Jared Nielson
CS 2420
Analysis Assignment 04

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

Christian Hansen was my programming partner. I submitted the source code.

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

One thing I learned from Christian is that it is ok, often times better, to use a simple solution rather than a complex one that might result in less code. One of our methods was failing a corner case test. I thought the best thing would be to rewrite the block of code that was causing the issue from the ground up. He said the best thing to do would be to use an if statement to handle that one particular condition. His approach was definitely the better of the two.

Although unrelated to the assignment Christian and I discussed the possibility of using github to manage our code base for our next pair programming assignment. I was vaguely familiar with version control systems and not very familiar with git at all. He gave me a really good explanation that helped explain things (something that several videos that I had watched previously couldn't do).

I was able to teach Christian a little bit about input streams in Java. This came up when we were reviewing the provided static method for reading a file.

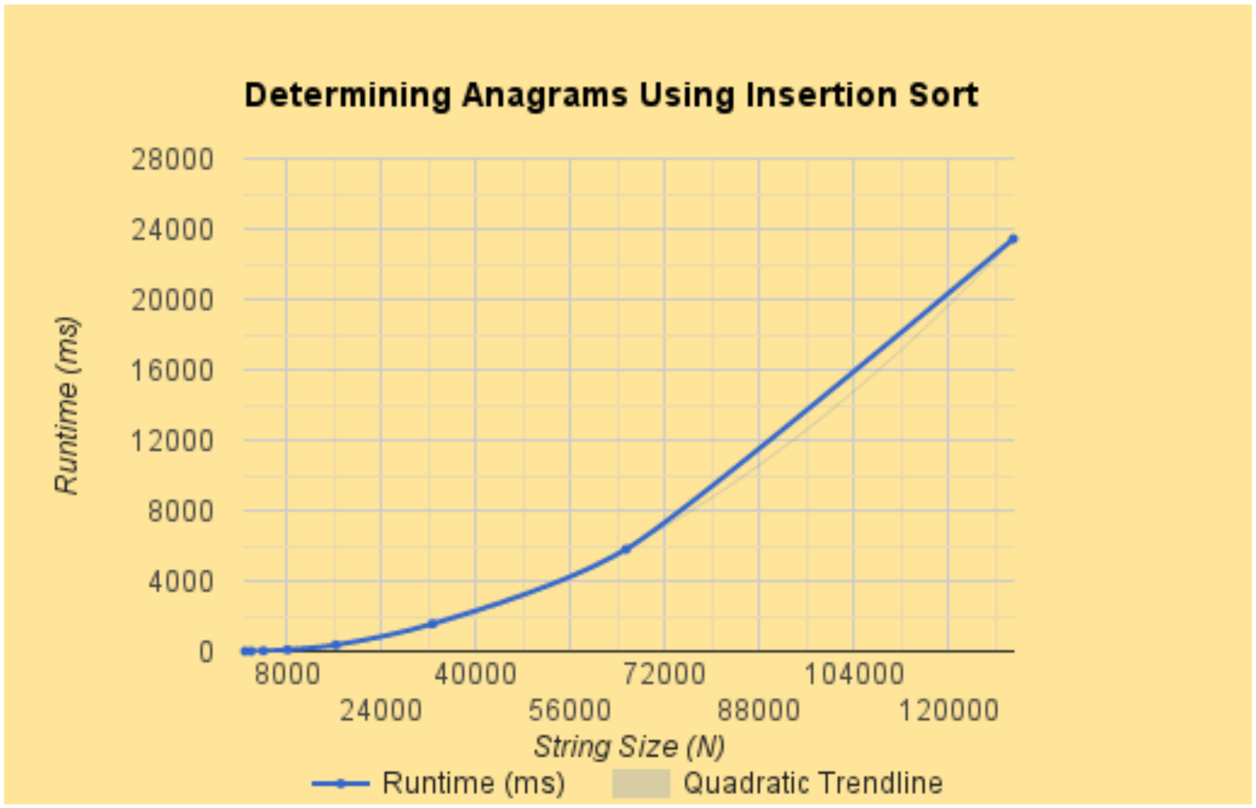
3. Evaluate your programming partner. Do you plan to work with this person again?

Christian is a great partner. He is smart, hard working and humble which, in my opinion, is the perfect combination for any programming partner. I'm looking forward to working with him on Assignment 05.

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. -Does the growth rate of the plotted running times match the Big-O behavior you predicted?

