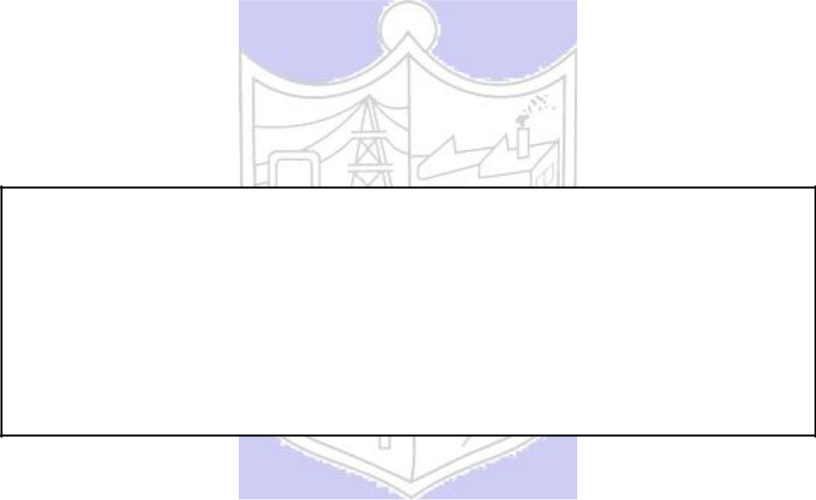
KJSCE/IT/SY BTECH/SEM III/DS/2022-23



**Experiment No. : 6**

**Title: Graph Traversal using appropriate data structure**

(A Constituent College of Somaiya Vidyavihar University)

KJSCE/IT/SY BTECH/SEM III/DS/2022-23

**Batch:A1**

**Roll No.: 16010421028**

**Experiment No.: 6**

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

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

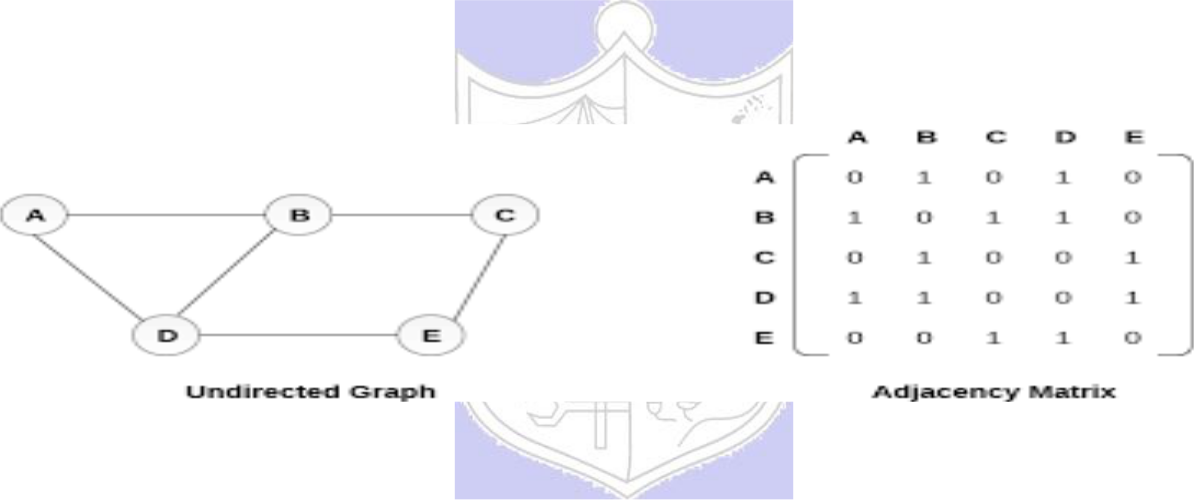
**Resources Used:** C/ C++ editor and compiler.

