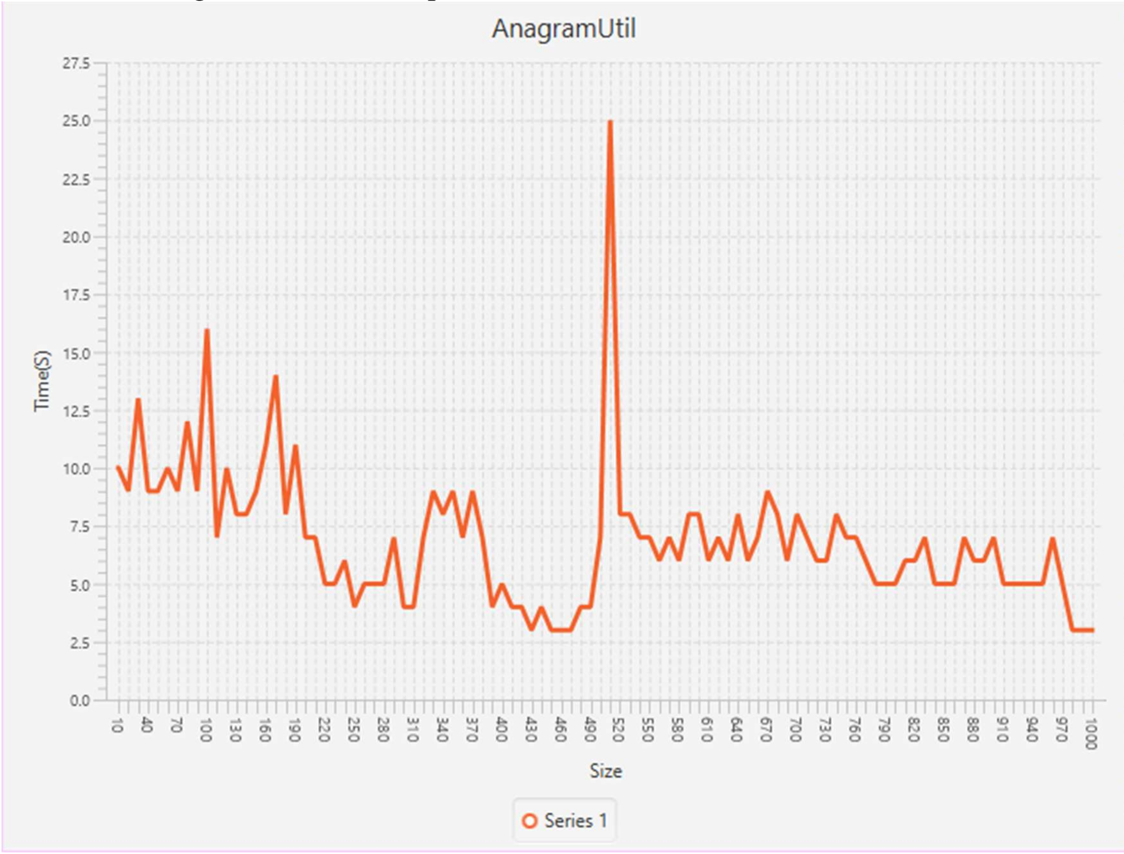
Nickolas Komarnitsky
U0717854
02/06/2017
2402
Assignment04

What is the Big-O behavior and why? Be sure to define N.
Plot the running time for various problem sizes

