CS 201

Fundamental Structures of Computer Science I

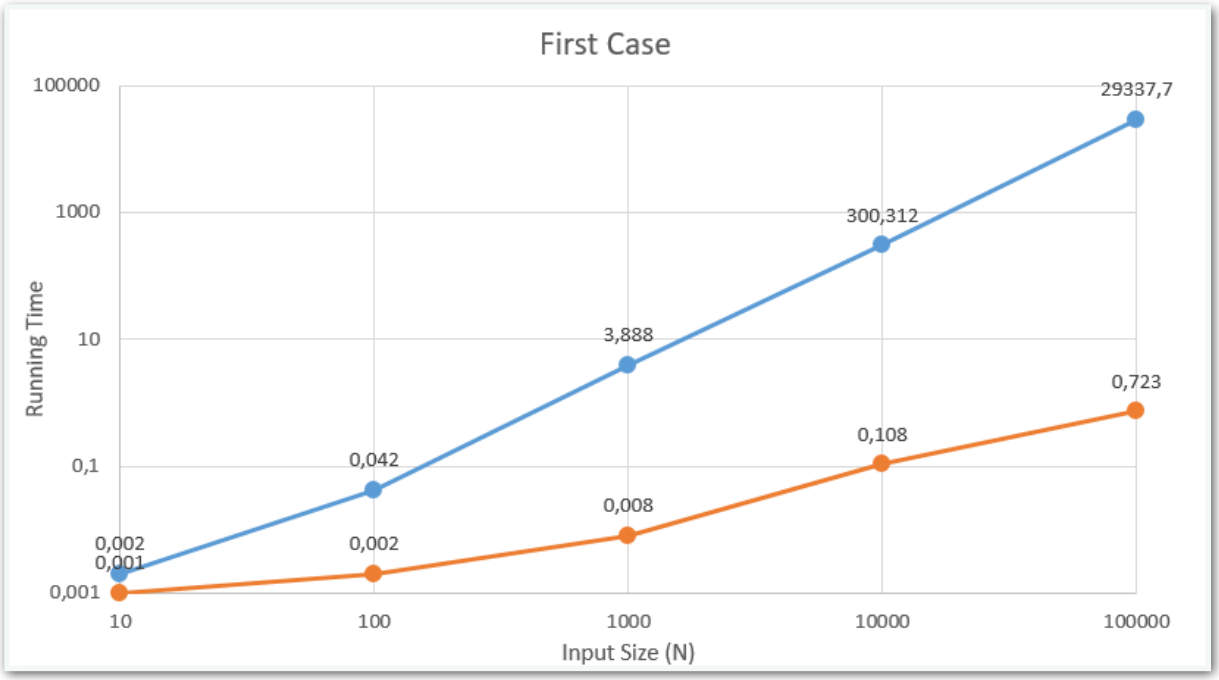
Homework Assignment 2

Name: Esad Ismail Tok

Student ID: 21801679

Section: 1

Plots for the Ordering Cases

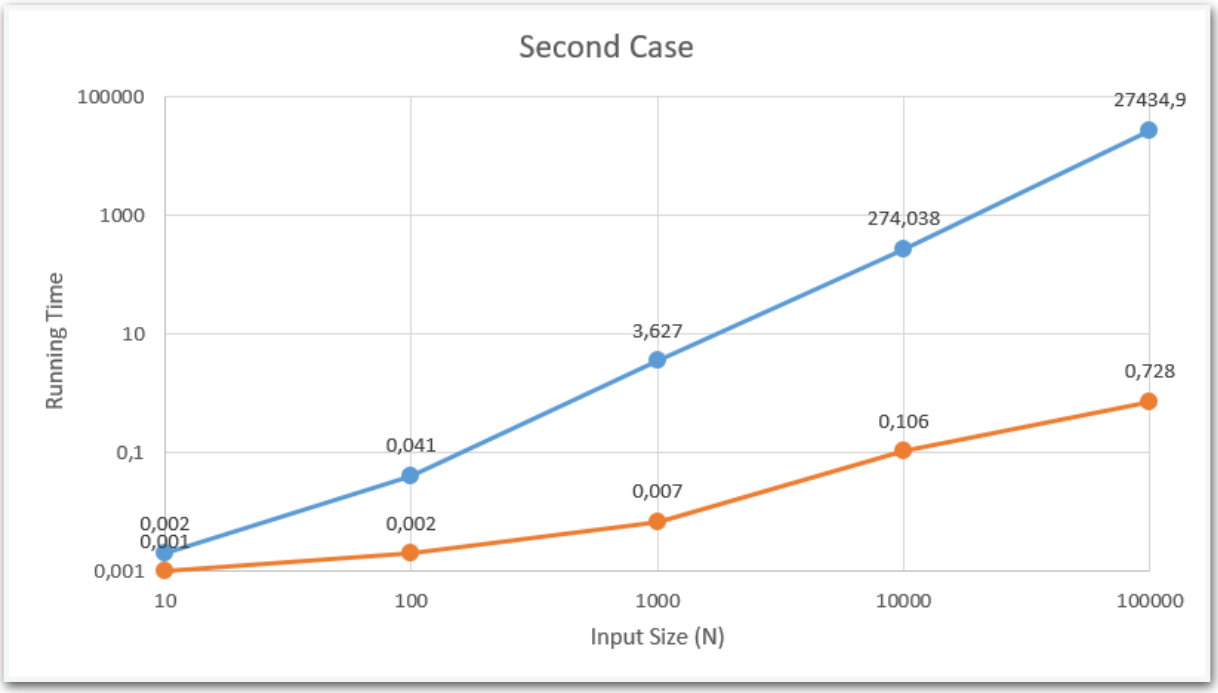


Plot 1

Plot 1 is the graph of the input size vs running time of the two algorithms in the first ordering case, in which all the items of the first array are smaller than all the items of the second array that will be merged.

The blue curve represents the first algorithm and the orange curve represents the second algorithm.

Plots for the Ordering Cases

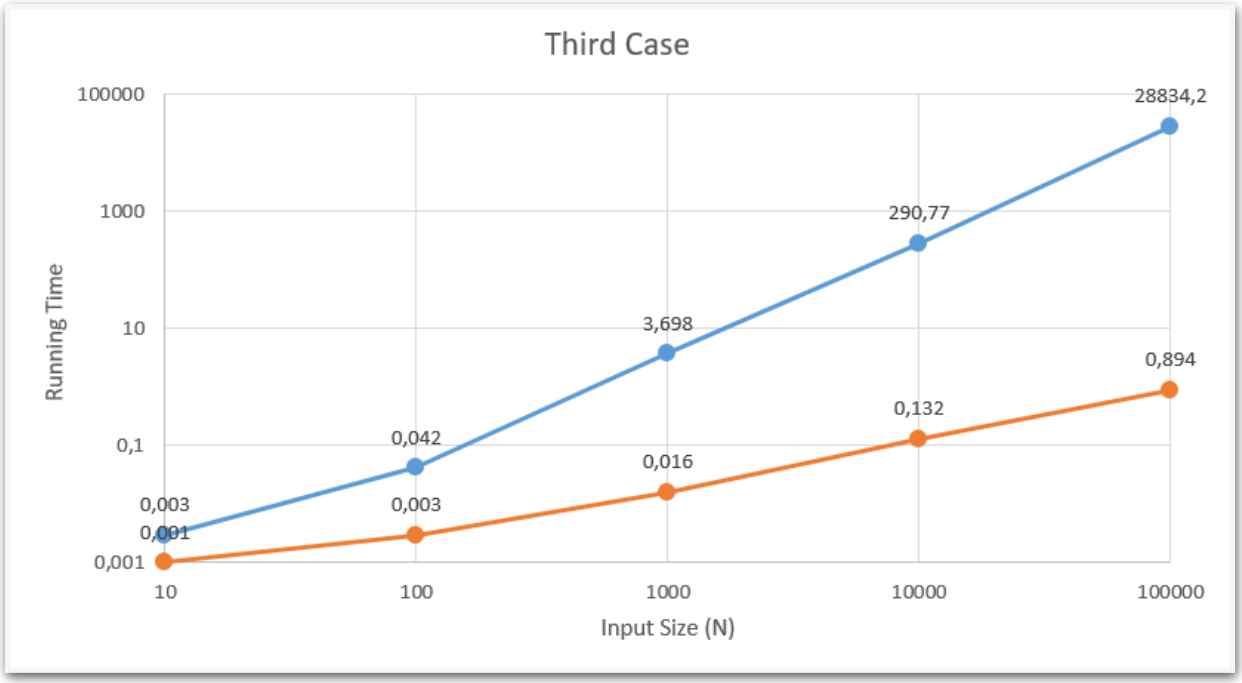


Plot 2

Plot 2 is the graph of the input size vs running time of the two algorithms in the second ordering case, in which all the items of the second array are smaller than all the items of the first array that will be merged.

The blue curve represents the first algorithm and the orange curve represents the second algorithm.

Plots for the Ordering Cases



Plot 3

Plot 3 is the graph of the input size vs running time of the two algorithms in the third ordering case, in which there is no relation between the items of the first and second arrays that will be merged such that all the items in both arrays are randomly large.

The blue curve represents the first algorithm and the orange curve represents the second algorithm.

Algorithm Time Complexities

Best Cases:

- Best case for the Algorithm 1 is the second case, in which all the items in the second array are smaller then all the items in the first array. The reason for this is that the inner loop of the first algorithm checks whether an item of second array is smaller than the items in the third array or not. Since in the second case all items in the second array are smaller then the first array and since all items of the first array are directly placed into the third array, the loop places the items of the second array at the beginning of the third array and therefore exits the loop before the other cases.
- Therefore the best case for Algorithm 2 is the second case. Since there is another nested for loop which iterates starting from the last element of the third array to open a place to the new item, the worst case time complexity for the first algorithm in the second case is *O(N^2)*.

