Name: Sumit Kalshetty

Sub: DSA(Lab). Roll no: 2125

Div: A

PRACTICAL NO:8

Problem Statement:

Represent a graph of city using adjacency matrix /adjacency list. Nodes should represent the

various landmarks and links should represent the distance between them. Find the shortest path using

Dijkstra's algorithm from single source to all destination.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PROGRAM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//#define INFINITY 9999

#include<iostream>

#include <limits.h>

#include<math.h>

#define max 10

using namespace std;

class graph

{

int g[max][max];

int n;

public:

graph()

{

}

void getgraph()

{

cout<<"\nEnter No. Of Vertices In Graph:";

cin>>n;

cout<<"Enter the weight of edge:\n";

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

{

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

{

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

cin>>g[i][j];

}

}

}

void displayg()

{

cout<<"\nAdjancy Matrix Is:\n";

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

{

cout<<"[";

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

{

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

}

cout<<"]\n";

}

}

void dijkstra()

{

int cost[max][max];

int v[max]={0};

int dist[max],sn=0,nn;

int pred[max];

int mind;

int i,j;

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

{

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

{

if(g[i][j]==0)

{

cost[i][j]=INT\_MAX;

}

else

{

cost[i][j]=g[i][j];

}

}

}

cout<<"\nCost Matrix Is:\n";

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

{

cout<<"[";

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

{

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

}

cout<<"]\n";

}

cout<<"\nlist of visited nodes is:\n";

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

{

cout<<i<<"="<<v[i]<<"\n";

}

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

{

dist[i]=cost[sn][i];

pred[i]=sn;

v[i]=0;

}

dist[sn]=0;

v[sn]=1;

int cnt=1;

while(cnt<=n-1)

{

mind=INT\_MAX;

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

{

if(dist[i]<=mind && !v[i])

{

mind=dist[i];

nn=i;

}

}

v[nn]=1;

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

{

if(!v[i])

{

if(mind+cost[nn][i]<dist[i])

{

dist[i]=mind+cost[nn][i];

pred[i]=nn;

}

}

}

cnt++;

}

//print

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

{

if(i!=sn)

{

cout<<"\n Distance Of Node"<<i<<"="<<dist[i];

cout<<"\n Path="<<i;

j=i;

do

{

j=pred[j];

cout<<"<-"<<j;

}while(j!=sn);

}

}

}

};

int main()

{

graph g;

g.getgraph();

g.displayg();

g.dijkstra();

return 0;

}