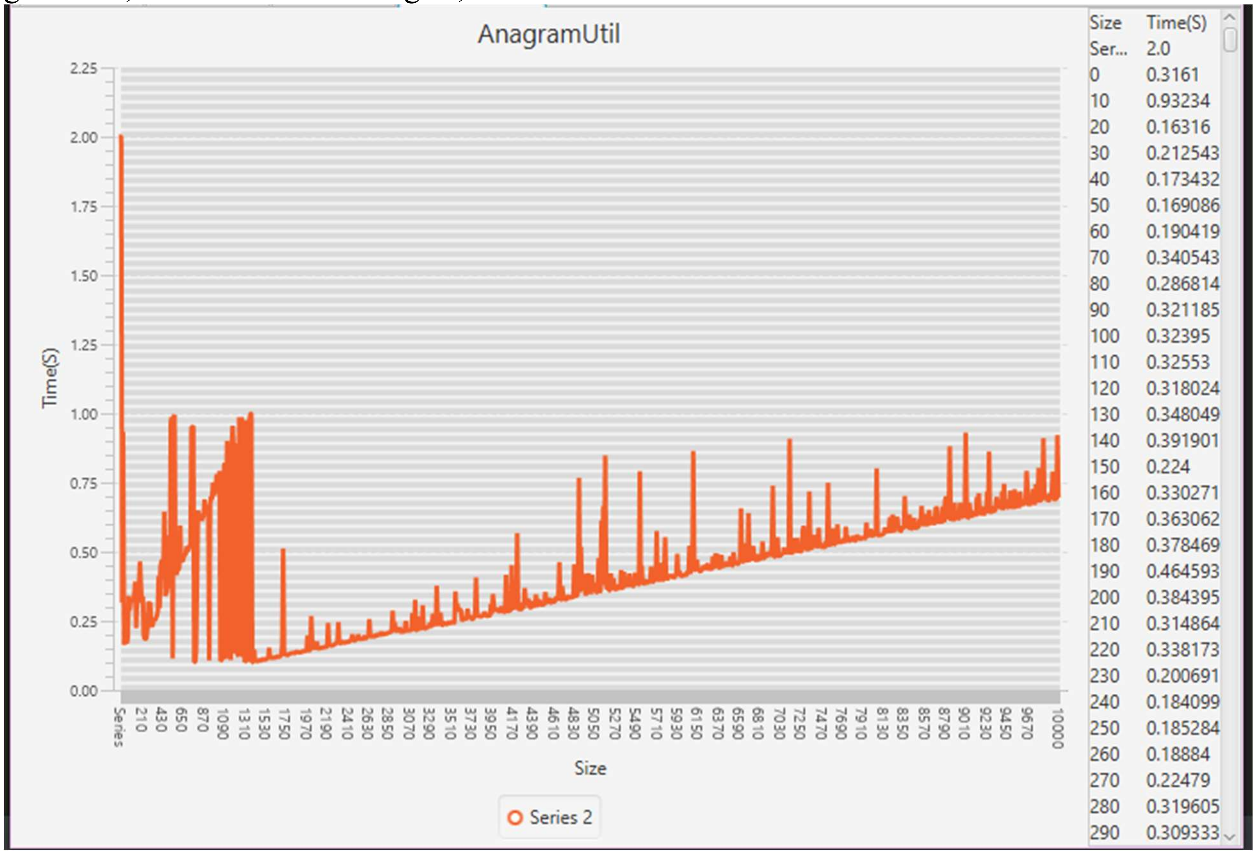


Si...	Tim...
10	10
20	9
30	13
40	9
50	9
60	10
70	9
80	12
90	9
100	16
110	7
120	10
130	8
140	8
150	9
160	11
170	14
180	8
190	11
200	7
210	7
220	5
230	5
240	6
250	4
260	5
270	5
280	5
290	7
300	4
310	4

The Big-O behavior is O(N), the run time increases as more and more operations are required. N is the length of the strings. The time required to check the two strings is increased because of the length of the string, but it is almost negligible in the long run.

