CS201 Homework Assignment 2

Name: Deniz Aydemir
ID: 22001859
Section: 1

1) Complexity of Each Algorithm

Algorithm_1:

First, we need to find the max number in the array. In the worst-case scenario, we look for all items in the array in O(n) time, where n is the array's length. Then, to move the max number to the beginning of the array, we spend O(1) time. Then, we look for the max number in the remaining n-1 elements in O(n) time in the worst-case scenario since O(n-1) = O(n). If we continue this process n/2 times, we sort the first half of the array, which is sufficient to find the median of the array. Therefore, this algorithm runs in O(n^2) time.

Algorithm_2:

then partitions the array into two parts such that one of the parts contains the values less than or equal to the pivot number and the other one contains values greater than the pivot number, in O(n) time, since algorithm needs to check every item in the array and rearrange their position in O(1) time. Then, the algorithm continues to partition the parts into sub-parts recursively. Depending on the taken pivot values, the recursive process takes O(log n) time on average since the algorithm approximately divides every part by 2. Therefore, the overall time complexity of the quick-sort is O(nlogn). However, if pivot values are chosen as the min or max value of the part that is partitioned, the recursive process takes O(n) time since we get one part with length n-1 and one part with length 1. Consequently, quick-sort takes O(n^2) time in a worst-case scenario.

Algorithm_3:

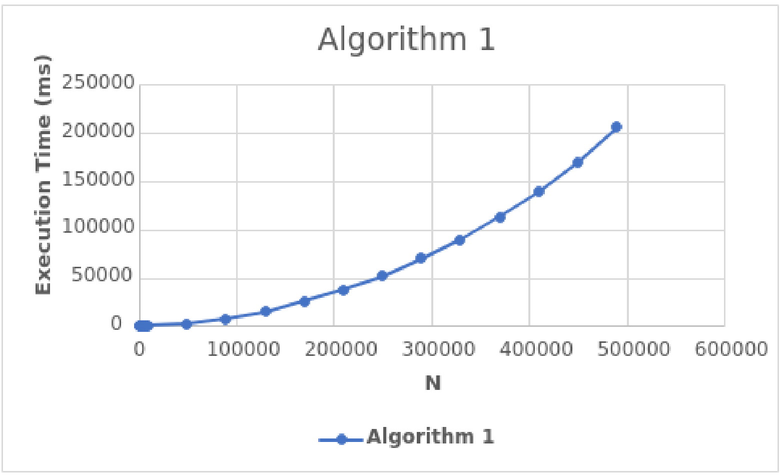
First of all, the algorithm finds a reasonable pivot value. The array is divided to form n / 5 sub-parts with length 5. Then, the algorithm sorts the sub-parts with quick-sort and finds the median of each subpart in O(n) time since sub-parts have a constant length (therefore, they are sorted with constant time O(1)) and there are n / 5 sub-parts. The algorithm continues recursively to determine the median of medians and acquire the pivot it will use in the next step. Then, the algorithm partitions the array according to the pivot it acquired in O(n) time and checks whether the current index of the pivot is equal to the index of the median, which is equal to (n+1) / 2. If it is, we found our median. If it is not, it means that our median index is less or greater than the pivot index. Hence, the algorithm has to evaluate the right or left parts as a new array until it finds the median. If the index of the pivot is smaller than the median's, the algorithm partitions the right part. Else if the index of the pivot is greater than the median's, the algorithm partitions the left part; since finding a reasonable median and partitioning takes O(n) time, the algorithm takes O(n) time.

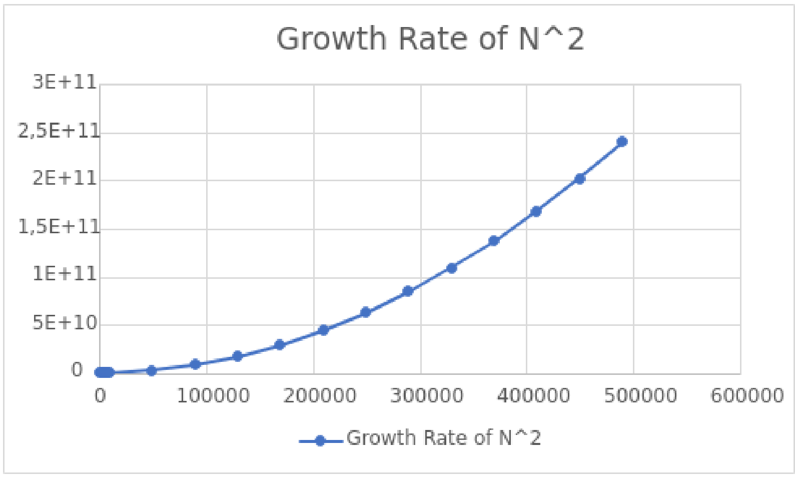
2) Table for Execution Times of Algorithms

