Selen Görgün

21802674

Section 1

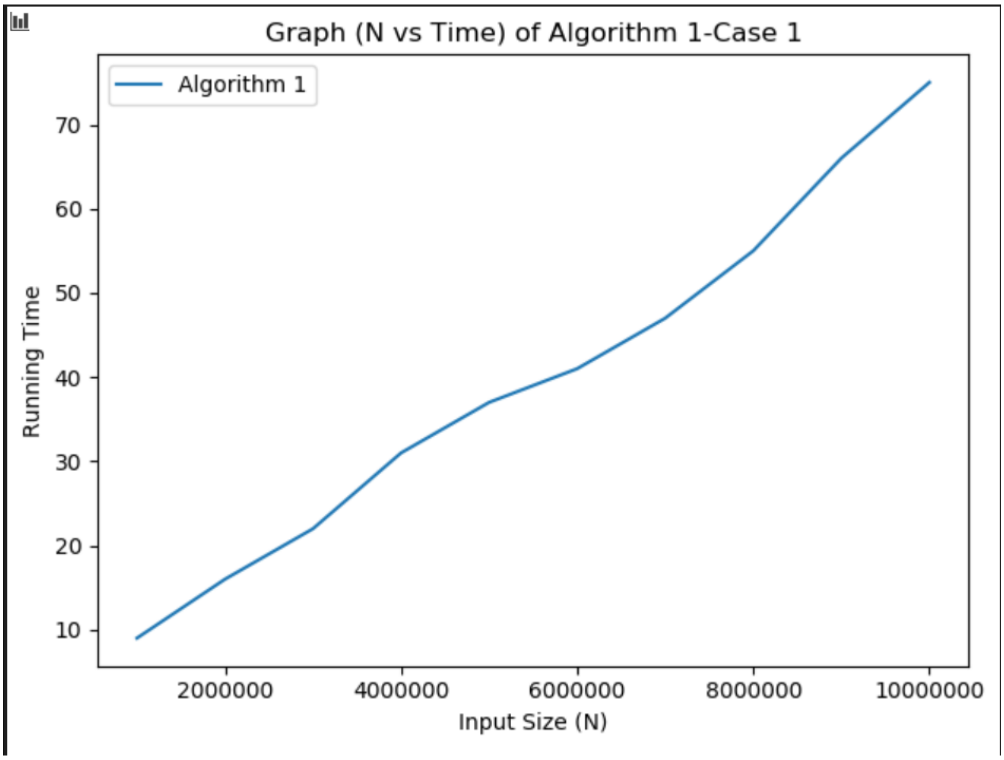
HW2

CS 201 Algorithm Time Complexity Analysis

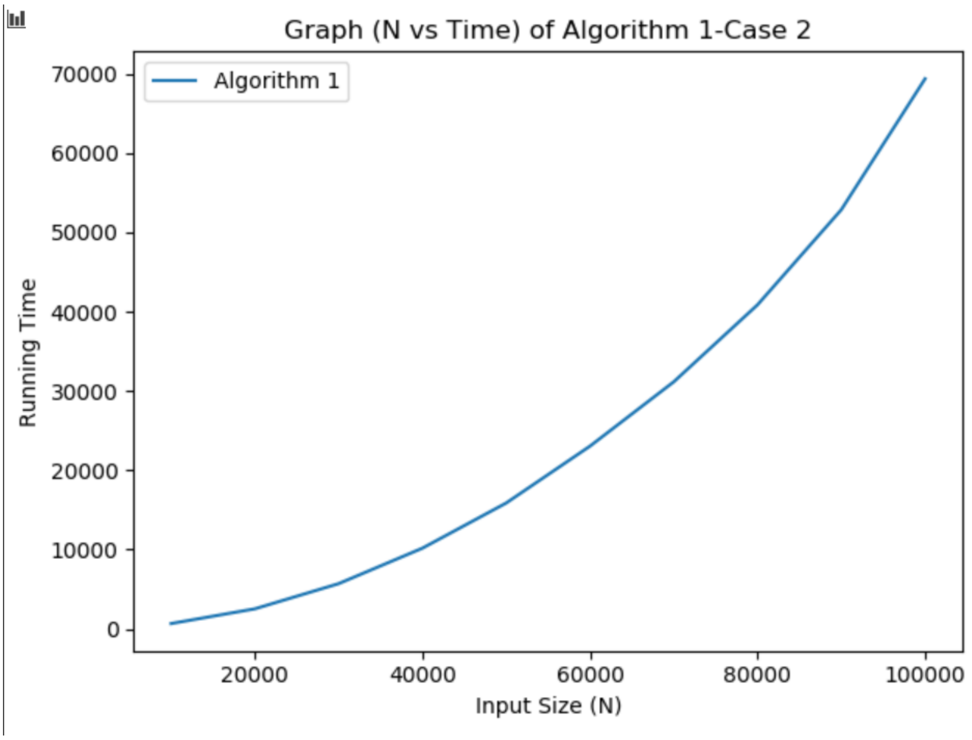
<u>Time(msec) table for Algorithm 1</u>

Input Size (n)	**Case 2**	**Case 3**
n = 10000	677	508
n = 20000	2534	2007
n = 30000	5708	4525
n = 40000	10181	8061
n = 50000	15875	12668
