**Assignment No : 8**

Name : Rasika Mahure

PRN No : 72018258B

Div : B

Roll No : S1951093

**Aim :-**

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**

#include<iostream>

using namespace std;

class graph

{

int g[20][20];

int e,v;

public:

void accept()

{

  int src,dest,cost,i,j;

  cout<<"\n Enter no. of vertices -";

  cin>>v;

  cout<<"\n Enter no. 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<<"\n Enter source and destination -";

     cin>>src>>dest;

     cout<<"\n Enter the cost of edges - ";

     cin>>cost;

     g[src][dest]=cost;

}

}

void display()

{

  int i,j;

  for(i=0;i<v;i++)

  {

    cout<<endl;

  for(j=0;j<v;j++)

  {

    cout<<g[i][j]<<"\t";

  }

  }

}

void djikstra(int start)

{

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

  for(i=0;i<v;i++)           //intialization of r[][]

  {

     for(j=0;j<v;j++)

     {

       if(g[i][j]==0)

        r[i][j]=9999;

       else

        r[i][j]=g[i][j];

     }

  }

  for(i=0;i<v;i++)//intialization of visited[],distance[],from[]

  {

     visited[i]=0;

     from[i]=start;

     distance[i]=r[start][i];

  }

     distance[start]=0;

     visited[start]=1;

     cnt=v;

  while(cnt>0)

  {

    mindst=9999;

      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<<"\n Distance of "<<i<<" from "<<start<<" is "<<distance[i]<<endl<<" path "<<i;

 j=i;

do

{

j=from[j];

cout<<"<-"<<j;;

 }while(j!=start);

}

}

};

int main()

{

  int s;

  graph g;

  g.accept();

  g.display();

  cout<<endl<<"Enter the starting vertex -";

  cin>>s;

  g.djikstra(s);

  cout<<endl;

  return 0;

}

**//OUTPUT**

Enter no. of vertices -7

Enter no. of edges 8

Enter source and destination -0 1

Enter the cost of edges - 2

Enter source and destination -1 2

Enter the cost of edges - 5

Enter source and destination -2 3

Enter the cost of edges - 6

Enter source and destination -3 4

Enter the cost of edges - 1

Enter source and destination -4 6

Enter the cost of edges - 8

Enter source and destination -4 5

Enter the cost of edges - 11

Enter source and destination -5 0

Enter the cost of edges - 10

Enter source and destination -0 6

Enter the cost of edges - 9

0 2 0 0 0 0 9

0 0 5 0 0 0 0

0 0 0 6 0 0 0

0 0 0 0 1 0 0

0 0 0 0 0 11 8

10 0 0 0 0 0 0

0 0 0 0 0 0 0

Enter the starting vertex -0

Distance of 0 from 0 is 0

path 0<-0

Distance of 1 from 0 is 2

path 1<-0

Distance of 2 from 0 is 7

path 2<-1<-0

Distance of 3 from 0 is 13

path 3<-2<-1<-0

Distance of 4 from 0 is 14

path 4<-3<-2<-1<-0

Distance of 5 from 0 is 25

path 5<-4<-3<-2<-1<-0

Distance of 6 from 0 is 9

path 6<-0