CS 330 – Bellman-Ford for Shortest Paths

deadline: 12/1/19 at 11:59 pm

In this exercise you will implement Bellman and Ford's dynamic programming algorithm for solving the single-source shortest paths problem. Given a directed graph with (possibly negative) weights on each edge and a source vertex s, you must find the shortest distance to every vertex. An algorithm for finding the shortest path from vertices s to t is given in *Algorithm Design*, Section 6.8, with pseudocode on page 294. There are additional details in the solutions for Lab 10. Your code should run in $\mathcal{O}(nm)$ time, where n is the number of vertices and m is the number of edges.

A separate version, based on "relaxing tense edges," was presented in lecture. One more detailed reference for it can be found in Section 24.1 of *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein. This textbook is commonly referred to by the abbreviation "CLRS" and is listed in the syllabus as a useful additional resource. Either presentation will work.

Submission Format

On Gradescope submit `bellman_ford.py` or `bellman_ford.java` (names must match exactly) reading a text file `input` and creating a text file `output` containing your solution. You may submit other files containing additional functions and classes. The autograder will put your code in the same directory as `input`, call it from the command line, and look for the file `output` in the same directory.

Input

The input format is identical to that used in Dijkstra's algorithm, with the exception that weights may now be negative. The file `input` will have $m + 3$ lines specifying a directed, weighted graph. The first line will be n, the number of vertices. The second line will be m, the number of edges. The third line will be the index s of the vertex from which to start. Every row after that will specify an edge by giving the starting node, the ending node, and the weight. The weights will all be integers. You may assume that there is a path to every node and that there are no negative cycles.

We have uploaded files corresponding to the visible test cases on Gradescope. They may help you test and debug locally. Note: passing these test cases does not guarantee your code is correct!

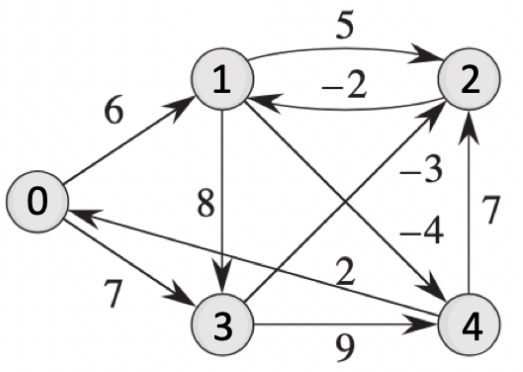
Output

Your algorithm must create a text file `output` listing the length of the shortest path to every vertex. The top line will correspond to vertex 0, the next line to vertex 1, and so on down to vertex $n - 1$.

Example on next page.

Example

Consider the following directed, weighted graph. We'll start measuring distances from node 0. The output file lists the correct shortest distances in order.



input:

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

output:

```
0
2
4
7
-2
```