Dan Ruley

u0956834

CS 2420

2/8/2019

Abstract:

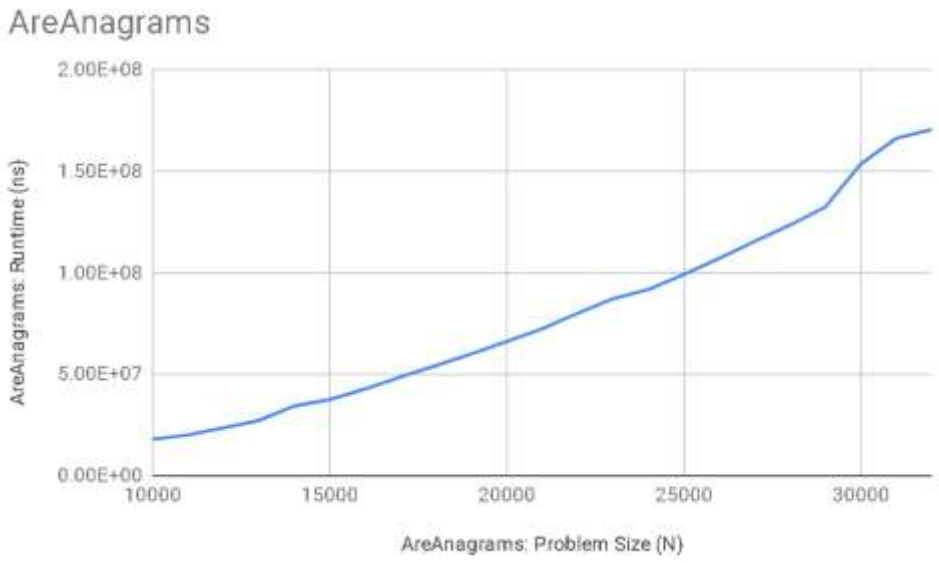
This document goes over the timing experiments and analysis of our AnagramChecker class. Overall, my partner and I worked well together and I would work with him in the future. Our program implements two static insertion sort methods: one that sorts a String lexicographically, and one that sorts a generic [T] array with an input comparator. The String sort method returns a String, because Strings are immutable so we must create a new String from the sorted version of the input, and return that. The [T] sort is a void method, because arrays are not immutable, and may directly modify the input T array. This is faster than making a copy, sorting it, and returning it since that incurs overhead.

Overall, our results were rather convincing that both our areAnagrams and getLargestAnagramGroup methods exhibit quadratic behavior.

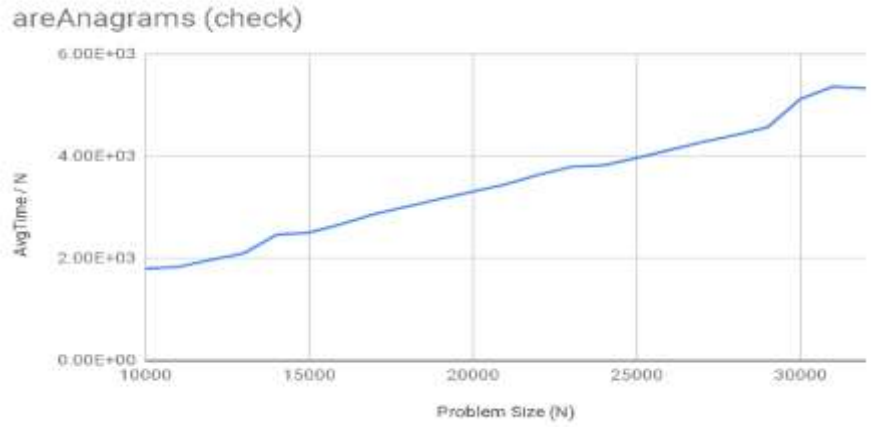
Experiment One (areAnagrams):

Timing breakdown: Twenty-two iterations ranging from N = 10,000, to N = 32,000. The inputs were two Strings built of randomly generated characters, of size N. We had the areAnagrams method loop 1000 times, and averaged the time across those 1000 iterations. We also subtracted overhead from the for loop, and variable assignments from the final time.

