**Lab Assignment #2 – Algorithm Analysis**

Due Date: Friday, Week 5

Purpose: The purpose of this Lab assignment is to:

1. Perform experimental analysis of algorithms
2. Explain and proof the running time of algorithms in terms of big-Oh notation

References: Read the course’s text chapter 4 and the lecture slides. This material provides the necessary information that you need to complete the exercises.

Be sure to read the following general instructions carefully:

- This assignment must be completed individually by all the students.

- You will have to **demonstrate your solution in a scheduled lab session** and upload the solution on **eCentennial** through the assignment link.

**Exercise 1**

1. Give a big-Oh characterization, in terms of **n**, of the running time of the **example1** method from **Exercises.java** class in Lesson 4 examples.
2. Give a big-Oh characterization, in terms of **n**, of the running time of the **example2** method from **Exercises.java** class in Lesson 4 examples.
3. Give a big-Oh characterization, in terms of **n**, of the running time of the **example3** method from **Exercises.java** class in Lesson 4 examples.
4. Give a big-Oh characterization, in terms of **n**, of the running time of the **example4** method from **Exercises.java** class in Lesson 4 examples.
5. Give a big-Oh characterization, in terms of **n**, of the running time of the **example5** method from **Exercises.java** class in Lesson 4 examples.

For each of the above questions, use comments in the code to provide the results and a brief explanation.

(4 marks)

**Exercise 2**

Perform an **experimental analysis** of the two algorithms *prefixAverage1* and *prefixAverage2*, from lesson examples. Optionally, visualize their running times as a function of the input size with a **log-log chart**. Use either Java or Python graphical capabilities for visualization. **Hint**: Choose representative values of the input size **n**, similar to *StringExperiment.java* from class examples.

(4 marks)

**Exercise 3**

For each of the algorithms *unique1* and *unique2* (**Uniqueness.java** class in Lesson 4 examples) which solve the element uniqueness problem, **perform an experimental analysis** to determine the largest value of **n** such that the given algorithm runs in one minute or less. **Hint**: Do a type of “binary search” to determine the maximum effective value of **n** for each algorithm.

(2 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Correct implementation of requirements:**   * Correct result of running time * Correct experimental analyses code and visualization * Explanation of proof or analyses when asked | 90% |
| **Friendly graphical display** | 10% |
| **Total** | 100% |

You must name your Eclipse project according to the following rule:

**YourFullname\_COMP254Labnumber\_Exercisenumber**.

Example: **JohnSmith\_ COMP254Lab2\_Ex1**

**Submission rules:**

Submit your modules as **zip files** that are named according to the following rule:

**YourFullname\_ COMP254Labnumber\_Exercisenumber.zip**

Example: **JohnSmith\_ COMP254Lab2\_Ex1.zip**

Use 7-zip to compress files (https://www.7-zip.org/download.html).