Name: Pratik Rameshwar Shinde

Roll No.: 2274

Class: SE-IT

Div: B-3

Subject: DSA LAB

Practical 8: Graph: Shortest Path Algorithm

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**********

```
#define INFINITY 9999
#include<iostream>
#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]=INFINITY;
                        }
                        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=INFINITY;
        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;
}
```

**********OUTPUT**********

```
[admin@fedora ~]$ g++ hfbdsa8.cpp [admin@fedora ~]$
./a.out

Enter No. Of Vertices In Graph:4 Enter the
weight of edge:
[0][0]=4
```

[0][1]=5
[0][2]=6
[0][3]=8
[1][0]=9
[1][1]=7
[1][2]=1
[1][3]=2
[2][0]=3
[2][1]=5
[2][2]=15
[2][3]=12
[3][0]=41
[3][1]=21
[3][2]=1
[3][3]=3

Adjancy Matrix Is:
[4 5 6 8]
[9 7 1 2]
[3 5 15 12]
[41 21 1 3]

Cost Matrix Is:
[4 5 6 8]
[9 7 1 2]
[3 5 15 12]
[41 21 1 3]
Distance Of Node1=5
Path=1<-0
Distance Of Node2=6
Path=2<-0
Distance Of Node3=7
Path=3←←1 0
[admin@fedora ~]$