Tutorial - 5

1) Difference between BFS and DFS

	BFS (Breadth First search)	DFS (Depth First search)
1.	Uses Queue data structure for finding the shortest path.	Uses stack data structure.
2.	It is more suitable for searching vertices which are closer to the given source.	It is more suitable when there are solutions away from the source.
3.	Considers all neighbours first and therefore not suitable for decision making trees.	In this we make a decision, then explore all paths this decision.
4.	Siblings are visited before the children.	Children are visited before the siblings.
5.	Can be used to find single source shortest path in an unweighted graph.	We traverse through more edges to reach a destination vertex from sour

Applications of BFS :-

1) Path finding algorithm is based on BFS or DFS.
2) Using GPS navigation system BFS is used to find neighbouring places.
3) In peer-to-peer network, BFS is used to find all neighbouring nodes.

Date.
Page No.

Applications of DFS.

1) Using DFS we can find path between two given vertices u & v.

2) If we perform DFS on unweighted graph, then it will create minimum spanning tree for all shortest-paths.

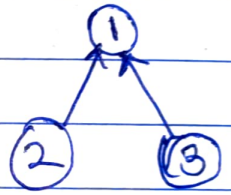
2. Queue is used to implement BFS as BFS searches the nodes with respect to their distance from the root. It requires to visit the child nodes in order their parents were discovered.
DFS is implemented using stack data structure. as DFS uses the idea of backtracking

3. Dense graph is a graph in which the number of edges is close to the maximal number of edges.
Sparse graph is a graph in which no. of edges is close to the minimal no. of edges.
If the graph is sparse, we should store it as a list of edges; and if a graph is dense we should store it as an adjacency matrix.

4. Detecting a cycle using BFS and DFS
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.
We use a parent array to keep track of the parent vertex so that we do not consider the visited parent as cycle.

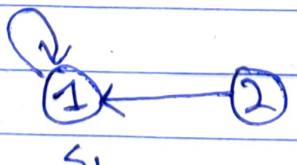
5. Disjoint set data structure contains a collection of disjoint or non overlapping sets. Disjoint set can be defined as the subsets where there is no common element between the two sets.
Operations which can be performed on disjoint sets are

1) Find : It determines in which subset a particular element is in and returns the representative of that particular. It follows parent nodes until it reaches the root.
eg:- $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{3\}$, $S_4 = \{4\}$, $S_5 = \{5\}$
Find (1) = 1
$S_1 = \{1, 2, 3\}$ $S_2 = \{4, 5\}$
Find (2) = 1

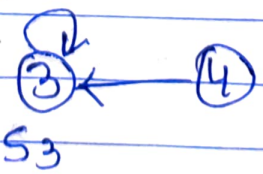


2) Union :- It merges two different subsets into a single subset and the representative of one set becomes representative of other.
Combines two trees into one by attaching one tree's root into the root of other.
$S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{3\}$ $S_4 = \{3\}$

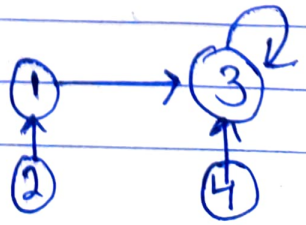
Union (1,2) = S_1



Union (3,4) = S_3

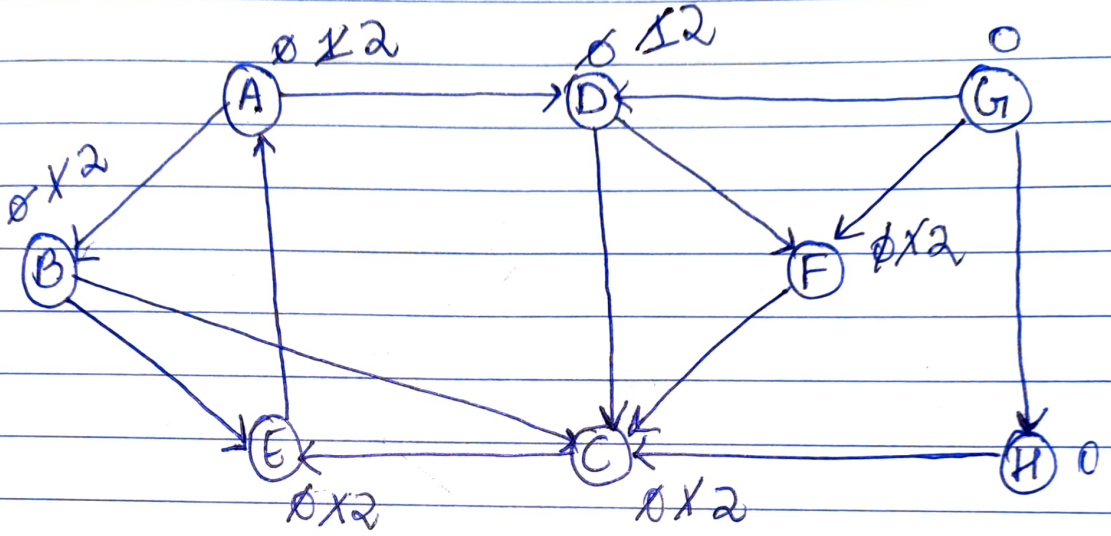


Union (S_1, S_3) =



3) Makeset :- It adds a new element. This element is placed into a new set containing only the new element, which is added to ~~the~~ data structures.

6.



Using BFS;

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

A → B → D → E → C → F

A → D → F