Assignment 03 Analysis

Rahul Ramkumar

September 21, 2016

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

   A: My programming partner was Lingxi Zhong, I submitted the source code to our program.

2. **Q:**What did you learn from your partner? What did your partner learn from you?

   A:Through my partner's troubleshooting of some of our timing code I came to better understand how generics should be used in Java. I think Lingxi learned some techniques for keeping things DRY with my attempt to refactor our timing code so we can reuse it for other assignments.

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

   A:Lingxi is a great motivated partner to work with who is good at using Java to help solve our issues when we have problems programming. I plan on working with him again.

4. **Q:**Analyze the run-time performance of the areAnagrams method.

   A: I believe that the behavior of our areAnagrams() method should have the same runtime performance as insertion sort, that is $O(n^2)$, where n is the length of the string given to our areAnagrams() method. It seems that the plotted growth rate agrees with this.

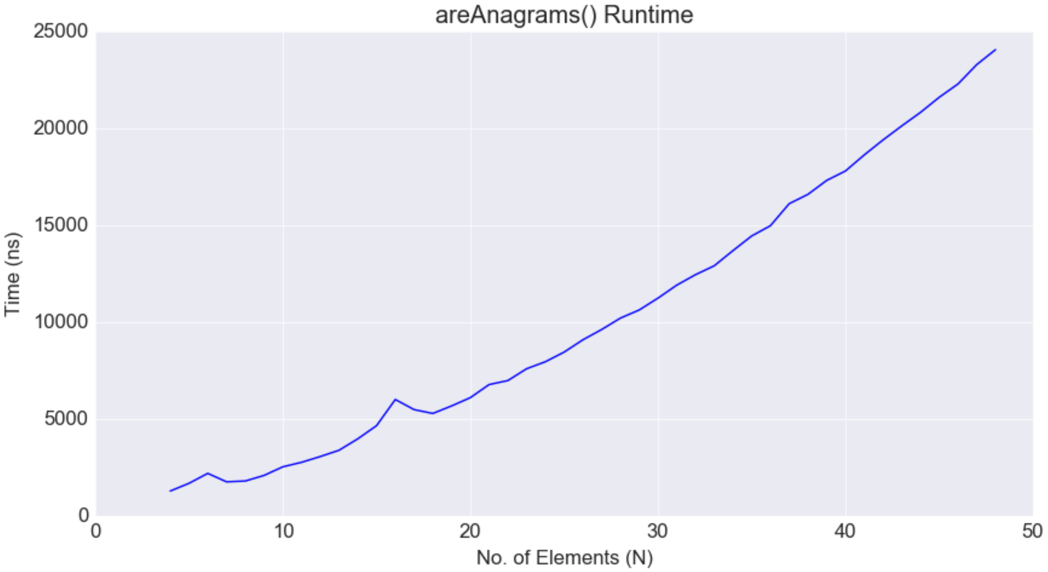

Figure 1: Plot of areAnagrams() method runtime.

5. **Q:**Analyze the run-time performance of the getLargestAnagramGroup method using your insertion sort algorithm.

   A: It seems that with our Insertion sort algorithm the getLargestAnagramGroup() method should have a runtime of $O(n^3)$ because the for every set of size n items you need to n number of $O(n^2)$ insertion sorts and $n \times n^2 = n^3$.

