**Week 6:Write a java program to implement Dijkstra’s algorithm for the Single source shortest path problem.**

import java.util.\*;

public class Dijkstra

{

public int distance[] = new int[10];

public int cost[][] = new int[10][10];

public void calc(int n,int s)

{

int flag[] = new int[n+1];

int i,minpos=1,k,c,minimum;

for(i=1;i<=n;i++)

{

flag[i]=0;

this.distance[i]=this.cost[s][i];

}

c=2;

while(c<=n)

{

minimum=9999;

for(k=1;k<=n;k++)

{

if(this.distance[k]<minimum && flag[k]!=1)

{

minimum=this.distance[k];

minpos=k;

}

}

flag[minpos]=1;

c++;

for(k=1;k<=n;k++)

{

if(this.distance[minpos]+this.cost[minpos][k] < this.distance[k] && flag[k]!=1 )

this.distance[k]=this.distance[minpos]+this.cost[minpos][k];

}

}

}

public static void main(String args[])

{

int nodes,source,i,j;

Scanner in = new Scanner(System.in);

System.out.println("Enter the Number of Nodes");

nodes = in.nextInt();

Dijkstra d = new Dijkstra();

System.out.println("Enter the Cost Matrix Weights:");

for(i=1;i<=nodes;i++)

for(j=1;j<=nodes;j++)

{

d.cost[i][j]=in.nextInt();

if(d.cost[i][j]==0)

d.cost[i][j]=9999;

}

System.out.println("Enter the Source Vertex :");

source=in.nextInt();

d.calc(nodes,source);

System.out.println("The Shortest Path from Source \t"+source+"\t to all other vertices are : \n");

for(i=1;i<=nodes;i++)

if(i!=source)

System.out.println("source :"+source+"\t destination :"+i+"\t MinCost is :"+d.distance[i]+"\t");

}

}