2) Graph traversal technique BFS (using queue)

#include<stdio.h>

#include<stdlib.h>

#define MAX 100

#define initial 1

#define waiting 2

#define visited 3

int n; /\*Number of vertices in the graph\*/

int adj[MAX][MAX]; /\*Adjacency Matrix\*/

int state[MAX]; /\*can be initial, waiting or visited\*/

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;

}/\*End of main()\*/

void BF\_Traversal()

{

int v;

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

state[v] = initial;

printf("\nEnter starting vertex for Breadth First Search : ");

scanf("%d", &v);

BFS(v);

}/\*End of BF\_Traversal()\*/

void BFS(int v)

{

int i;

insert\_queue(v);

state[v] = waiting;

while(!isEmpty\_queue())

{

v = delete\_queue( );

printf("%d ",v);

state[v] = visited;

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

{

/\*Check for adjacent unvisited vertices \*/

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

{

insert\_queue(i);

state[i] = waiting;

}

}

}

printf("\n");

}/\*End of BFS()\*/

void insert\_queue(int vertex)

{

if(rear == MAX-1)

printf("\nQueue Overflow\n");

else

{

if(front == -1) /\*If queue is initially empty \*/

front = 0;

rear = rear+1;

queue[rear] = vertex ;

}

}/\*End of insert\_queue()\*/

int isEmpty\_queue()

{

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

return 1;

else

return 0;

}/\*End of isEmpty\_queue()\*/

int delete\_queue()

{

int del\_item;

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

{

printf("\nQueue Underflow\n");

exit(1);

}

del\_item = queue[front];

front = front+1;

return del\_item;

}/\*End of delete\_queue() \*/

void create\_graph()

{

int i,max\_edges,origin,destin;

printf("\nEnter number of vertices : ");

scanf("%d",&n);

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

for(i=1; i<=max\_edges; i++)

{

printf("\nEnter edge %d( -1 -1 to quit ) : ",i);

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

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

break;

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

{

printf("\nInvalid edge!\n");

i--;

}

else

{

adj[origin][destin] = 1;

}

}

}

Output:

