mark the index in recursion stack.

(4) Find all the vertices which are not visited & are adjacent to the current node. Recursively call the function for those vertices, if the recursive function returns true, return true.

(5) If the adjacent vertices are already marked in the recursion stack, then return true.

(6) Create a wrapper class, that calls the recursive function for all the vertices and if any function returns true, return true, else if for all vertices the function returns false, return false.

Q5) What do you mean by disjoint set data structure? Explain 2 operations along with examples. Which can be performed on disjoint sets.

→ Disjoint set is basically a group of sets where no item can be in more than one set. It supports Union and find operations on Subsets.

Assume that you have a set of 'N' elements that are divided into further subsets and you have to track the connectivity of each element in a specific

Subset or connectivity of subsets with each other. You can use the Union-find algorithm (Disjoint set union) to achieve this,

Operations on Disjoint Set.

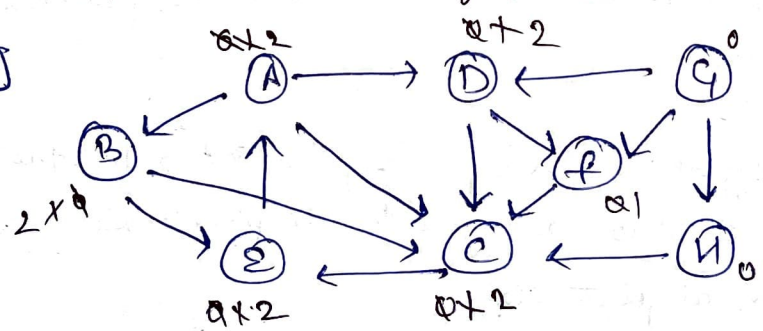
Let $S_1 = \{1, 2, 3\}$ and $S_2 = \{4, 5, 6\}$

- find(): It is used to find in which subset a particular element is present & returns the representation of that particular set.
  eg:- find(1) = S_1, find(5) = S_2

- union(): It merges two different subsets into a single subset & representative of one set becomes representative of the other.
  eg:- $S_1 \cup S_2 \Rightarrow S_3 = \{1, 2, 3, 4, 5, 6\}$

Q6) Run BfS and DfS on graph given below:

BfS



Source: B, Destination f

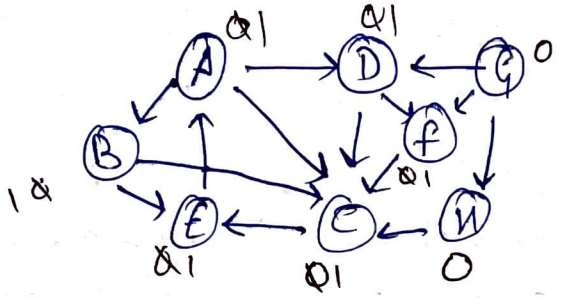
Queue

NODE	B	E	C	A	D	f
PARENT	-	B	B	E	A	D

Front → Rear →

∴ Path, P: B → E → A → D → f

DfS

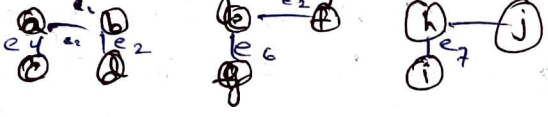


Node Processed	Stack ←
-	B
B	E C
E	A C
A	D C C
D	C f C C
f	C C

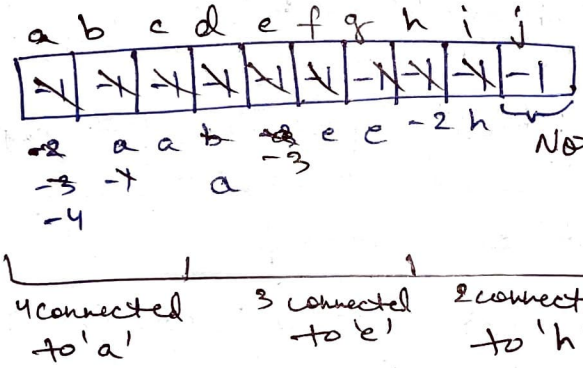
Source: B, Destination: f

∴ Path, P: B → E → A → D → f

Q7) find out the no. of connected components vertices in each component using disjoint set data structure in the given graph

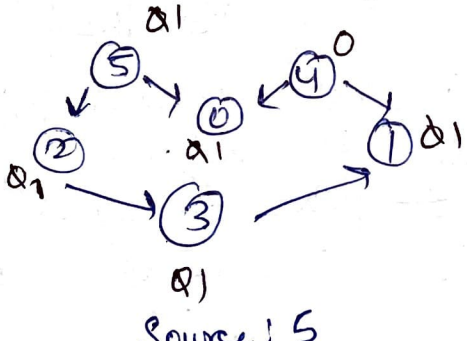


→



Q8) Apply topological sorting & DfS on the given graph:

DfS



Node Processed	Stack ←
-	5
5	2 0
2	3 0
3	1 0
1	0
0	-

Source: 5

∴ DfS traversal: 5 → 2 → 3 → 1 → 0

Topological Sort

5, 2, 4, 0, 3, 1

Q9) Heap data structure can be used to implement priority queue. Name few graph algorithms where you need to use priority queue and why?

→ Heaps are great for implementing a priority queue because of the largest and smallest element being at the root of tree for a max heap and a min heap respectively.

We use a max heap for a max-priority queue & a min heap for a min-priority queue.

Application of Priority queue

① Dijkstra's shortest path algorithm: When the graph is stored in the form of adjancy list or matrix, priority queue can be used to extract minimum weighted path efficiently when implementing the algorithm.

② Prim's algorithm: Priority queue is used to implement Prim's algorithm to store keys of node & extract minimum key node at every step.

③ Data Compression: Priority queue is used in Huffman codes which is used to compress data.

Q10) What is the difference between max heap & min heap?

→

Min Heap	Max Heap
① In a min heap, key present at the root node must be less than or equal to the keys present at all of its children.	In a Max heap, key present at the root node must be greater than or equal to the keys present at all of its children.
② The minimum key element is present at the root.	The maximum key element is process at the root.
③ It was the ascending priority	It use the descending priority