Adjacency Matrix:

//using adj matrix -BFS(Que)

#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];

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;

}

//display function

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";

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 <<"Enter initial vertex";

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;

}

}

(Adjecency List)

#include<iostream>

using namespace std;

#define MAX 10

#define TRUE 1

#define FALSE 0

// declaring an adjacency list for storing the graph

class lgra

{

private:

struct node1

{

int vertex;

struct node1 \*next;

};

node1 \*head[MAX];

int visited[MAX];

public:

//static int nodecount;

lgra();

void create();

void dfs(int);

};

//constructor

lgra::lgra()

{

int v1;

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

visited[v1]=FALSE;

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<<"\nEnter the Edge of a graph\n";

cin>>v1>>v2;

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

cout<<"Invalid Vertex Value\n";

else

{

//creating link from v1 to v2

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;

}

//creating link from v2 to v1

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');

}

//dfs function

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 from where you want to traverse:";

cin>>v1;

if(v1>=MAX)

cout<<"Invalid Vertex\n";

else

{

cout<<"The Dfs of the graph:";

g.dfs(v1);

}

}