n = 60000	23044	18237
n = 70000	31154	24723
n = 80000	40866	32230
n = 90000	52854	44124
n = 100000	69364	54284

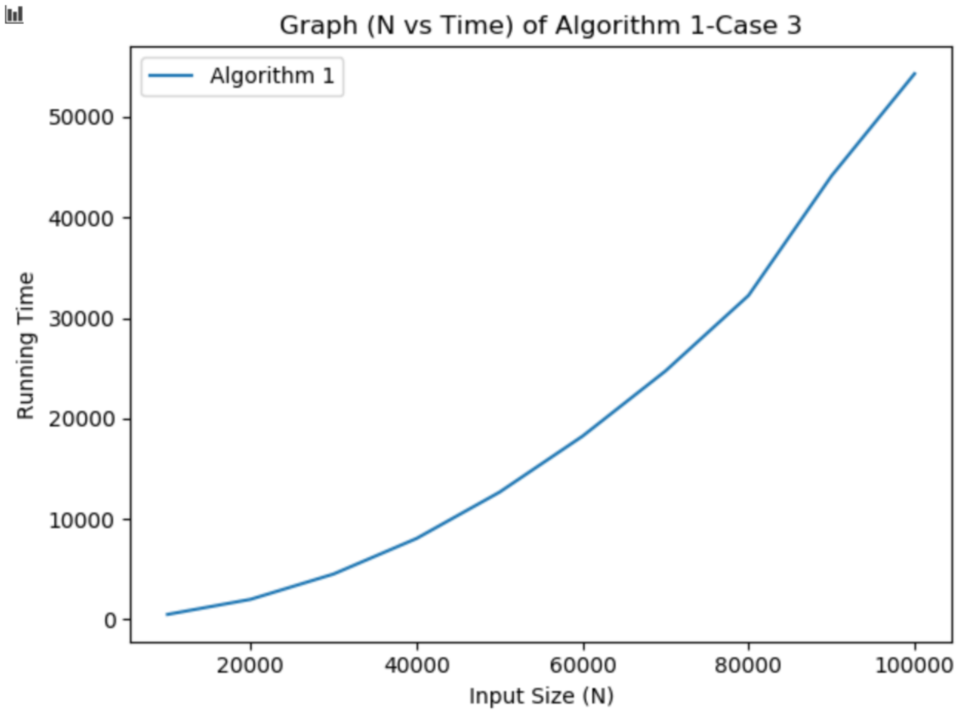
Input Size (n)	**Case 1**
n = 1000000	9
n = 2000000	16
n = 3000000	22
n = 4000000	31
n = 5000000	37
n = 6000000	41
n = 7000000	47
n = 8000000	55
n = 9000000	66
n = 10000000	75