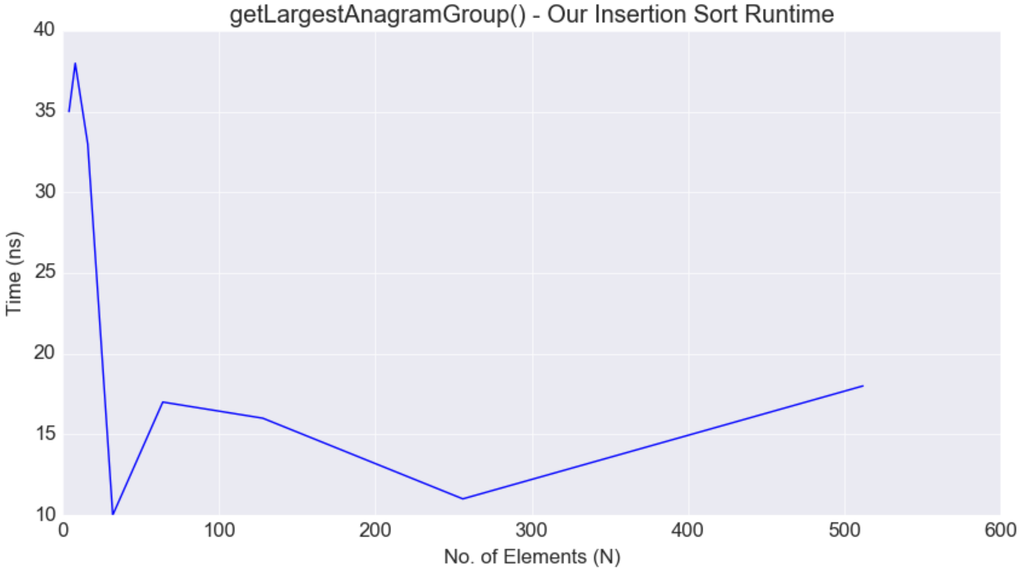

Figure 2: Plot of getLargestAnagramGroup() method with our Insertion sort runtime.

6. **Q:**What is the run-time performance of the getLargestAnagramGroup method if we use Java's sort method instead?

   A: Because Java's method uses a quicksort with a runtime performance of $O(n \log n)$, for every set of size n items you will need to do n number of $O(n \log n)$ which will result in a time complexity of $O(n^2 \log n)$ which should be quicker than our implementation with insertion sort as $\log n < n$ so $O(n^2 \log n) < O(n^3)$.

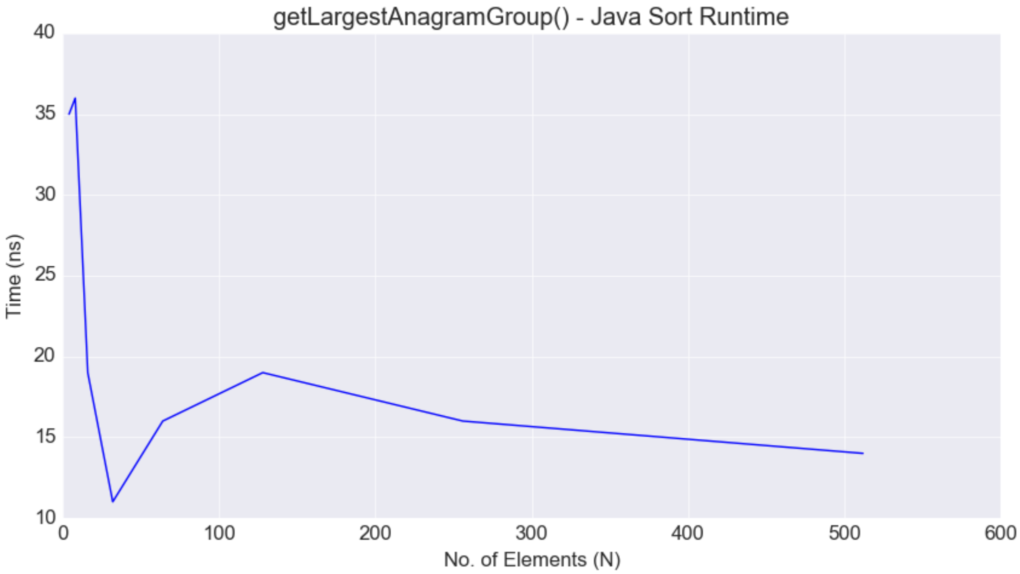

Figure 3: Plot of getLargestAnagramGroup() method with Java's built-in sort runtime.

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

   A: About 4-5 hours.