

**Batch: B-1 Roll No.: 16010422234 Name: Chandana Ramesh Galgali**

**Experiment No.: 6**

**Aim:** Implement a menu driven program to represent a graph and traverse it using BFS technique.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources Used:** C/ C++ editor and compiler.

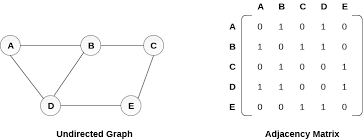
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory:**

**Graph**

Given an undirected graph G= (V,E) and a vertex V in V(G), then we are interested in visiting all vertices in G that are reachable from V i.e. all vertices connected to V. There are two techniques of doing it namely Depth First Search (DFS) and Breadth First Search(BFS).

**Graph Representation using Adjacency Matrix**

****

**Depth First Search**

The procedure of performing DFS on an undirected graph can be as follows :

The starting vertex v is visited. Next an unvisited vertex w adjacent to v is selected and a depth first search from w is initiated. When a vertex u is reached such that all its adjacent vertices have been visited, we back up to the last vertex visited which has an unvisited vertex w adjacent to it and initiate a depth first search from w. the search terminates when no unvisited vertex can be reached from any of the visited ones. Given an undirected graph G=(V,E) with n vertices and an array visited[n] initially set to false, this algorithm, dfs (v) visits all vertices reachable from v. Visited is a global array.

**Breadth First Search**

Starting at vertex v and making it as visited, BFS visits next all unvisited vertices adjacent to v. then unvisited vertices adjacent to there vertices are visited and so on. A breadth first search of G is carried out beginning at vertex v as bfs (v). All vertices visited are marked as visited [i]=true. The graph G and array visited are global and visited is initialized to false. Initialize, addqueue, emptyqueue, deletequeue are the functions to handle operations on queue.

**Algorithm :**

Implement the static linear queue ADT, Represent the graph using adjacency matrix and implement following pseudo code for BFS.

***Pseudo Code: bfs (v)***

*initialize queue q*

*visited [v] = true*

*addqueue(q,v)*

*while not emptyqueue*

*v=deletequeue(q)*

*add v into bfs sequence*

*for all vertices w adjacent to v do*

*if not visited [w] then*

*addqueue (q,w)*

*visited [w]=true*

**Results:**

#include <stdio.h>

#define MAX 20

void bfs(int adj[][MAX],int visited[],int start)

{

int queue[MAX],rear = -1,front = -1, i;

queue[++rear] = start;

visited[start] = 1;

while(rear != front)

{

start = queue[++front];

if(start == 4)

{

printf("5\t");

}

else

{

printf("%c \t",start + 65);

}

for(i = 0; i < MAX; i++)

{

if(adj[start][i] == 1 && visited[i] == 0)

{

queue[++rear] = i;

visited[i] = 1;

}

}

}

}

int main()

{

int visited[MAX] = {0};

int adj[MAX][MAX], i, j;

printf("\nEnter the adjacency matrix: ");

for(i = 0; i < MAX; i++)

{

for(j = 0; j < MAX; j++)

{

scanf("%d ", &adj[i][j]);

}

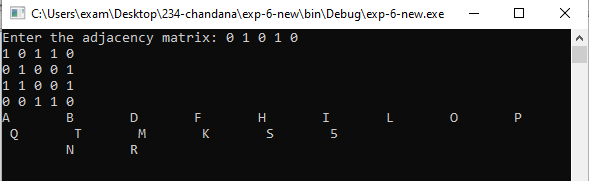
bfs(adj,visited,0);

printf("\n");

}

return 0;

}

****

A program depicting the BFS using adjacency matrix and capable of handling all possible boundary conditions and the same is reflected clearly in the output.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Outcomes: Apply linear and non-linear data structure in application development.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conclusion:** The experiment was successful in implementing a menu-driven program to represent a graph and traverse it using the BFS technique. The program exhibited the desired behavior and functionality, allowing the user to interact with the graph and perform BFS traversal.

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References:**

**Books/ Journals/ Websites:**

* Y. Langsam, M. Augenstin and A. Tannenbaum, “Data Structures using C”, Pearson Education Asia, 1st Edition, 2002.
* Vlab on BFS