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

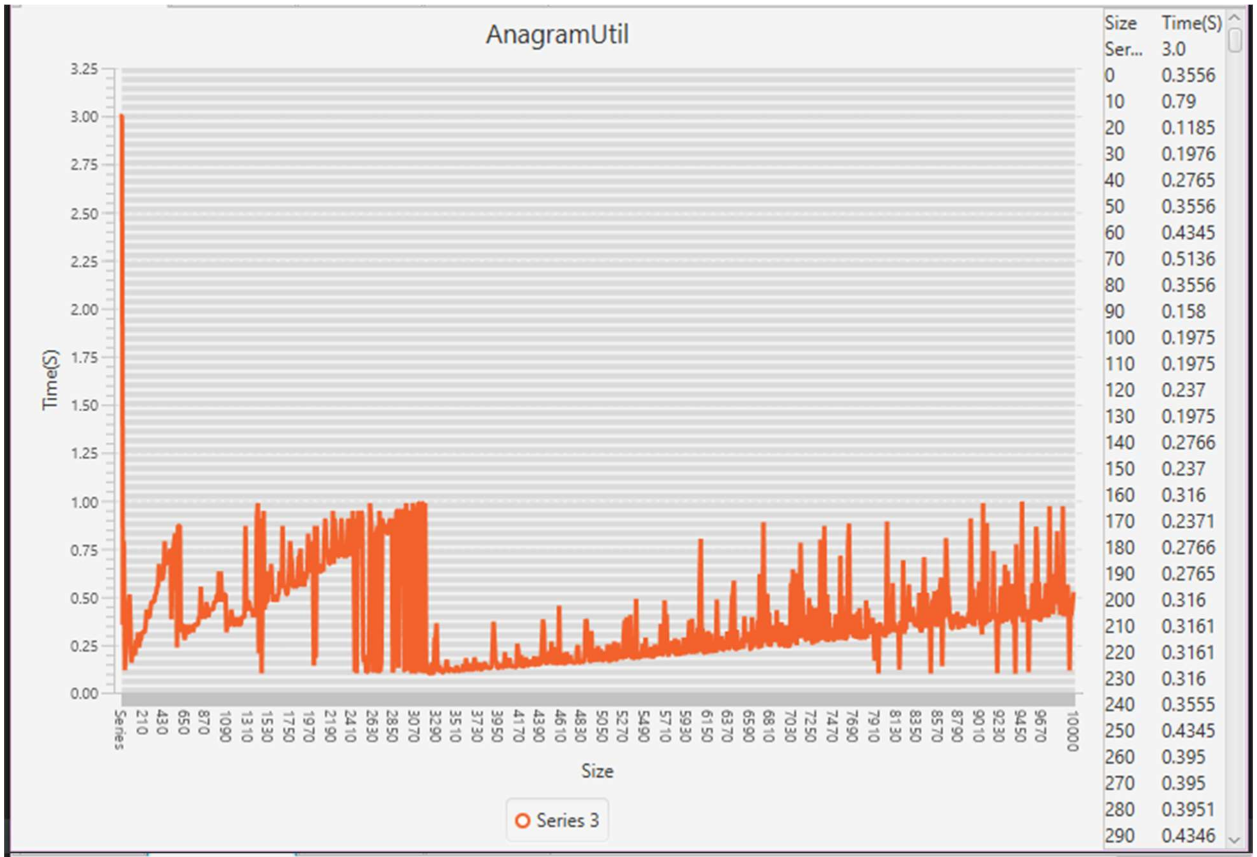
Analyze the run-time performance of the getLargestAnagramGroup method using your insertion sort algorithm. (Use the same list of guiding questions as above.) Note that in this case, N is the number of words, not the length of words. Finding the largest group of anagrams involves sorting the entire list 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 (write a function to randomly build a string of characters). If you use the random word generator, use modest word lengths, such as 5-15 characters.



