**//BELLMAN FORD**

import java.util.Scanner;

public class Main

{

private int D[];

private int num\_ver;

public static final int MAX\_VALUE = 99;

public Main(int num\_ver)

{

this.num\_ver = num\_ver;

D = new int[num\_ver + 1];

}

public void BellmanFordEvaluation(int source, int A[][])

{

for (int node = 1; node <= num\_ver; node++)

{

D[node] = MAX\_VALUE;

}

D[source] = 0;

for (int node = 1; node <= num\_ver - 1; node++)

{

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

if (A[sn][dn] != MAX\_VALUE)

{

if (D[dn] > D[sn]+ A[sn][dn])

D[dn] = D[sn] + A[sn][dn];

}

}

}

}

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

if (A[sn][dn] != MAX\_VALUE)

{

if (D[dn] > D[sn]+ A[sn][dn])

{System.out.println("The Graph contains negative edge cycle");

System.exit(0);}

}

}

}

System.out.println("distances are");

for (int vertex = 1; vertex <= num\_ver; vertex++)

{

System.out.println("from " + source + " to "+ vertex + " is " + D[vertex]);

}

}

public static void main(String[ ] args)

{

int num\_ver = 0;

int source;

Scanner scanner = new Scanner(System.in); System.out.println("Enter the number of vertices");

num\_ver = scanner.nextInt();

int A[][] = new int[num\_ver + 1][num\_ver + 1];

System.out.println("Enter the adjacency matrix");

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

A[sn][dn] = scanner.nextInt();

if (sn == dn)

{

A[sn][dn] = 0;

continue;

}

if (A[sn][dn] == 0)

{

A[sn][dn] = MAX\_VALUE;

}

}

}

System.out.println("Enter the source vertex");

source = scanner.nextInt();

Main b = new Main (num\_ver); b.BellmanFordEvaluation(source, A); scanner.close();

}

}