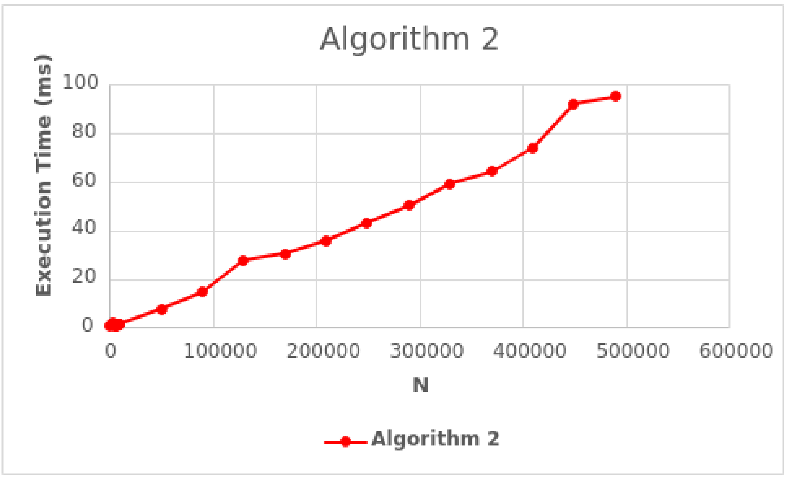
N	Algorithm 1 (ms)	Algorithm 2 (ms)	Algorithm 3 (ms)
1000	4,262	0,576	0,483
2000	16,544	1,165	0,84
3000	36,081	1,833	1,151
4000	39,509	0,944	0,59
5000	28,001	0,746	0,424
6000	30,95	0,82	0,451
7000	40,294	0,853	0,271
8000	52,861	1,046	0,679
9000	67,762	1,236	0,684
10000	83,057	1,404	0,777
50000	2025,21	7,542	3,656
90000	6568,23	14,346	6,178
130000	14291,5	27,565	9,435
170000	24936,5	30,242	12,987
210000	37031,4	35,717	15,316
250000	51206,4	43,124	17,911
290000	69405	50,115	20,309
330000	89266,9	59,209	22,466
370000	112284	64,094	24,978
410000	138968	73,935	28,146
450000	169190	92,107	34,058
490000	205884	95,225	34,226

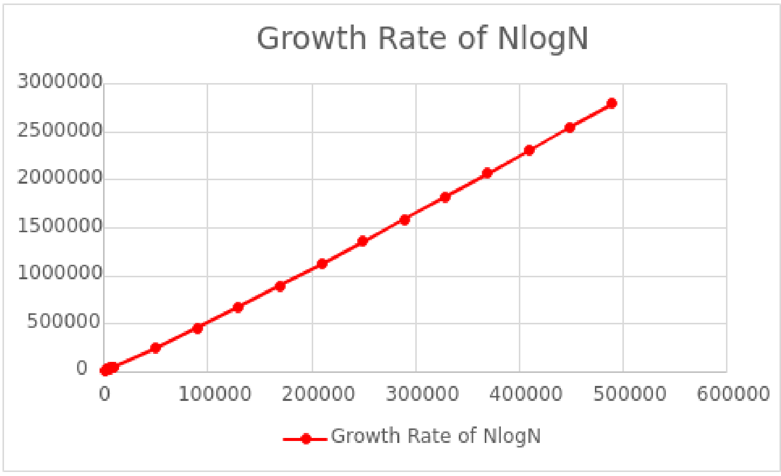
3.1) Plots for Algorithms and Expected Growth Rates Separately

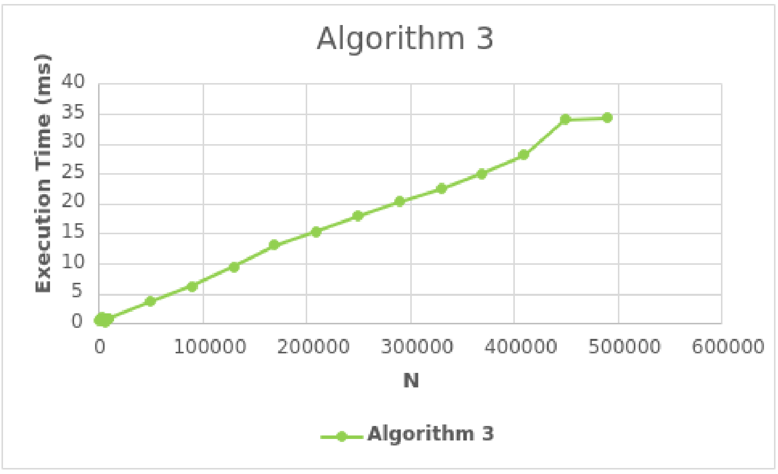
(Shapes of the graphs of growth rates are not the ideal, especially N^2, since spaces between vertical values and horizontal values are not same. Making every space have same value resulted in abnormal graph sizes, therefore I avoided it. Graphs of NlogN and N may seem identical but values in vertical axis are increasing differently.)

