CSE 100 Algorithm Design and Analysis Lab 10 (20 points)
Due date: December 6, 23:59

The Bellman-Ford Algorithm

In this assignment, you are asked to implement the Bellman-Ford Algorithm which solves the single-source shortest-paths problem. Specifically, you are given as input a directed graph $G = (V, E)$ with weight $w(u, v)$ on each edge $(u, v) \in E$ along with a source vertex $s \in V$. Edges may have negative weights.

The input has the following format. There are two integers on the first line. The first integer represents the number of vertices $|V|$. The second integer is the number of edges $|E|$. Vertices are indexed by $0, 1, \ldots, |V| - 1$. On the following $|E|$ lines, there are three integers $u, v, w(u, v)$ representing an edge (u, v) with weight $w(u, v)$. Vertex 0 is the source vertex. The output falls into two possible cases:

Case (i): There is no negative-weight cycle reachable from s. In this case, you must output TRUE on the first line, followed by the shortest distance from s to each vertex in the graph. More precisely, you must output TRUE, $\delta(0, 0)$, $\delta(0, 1)$, $\ldots$, $\delta(0, |V| - 1)$, one per line. Recall that $\delta(u, v)$ denotes the shortest distance from u to v. If a vertex v is not reachable, output INFINITY in place of $\delta(0, v)$.

Case (ii): There is a negative-weight cycle reachable from s. You must output FALSE.

Examples of input and output

```
Input 1
6 10
0 1 6
1 2 5
1 3 -4
1 4  8
2 1 -2
3 0 2
3 2 7
3 4 9
4 0 7
5 2 5

Output 1
TRUE
0
6
9
2
11
INFINITY

Input 2
6 11
```

```
0 1 6
1 2 5
1 3 -4
1 4  8
2 1 -2
3 0 2
3 2 7
3 4 9
3 5 -14
4 0 7
5 2 5

Output 2
FALSE
```

Submission Submit the source code `BellmanFord.cpp` through the assignments page of CatCourses by the deadline. Be careful since CatCourses strictly enforces the assignment deadline. You may be asked to compile, run, and explain the code to the TA to prove that you understand what you wrote.

Grading We provide 10 test cases for you to try your implementation. Each of them is valued at 1 point if executed correctly. We will use 10 additional test cases to check that your implementation is general enough. These are not provided to you and are also valued at 1 point each.

Important Note We will use plagiarism software to detect cheating. Offenders will be subjected to the UCM Academic Honesty Policy which states: *if any violation of the UCM Academic Honesty Policy is suspected in a course, the instructor of record must fill out the Faculty Report for Academic Misconduct.*