**Experiment no:06**

Program code:

Adjacency matrix:

#include<iostream>

#include<stdlib.h>

using namespace std;

int cost[10][10],i,j,k,n,qu[10],front,rear,v,visit[10],visited[10];

int stk[10],top,visit1[10],visited1[10];

int main(){

int m;

cout<<"enter no of vertices:";

cin>>n;

cout<<"enter no of edges:";

cin>>m;

cout<<"\nEDGES\n";

for(k=1;k<=m;k++){

cin>>i>>j;

cost[i][j]=1;

cost[j][i]=1;

}

cout<<"the adjacency matrix of the graph is"<<endl;

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

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

cout<<" "<<cost[i][j];

}

cout<<endl;

}

cout<<"enter initial vertex of BFS:";

cin>>v;

cout<<"the BFS of the graph is:\n";

cout<<v<<" ";

visited[v]=1;

k=1;

while(k<n){

for(j=1;j<=n;j++)

if(cost[v][j]!=0&& visited[j]!=1&& visit[j]!=1){

visit[j]=1;

qu[rear++]=j;

}

v=qu[front++];

cout<<v<<" ";

k++;

visit[v]=0;visited[v]=1;

}

cout<<"\n enter initial vertex of DFS:";

cin>>v;

cout<<"the DFS of the graph is\n";

cout<<v<<" ";

visited[v]=1;

k=1;

while (k<n){

for(j=n;j>=1;j--)

if(cost[v][j]!=0&&visited1[j]!=1&&visit1[j]!=1){

visit1[j]=1;

stk[top]=j;

top++;

}

v=stk[--top];

cout<<v<<" ";

k++;

visit1[v]=0;visited1[v]=1;

}

cout<<endl;

return 0;

}

**Output:**

enter no of vertices:5

enter no of edges:4

EDGES

1 2

1 3

2 1

2 4

the adjacency matrix of the graph is

0 0 0 0 0

0 0 1 1 0

0 1 0 0 1

0 1 0 0 0

0 0 1 0 0

enter initial vertex of BFS:1

the BFS of the graph is:

1 2 3 4 0

enter initial vertex of DFS:2

the DFS of the graph is

2 1 2 3 4

=== Code Execution Successful ===

**Adjacency list:**

Program code:

#include<iostream>

using namespace std;

#define MAX 10

#define TRUE 1

#define FALSE 0

class lgra{

private:

struct node1{

int vertex;

struct node1\*next;

};

node1\*head[MAX];

int visited[MAX];

public:

lgra();

void create();

void dfs(int);

};

lgra::lgra(){

int v1;

for(v1=0;v1<MAX;v1++)

head[v1]=NULL;

}

void lgra::create(){

int v1,v2;

char ans;

node1\*N,\*first;

cout<<"enter the vertices no beginning with 0";

do{

cout<<"\n enter the edge of a graph\n";

cin>>v1>>v2;

if(v1>=MAX||v2>=MAX)

cout<<"invalid vertex value\n";

else{

N=new node1;

if(N==NULL)

cout<<"insufficient memory\n";

N->vertex=v2;

N->next=NULL;

first=head[v1];

if(first==NULL)

head[v1]=N;

else{

while(first->next!=NULL)

first=first->next;

first->next=N;

}

N=new node1;

if(N==NULL)

cout<<"insufficient memory\n";

N->vertex=v1;

N->next=NULL;

first=head[v2];

if(first==NULL)

head[v2]=N;

else{

while(first->next!=NULL)

first=first->next;

first->next=N;

}

}

cout<<"\n want to add more edges?(y/n)";

cin>>ans;

}while(ans=='y');

}

void lgra::dfs(int v1){

node1\*first;

cout<<endl<<v1;

visited[v1]=TRUE;

first=head[v1];

while(first!=NULL)

if (visited[first->vertex]==FALSE)

dfs(first->vertex);

else

first=first->next;

}

int main(){

int v1;

lgra g;

g.create();

cout<<endl<<"enter the vertex where you want to traverse:";

cin>>v1;

if(v1>=MAX)

cout<<"invalid vertex\n";

else{

cout<<"the dfs of the graph:";

g.dfs(v1);

}}

**Output:**

enter the vertices no beginning with 0

enter the edge of a graph

1 2

want to add more edges?(y/n)y

enter the edge of a graph

1 3

want to add more edges?(y/n)y

enter the edge of a graph

3 4

want to add more edges?(y/n)y

enter the edge of a graph

3 5

want to add more edges?(y/n)n

enter the vertex where you want to traverse:3

the dfs of the graph:

3

1

5