Overall, our data looks quite consistent with N^2 behavior. The plot of areAnagrams runtime shows a nice quadratic curve:



A check analysis also confirms the O(N^2) behavior for areAnagrams:

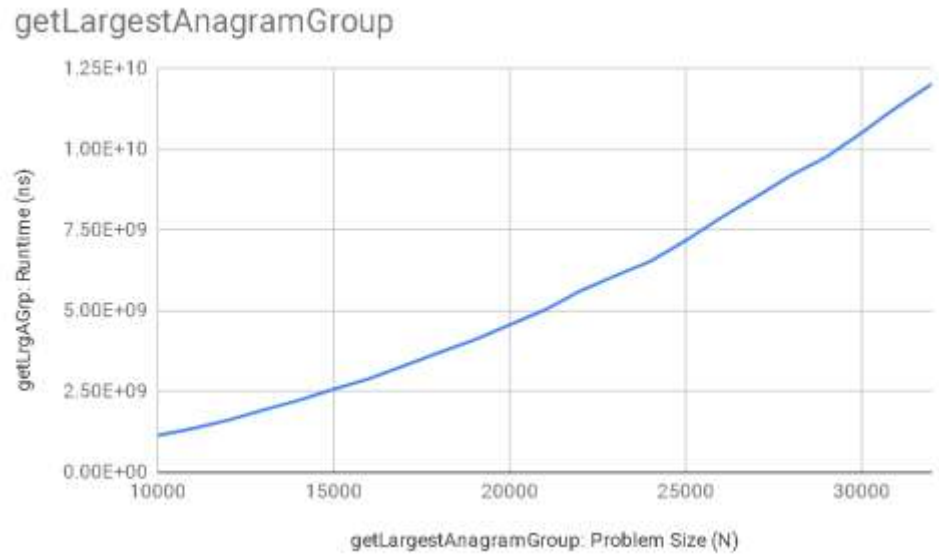


Clearly, the O(N) check values are steadily increasing, meaning O(N) is an underestimate of the O(N) behavior. O(N^3) check values are converging to zero, showing that it is an overestimate. O(N^2) are converging to a positive number, so it's the clear match here.

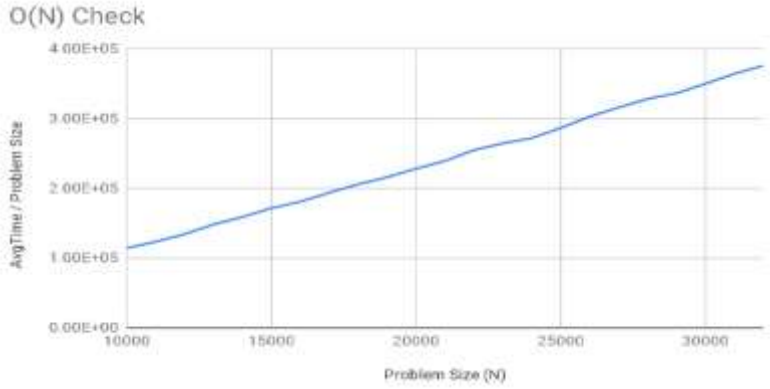
Experiment Two (getLargestAnagramGroup):

Timing breakdown: Twenty-two iterations ranging from N = 10,000, to N = 32,000. The inputs were an array of random Strings, of size N. We had the getLargestAnagramGroup method loop 1000 times, and averaged the time across those 1000 iterations (using a temp copy of the master array each time to be sure we were not sorting an already sorted array). We also subtracted overhead from the for loop, copying the array, and variable assignments from the final time.

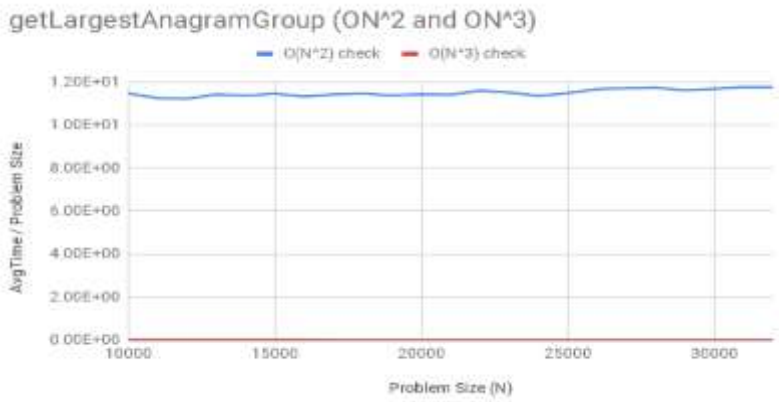
Overall, our data looks quite consistent with O(N^2) behavior. The plot of getLargestAnagram runtime shows a nice quadratic curve:



