Lauren McFaden (903881469)

Dr. Hendrix  
Comp. 2210

9/25/17

**Assignment 3: Empirical Analysis – Time Complexity**

*Introduction/ Background Information*

For this experiment, the goal is to develop and perform an experiment that allows us to empirically discover and analyze the big-Oh time complexity first mentioned in lecture. Empirical analysis is “analyze running time based on observations and experiments.” Empirical analysis involves using the scientific method during our experiments. The big-Oh complexity allows us to explore the efficiency of different algorithms presented. The results from the big-Oh time complexity allows us to perform calculations to have concrete proof of the efficiency of a specific algorithm. For this experiment we will use our own Banner ID numbers in the methodToTime() method in the ProvidedClass. In addition, I will be using my own personal system that may have a potential impact on the efficiency of the algorithm.

*Procedure*

When beginning this experimental procedure I began with using the source code provided and performed an initial run using my own input banner ID number 903881469. After performing the initial code I ended up with a problem size and the elapsed time, and using the data I needed to determine the ratio between the elapsed time and the current time in addition to the log ratio. The ratio is used to show us that if the problem size has successfully doubled then the method’s running time can be determined by dividing the two time. In addition, he log ratio which can be written as log(elapsed time/ current time)/log(2) is used as a calculation for the formula K = log2 R. After obtaining all of the data for the big-Oh I analyzed that data with the intention to see if I could repeat this process and obtain the same data. This was the challenging aspect in the experiment understanding why the data would change every time I ran it. After several runs of the data and determine the ratios and log ratios it became clear that the ratios were precise with one another revealing the efficiency of the big-Oh time complexity.

*Data Collected*

Initial ProvidedClient.java output:

----jGRASP: process ended by user.

----jGRASP exec: java ProvidedClient

Problem size = 2 Elapsed time = 0.110620883

Problem size = 4 Elapsed time = 0.047990411

Problem size = 8 Elapsed time = 0.169313662

Problem size = 16 Elapsed time = 0.627369465

Problem size = 32Elapsed time = 2.044639534

Problem size = 64 Elapsed time = 7.248580708

Problem size = 128 Elapsed time = 28.918467352

Problem size = 256 Elapsed time = 118.791811437

Problem size = 512 Elapsed time = 480.262663851

Problem size = 1024 Elapsed time = 1907.176016753

----jGRASP: operation complete.

This is the initial data retrieved from the source code ProvideClient.java. I then took the data as is and determined the:

* R value = “the third column (R) is the ratio (i.e., T imei/T imei−1)”
* K value = “n (k) is log2 R” which can be translated as LOG(R)/LOG(2)

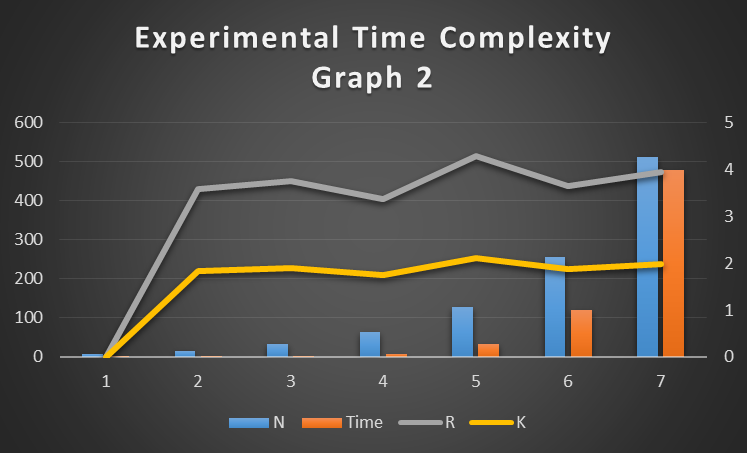
Taking that data and placing it into a chart that consists of each row in this table records data for a given run of some method being timed. Then the first column (N) records the problem size, the second column (Time) records the time taken for the method to run on this problem size. The R value is the third column and finally the K value is in the fourth column. For each problem size, the elapsed time, the ratio, and the log ratio was made into a chart and visual graph as seen below. It is important to see how the R value has a trend around 3.5-4 and the K value trends around 2. Having the ratios placed around the same values reveals that the efficiency for the problem size remains fairly similar with only a slight increases for the larger problem sizes, but overall remaining consistent to discovering the big-Oh time complexity.



*Rounding the Data*

After getting an idea of the rough values made sure to use the System.nanoTime() to regenerate the data expressed in seconds. The chart and graph created from this data strongly resembles the previous graph even though it was a different run.





*Conclusion*

In the end of the experiment we can come to the conclusion that it is extremely unlikely to get the exact same results from the same code running. Every time I ran the code I discovered that the numbers would vary even though the process was the same. Though as the experiment went on I discovered that even if the data was not the exact same the ratio and the log ratio remained consistent with what the big-Oh time complexity. I saw that the main way to understand the efficiency is with the ratio and the log ratio because those are the means to tell of efficient the algorithm is. In conclusion, while analyzing the algorithm I discovered while repeating the experiment that the efficiency created precise time ration and log ratio proving the efficiency of the big-Oh time complexity.