**WEEK 10**

**AIM: Traverse a graph using BFS & DFS techniques**

In this exercise, I implemented BFS and DFS traversals on the given graph.

**Variables Used:**

Node of the graph: node

Data part of the node: vertex

Edge linking of node: next

Graph using adjacency list representation: graph

Total number of vertices: totalVertices

Adjacency list of graph: adjList

Storing the status of visiting nodes: visited

Queue for the bfs: q

Stack for the dfs: s

**Input Graph:**



**Breadth First Search (BFS):**

Breadth first search is a graph traversal algorithm that starts traversing the graph from the root node and explores all the neighboring nodes. Then, it selects the nearest node and explores all the unexplored nodes. The algorithm follows the same process for each of the nearest nodes until it finds the goal.

**Algorithm:**

Step 1: Take an Empty Queue.

Step 2: Select a starting node (visiting a node) and insert it into the Queue and set the corresponding visited value to True.

Step 3: Provided that the Queue is not empty, extract the node from the Queue.

Step 4: Print the extracted node.

Step 5: Insert unvisited successor nodes (visited = False) of the extracted node into the Queue and set their corresponding visited value to True.

**Time Complexity:** O( V + E ) // V: number of vertices

// E: number of edges

**Sample Input:**

5

0 1

0 2

1 2

2 0

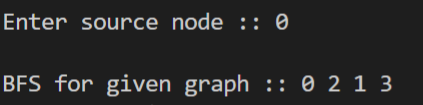
2 3

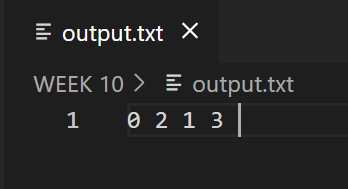
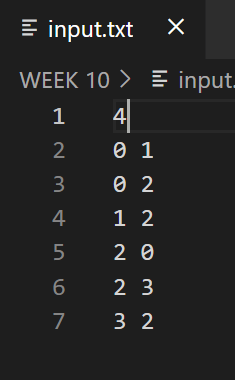
3 2

4 4

source vertex: 0

**Sample output:** 0 2 1 3





**Depth First Search (DFS):**

Depth first search (DFS) algorithm starts with the initial node of the graph G and then goes deeper and deeper until we find the goal node or the node which has no children. The algorithm then backtracks from the dead-end towards the most recent node that is yet to be completely unexplored.

**Algorithm:**

Step 1: SET visited = False for each node in G

Step 2: Push the starting node u on the stack and set its visited = 1

Step 3: Repeat Steps 4 and 5 until STACK is empty

Step 4: Pop the top node u.

Step 5: Push on the stack all the successor node of u that are in the ready state (whose visited = False) and set their  
Visited = True

**Time Complexity:** O( V + E ) // V: number of vertices

// E: number of edges

**Sample Input:**

5

0 1

0 2

1 2

2 0

2 3

3 2

4 4

**Sample output:** 0 2 1 3

