**Name:Shraddha Zoman**

**Roll no:TYCOD298**

**Problem Statement:** You have been given a network of ‘N’ nodes from 1 to ‘N’ and ‘M’ edges. For each edge, you are given three values (ui, vi, wi) where “ui” and “vi” denote the nodes and “wi” denotes an integer value which represents the time taken by a signal to travel from “ui” to “vi”. Now, you are supposed to find the time which a signal takes to travel from a given node ‘K’ to all nodes. If it is impossible for all nodes to receive the signal then print -1. Implement the given Network Delay Time using Dijkstra’s algorithm

**Program:**

import java.util.\*;

public class assign4 {

    static int mincost(int cost[], boolean vis[])

    {

        int min=Integer.MAX\_VALUE;

        int idx=-1;

        for(int i=0; i<cost.length;i++)

        {

            if(!vis[i]&&cost[i]<=min)

            {

                min=cost[i];

                idx=i;

            }

        }

        return idx;

    }

    static void shortestpath(int graph[][], int src, int n)

    {

        int cost[] =new int[n];

        boolean vis[]=new boolean[n];

        for(int i=0;i<n;i++)

        {

            cost[i]=Integer.MAX\_VALUE;

            vis[i]=false;

        }

        cost[src]=0;

        for(int i=0;i<n;i++)

        {

            int x=mincost(cost,vis);

            vis[x]=true;

            for(int j=0;j<n;j++)

            {

                if(!vis[j]&&graph[x][j]!=0&&cost[x]!=Integer.MAX\_VALUE&&cost[x]+graph[x][j]<cost[j])

                {

                    cost[j]=graph[x][j]+cost[x];

                }

            }

        }

        for(int i=0;i<cost.length;i++)

        {

            System.out.println(i+" "+cost[i]);

        }

    }

    public static void main(String args[])

    {

        Scanner sc= new Scanner(System.in);

        System.out.println("Enter no of vertex");

        int v=sc.nextInt();

        int graph[][]=new int[v][v];

        for(int i=0;i<v;i++)

        {

            for(int j=0;j<v;j++)

            {

                System.out.println("Enter edge["+i+"]"+"["+j+"]");

                graph[i][j]=sc.nextInt();

            }

        }

        System.out.println("Shortest Path is ");

        shortestpath(graph,0,v);

    }

}

**Output**

PS D:\Java> javac assign4.java

PS D:\Java> java assign4.java

Enter no of vertex

6

Enter edge[0][0]

0

Enter edge[0][1]

2

Enter edge[0][2]

4

Enter edge[0][3]

0

Enter edge[0][4]

0

Enter edge[0][5]

0

Enter edge[1][0]

2

Enter edge[1][1]

0

Enter edge[1][2]

1

Enter edge[1][3]

7

Enter edge[1][4]

0

Enter edge[1][5]

0

Enter edge[2][0]

4

Enter edge[2][1]

1

Enter edge[2][2]

0

Enter edge[2][3]

0

Enter edge[2][4]

3

Enter edge[2][5]

0

Enter edge[3][0]

0

Enter edge[3][1]

7

Enter edge[3][2]

0

Enter edge[3][3]

0

Enter edge[3][4]

2

Enter edge[3][5]

1

Enter edge[4][0]

0

Enter edge[4][1]

0

Enter edge[4][2]

3

Enter edge[4][3]

2

Enter edge[4][4]

0

Enter edge[4][5]

5

Enter edge[5][0]

0

Enter edge[5][1]

0

Enter edge[5][2]

0

Enter edge[5][3]

1

Enter edge[5][4]

5

Enter edge[5][5]

0

Shortest Path is

0 0

1 2

2 3

3 8

4 6

5 9