BFS: Code🡪

#include<bits/stdc++.h>

using namespace std; class Graph{

int no\_of\_nodes=0;

int no\_of\_edges=0; bool visited\_nodes[50]; int adj\_matrix[50][50]; public:

Graph(int v){

no\_of\_nodes=v;

for(int i=0;i<no\_of\_nodes;i++) {

visited\_nodes[i]=false;

for(int j=0;j<no\_of\_nodes;j++){

adj\_matrix[i][j]=0;

}

}

}

void insert\_edges(){

int x,y;

cout<<"Enter number of edges = "; cin>>no\_of\_edges;

for(int i=0;i<no\_of\_edges;i++){

cout<<"Enter edge : ";

cin>>x>>y;

adj\_matrix[x][y]=adj\_matrix[y][x]=1;

}

}

void BFS(int starting\_node){

queue <int>q;

visited\_nodes[starting\_node]=true;

q.push(starting\_node);

// Find all adjecent of this node;

while(!q.empty()){

int temp = q.front(); cout<<q.front()<<" ";

q.pop();

for(int i=0;i<no\_of\_nodes;i++){

if(adj\_matrix[temp][i]==1 && !visited\_nodes[i]){

visited\_nodes[i]=true;

q.push(i);

}

}

}

}

};

int main() {

int ver,s;

cout<<"Enter number of nodes = "; cin>>ver;

Graph g(ver);

g.insert\_edges();

cout<<"Enter starting node : "; cin>>s;

g.BFS(s);

}

Output:

Enter number of nodes = 9

Enter number of edges = 8

Enter edge : 0 1

Enter edge : 0 2

Enter edge : 0 3

Enter edge : 1 4

Enter edge : 2 5

Enter edge : 2 6

Enter edge : 6 7

Enter edge : 7 8

Enter starting node : 0

0 1 2 3 4 5 6 7 8

DFS Code:

#include<bits/stdc++.h>

using namespace std;

class Graph{

int no\_of\_nodes=0;

int no\_of\_edges=0;

bool visited\_nodes[50];

int adj\_matrix[50][50];

public:

Graph(int v){

no\_of\_nodes=v;

for(int i=0;i<no\_of\_nodes;i++) {

visited\_nodes[i]=false;

for(int j=0;j<no\_of\_nodes;j++){

adj\_matrix[i][j]=0;

}

}

}

void insert\_edges(){

int x,y;

cout<<"Enter number of edges = ";

cin>>no\_of\_edges;

for(int i=0;i<no\_of\_edges;i++){

cout<<"Enter edge : ";

cin>>x>>y;

adj\_matrix[x][y]=adj\_matrix[y][x]=1;

}

}

bool are\_all\_visited(){

for(int i=0;i<no\_of\_nodes;i++){

if(visited\_nodes[i]==0)

return false;

}

return true;

}

void DFS(int starting\_node){

stack<int> s; s.push(starting\_node);

while(!s.empty() && !are\_all\_visited()){

int temp=s.top();

cout<<temp<<" "; visited\_nodes[temp]=true;

s.pop();

for(int i=0;i<no\_of\_nodes;i++){

if(adj\_matrix[temp][i]==1 && !visited\_nodes[i]){

s.push(i);

}

}

}

}

};

int main() {

int ver,s;

cout<<"Enter number of nodes = "; cin>>ver;

Graph g(ver);

g.insert\_edges();

cout<<"Enter starting node : "; cin>>s;

g.DFS(s);

}

Output:

Enter number of nodes = 9

Enter number of edges = 8

Enter edge : 0 1

Enter edge : 0 2

Enter edge : 0 3

Enter edge : 1 4

Enter edge : 2 5

Enter edge : 2 6

Enter edge : 6 7

Enter edge : 7 8

Enter starting node : 0

0 3 2 6 7 8 5 1 4