Graph (N vs Time) of Algorithm 1-Case 1
Algorithm 1
Running Time
Input Size (N)
10
20
30
40
50
60
70
2000000
4000000
6000000
8000000
10000000

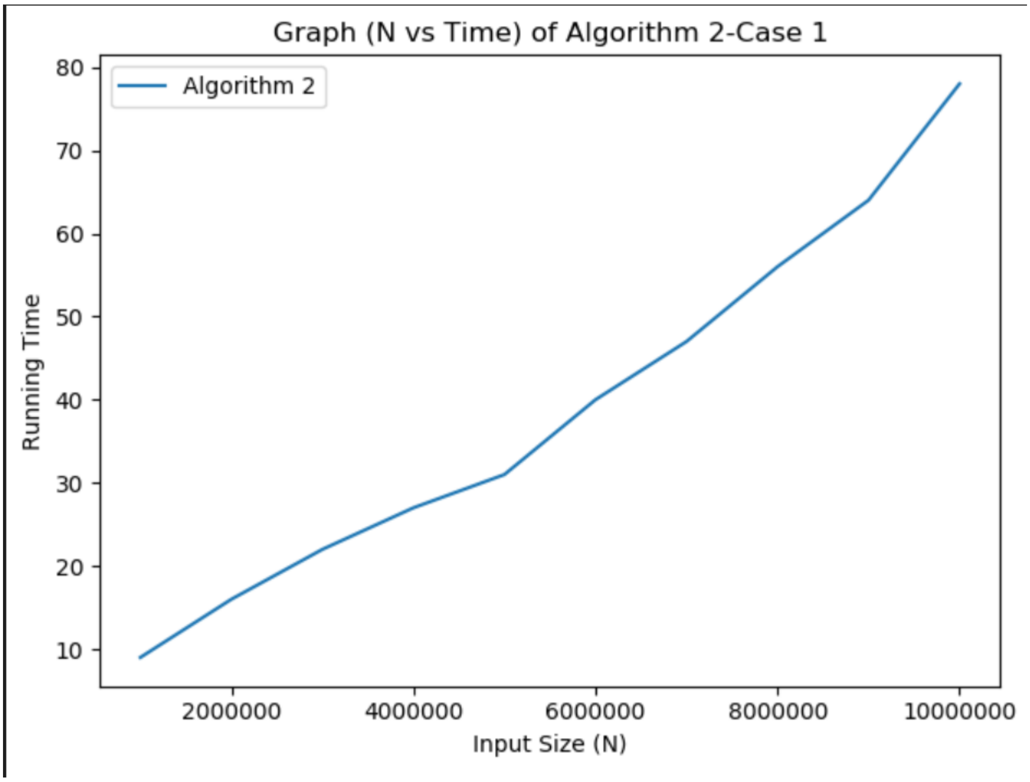


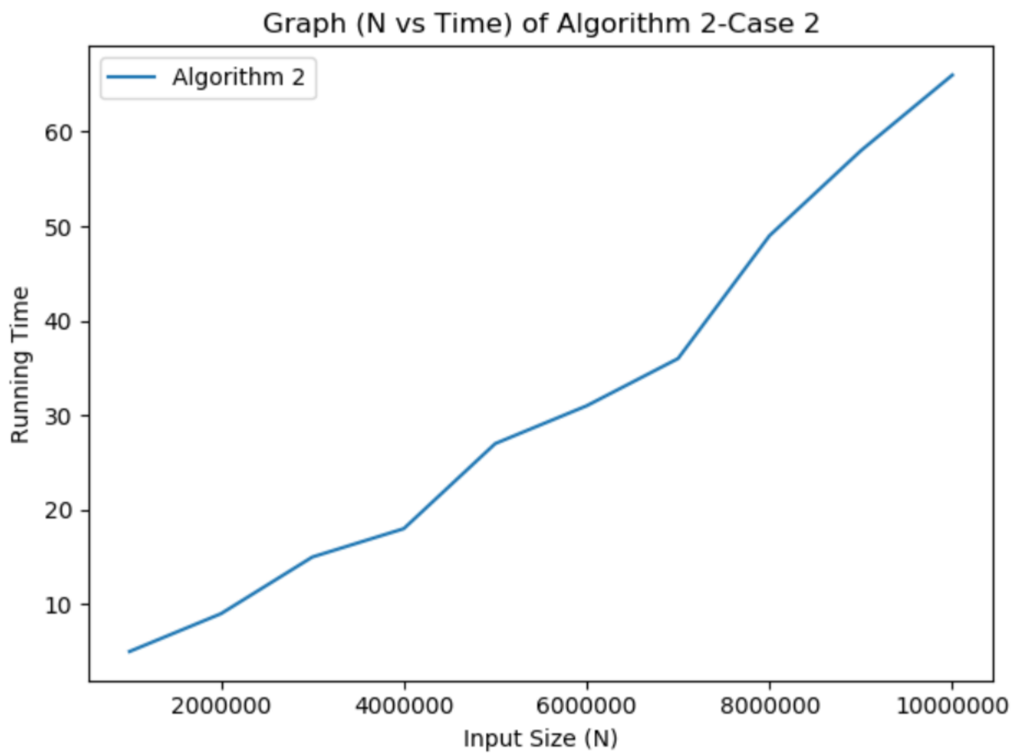
Graph (N vs Time) of Algorithm 1-Case 2
Algorithm 1
Running Time
Input Size (N)
0
10000
20000
30000
40000
50000
60000
70000
20000
40000
