**Tutorial-2**

1. Implement BFS and DFS for given graph. (Graph is given in Lab session).

Diagram

Description automatically generated

**BFS :: CODE::**

#include <stdio.h>

#include<stdlib.h>

#define n 8

int a[8][8]={

{0,1,1,1,0,0,0,0},

{1,0,1,0,1,0,0,0},

{1,1,0,0,0,1,0,0},

{1,0,0,0,0,0,1,1},

{0,1,0,0,0,1,0,0},

{0,0,1,0,1,0,0,0},

{0,0,0,1,0,0,0,1},

{0,0,0,1,0,0,1,0}

};

int front = -1;

int rear = -1;

int visit[8];

int q[10];

void enqueue(int item) {

if (rear == 9) {

printf("QUEUE FULL");

} else {

rear = rear + 1;

q[rear] = item;

if (front == -1) {

front = front + 1;

}

}

}

int dequeue() {

int item;

if ((front == -1)) {

return -1;

} else {

item = q[front];

if (front == rear) {

front = -1;

} else {

front = front + 1;

}

return item;

}

}

void bfs(int s) {

int p, i;

enqueue(s);

visit[s] = 1;

loop:

p = dequeue();

while (p != -1) {

printf("%d->", p);

for (i = 0; i <= n - 1; i++) {

if ((a[p][i] == 1) && (visit[i] == 0)) {

enqueue(i);

visit[i] = 1;

}

}

goto loop;

}

}

void main() {

int i, j, s;

printf("Adj matrix: \n");

for (i = 0; i <= n - 1; i++) {

for (j = 0; j <= n - 1; j++) {

printf("%d", a[i][j]);

}

printf("\n");

}

printf("\nenter the sourcevector: ");

scanf("%d", & s);

printf("BFS TRAVERSALIS: ");

bfs(s);

}

**::Output::**

Graphical user interface, text

Description automatically generated

**DFS :: CODE ::**

#include<stdio.h>

#include<stdlib.h>

#define n 8

int a[8][8]={

{0,1,1,1,0,0,0,0},

{1,0,1,0,1,0,0,0},

{1,1,0,0,0,1,0,0},

{1,0,0,0,0,0,1,1},

{0,1,0,0,0,1,0,0},

{0,0,1,0,1,0,0,0},

{0,0,0,1,0,0,0,1},

{0,0,0,1,0,0,1,0}

};

int stack[20];

int visit[8];

int top = -1;

void dfs(int s);

void push(int iteam);

int pop();

int main() {

int i, j, s;

printf("\n\n THE ADJACENCY MATRIX IS \n\n");

for (i = 0; i <= n - 1; i++) {

for (j = 0; j <= n - 1; j++) {

printf("%d", a[i][j]);

}

printf("\n");

}

printf("\n\n ENTER THE SOURSE VERTEX: ");

scanf("%d", & s);

dfs(s);

return 0;

}

void dfs(int s) {

int i, k;

push(s);

visit[s] = 1;

k = pop();

if (k != -1) {

printf("%d", k);

visit[k] = 1;

}

while (k != -1) {

for (i = n - 1; i >= 0; i--) {

if (a[k][i] == 1 && visit[i] == 0) {

push(i);

}

}

k = pop();

if (k != -1) {

if (visit[k] == 0) {

printf("%d", k);

visit[k] = 1;

}

}

}

}

void push(int item) {

if (top == 9) {

printf("stack overflow");

} else {

top = top + 1;

stack[top] = item;

}

}

int pop() {

int k;

if (top == -1) {

return -1;

} else {

k = stack[top];

top = top - 1;

return k;

}

}

**:: Output ::**

**Text

Description automatically generated with medium confidence**