**Ex:11 - Bellman-Ford algorithm**

**Name:** Athithraja R

**Reg.no:** 2022503702

**GFG.java**

import java.util.Scanner;

import java.util.Arrays;

class GfG {

static int[] bellmanFord(int V, int[][] edges, int src) {

int[] dist = new int[V];

Arrays.fill(dist, Integer.MAX\_VALUE); // Set all distances to infinity initially

dist[src] = 0;

// Relaxation of edges V-1 times

for (int i = 0; i < V - 1; i++) { // V-1 iterations

for (int[] edge : edges) {

int u = edge[0];

int v = edge[1];

int wt = edge[2];

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

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

}

}

}

// Check for negative weight cycle

for (int[] edge : edges) {

int u = edge[0];

int v = edge[1];

int wt = edge[2];

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

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

return new int[]{-1}; // Return -1 to indicate negative cycle

}

}

return dist;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of vertices: ");

int V = sc.nextInt();

System.out.print("Enter the number of edges: ");

int E = sc.nextInt();

int[][] edges = new int[E][3];

System.out.println("Enter the edges (u, v, weight) for each edge:");

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

edges[i][0] = sc.nextInt(); // u

edges[i][1] = sc.nextInt(); // v

edges[i][2] = sc.nextInt(); // weight

}

System.out.print("Enter the source vertex: ");

int src = sc.nextInt();

// Get the shortest distances

int[] ans = bellmanFord(V, edges, src);

if (ans[0] != -1) { // If there is no negative cycle

System.out.println("Shortest distances from vertex " + src + " are:");

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

if (ans[i] == Integer.MAX\_VALUE) {

System.out.println("Vertex " + i + " is unreachable from source " + src);

} else {

System.out.println("Distance to vertex " + i + ": " + ans[i]);

}

}

}

sc.close();

}

}

**Output:**

