CIS 315, Intermediate Algorithms
Winter 2019

Assignment 2

due January 28, 2019

1 Description

For this assignment, you are to write a program which will take the description of a weighted directed acyclic graph from standard input and write to standard output two numbers measuring the graph. The program will need to compute (i) the length of the longest path from 1 to n, and (ii) the number of distinct paths from 1 to n that have the longest length found in part (i).

You may write your program in either Java, Python, C, or C++. Other languages need to be approved by the instructor. You may store the graph with either an adjacency list or an adjacency matrix. Your program should implement a linear time algorithm and will be tested on very large graphs, so you cannot possibly list them all. As discussed in class, "linear time" (with respect to the input size) here means $O(n+m)$ if using an adjacency list, and $O(n^2)$ if using adjacency matrix.

2 Sample I/0

The first line of the input is a line containing two integers, N and M. N is the number of nodes - the nodes are $1, 2, 3, \ldots, N$ - and M is the number of edges. There follow M lines describing each edge. An edge description is a line containing three integers I J W: $I \leq J$ (ensuring that the graph is acyclic) and $W > 0$.

input file *inSample1.txt*

```
5 7
1 2 7
1 3 10
2 4 8
2 5 3
3 4 5
3 5 10
4 5 5
```

Expected output:

```
longest path: 20
```

```
number of longest paths: 3
```

input file *inSample2.txt*

```
10 12
1 2 1
1 3 1
2 4 1
3 4 1
4 5 1
4 6 1
5 7 1
6 7 1
7 8 1
7 9 1
8 10 1
9 10 1
```

Expected output:

```
longest path: 6
number of longest paths: 8
```

3 Testing Protocol

We will test your program by running your program at the command line. You will need to use **standard input**. Do not pass in the name of the file as an argument - do not encode the name of your input file in your program. We will run your program on several different test files, some of which may be generated by other programs and piped into yours.

We will not attempt to cause overflow on the number of paths - that number should fit into an `int`. If your program seems slow, which is often caused by trying to list all possible paths individually (exponential time!), be aware that we will test it on a graph with many paths.

You may want to consider the case where the graph is not connected or where there is no path from node 1 to node N. However, we will not test your program on any tricky inputs: there will be a path from node 1 to all other nodes, and from all nodes to node N.

4 Submission

Post your work to Canvas, as with HW0. Submit only a single source file