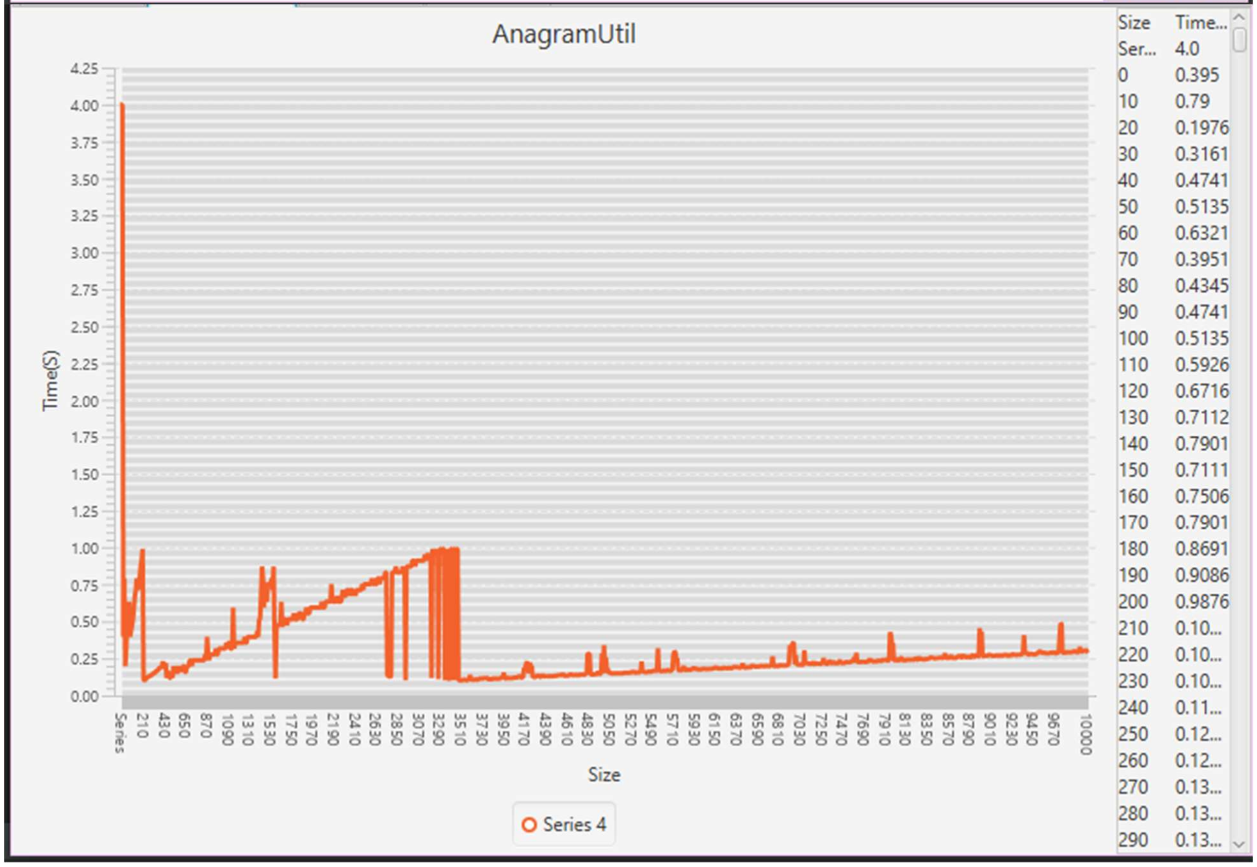
Size	Time(S)
Ser...	2.0
0	0.3161
10	0.93234
20	0.16316
30	0.212543
40	0.173432
50	0.169086
60	0.190419
70	0.340543
80	0.286814
90	0.321185
100	0.32395
110	0.32553
120	0.318024
130	0.348049
140	0.391901
150	0.224
160	0.330271
170	0.363062
180	0.378469
190	0.464593
200	0.384395
210	0.314864
220	0.338173
230	0.200691
240	0.184099
250	0.185284
260	0.18884
270	0.22479
280	0.319605
290	0.309333

The get largest anagram group takes only a few seconds, even with an array of 10000 items. It ends up with there being very fast times, but only in some cases. It does best with around 1000 items in the array.

Analyze the run-time performance of the insertionSort method using an array of strings and an array of integers. Does the speed of computing match the speed of the getLargestAnagramGroup? Explain why/why not.
Yes, they match because the insertion sort is the longest part of the algorithm. The has very little effect on the time

