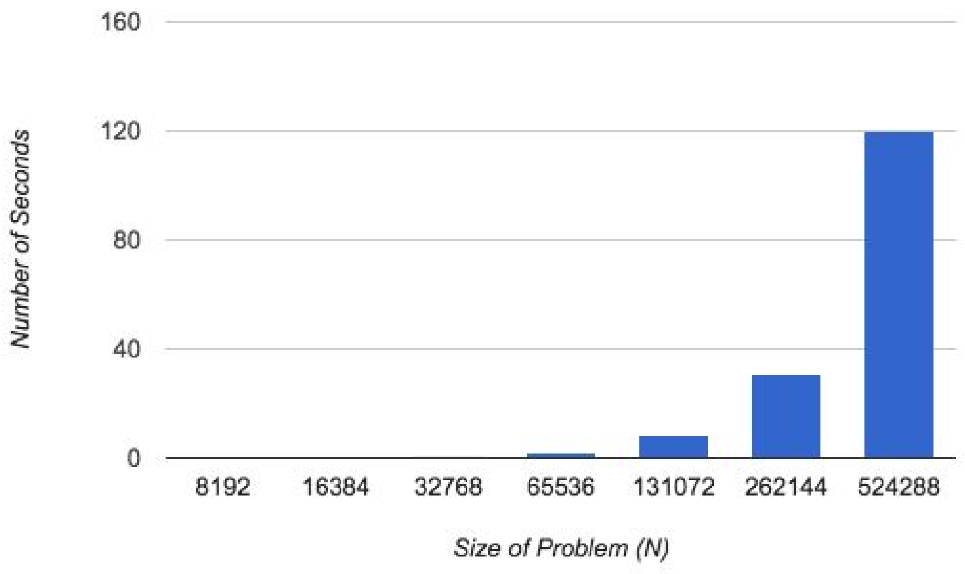
Nicholas Kerr
CS 2420
assignment04

1. Who is your programming partner? Which of you submitted the source code of your program?
My programming partner is Andrew Worley, u0651238. He submitted the code.

2. What did you learn from your partner? What did your partner learn from you?
I learned a bunch from my partner, he was the stronger coder for sure. He was very helpful and good at explaining things when I had questions. I think I may have been better at analysis and how long to expect things to take. Overall I'm glad we were partners.

3. Evaluate your programming partner. Do you plan to work with this person again?
He was good. I enjoyed working with him and he was patient with me. I would work with him again, but I can also see value in working with other people.

4. Analyze the run-time performance of the areAnagrams method.
-What is the Big-O behavior and why? Be sure to define N.
For the areAnagrams method the problem size (referred to as N) is the number of characters in the strings being compared. After pruning the behavior to asymptotic behavior, areAnagrams is O(N^2). The majority of the complexity for this method comes from comparing each item in a character array to each other item in the array. Because the insertionSort compares each element to the whole list it matches normal insertionSort complexity. Comparing all the elements of an array to each element in an array means looking at each item N^2 times. The rest of the operations are either singular or less than N, which all drops off as limits are approached in asymptotic behavior.
-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.)



Nicholas Kerr
CS 2420
assignment04

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

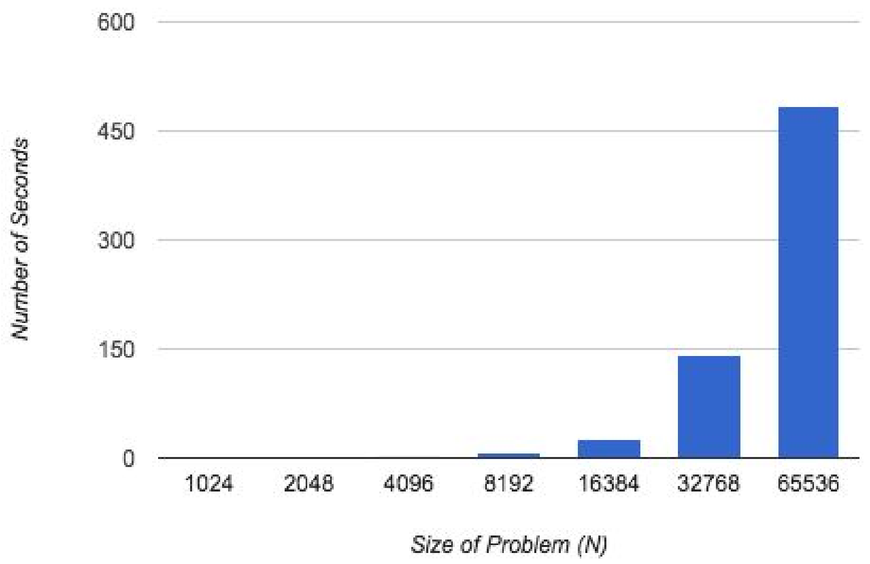
Yes it does. As the size of the problem doubles, the amount of time required quadruples. If O(N^2) then doubling the problem means (2N). Then (2N)^2 becomes 4N^2, which shows that as the size of N doubles the complexity quadruples. Because the complexity has quadrupled so has the time.

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.

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

For the getLargestAnagramGroup method the problem size (referred to as N) is the number of “words” in the array being searched. There are operations that happen once, or less then N times, as well as a while loop inside of a while loop. Because nested loops using insertionSort compare each element to each other element in the array there are N^2 operations in the insertionSort. Asymptotic behavior allows us to ignore the other aspects of complexity compared to N^2, which makes the complexity O(N^2).

-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.)

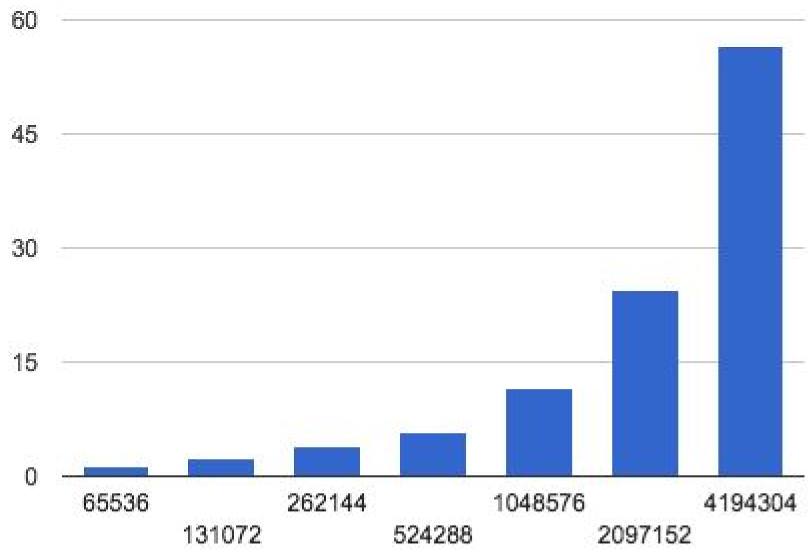


Nicholas Kerr
CS 2420
assignment04

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

Yes it does. For N^2 complexity we would expect the time to quadruple as the problem size doubles, which is what the graph shows based on the data from the simulation.

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.)



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

The complexity seems to be N Log(N), it more then doubles each time. Separately, the search function seems to be much faster than our implementation. It did 4 million records in just a little more time then our insertionSort at 16K records.

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

At least 20, learning analysis took some time. I hope it takes less time in the future, as I get better at identifying tests and how to graph them.

Programming partners are encouraged to collaborate on the answers to these questions. However, each partner must write and submit his/her own solutions.

Upload your solution (in .pdf format only) to the assignment 4 page by 12 pm on September 21.