**Code for Bellman Ford:**

import java.util.\*;

class Main {

static class Edge {

int src, dest, weight;

}

static class Graph {

int V, E;

Edge edge[];

}

static Graph createGraph(int V, int E) {

Graph graph = new Graph();

graph.V = V;

graph.E = E;

graph.edge = new Edge[graph.E];

for (int i = 0; i < graph.E; i++) {

graph.edge[i] = new Edge();

}

return graph;

}

static boolean isNegCycleBellmanFord(Graph graph, int src) {

int V = graph.V;

int E = graph.E;

int[] dist = new int[V];

// Step 1: Initialize distances from src

// to all other vertices as INFINITE

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

dist[i] = Integer.MAX\_VALUE;

dist[src] = 0;

printArr(dist,V);

// Step 2: Relax all edges |V| - 1 times.

for (int i = 1; i <= V - 1; 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;

System.out.print("\nCurrent Iteration : ");

printArr(dist,V);

}

}

}

// Step 3: check for negative-weight cycles.

// The above step guarantees shortest distances

// if graph doesn't contain negative weight cycle.

// If we get a shorter path, then there

// is a cycle.

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

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

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

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

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

return true;

}

return false;

}

static void printArr(int dist[], int V)

{

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

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

if(dist[i] == 2147483647) // Garbage value

{

System.out.println(i + "\t\tInfinity");

continue;

}

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

}

}

public static void main(String[] args) {

int V = 5, E = 6;

Graph graph = createGraph(V, E);

// add edge 0-1 (or A-B in above figure)

graph.edge[0].src = 0;

graph.edge[0].dest = 1;

graph.edge[0].weight = 2;

graph.edge[1].src = 0;

graph.edge[1].dest = 2;

graph.edge[1].weight = 4;

graph.edge[2].src = 1;

graph.edge[2].dest = 3;

graph.edge[2].weight = 2;

graph.edge[3].src = 2;

graph.edge[3].dest = 4;

graph.edge[3].weight = 3;

graph.edge[4].src = 2;

graph.edge[4].dest = 3;

graph.edge[4].weight = 4;

graph.edge[5].src = 4;

graph.edge[5].dest = 3;

graph.edge[5].weight = -5;

if (isNegCycleBellmanFord(graph, 0))

System.out.println("\nYes");

else

System.out.println("\nNo");

}

}

**Output:**