60000
80000
100000

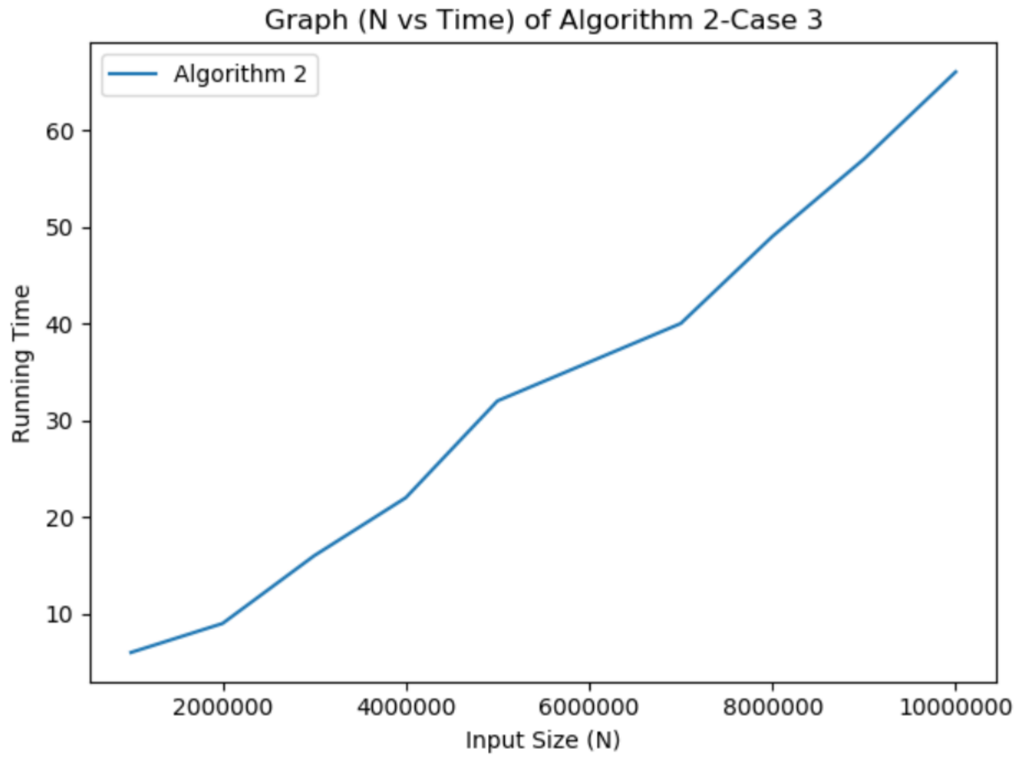


<u>Time(msec) table for Algorithm 2</u>

Input Size (n)	**Case 1**	**Case 2**	**Case 3**
n = 1000000	9	5	6
n = 2000000	16	9	9
n = 3000000	22	15	16
n = 4000000	27	18	22
n = 5000000	31	27	32
n = 6000000	40	31	36
n = 7000000	47	36	40
n = 8000000	56	49	49
n = 9000000	64	58	57
n = 10000000	78	66	66