- Best case for the Algorithm 2 is one of the first and second cases. Those two cases can be considered as the average cases as well but since there is no better case in this algorithm those two cases are considered as the best case. In the first and second cases, all items in one of the arrays are smaller than the other and therefore the algorithm can execute the first loop quicker and then skip to the next consecutive loop.
- Since the first loop which has more consecutive codes to be executed compared to the other loops is executed with less iteration because all the items of one of the arrays are smaller than the other, both first and second cases can be considered as best cases. The worst case time complexity in these cases is *O(N)* because there is no nested loops but only consecutive loops.

Average Cases:

- Average case for the Algorithm 1 is the third case in which there is no relation between the items in the arrays and they are randomly greater or smaller than each other. In this case the correct place to put an item of the second array in third array can be found near the middle of the iteration of the loop.

- Therefore the third case is the average case for the Algorithm 1 and the worst case time complexity of the Algorithm 1 in the third case is *O(N^2)* again because there is nested loop to be iterated.

- Average case for the Algorithm 2 can be again considered as the first or second case. According to the *Plot 1* and *Plot 2,* the running times of the both cases are similar for the Algorithm 2 in both graphs. Those two cases are the average cases among the three cases however since there is no better case in terms of running time, those two cases can be considered as the best case as well.
- The worst case time complexity for the average case in Algorithm 2 is *O(N)* again because there is no nested loops to be iterated and there are just consecutive loops.

Worst Cases:

- Worst case for the Algorithm 1 is the first case, in which all the items of the first array are smaller than the second array. Since the algorithm iterates the items of the second array until it finds a greater item in the third array which includes the items of the first array, and since there is no item in the first array that is greater than the second array, the loop executes until the end of all items and then exits
- Therefore the inner loop iterates all the items of the third array and cannot find an item that is greater than any item of the second array. So the first case is the worst case for the Algorithm 1 and the worst case time complexity is *O(N^2)* again since there is a nested loop.

- Worst case for the Algorithm 2 is the third case in which all the items of the arrays are randomly smaller or grater than each other. Since there is no specific order between the both array items, the algorithm cannot exit the first loop quicker as it does in the first or second case. Moreover the first loop in this algorithm includes more consecutive statements to be executed and the running times of the third case are greater than its counterparts of the first and second cases according to the *Plot 1, Plot 2, and Plot 3*. Therefore third case is the worst case for the Algorithm 2.
- The worst case time complexity for the Algorithm 2 in the third case is again *O(N)* since there is no nested loops but only consecutive loops.

Computer Specifications

Operating System:	macOS Catalina, version 10.15.6
Processor:	2,8 GHz quad-core Intel Core i7
RAM:	16 GB 2133 MHz LPDDR3
Storage:	256 GB SSD
Graphics:	Intel HD Graphics 630 1536 MB

Plot for the Theoretical Worst Case Growth Rates

Plot 4

Plot 4 is the graph of the input size vs running time that represents the theoretical worst case growth rates of Algorithm 1 and Algorithm 2 according to the same N inputs.

The blue curve represents the first algorithm and the orange curve represents the second algorithm.

Discussion

Both theoretical and experimental worst case growth rates of the two algorithms have similar patterns. Second algorithm having *O(N)* time complexity seems clearly more desirable than the first algorithm having *O(N^2)* time complexity. In both theoretical and experimental worst case plots, there is no considerable amount of difference between the running times of the algorithms before the input size gets the value of 100. That is because both algorithms can be executed in a short time and therefore it is hard to observe the running time differences between the algorithms when the input size is less than 100.
On the other hand, as the input size grows, the running time differences between Algorithm 1 and Algorithm 2 becomes more visible and in both theoretical and experimental plots the running times in the last input sizes are considerably far from each other. The expected difference according to the theoretical plots can also be proved by looking the experimental plots which proves how Algorithm 1 is executed by taking great amount of time as the input size increases.
The running times of the both algorithms in the experimental plots are similar to the ones in the theoretical plots when the input sizes are not so large. However as the input size increases to the greater numbers such as 10000 the theoretical running time values become larger compared to their experimental counterparts, even though the growth patterns of both plots remain similar. Those running time differences between the theoretical and experimental plots in large input sizes can be caused by several variables such as computer specifications or even the implementations of the specific algorithms.
To sum up, by observing the worst case growth rate patterns of the two algorithms, it is more clear that the second algorithm having *O(N)* time complexity is considerably feasible than the first algorithm having *O(N^2)* time complexity and the theoretical worst case growth rate plots also supports that conclusion. Moreover, although theoretical running time values can be larger than the experimental running time values in large input sizes, the experimental growth rate patters of both algorithms are similar to their theoretical counterparts.