**Lab Assignment #2 – Algorithm Analysis**

Due Date: By the dropbox deadline.

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.

**Instructions:**

You **MUST** create a short demo video of your solution. Do not show yourself in the video. Upload your video in your personal youtube account or google-drive account and share it with the instructor **only**. Do not share it publicly. During submission at the dropbox, **write the link of your video** in the **Comments** **box** (present near the bottom of the submission page). Next, create a zip file of your solution as mentioned below in section **Submission Rules**, upload that zip file, and submit.

You **must** name a relevant Eclipse project according to the following rule:

YourFirstname\_YourLastname\_COMP254\_Labnumber\_ExerciseNumber.

Example: If student name is John Smith, the name of Eclipse project for Ex1 of Lab1 should be **John\_Smith\_COMP254\_Lab1\_Ex1**

**Submission Rules:**

Compress all your Eclipse projects as a **single** **zip** filethat is named according to the following rule: YourFirstname\_YourLastname\_COMP254\_Labnumber.zip

Example: **John\_Smith\_COMP254\_Lab1.zip**

Submit the above single zip file using the procedure mentioned in section **Instructions** above.

**Evaluation:**

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

**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.

**Ans**- There is one “for” loop for n and code will run for n time in for loop that’s why we can say that running time complexity is

**O(n).**

1. 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.

**Ans-** There is one “for” loop for n and code will run for n/2 time in for loop cause ni is increment by 2 that’s why we can say that running time complexity is **n/2**  and Big-Oh will be

**O(n) .**

1. 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.

**Ans-** Code will execute  1+2+3+4...+n times = (n² + n)/2 = n²/2 + n/2. Cause of nested loop .

And by simplify O(n2 /2 +n/2) and we only consider higher degree to represent big oh

So answer will be

**O(n2)**

1. 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.

**Ans-** There is one “for” loop for n and code will run for n time in for loop that’s why we can say that running time complexity is

**O(n).**

1. 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.

**Ans-**  There is 3 nested loop

1. N times
2. N times
3. 0 to

Inside 2 loop is same as question c we can convert into 2 loop

1. N times
2. O(N2)

Now we can use these two nested loops and calculate

Answer is

**O(n3)**

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

(5 marks)

**Exercise 2**

Perform an **experimental analysis** of the two algorithms *prefixAverage1* and *prefixAverage2*, posted in the **eCentennial** module “Lesson Examples (from textbook)”. Visualize their running times as a function of the input size with a **log-log chart** (A log-log chart is a two-dimensional graph where the x-coordinates and y-coordinates are logarithmic values [base of log should be same in both axes]). A log calculator can be found in <https://www.calculator.net/log-calculator.html>. You **must** use Java graphical capabilities for visualization—In this regard, you **must** use the external libraries **jcommon-1.0.23.jar** and **jfreechart-1.0.19.jar**. These two libraries are present inside the zip file libraries.zip. Download libraries.zip from the assignment link of **eCentennial**. Unzip it and use the two libraries. You **must not use any other external libraries.** Also note that if you search the internet, you may find examples on how to create a two-dimensional graph with cartesian co-ordinates [i.e. a X-Y graph] using the APIs provided in these two libraries. **If you use the code from the internet, please acknowledge the website/author** as comments in your code. There will be **no** **deduction of marks** if you do so. (**Hint**: Choose representative values of the input size **n**, similar to *StringExperiment.java* in Lesson4Examples posted in the **eCentennial** module “Lesson Examples (from textbook)”).

(5 marks)