Ex. No. 9a **Breadth First Search**

Date:

**Aim**

To create adjacency matrix of the given graph and to perform breadth first search traversal.

**Algorithm**

1. Start

2. Obtain Adjacency matrix for the given graph

3. Define a Queue of size total number of vertices in the graph

4. Select any vertex as starting point for traversal. Visit that vertex and insert it into the Queue.

5. Visit all the adjacent vertices of the verex which is at front of the Queue which is not visited and insert them into the Queue.

6. When there is no new vertex to be visit from the vertex at front of the Queue then delete that vertex from the Queue.

7. Repeat step 5 and 6 until queue becomes empty.

8. When queue becomes Empty, then produce final spanning tree by removing unused edges from the graph.

9. Stop

**Program**

/\* Graph Traversal – BFS \*/

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

#define initial 1

#define waiting 2

#define visited 3

int n;

int adj[MAX][MAX];

int state[MAX];

void create\_graph();

void BF\_Traversal();

void BFS(int v);

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

void insert\_queue(int vertex);

int delete\_queue();

int isEmpty\_queue();

int main()

{

create\_graph();

BF\_Traversal();

return 0;

}

void BF\_Traversal()

{

int v;

for(v=0; v<n; v++)

state[v] = initial;

printf("Enter Start Vertex for BFS: ");

scanf("%d", &v);

BFS(v);

}

void BFS(int v)

{

int i;

insert\_queue(v);

state[v] = waiting;

printf("BFS Traversal : ");

while(!isEmpty\_queue())

{

v = delete\_queue( );

printf("%d ", v);

state[v] = visited;

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

{

if(adj[v][i] == 1 && state[i] == initial)

{

insert\_queue(i);

state[i] = waiting;

}

}

}

printf("\n");

}

void insert\_queue(int vertex)

{

if(rear == MAX-1)

printf("Queue Overflow\n");

else

{

if(front == -1)

front = 0;

rear = rear+1;

queue[rear] = vertex ;

}

}

int isEmpty\_queue()

{

if(front == -1 || front > rear)

return 1;

else

return 0;

}

int delete\_queue()

{

int delete\_item;

if(front == -1 || front > rear)

{

printf("Queue Underflow\n");

exit(1);

}

delete\_item = queue[front];

front = front+1;

return delete\_item;

}

void create\_graph()

{

int count,max\_edge,origin,destin;

printf("Enter number of vertices : ");

scanf("%d", &n);

max\_edge = n \* (n-1);

for(count=1; count<=max\_edge; count++)

{

printf("Enter edge %d( -1 -1 to quit ) : ",count);

scanf("%d %d", &origin, &destin);

if((origin == -1) && (destin == -1))

break;

if(origin>=n || destin>=n || origin<0 || destin<0)

{

printf("Invalid edge!\n");

count--;

}

else

adj[origin][destin] = 1;

}

}

**Output**

**Result**

Thus Breadth First Traversal is executed on the given graph.