

Date: 21/01/2022

Sub.: AI (LP2)

Roll No.: 31149

Batch: M1

ASSIGNMENT 1.\* Problem Statement:

Implement Depth First Search and Breadth First Search Algorithm. Use an undirected graph and develop a recursive algorithm for searching all vertices of graph or tree data structure.

\* Objectives:

To learn and implement depth first search and breadth first search algorithms for undirected graph.

\* Outcomes:

To implement and learn applications of DFS and BFS algorithms.

\* Theory:

Unlike linear data structures which have only one way to traverse them, trees or graphs can be traversed in different ways. Generally there are two widely used traversals for trees/graphs, which are:

① Breadth First Search (BFS)

② Depth First Search (DFS)

i) BFS:

This algorithm is used to visit all the nodes of the graph. In this algorithm, one node is selected and then all of the adjacent nodes are visited one by one. After completing all the adjacent vertices, it moves further to check vertices and



their adjacent vertices again.

## ii) Depth First Search:

This algorithm works by using given starting vertex. When an adjacent vertex is found, we move to its adjacent vertex first and try to traverse in the same manner. We use stack data structure implicitly or explicitly when using a non-recursive approach.

### \* Algorithm:

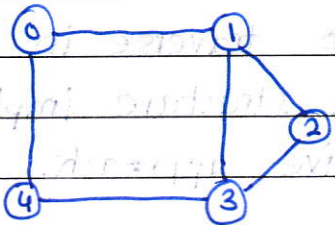
#### 1) BFS: (@ Queue Data Structure)

- ① Set status of all nodes as not visited.
- ② Take input as starting vertex of traversal.
- ③ Set status of this vertex as visited.
- ④ Push this vertex in the queue.
- ⑤ While queue is not empty
  - a) Print front of queue and remove it.
  - b) Move to adjacent of current vertex.
  - c) If adjacent is not visited then enqueue it to queue.
  - d) Set status of current vertex as visited.
- ⑥ End

#### 2) DFS: (Stack Data Structure (Implicit Use))

- ① Take starting node as input.
- ② Set status of current node as visited.
- ③ Print current vertex.
- ④ For all elements in adjacency list corresponding to the current vertex,
  - a) If vertex is not visited, then call DFS function recursively with this vertex.
- ⑤ End

\* Test Cases:

Input Graph	Output	Result
	i) BFS: 0 1 4 2 3	Pass
	ii) DFS: 0 1 2 3 4	Pass

\* Conclusion:

Thus we have successfully created graph data structure and implemented Depth First and Breadth First traversal algorithms for the same.