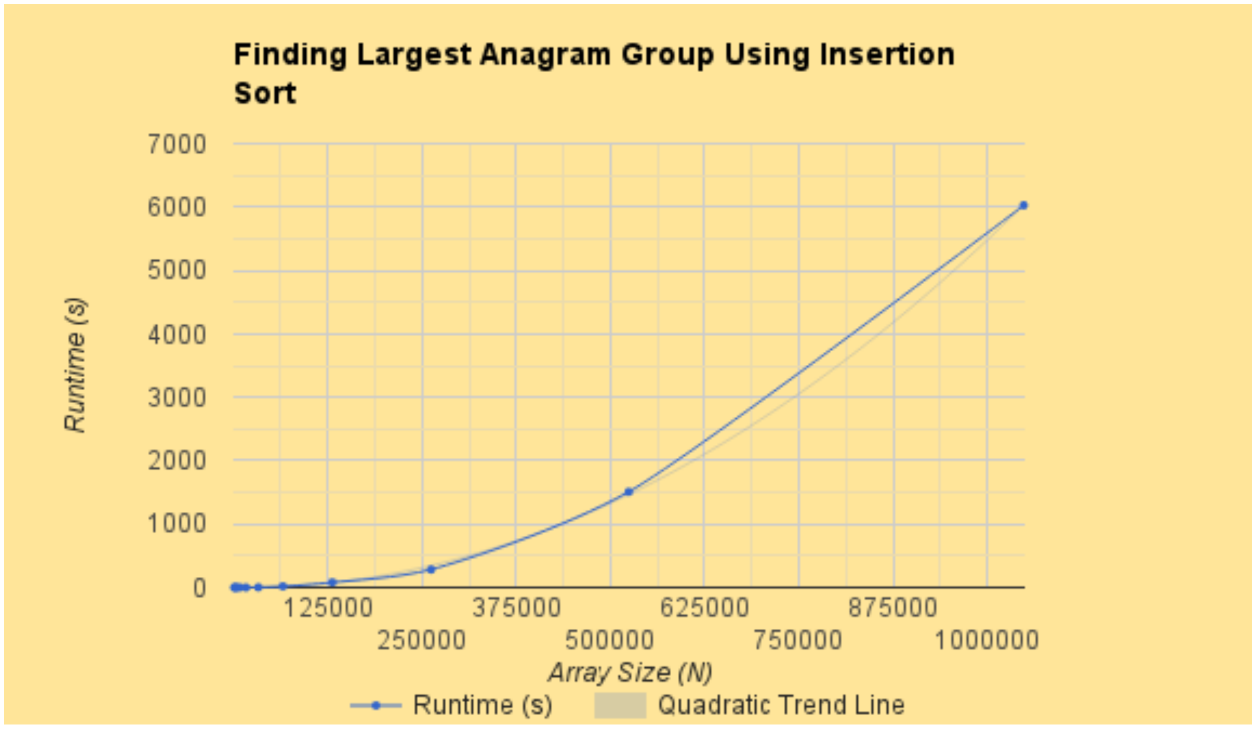
The Big-O behavior of the areAnagrams method is O(N^2) where N is the length of the strings passed in to the areAnagrams method (assuming they have equal length). The reason for this is that areAnagrams performs an insertion sort on each string and then calls the strings equals method to determine if they are the same. Insertion sort has a Big-O complexity of O(N^2) while the equals method has a linear complexity. The asymptotic behavior of the areAnagrams method is determined by the insertion sort and so it has a complexity of O(N^2).

A plot of the runtime test is listed below. On the plot is a quadratic regression that matches nicely with the values from the test indicating that the Big-O behavior of areAnagrams is indeed O(N^2).



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

The run-time performance of the getLargestAnagramGroup method is O(N^2) with N being the size of the array of strings. The reason for this is again the method uses an insertion sort on the array of strings which has an O(N^2) complexity, this complexity determines the asymptotic behavior of the getLargestAnagramGroup. Below is a plot of the running time for the getLargestAnagramGroup method with various array sizes. With each doubled sample size the run time quadruples (approximately) indicating N^2 behavior. A quadratic regression is plotted behind the runtime curve that matches nicely providing further evidence of quadratic complexity.



6. What is the run-time performance of the getLargestAnagramGroup method if we use Java's sort method instead? How does it compare to using insertion sort? (Use the same list of guiding questions as in #4.)

Java 8's sort method uses a variation on merge sort that has NlogN complexity for a randomly sorted array. This means that using Java 8's sort method in the getLargestAnagramGroup method would greatly improve the runtime efficiency. Instead of having a Big-O complexity of O(N^2) it would have a Big-O complexity of O(NlogN).

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

We spent approximately 8 hours on this assignment not including running our timing tests.