Axel Kerksiek
CS 2420
U0691509
09/20/2016

Assignment 4: Anagrams

Analysis Document

Question 1: Who is your programming partner? Which of you submitted the source code of your program?
Answer 1: My programming partner was Nathan Milot. He submitted the code.

Question 2: What did you learn from your partner? What did your partner learn from you?
Answer 2: Every time I work with Nathan I learn a lot about syntax in object oriented programing. He really knows how to code well and knows how to code correctly. As for what he learned from me, I'm not sure. However, I do think that we did a good job of bouncing ideas off of each other and solving the problem together.

Question 3: Evaluate your programming partner. Do you plan to work with this person again?
Answer 3: Nathan is a great partner. His schedule and mine match up pretty well so we always have item to work on the project. We get the work done well before the deadline so that we enough time to test and debug. I would definitely work with Nathan again. We are planning on working on the next project together.

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

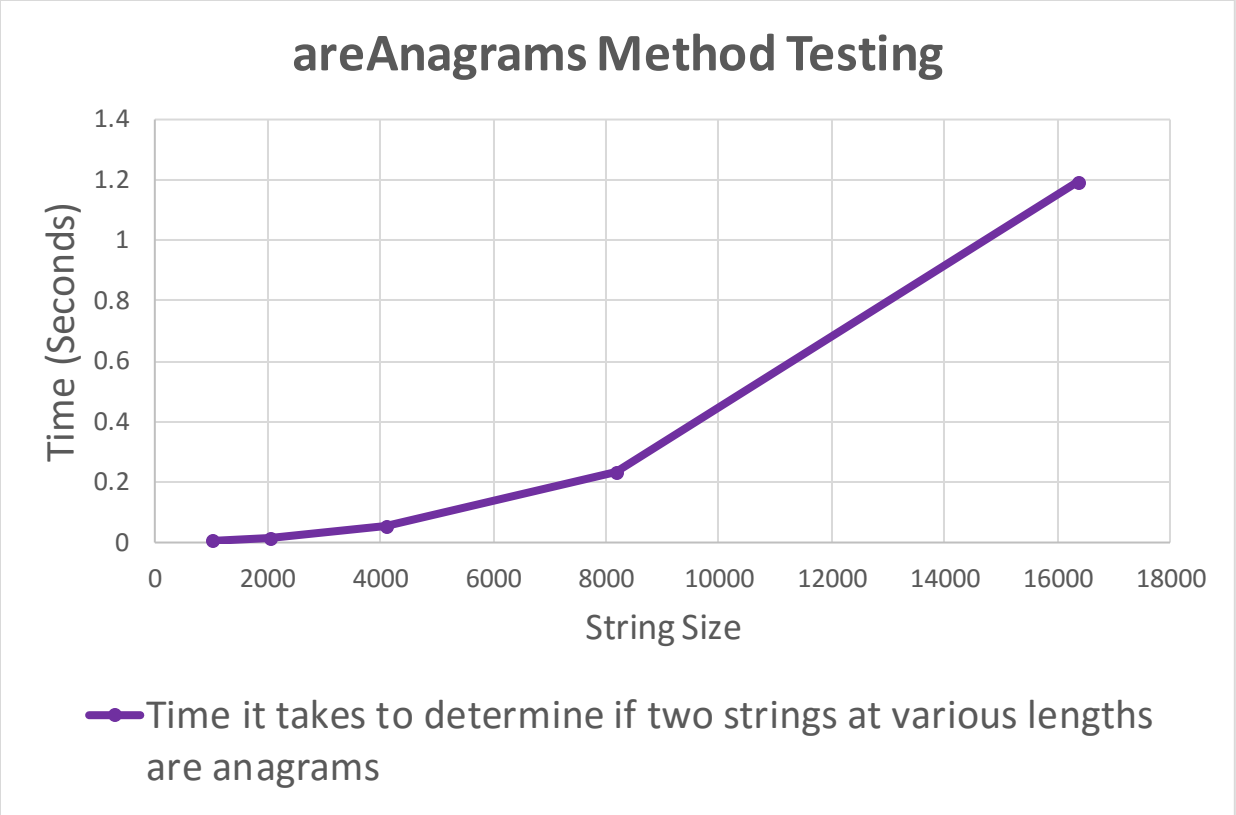
-What is the Big-O behavior and why? Be sure to define N.

