COS 485: Program #1 – Longest Row in O(N) comparisons

Objectives: Designing, coding and testing your first algorithm for COS 485.

We have NxN matrix (2D array) that contains just 1's and 0's. They are stored so that in each row, all the 1's come before all the 0's in that row. Assuming the matrix is already in memory, write an algorithm running in O(N) time that finds the length of the row that contains the most 1's.

The programming assignments for this course will be written in Java and developed and tested within a highly supportive system called the Scaffold that was built to allow you to focus solely on designing your algorithm. It reads input files, runs a series of test cases, shows visualizations of your results, evaluates their correctness, times their execution times, plots the times on a graph, and produces a summary report.

Setting up the project in Eclipse:

- Make a new java project. Use java 11 or java 17.
- Download the jar files from Brightspace, unzip them, and copy them into your project *(not into any package)*
  - **LongestRowTester.jar**
  - **Scaffold.jar**
- Configure the build path: Project > Properties > Java Build Path > Libraries > Classpath > Add JARs... > *add the jar files*
- Make a new package: **student** *(you must use this name because the tester looks here)*
- Copy the starting code .java file into the student package. There should not be any compiler errors.
  - **LongestRow.java**
- Now make a run configuration: Run > Run Configuration... > Java Application > 'New' button >
  Name: *give a name for this run configuration*
  Project: *the name of your project*
  Main class: **tester/LongestRowTester** *(check capitalization & spelling if it can't find main)*
- Run *(The starting code is working code, although it does not find correct answers.)*

Your Task:

Design an algorithm to solve the task, then modify LongestRow.java to implement your algorithm.

Notes about the Scaffold system:

1. You can use the debugger. If you want to debug just a specific test case, put it in the Arguments tab of the run configuration. E.g.: test1.txt
2. The execution times reported are wall clock times. These may be affected by other processes running on your computer.
3. The graph tab plots the measured execution times in relation to common asymptotic growth rates. Execution times on real problems are sometimes much faster than worst case behavior. The graph is not a substitute for worst case analysis.

What to turn in:

Written Report turned in through Brightspace (must be .doc, .docx, or .pdf)

1. A brief English description of your algorithm
2. An analysis of its worst case execution time
3. A screen shot of the report tab
4. A screen shot of your results for test3.txt

Electronic Submit

From a Unix machine in the lab run the program “submit” to submit your files.
Submit your source code (.java files) and compiled code (.class files) to the directory: **prog1**

Grading:

- 10 points – description of your algorithm
- 10 points – analysis of worst case execution time
- 60 points – finding the longest row
- 20 points – meeting the O(N) execution time goal