COMP 2210 Assignment 3 Part 1
Group 28

Tyler Jewell **Jentry Chesnut**

February 23, 2016

Abstract

The time complexity of an algorithm can be described as the running time of an algorithm. The lab focuses on an empirical approach to analyzing the data and determining what the time complexity of the algorithm is. This means the experiment to find the big-Oh running time of TimingLab.timeTrial() method is subjective based on the system used to run the algorithm and the type of algorithm used in the method.

1. Problem Overview

In the lab, the problem of time complexity and, indirectly, scalability are studied with the main goal being able to discover the big-Oh notation of the TimingLab.timeTrial() method by using the data gathered from the TimingLabClient.java class and the a3resources.jar file. The other goals are to record the ratio and the value k using the formula detailed in Formula 1 below and record the data in a graph of N (Problem size) vs. Time (The time it took per problem size in seconds). In the experiment, we reduced the initial problem size by ½ in order to get enough data to make a well thought out conclusion about the data without the program running for 3.10 days.

$$T(N) \propto N^K \Rightarrow T(2N)/T(N) \propto (2N)^k/N^k = 2^k N^k/N^k = 2^k$$

Formula 1

2. Experimental Procedure

The machine used for this data is a Windows 8.1 OS with an Intel i5 processor and 12 GB of RAM running JDK 8 and JRE 8.

```
for (int i = 0; i < 5; i++) {
                        start = System.nanoTime();
                        tl.timeTrial(N);
                        elapsedTime = (System.nanoTime() - start) / BILLION;
                        System.out.print(N + "\t");
                        System.out.printf("%4.3f\n", elapsedTime);
                        N = N * 2;
        }
```

Snippet 1: Calling loop of timeTrial() method

The snippet of code shown above is used to determine the running time and print the problem size, N, alongside the running time in seconds. This section of code runs a total of five times, with each iteration doubling the problem size from the last run. Using the running time

gathered from the experiment, we calculated the ratio, and using the ratio we were able to calculate the value k, which is the exponent that correlates to the big-Oh notation for the algorithm. In order to calculate the ratio, the formula, $T(2N)/T(N)$, was utilized, and by performing basic algebra we were able to solve for the value k by using the formula, $log_2(T(2N)/T(N))$. These calculations were performed outside of the code by using a scientific calculator and excel as a tool to graph this data and put it in a table. The formula for percent error is also utilized in order to determine how close the approximations found in the experiment are to their theoretical expectation, $\%Error = |(Theoretical\ value - Experimental\ value)\ /\ (Theoretical\ value)| * 100$.

3. Data Collection and Analysis

Table 1: Running time data and calculations

N	Elapsed Time (sec)	R	k
4	0.328	—	—
8	3.72	11.341	3.503
16	58.792	15.804	3.982
32	938.025	15.955	3.996
64	15275.68	16.285	4.025

The data collected is found above in Table 1. What this data entails is the problem size, the amount of time elapsed in seconds, the ratio, and the constant value. To discover the big-Oh notation it is necessary to observe the value that the constant k is approaching, which in our groups case was four, which reveals the time complexity as $T(N) \propto N^4$ or O(N^4). Essentially what this means is that the expected runtime is growing by a power of four every time in the worst case. Calculation 1 shows that the runtime can be approximated rather closely.

$$T(64) = 15{,}275.68$$
$$R = 16.285$$
$$T(64)(R) = T(128)$$
$$15{,}275.68(16.285) = 248764.449 \approx N^4(.001) = 128^4(.001) = 268435.456$$
$$\%Error = |(268{,}435.356 - 248{,}764.449)\ /\ (268{,}435.356)| * 100$$
$$\%Error = 7.33\ \%$$

Calculation 1: This is a calculation to find an approximation for T(N) for any N, but in this case, T(128) is used. 248,764.449 seconds is approximately the same as 268,435.356. The percent error shows that the experimental value is within 7.33% of the theoretical value.

Figure 1 below, expresses the time complexity in a visual format, which aids in grasping what this time complexity means for the efficiency of the algorithm as the problem size continually doubles.

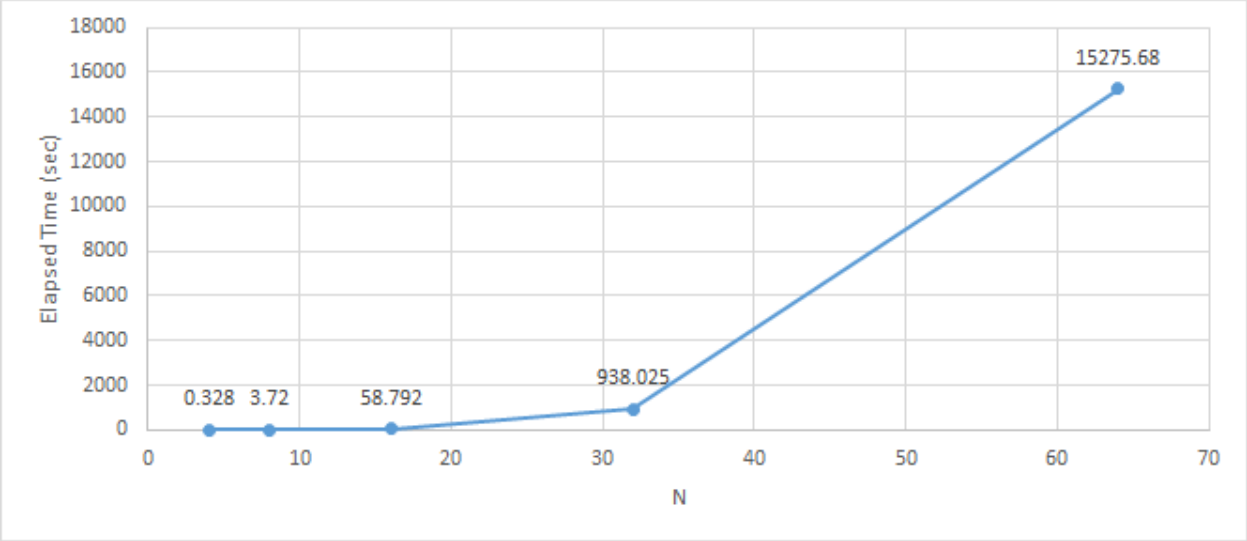

Figure 1: Graph of Time vs. N

4. Interpretation

Based on the data above and procedures given, we have concluded that the time complexity of TimingLab has a big-O notation of $O(N^4)$. We've come to this conclusion using the following using Fomula 1. By starting with a problem size of 4 and using the specific tools mentioned in the procedure, the time it takes to run the problem goes from 0.328 seconds for a problem size of 4 to 4.24 hours for a problem size of 64. To go further into the problem, we calculated a theoretical answer for a problem size of 128. According to the formula, plugging in 128 will theoretically give you an estimate of 69 to 72 hours. Overall, the data gathered enabled us to determine that it doesn't matter what system an algorithm is being ran on because if enough data is gathered it will always trend toward its big-Oh notation.