Osama Kergaye
CS 2420
9/14/2016
Dr. Meyer

Assignment 03 Analysis Document

1. Who is your programming partner? Which of you submitted the source code of your program?

Samuel Bridge was my programming partner, and I (Osama Kergaye) submitted the code.

2. What did you learn from your partner? What did your partner learn from you?

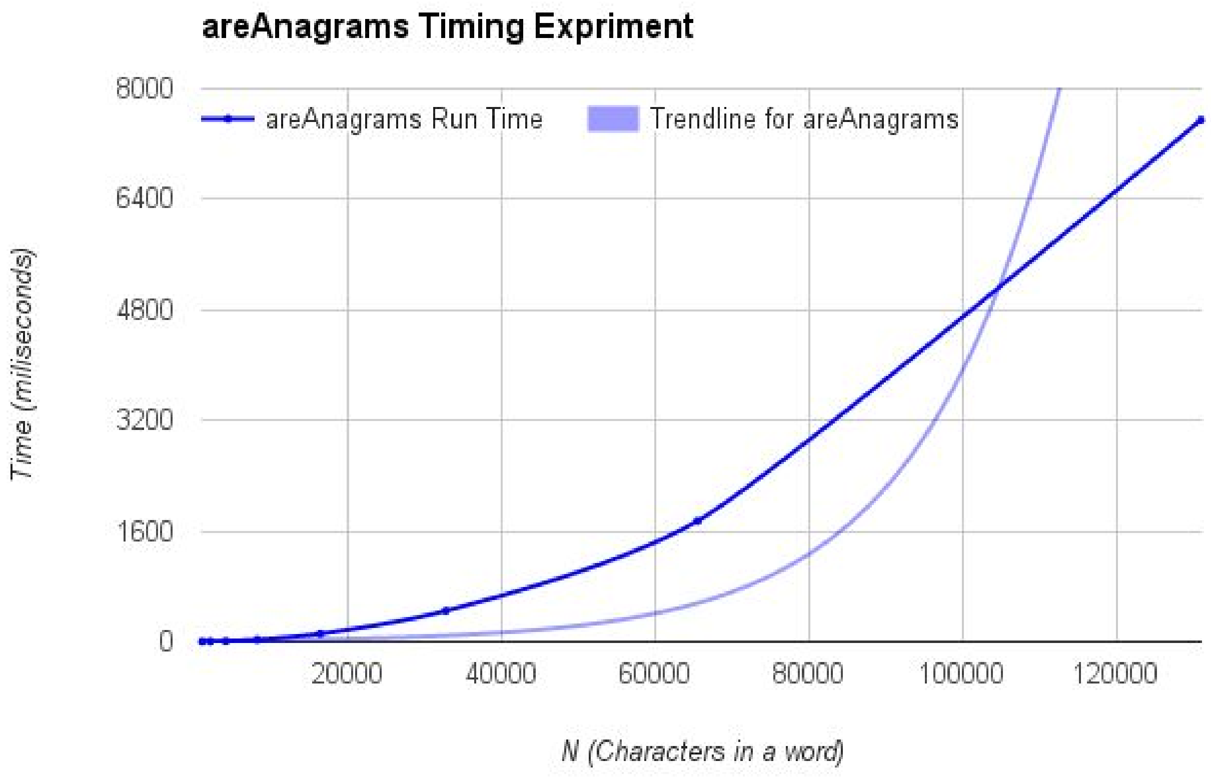
I learned better debugging and problem solving techniques from Samuel. He may have learned better Junit testing from me.

3. Evaluate your programming partner. Do you plan to work with this person again?

Samuel was a fantastic programming partner, and I do intend to program with him again.

