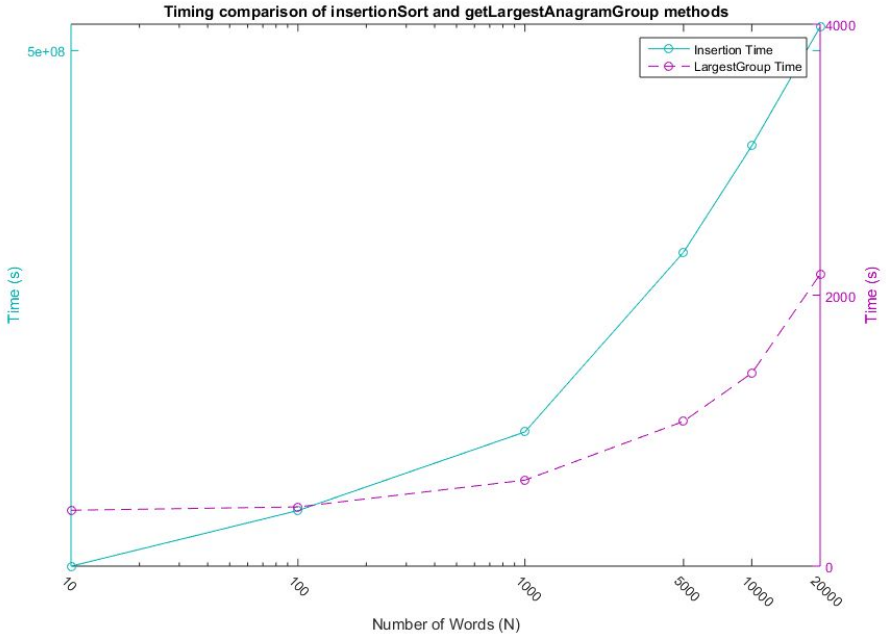
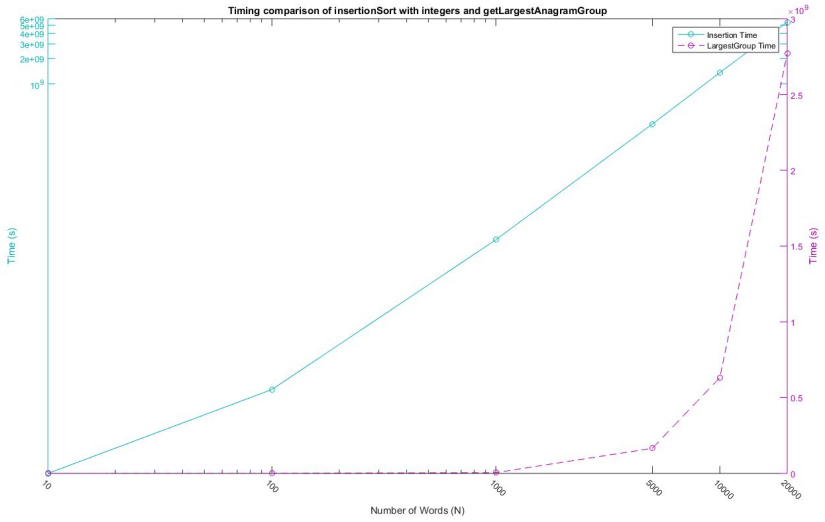
Tony Diep
CS 2420
Assignment 4 Anagrams
2/9/17

The value, N, represents the number of words in an array; they were randomly generated prior to being searched for anagrams and anagram groups. The N values tested range from ten to two-thousand words in an array, and each test was run one-hundred times to obtain an accurate average value. When looking at the plot below, one can see that the time it took the insertion sort to process its task is displayed as a Big-O behavior of Θ(N^2). This was determined based on the somewhat steady uprising of the data set that resemble a quadratic in trend. The greatest anagram group, other the other hand, displayed a Big-O behavior of Θ(N*log(N)). In order to see this, it should be noted that the y-axis associated with the insertion sort time is represented in a semi-log fashion. It was initially predicted that the insertion sort would follow a linear or quadratic trend, as it was found to be. I did not predict the N*log(N) efficiency of the greatestAnagramGroup, however; this was very impressive.

The second plot compares the run-time performance of insertionSort using an array of strings and an array of integers. The case with integers is represented in the legend as the insertion time. The data suggests that as the number of integers in an array progresses the time it takes to process the task progresses in a linear manner. The complexity is therefore found to be Θ(N). This is quicker than the processing time of an array of string. This is due to the fact that integers take up less space in memory than a string which is composed of individual chars. When comparing plot one, it insertionSort was faster to process for smaller sized arrays, and then decreased in efficiency when handling the larger arrays; the outcome with the large arrays was similar to that of the getLargestAnagramGroup.

Observing the final plot, one will see a comparison between the insertion sort that we wrote and the sort that Java uses. Both of which are processing string arrays. It would seem that our sort is quicker at processing some of the smaller array sizes, while Java's sort is more efficient handling the larger arrays. Both decrease in efficiency when handling the 10,000 and 20,000 sized array. We determined that our Big-O behavior is that of Θ(N), while Java's may be Θ(N^2). It was expected that Java's sort would be more efficient for the majority of tasks, but ours was similar in efficiency with smaller arrays.


Timing comparison of insertionSort and getLargestAnagramGroup methods
Insertion Time
LargestGroup Time
Time (s)
Time (s)
5e+08
4000
2000
0
10
100
1000
5000
10000
20000
Number of Words (N)


Timing comparison of insertionSort with integers and getLargestAnagramGroup
Insertion Time
LargestGroup Time
Time (s)
Time (s)
Number of Words (N)