A check analysis also confirms an O(N^2) runtime for getLargestAnagramGroup:



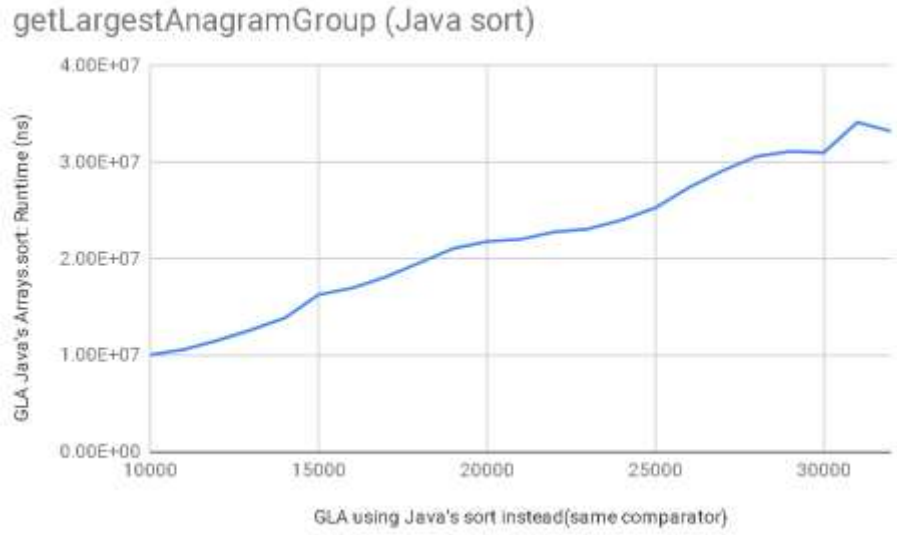
Clearly, the O(N) check values are steadily increasing, meaning O(N) is an underestimate of the O(N) behavior. O(N^3) check values are converging to zero, showing that it is an overestimate. O(N^2) are converging to a positive number, so it's the clear match here.



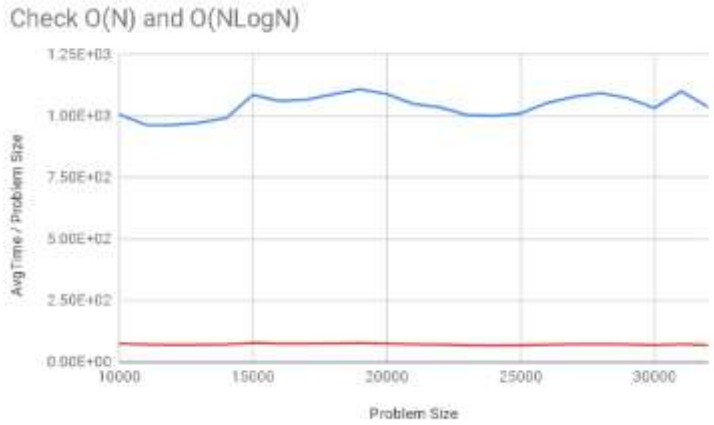
Experiment Three (getLargestAnagramGroup with Java's Arrays.sort method):

Timing breakdown: Timing protocol exactly the same as Experiment two.

The difference here is that we changed the sort in getLargestAnagramGroup from our insertion sort to Java's Arrays.sort. Java uses a mergesort algorithm in its sort method, which has an O(NlogN) time. Our data ended up looking more like O(N), probably because we had to use a relatively modest N due to hardware limitations and it taking quite long to time very large N values. So, the LogN behavior would not be very evident as Log base 2 of ~10000-30000 is still relatively modest.



A check analysis shows:



We could argue this is evidence of convergence for both cases. But as NLogN is very close to zero, it's likely this is an overestimate *for these relatively small values of N.* If much larger values of N were to be tested, likely O(NLogN) behavior would start dominating.

Conclusion:

Overall, results of our timing experiments were consistent with our expectations. areAnagrams and getLargestAnagramGroup exhibited lovely quadratic behavior, as expected for insertion sort. Using Java's mergesort instead, we saw somewhat O(N) behavior, but if we had to processing speed to do tests on much larger values of N, likely it would exhibit O(NLogN behavior).