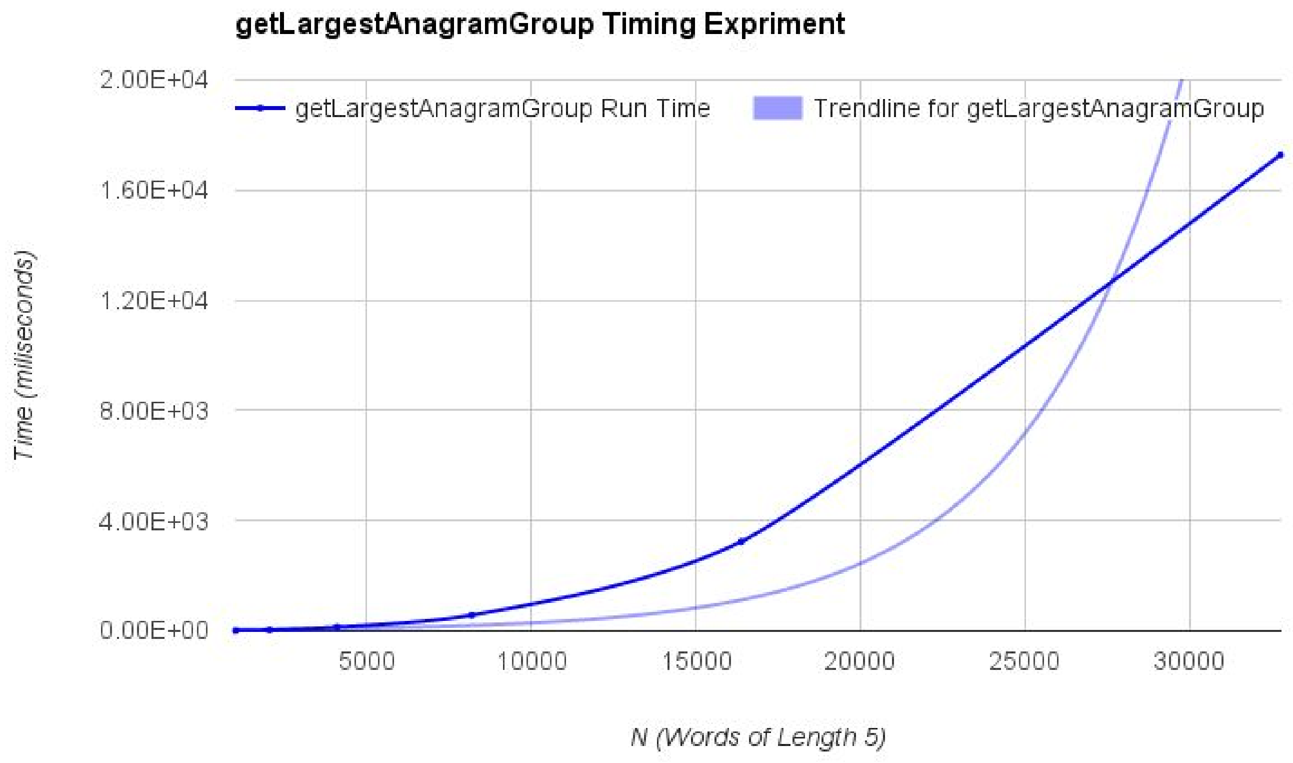
4. Analyze the run-time performance of the areAnagrams method.
-What is the Big-O behavior and why? Be sure to define N. -Plot the running time for various problem sizes (up to you to choose problem sizes that sufficiently analyze the problem). (NOTE: The provided AnagramTester.java contains a method for generating a random string of a certain length.) -Does the growth rate of the plotted running times match the Big-O behavior you predicted?

After analyzing areAnagrams the big O behavior was O(N^2), which was my guess. This was likely do to it calling insertion sort, because insertion sort has to fully iterate two nested loops and this creates an N times N resulting in N^2. N, in this case, is the length of the words to be compared. The growth rate does indeed match my predicted behavior.