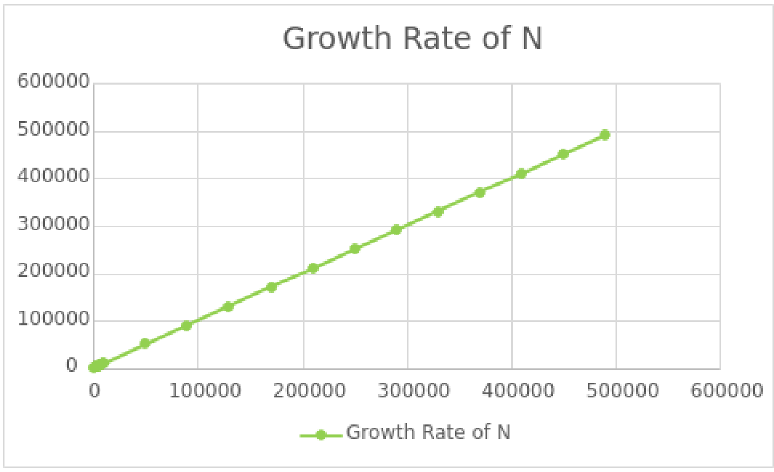






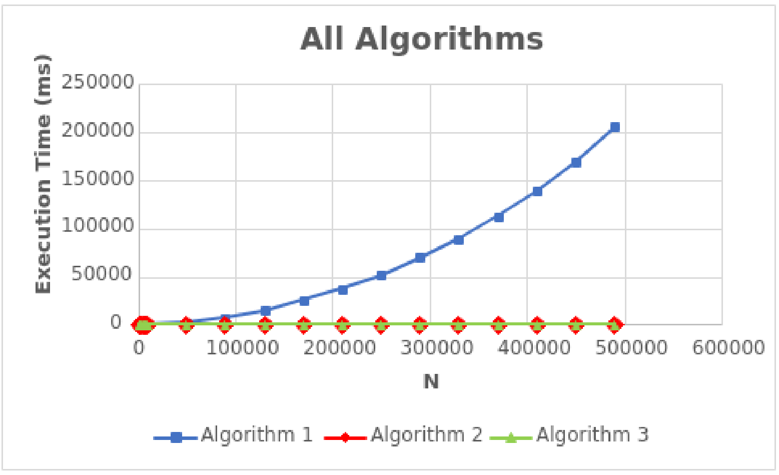


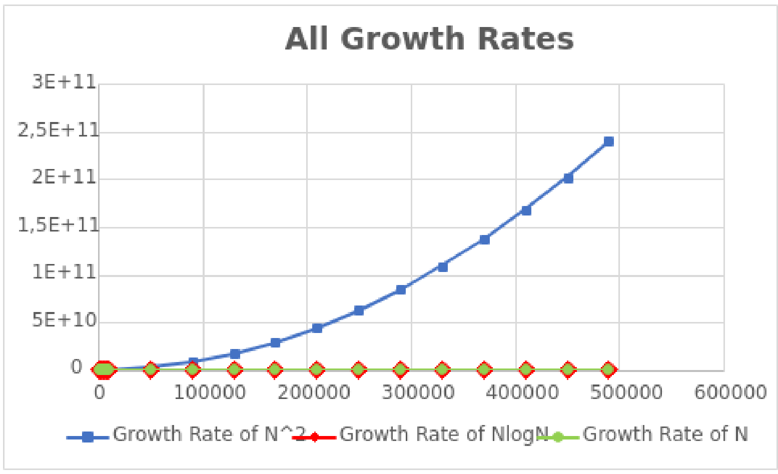




3.2) Plots for Algorithms and Expected Growth Rates Together

(Since the execution times for Algorithm 1 is too large relative to Algorithm 1 and 2, the graph does not give so much detail about other two Algorithms. This situation can also be observable in the graph of growth rates.)





4) Comparison of Expected Results and Obtained Results

The results I got from my computer are roughly similar to the expected values. Among the three algorithms, the graph of Algorithm 1 is the most similar to its expected graph. This situation may result from due to fact that its results (execution times) are too high to reflect the differences with its expected graph. On the other hand, Algorithm 1 and Algorithm 2 have some differences from their expected graphs which are worthy of note. In Algorithm 2's case, the quick-sort algorithm chooses random pivots which some of them undesirable, like the min or max value of the array. This situation could be the reason for small inconsistencies in the slope of Algorithm 2. Moreover, the array itself is a factor of execution time. For instance, the stage before the last stage in both Algorithm 1 and Algorithm 2 has an execution time as nearly as the last stage in both graphs. Array taken in this stage may have a hard-to-reach median index, which results in more execution time than expected.

5) Specification of the Computer I Use

Operating System: Linux Mint 21 Cinnamon 5.4.12

CPU: Intel© Core™ i3-10110U CPU @ 2.10GHz × 2

RAM: 2 x 4GB SK Hynix 2400MHz DDR4