-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?

Answer 4:

- The Big-O behavior of areAnagrams is O(N^2). The worst and average case for insertion sort is O(N^2) and the best case is O(N). The reason why the areAnagrams method is O(N^2) is because we are testing strings anywhere from 2^10 to 2^14 which is around 1,000 – 16,000 characters long. Because these strings are not sorted and they are not small N, it falls into average and worst case scenario of O(N^2).

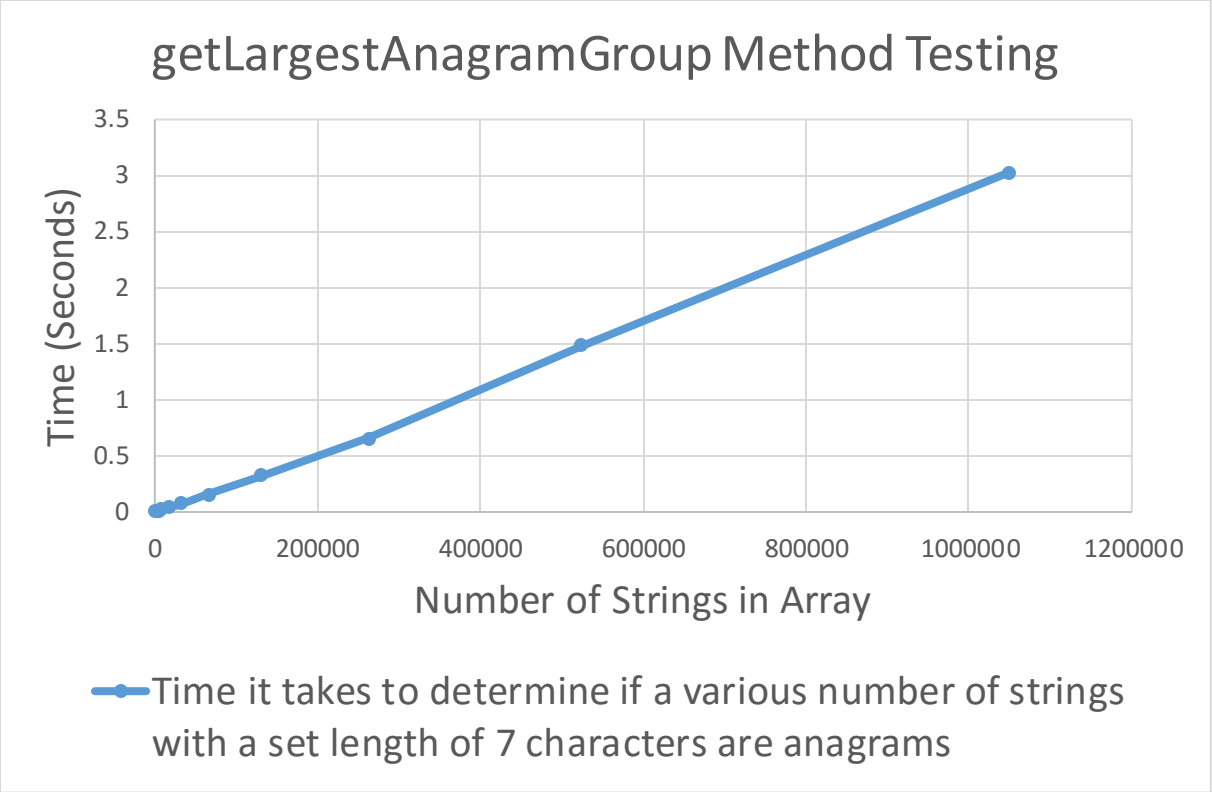


- Yes, the growth rate is what we predicted it would be. It is exponential.

Question 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.

Answer 5:

- The Big-O behavior of the getLargestAnagramGroup method is O(N). The worst and average case for insertion sort is O(N^2) and the best case is O(N). The reason why the getLargestAnagramGroup method is O(N) is because we are testing strings with a pretty small size N of 7 characters. Because these strings are small N the insertion Sort falls into the best case scenario which is O(N). The number of words we are comparing only gives the complexity a larger constant number in front however that will not change the general slope of the Big-O behavior.