Graph (N vs Time) of Algorithm 2-Case 1
Algorithm 2
Running Time
80
70
60
50
40
30
20
10
2000000
4000000
6000000
8000000
10000000
Input Size (N)


Graph (N vs Time) of Algorithm 2-Case 2
Algorithm 2
Running Time
60
50
40
30
20
10
2000000
4000000
6000000
8000000
10000000
Input Size (N)



Algorithm 1 Analysis

In algorithm 1, all items in array 1 is added to array 3 in the same order firstly. After that, items in array 2 are being added to array 3. While doing this process, items in array 2 and array 3 are compared. Comparison from the beginning of array 2 and the end of array 3. If any of the items in array 2 are bigger than or equal to the items of array 3, the items in array 3 are shifted to make place for the item from array 2. If the item in array is smaller than the all of the items in array 3, all the items in array 3 are shifted one index higher and the item in array 2 is inserted to the first index of array 3.

The *best case* for algorithm 1 is *Case 1* which is the case where all the items in array 1 is smaller than array 2. In this case, since all the items in array 1 is added to array 3, all items in array 3 are smaller than array 2. So, the last item of array 3 is smaller than the first item item of array 2. In this case, it is not needed to shift any items in array 3 to insert any item of array 2. All array 2 items will be added to the end of array 3 in the same order. Time complexity for the best case is O(n).

The *worst case* for algorithm 1 is *Case 2* which is the case where all the items in array 2 are smaller than array 1. In this case, since all the items in array 2 are smaller than the items in array 3, all of the items in array 3 needs to be shifted to make place for the items in array 2. Time complexity for the worst case is O(n^2).

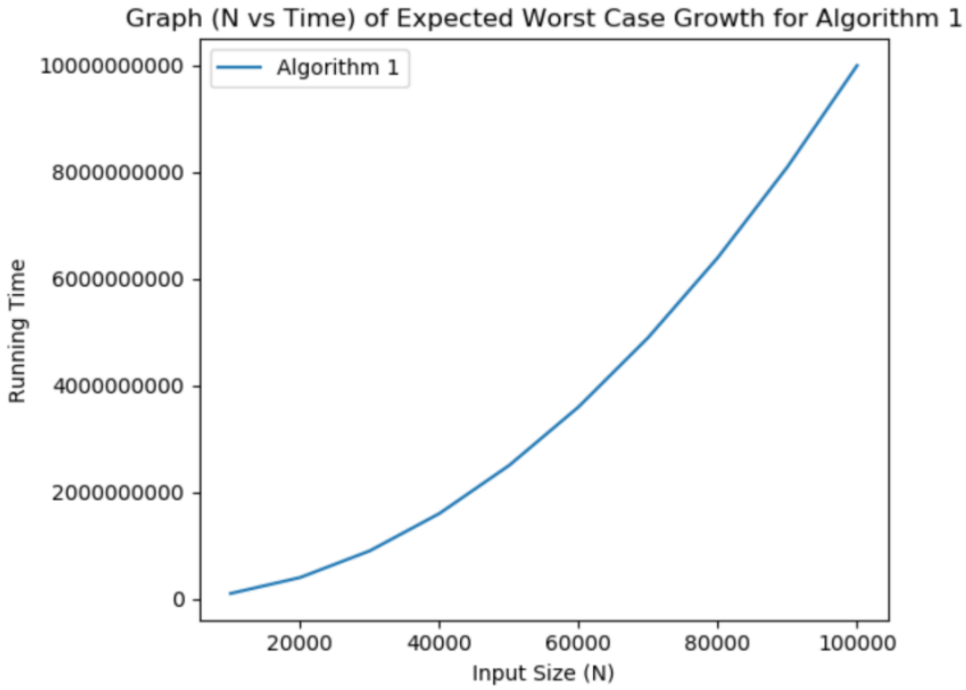
In *Case 3*, some items of array 3 are being shifted and some items are not since there is no ordering between the items of array 1 and array 2.

