**Code:**

#include <bits/stdc++.h>

using namespace std;

struct Edge {

int src, dest, weight;

};

class Graph {

int V, E;

vector<Edge> edges;

public:

Graph(int vertices, int edgesCount) : V(vertices), E(edgesCount) {}

void addEdge(int src, int dest, int weight) {

edges.push\_back({src, dest, weight});

}

void bellmanFord(int startVertex) {

vector<int> distance(V, INT\_MAX);

distance[startVertex] = 0;

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

for (const Edge& edge : edges) {

if (distance[edge.src] != INT\_MAX && distance[edge.src] + edge.weight < distance[edge.dest]) {

distance[edge.dest] = distance[edge.src] + edge.weight;

}

}

}

for (const Edge& edge : edges) {

if (distance[edge.src] != INT\_MAX && distance[edge.src] + edge.weight < distance[edge.dest]) {

cout << "Graph contains negative weight cycle. Bellman-Ford does not work for such graphs.\n";

return;

}

}

cout << "Vertex Distance from Source\n";

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

cout << i << "\t\t" << distance[i] << endl;

}

}

};

int main() {

int V, E;

cout << "Enter the number of vertices and edges: ";

cin >> V >> E;

Graph graph(V, E);

cout << "Enter the edges (source, destination, weight):\n";

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

int src, dest, weight;

cin >> src >> dest >> weight;

graph.addEdge(src, dest, weight);

}

int startVertex;

cout << "Enter the starting vertex: ";

cin >> startVertex;

auto start = chrono::steady\_clock::now();

graph.bellmanFord(startVertex);

auto end = chrono::steady\_clock::now();

auto diff = end - start;

cout <<"Execution time: "<< chrono::duration <double, milli> (diff).count() << " ms\n";

return 0;

}

**Output:**

