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

My programming partner was Andy Dao. Andy will be submitting the source code.

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

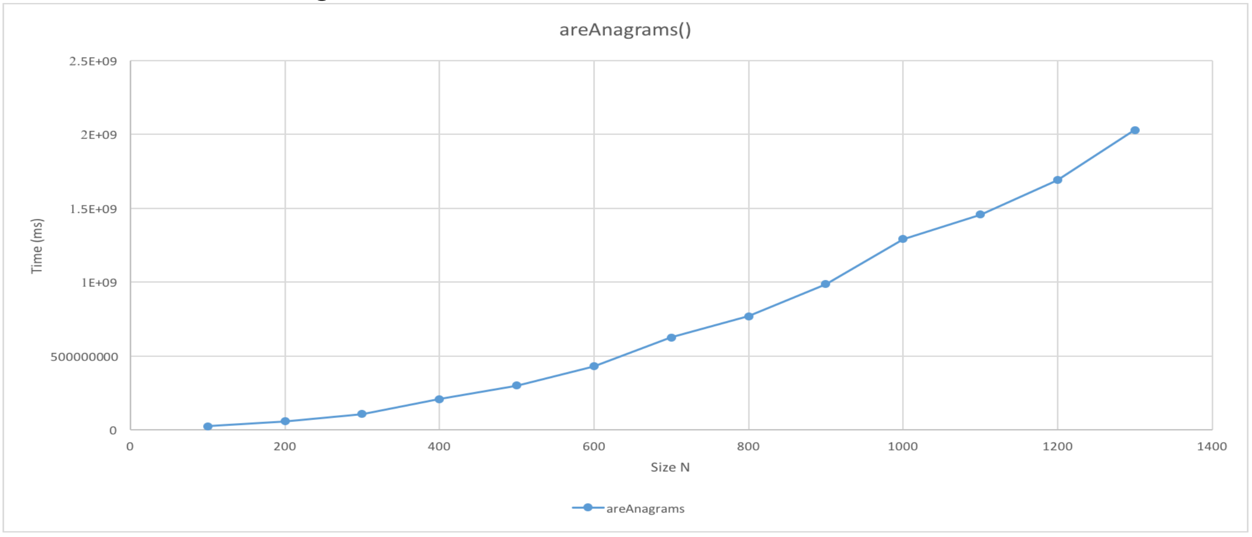
Andy definitely made me more cautious and more detail oriented about checking my code. Before I thought our areAnagrams method was O(N^2) but after his persistent caution I realized that It's always good to have a partner peer review you're assignment and double check your work.

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

Andy's a cool guy. Very reliable. I like working with him as a partner.

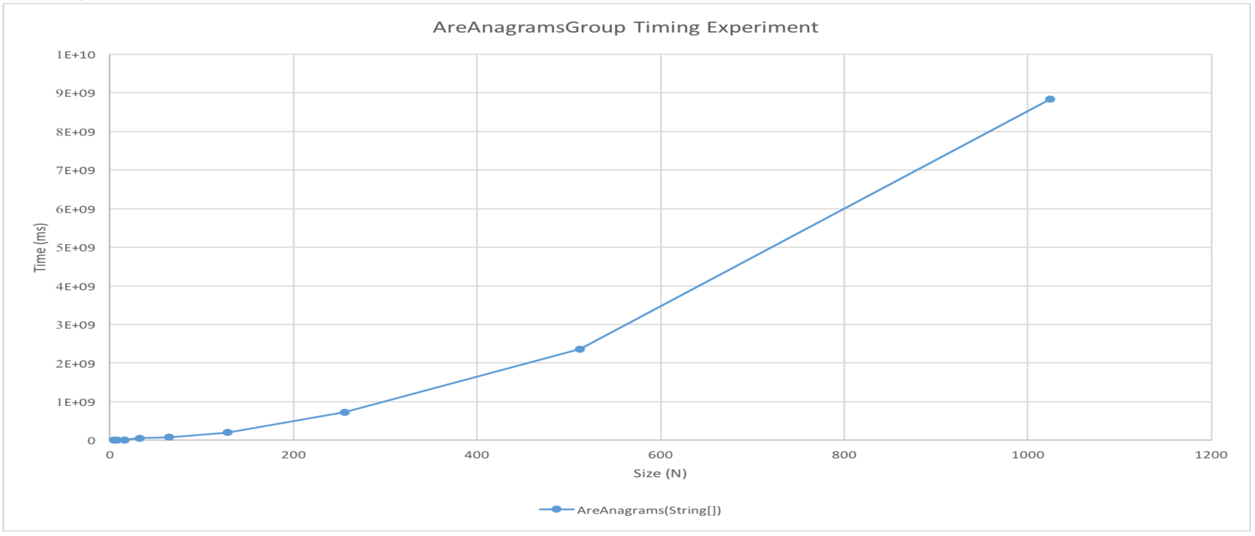
4. Analyze the run-time performance of the areAnagrams method.
-What is the Big-O behavior and why? Be sure to define N. -Plot the running time for various problem sizes (up to you to choose problem sizes that sufficiently analyze the problem). (NOTE: The provided AnagramTester.java contains a method 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?