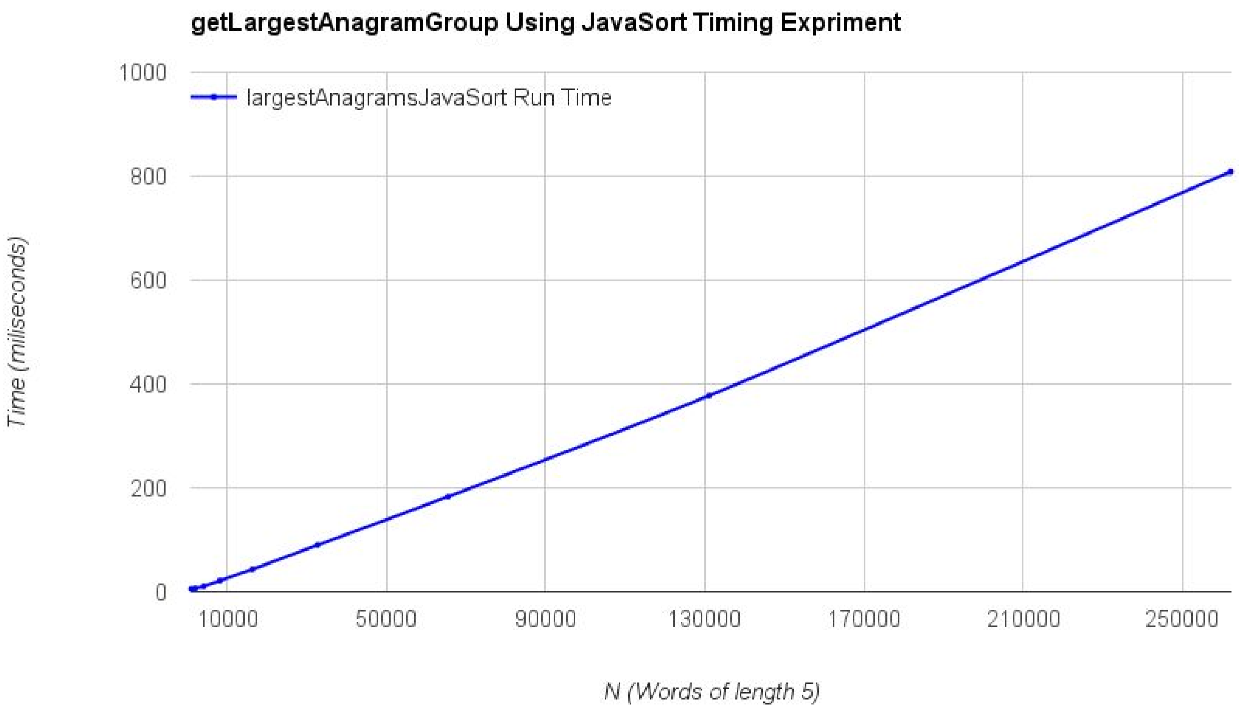
5.Analyze the run-time performance of the getLargestAnagramGroup method using your insertion sort algorithm. (Use the same list of guiding questions as in #4.) Note that in this case, N is the number of words, not the length of words. Finding the largest group of anagrams involves sorting the entire list of words based on some criteria (not the natural ordering). To get varying input size, consider using the very large list of words linked on the assignment page, save it as a file, and take out words as necessary to get different problem sizes, or use a random word generator, provided in AnagramTester.java. If you use the random word generator, use a modest word length, such as 5-15 characters.

After analyzing getLargestAnagramGroup the big O behavior was O(N^2), which was my guess. This was likely do to it calling insertion sort, because insertion sort has to fully iterate two nested loops and this creates an N times N resulting in N^2. N, in this case, is the amount of words to be compared, each having a length of 5 and are randomly generated. The growth rate does indeed match my predicted behavior.



6. What is the run-time performance of the getLargestAnagramGroup method if we use Java's sort method instead (http://docs.oracle.com/javase/6/docs/api/java/util/Arrays.html)? How does it compare to using insertion sort? (Use the same list of guiding questions as in #4.)

The runtime using Java's sort method is O(N log(N)), which is what I thought it might be because it had to be faster than what we programmed. This is because java invokes mergesort when accepting a comparator, meaning it uses a divide and conquer algorithm. This type of algorithm, we learned in class has a dividing portion that is O(log(N)) and an iterative portion that is O(N). This results in a runtime that is much faster than insertion sort. N in this case are words of length 5. The running time matches my guess.



7. How many hours did you spend on this assignment?

We spent about 25 hours on this assignment.