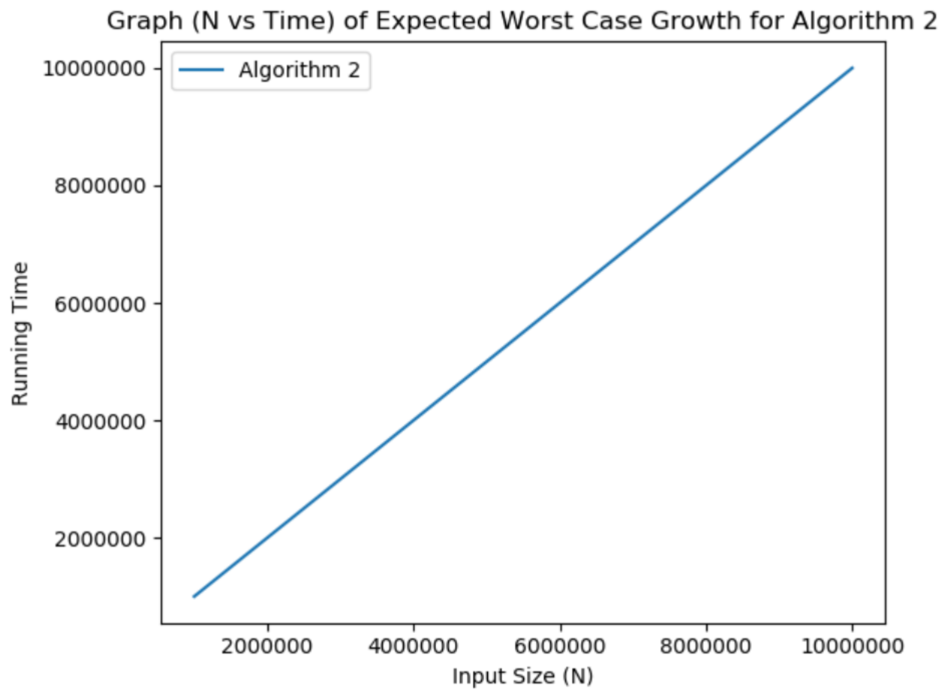
Algorithm 2 Analysis

In algorithm 2, items in array 1 and array 2 are being compared and the smallest is being added to array 3 until all of the items in one of the arrays are added. After this comparison, the items in the array in which that have still items left to add are being added to array 3.

Time complexity for the *best case* and *worst case* is O(n). For *Case 1*, all the items in array 1 are going to be added to array 3 and then, all the items in array 2 are going to be added to array 3. For *Case 2*, items of array 2 are going to be added first. For *Case 3*, items are going to be added until all of the items in one of the arrays are added to array 3. After that, items that are not added yet are going to be added. Time complexity for all of the cases is O(n). This is why the difference between the execution times of the three cases is very little.

Expected Worst Case Growth Rates Obtained from Theory





<u>Comparison of Expected Growth Rates and Experimental Growth Rates</u>

The experimental growth rates graphs are not as perfect as the expected growth rates graphs. The reason of this might be that there are not enough output data to form a better graph. The number of the output data and the perfection of the graphs are directly proportional. Also, the output data changes according to the computer used in the experiment.

<u>Computer Specifications</u>

Processor: Intel® Coffee Lake Core™ i7-9750H 6C/12T; 12MB L3; 8GT/s; 2.6GHz > 4.5GHz; 45W; 14nm

Memory: 16GB (2x8GB) DDR4L 1.2V 2666MHz SODIMM