The Big-O behavior of our areAnagrams method according to the timing methods had a Big-O notation of N. This was the predicted N behavior of our method for the areAnagram. This may be because the way our method looks through N elements only once comparing the left to the right throughout the array. Our code has two iterations throughout the sort method N made us predict N^2. But the timing concludes for a larger size N we will get a more detailed N behavior for our areAnagrams method.



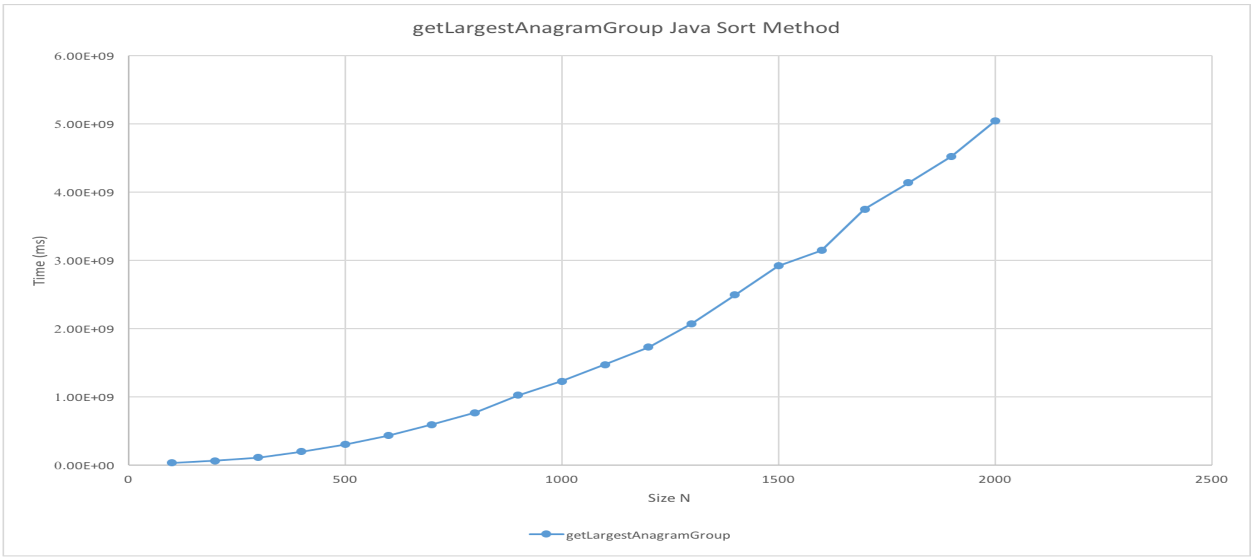
5.Analyze the run-time performance of the getLargestAnagramGroup method using your insertion sort algorithm. (Use the same list of guiding questions as in #4.) Note that in this case, N is the number of words, not the length of words. Finding the largest group of anagrams involves sorting the entire 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.

The Big-O behavior expected of our getLargestAnagramGroup method was predicted to be N^2. This prediction was very similar to our plot times listed in our getLargestAnagramGroup. Largely the two loops implemented in order to basically search through N elements.



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 for Java's Sort Method compared to our insertion sort method was greater with O(NlogN) performance. This is due to the fact that since Java 6 java has utilized a mergesort algorithm to complete their sort method.



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

I spent about 15 hours on this assignment.