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

**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

(A Constituent College of Somaiya Vidyavihar University)

KJSCE/IT/SY BTECH/SEM III/DS/2022-23

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)***

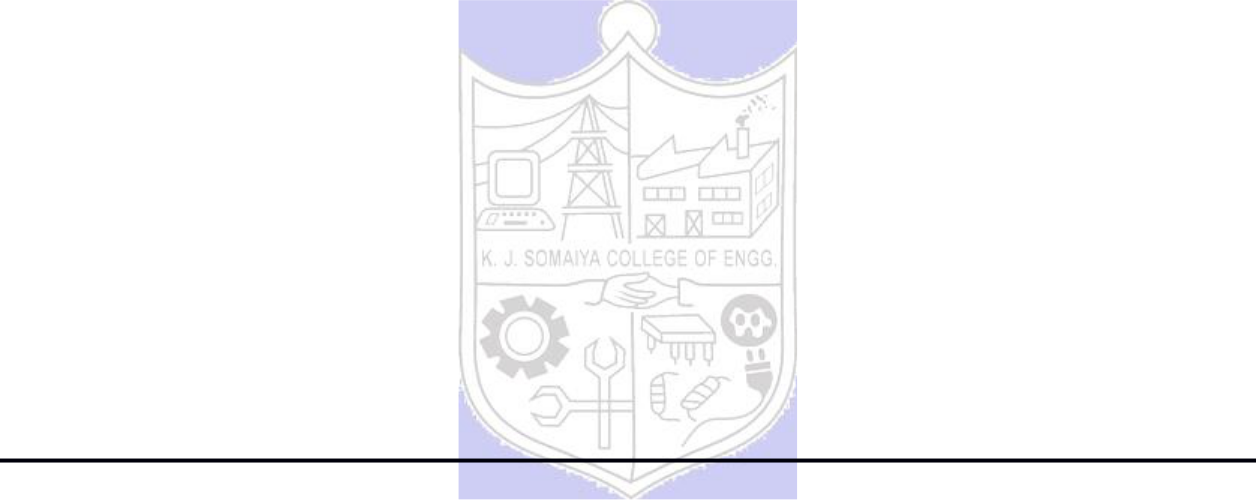
*initialize queue q*

*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:**

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.

#include<stdio.h>

int ary[20][20],queue[20],visit[20],n,front=-1,rear=-1;

void enqueue(int v){

    queue[rear] = v;

}

void dequeue(){

    front += 1;

}

int isEmpty(){

    if(queue[front]==0){

        return 1;}

    else{

        return 0;}

}

int isFull(){

    if(rear==100){

        return 1;}

    else{

        return 0;}

}

int peek(){

    return queue[front];

}

void bfs(int a)

{

    for (int i = 0; i < n; i++)

    {

        if (ary[a][i] != 0 && visit[i] == 0)

        {

            rear = rear + 1;

            enqueue(i);

            visit[i] = 1;

            printf("%d ", i);

        }

    }

    dequeue() ;

    if (!isEmpty()){

        bfs(peek());}

}

void main()

{

    int v;

    printf("\n NUMBER OF VERTICES ARE:");

    scanf("%d",&n);

    for (int i=0;i<n;i++)

    {

        visit[i]=0;

    }

    printf("\n GRAPH DATA IN MATRIX FORM:\n");

    for (int i=0;i<n;i++){

        for (int j=0;j<n;j++){

            scanf("%d",&ary[i][j]);}}

    printf("\n STARTING VERTEX IS:");

    scanf("%d",&v);

    front = rear=0;

    queue[rear]=v;

    printf("\n BFS TRAVERSAL IS: ");

    visit[v]=1;

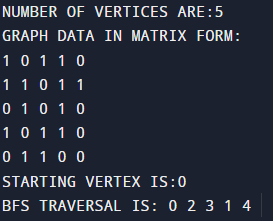
    printf("%d ",v);

    bfs(v);

    if(rear != n-1){

        printf("\nBFS NOT POSSIBLE \n");}

}

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

**Outcomes: CO2: Apply linear and non-linear data structure in application development**



**Conclusion: We learnt about Graph and its various traversal techniques and implemented the same.**

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

(A Constituent College of Somaiya Vidyavihar University)

KJSCE/IT/SY BTECH/SEM III/DS/2022-23

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

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

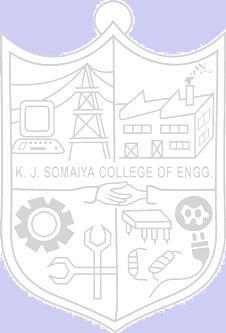
**References:**

**Books/ Journals/ Websites:**

* Y. Langsam, M. Augenstin and A. Tannenbaum, “Data Structures using C”, Pearson

Education Asia, 1st Edition, 2002.

* Vlab on BFS



(A Constituent College of Somaiya Vidyavihar University)