/\*Write a function to get the number of vertices in an undirected graph and its edges. You may assume that no edge is input twice. i. Use adjacency list representation of the graph and find runtime of the function ii. Use adjacency matrix representation of the graph and find runtime of the function

/\* C++ Program to Implement Graph using Adjacency List, Adjacency matrix, Traverse using BFS and DFS \*/

#include <iostream>

#include <cstdlib>

#include<stdlib.h>

#include<list>

using namespace std;

int visited[20],visit[20],stk[20],top=0,qu[12],front,rear;

/\* Adjacency List Node \*/

struct AdjListNode

{

int dest;

struct AdjListNode\* next;

};

/\* Adjacency List \*/

struct AdjList

{

struct AdjListNode \*head;

};

/\* Class Graph \*/

class Graph

{

private:

int V,k;

struct AdjList\* array;

int adjM[10][10];

list<int>\* adj;

public:

Graph(int V)

{

this->V = V;

array = new AdjList [V];

for (int i=1;i<=V; ++i)

{

array[i].head = NULL;

for(int j=1; j<=V; ++j)

{

adjM[i][j]=0;

}

}

adj = new list<int>[V];

for (int i=1; i<=V; ++i)

visited[i]=0;

}

/\* Creating New Adjacency List Node \*/

AdjListNode\* newAdjListNode(int dest)

{

AdjListNode\* newNode = new AdjListNode;

newNode->dest = dest;

newNode->next = NULL;

return newNode;

}

/\*Add Edge to Graph \*/

void addEdge()

{

int src,des;

cout<<"\n Enter Edge ";

cout<<"\n From -> ";

cin>>src;

cout<<"\n To -> ";

cin>>des;

adjM[src][des]=1;

adjM[des][src]=1;

adj[src].push\_back(des);

adj[des].push\_back(src);

AdjListNode\* newNode = newAdjListNode(des);

newNode->next = array[src].head;

array[src].head = newNode;

newNode = newAdjListNode(src);

newNode->next = array[des].head;

array[des].head = newNode;

}

/\* Print the graph\*/

void printGraph()

{

int v;

cout<<"\n Adjucency Matrix is : \n ";

for (int i=1; i<=V; ++i)

{

for(int j=1; j<=V; ++j)

{

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

}

cout<<"\n";

}

for (v=1; v<=V;++v)

{

AdjListNode\* p = array[v].head;

cout<<"\n Adjacency list of vertex "<<v<<"\n head ";

while(p)

{

cout<<"-> "<<p->dest;

p = p->next;

}

cout<<endl;

}

}

/\* Traverse using DFS \*/

void DFS(int u)

{

for (int i=0;i<=V; ++i)

visited[i]=0;

int j;

cout <<"ORDER OF VISITED VERTICES";

cout << u <<" ";

visited[u]=1;

k=1;

while(k<V)

{

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

{

if(adjM[u][j]!=0 && visited[j]!=1 && visit[j]!=1)

{

visit[j]=1;

stk[top]=j;

top++;

}

}

u=stk[--top];

cout<<u << " ";

k++;

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

}

}

/\* Traverse using BFS \*/

void bfs(int v)

{

for (int i=0;i<=V; ++i)

visited[i]=0;

cout <<"Visitied vertices\n";

cout << v;

visited[v]=1;

k=1;

while(k<V)

{

for(int j=1;j<=V;j++)

if(adjM[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;

}

}

};

/\* \* Main \*/

int main()

{

int ver,ed,ch,i,j,s;

cout<<"Enter Count of No of Vertices";

cin>>ver;

cout<<"Enter Count of No of Egdes";

cin>>ed;

Graph gh(ed);

do

{

cout<<"\n Select option :";

cout<<"\n\t 1. Create graph \n \t2.Display graph \n\t3.DFS \n \t 4.BFS \n \t 5.Exit";

cin>>ch;

switch(ch)

{

case 1:

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

gh.addEdge();

break;

case 2:

// print the adjacency list representation of the above graph

gh.printGraph();

break;

case 3:

cout<<"\n select Initial Vertex";

cin>>s;

// print the adjacency list representation of the above graph

gh.DFS(s);

break;

case 4:

cout<<"\n select Initial Vertex";

cin>>s;

// print the adjacency list representation of the above graph

gh.bfs(s);

break;

default : cout<<"\n Enter correct choice";

}

}while(ch!=5);

return 0;

}

/\* OUTPUT

Enter Count of No of Vertices 4

Enter Count of No of Egdes 4

Select option :

1. Create graph

2.Display graph

3.DFS

4.BFS

5.Exit

1

Enter Edge

From -> 0

To -> 1

Enter Edge

From -> 0

To -> 2

Enter Edge

From -> 1

To -> 3

Enter Edge

From -> 2

To -> 3

Select option :

1. Create graph

2.Display graph

3.DFS

4.BFS

5.Exit

2

Adjucency Matrix is :

0 0 1 0

0 0 1 0

1 1 0 0

0 0 0 0

Adjacency list of vertex 1

head -> 3-> 0

Adjacency list of vertex 2

head -> 3-> 0

Adjacency list of vertex 3

head -> 2-> 1

Adjacency list of vertex 4

head

Select option :

1. Create graph

2.Display graph

3.DFS

4.BFS

5.Exit

3

select Initial Vertex 0

ORDER OF VISITED VERTICES0 1 3 2

Select option :

1. Create graph

2.Display graph

3.DFS

4.BFS

5.Exit

4

select Initial Vertex 1

Visitied vertices

13 2 0

Select option :

1. Create graph

2.Display graph

3.DFS

4.BFS

5.Exit

5

Enter correct choice

Process returned 0 (0x0) execution time : 48.867 s

Press any key to continue.

\*/