**CIS 315, Intermediate Algorithms**
**Spring 2019**

# Assignment 2

*due April 22, 2019*

## 1 Description

For this assignment, you are to write a program which will take the description of a series of unweighted directed acyclic graphs from **standard input** and write to **standard output** four different measures for each graph. The measures the program will need to compute are

1. the length of the shortest path from node 1 to node n
2. the number of distinct shortest paths from 1 to n of the length above
3. the length of the longest path from 1 to n
4. the number of distinct longets paths from 1 to n of the length above

You may write your program in either Java, Python, C, or C++. Other languages need to be approved by the TA or 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 enumerate all possible paths. Linear time here means $O(n + m)$ if using an adjacency list, and $O(n^2)$ if using adjacency matrix.

## 2 Sample Input

The input will be a text file. The graph description starts with the integers $N$ and $M$, each on a separate line. $N$ is the number of nodes (numbered $1, 2, \ldots, N$) and $M$ is the number of edges ($N \geq 1, M \geq N - 1$). Following that will be $M$ lines of the form $I$ $J$, indicating that there is an edge from node $I$ to node $J$ It will always be the case that $1 \leq I < J \leq N$, which ensure that the graph is a DAG and that the topological ordering is $1, 2, \ldots, N$.

## 3 Examples

**Graph 1**

*input:* from file inSmall1.txt

```
5
10
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
4 5
```

*output:*

```
Shortest path: 1
Number of short paths: 1
Longest path: 4
Number of long paths: 1
```

**Graph 2**

*input:* from file inSmall3.txt

```
13
18
1 2
1 3
1 4
2 4
3 4
4 5
4 6
5 7
6 7
7 8
7 9
7 10
8 10
9 10
10 11
10 12
11 13
12 13
```

*output:*

```
Shortest path: 6
Number of short paths: 4
Longest path: 8
Number of long paths: 16
```

## 4 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.

## 5 Submission

Post a copy of your *.java* (or *.py*, *.c*, or *.cpp*) program to Canvas by one second before midnight of the due date. The file name does not matter, but it should be a single file.