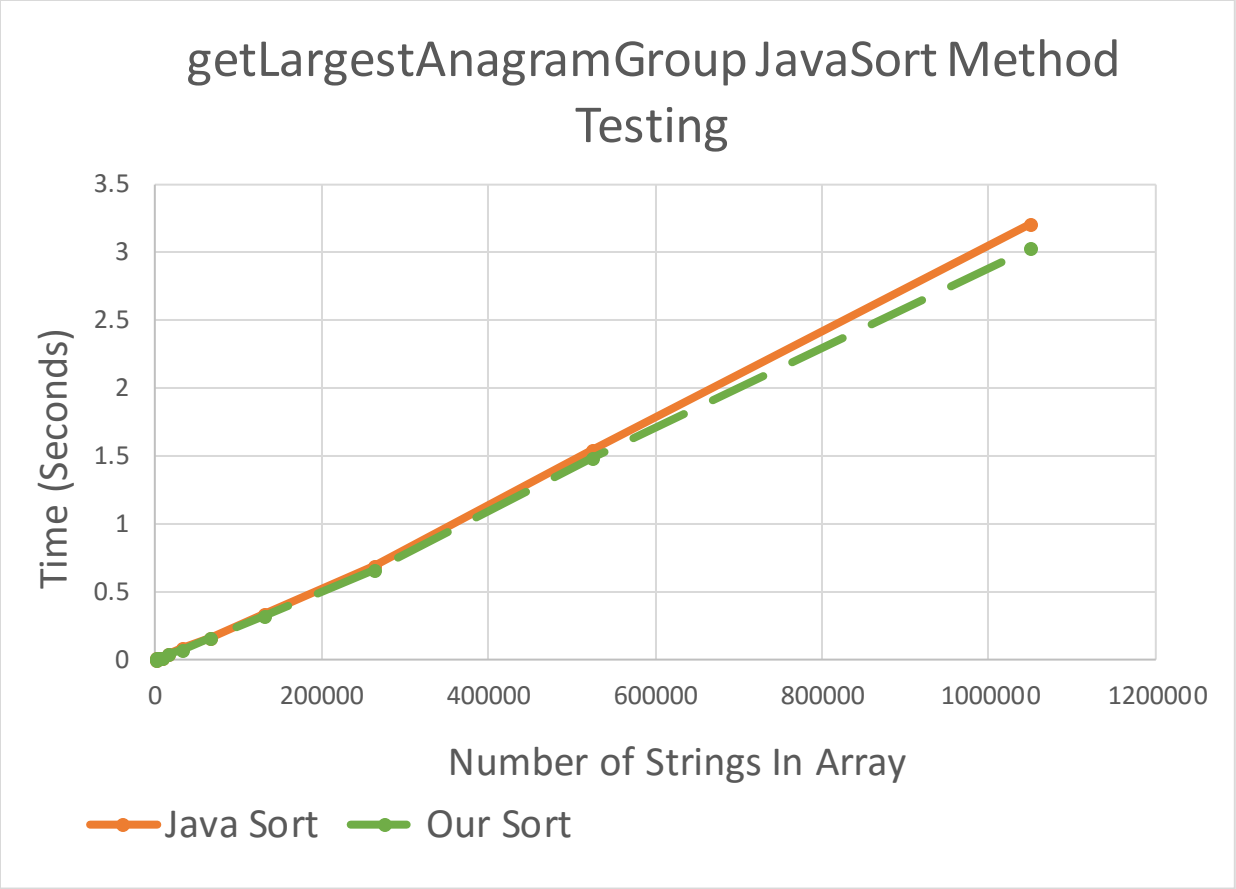


- Yes, the growth rate is what we predicted it would be. It is linear.

Question 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.)

Answer 6:

- The Big-O behavior of the getLargestAnagramGroup method using Java's sort is O(N). The worst and average case for insertion sort is O(N^2) and the best case is O(N). It is best case scenario because out string size is 7 characters long which is a pretty small N size. We noticed that our insertion sort is a little faster than Java's sort. As the number of strings in the array increases, our method seems to outperform Java's sort.



- Yes, the growth rate is what we predicted it would be. It is linear. The data also supports our conclusion that our insertion sort is better than Javas!

Question 7: How many hours did you spend on this assignment?
Answer 7: Around 10 hours.