Represent a given graph using adjacency matrix /adjacency list and find the

shortest path using Dijkstra's algorithm (single source all destination).

INPUT:

#include <iostream>

using namespace

std; class graph{

int g[20][20];

int e,v;

public:

void accept(); void

display(); void

dijkstra(int start);

}; void graph:: accept(){ int src, dest,

cost, i,j; cout<<"Enter the number of

vertices: "; cin>>v;

cout<<"Enter the number of edges: ";

cin>>e; for(i=0; i<v; i++){

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

g[i][j]=0;

} } for(i=0; i<e; i++){

cout<<"\nEnter source vertex: ";

cin>>src;

cout<<"Enter destination vertex: ";

cin>>dest;

cout<<"Enter the cost of the edge: ";

cin>>cost; g[src][dest]=cost;

g[dest][src]=cost;

} } void

graph::display(){

int i,j; for(i=0;

i<v; i++){

cout<<endl;

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

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

}

} }

void graph::dijkstra(int start){

int r[20][20],

visited[20],distance[20],from[20],i,j,cnt,mindst,next;

for(i=0; i<v; i++){ for(j=0; j<v; j++){

if(g[i][j]==0){ r[i][j]=999;

}

else{

r[i][j]=g[i][j];

}

} } for(i=0;

i<v; i++){ visited[i]=0;

from[i]=start;

distance[i]=r[start][i];

}

distance[start]=0;

visited[start]=1;

cnt=v; while(cnt>0){

mindst=999;

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

if((mindst>distance[i]) && visited[i]==0){

mindst=distance[i]; next=i;

} }

visited[next]=1; for(i=0; i<v; i++){

if(visited[i]==0 &&

distance[i]>(mindst+r[next][i])){

distance[i]=mindst+r[next][i];

from[i]=next;

}

} cnt--

; }

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

cout<<"\nDistance of "<<i<< " from "<<start<<" is

"<<distance[i]<<endl<<"Path "<<i;

j=i; do{

j=from[j]; cout<<"<-

"<<j;

}

while(j!=start);

} } int

main() {

graph g;

int s;

g.accept();

g.display();

cout<<"\nEnter the starting vertex: ";

cin>>s;

g.dijkstra(s); return

0;

}

OUTPUT:

Enter the number of vertices: 6

Enter the number of edges: 8

Enter source vertex: 0

Enter destination vertex: 1

Enter the cost of the edge: 7

Enter source vertex: 1

Enter destination vertex: 2

Enter the cost of the edge: 9

Enter source vertex: 2

Enter destination vertex: 3

Enter the cost of the edge: 1

Enter source vertex: 4

Enter destination vertex: 3

Enter the cost of the edge: 5

Enter source vertex: 5

Enter destination vertex: 4

Enter the cost of the edge: 10

Enter source vertex: 0

Enter destination vertex: 5

Enter the cost of the edge: 12

Enter source vertex: 1

Enter destination vertex: 5

Enter the cost of the edge: 2

Enter source vertex: 4

Enter destination vertex: 2

Enter the cost of the edge: 4

0 7 0 0 0 12

7 0 9 0 0 2

0 9 0 1 4 0

0 0 1 0 5 0

0 0 4 5 0 10

12 2 0 0 10 0

Enter the starting vertex: 0

Distance of 0 from 0 is 0

Path 0<-0

Distance of 1 from 0 is 7

Path 1<-0

Distance of 2 from 0 is 16

Path 2<-1<-0

Distance of 3 from 0 is 17

Path 3<-2<-1<-0

Distance of 4 from 0 is 19

Path 4<-5<-1<-0

Distance of 5 from 0 is 9

Path 5<-1<-0