***Bellman Ford***

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Main {

class Edge {

int src, dest, weight;

Edge()

{

src = dest = weight = 0;

}

};

int V, E;

Edge edge[];

Main(int v, int e)

{

V = v;

E = e;

edge = new Edge[e];

for (int i = 0; i < e; ++i)

edge[i] = new Edge();

}

void BellmanFord(Main graph, int src)

{

int V = graph.V, E = graph.E;

int dist[] = new int[V];

for (int i = 0; i < V; ++i)

dist[i] = Integer.MAX\_VALUE;

dist[src] = 0;

for (int i = 1; i < V; ++i) {

for (int j = 0; j < E; ++j) {

int u = graph.edge[j].src;

int v = graph.edge[j].dest;

int weight = graph.edge[j].weight;

if (dist[u] != Integer.MAX\_VALUE && dist[u] + weight < dist[v])

dist[v] = dist[u] + weight;

}

}

for (int j = 0; j < E; ++j) {

int u = graph.edge[j].src;

int v = graph.edge[j].dest;

int weight = graph.edge[j].weight;

if (dist[u] != Integer.MAX\_VALUE && dist[u] + weight < dist[v]) {

System.out.println("Graph contains negative weight cycle");

return;

}

}

printArr(dist, V);

}

void printArr(int dist[], int V)

{

System.out.println("Vertex Distance from Source");

for (int i = 0; i < V; ++i)

System.out.println(i + "\t\t" + dist[i]);

}

public static void main(String[] args)

{

int V = 5;

int E = 8;

Main graph = new Main(V, E);

graph.edge[0].src = 0;

graph.edge[0].dest = 1;

graph.edge[0].weight = -1;

graph.edge[1].src = 0;

graph.edge[1].dest = 2;

graph.edge[1].weight = 4;

graph.edge[2].src = 1;

graph.edge[2].dest = 2;

graph.edge[2].weight = 3;

graph.edge[3].src = 1;

graph.edge[3].dest = 3;

graph.edge[3].weight = 2;

graph.edge[4].src = 1;

graph.edge[4].dest = 4;

graph.edge[4].weight = 2;

graph.edge[5].src = 3;

graph.edge[5].dest = 2;

graph.edge[5].weight = 5;

graph.edge[6].src = 3;

graph.edge[6].dest = 1;

graph.edge[6].weight = 1;

graph.edge[7].src = 4;

graph.edge[7].dest = 3;

graph.edge[7].weight = -3;

graph.BellmanFord(graph, 0);

}

}