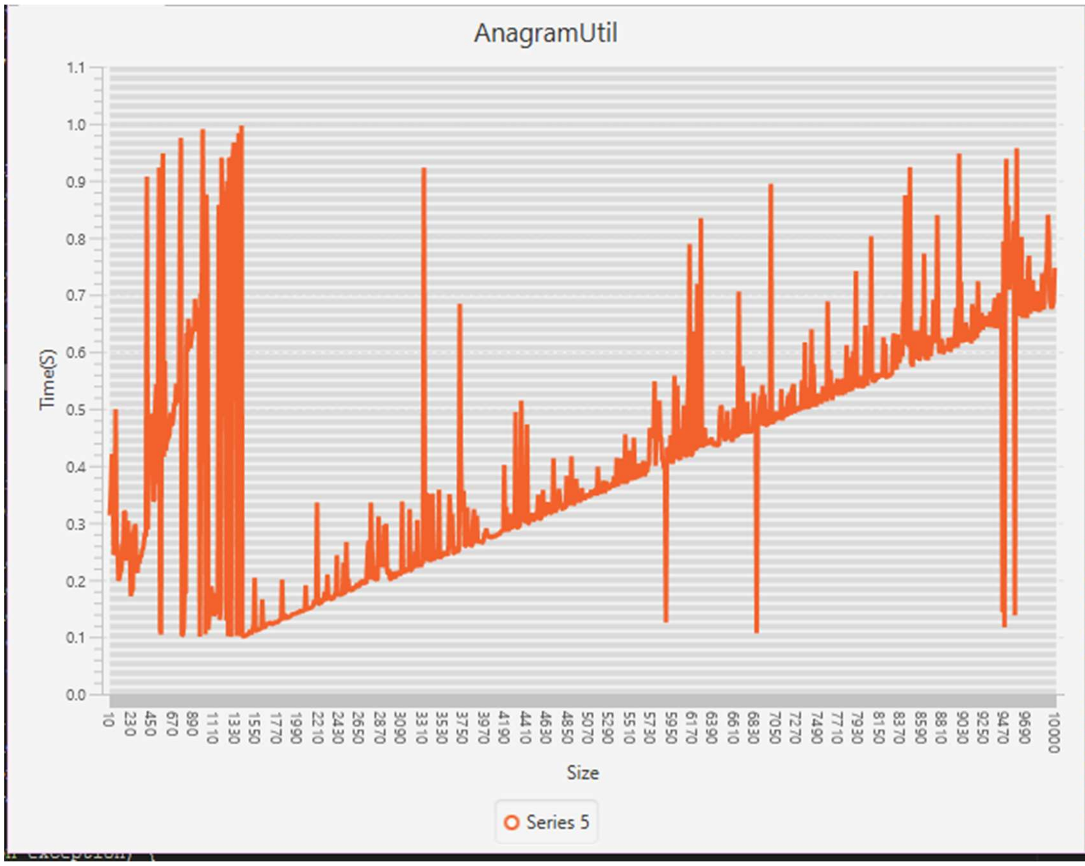


Size	Time(S)
Ser...	3.0
0	0.3556
10	0.79
20	0.1185
30	0.1976
40	0.2765
50	0.3556
60	0.4345
70	0.5136
80	0.3556
90	0.158
100	0.1975
110	0.1975
120	0.237
130	0.1975
140	0.2766
150	0.237
160	0.316
170	0.2371
180	0.2766
190	0.2765
200	0.316
210	0.3161
220	0.3161
230	0.316
240	0.3555
250	0.4345
260	0.395
270	0.395
280	0.3951
290	0.4346



Size	Time...
Ser...	4.0
0	0.395
10	0.79
20	0.1976
30	0.3161
40	0.4741
50	0.5135
60	0.6321
70	0.3951
80	0.4345
90	0.4741
100	0.5135
110	0.5926
120	0.6716
130	0.7112
140	0.7901
150	0.7111
160	0.7506
170	0.7901
180	0.8691
190	0.9086
200	0.9876
210	0.10...
220	0.10...
230	0.10...
240	0.11...
250	0.12...
260	0.12...
270	0.13...
280	0.13...
290	0.13...

What is the run-time performance of the getLargestAnagramGroup method if we use Java's ArrayList sort method instead of our own (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 above.)
It runs a lot faster with the slowest time under 1 second. Using insertion sort takes over 1 second in most cases.



Size	Time(S)
10	0.317234
20	0.360296
30	0.420346
40	0.41521
50	0.244148
60	0.247308
70	0.499357
80	0.253235
90	0.262321
100	0.198321
110	0.203457
120	0.214123
130	0.216889
140	0.246914
150	0.269037
160	0.32079
170	0.318024
180	0.233876
190	0.252839
200	0.303802
210	0.25916
220	0.283259
230	0.171456
240	0.180543
250	0.182914
260	0.283259
270	0.296296
280	0.294716
290	0.217284
300	0.212543
310	0.243753