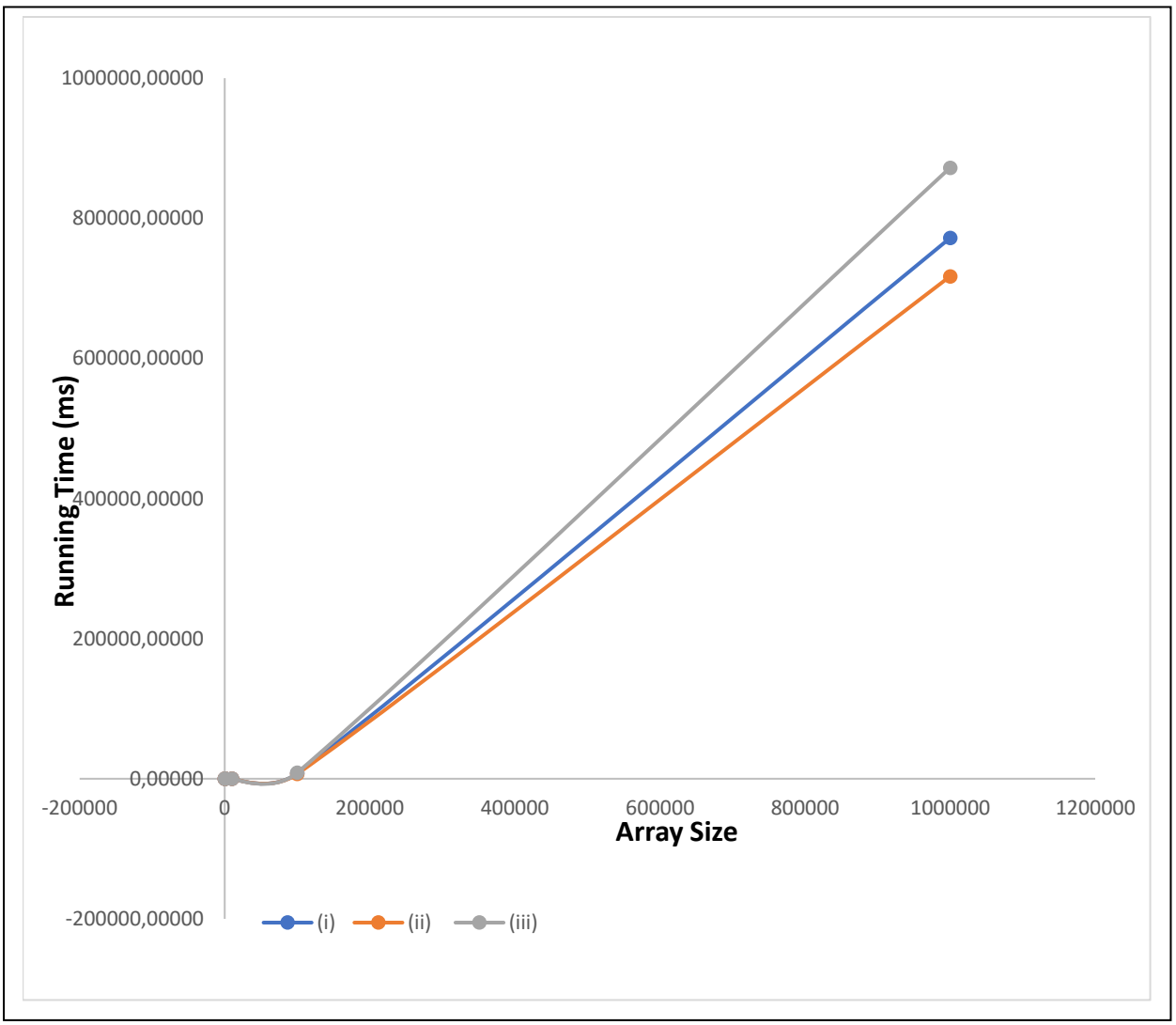
Burak Yetiştiren
Section-2
21802608
CS201
HW2

Computer Specifications
Model: MSI GF75 Thin 8SC-206XTR
Processor: 4.1 GHz Intel Core i7
Memory: 8GB 2666 MHz DDR4
Storage: 5400 RPM 1TB HDD 256 GB SSD

| Array Size | (i) | (ii) | (iii) |
|---|---|---|---|
| 10 | 0,00020 | 0,00020 | 0,00030 |
| 1000 | 0,78000 | 0,67000 | 0,86000 |
| 10000 | 77,07000 | 65,79000 | 85,13000 |
| 100000 | 7721,00000 | 6889,00000 | 8636,00000 |
| 1000000 | 771889,00000 | 717182,00000 | 872167,00000 |

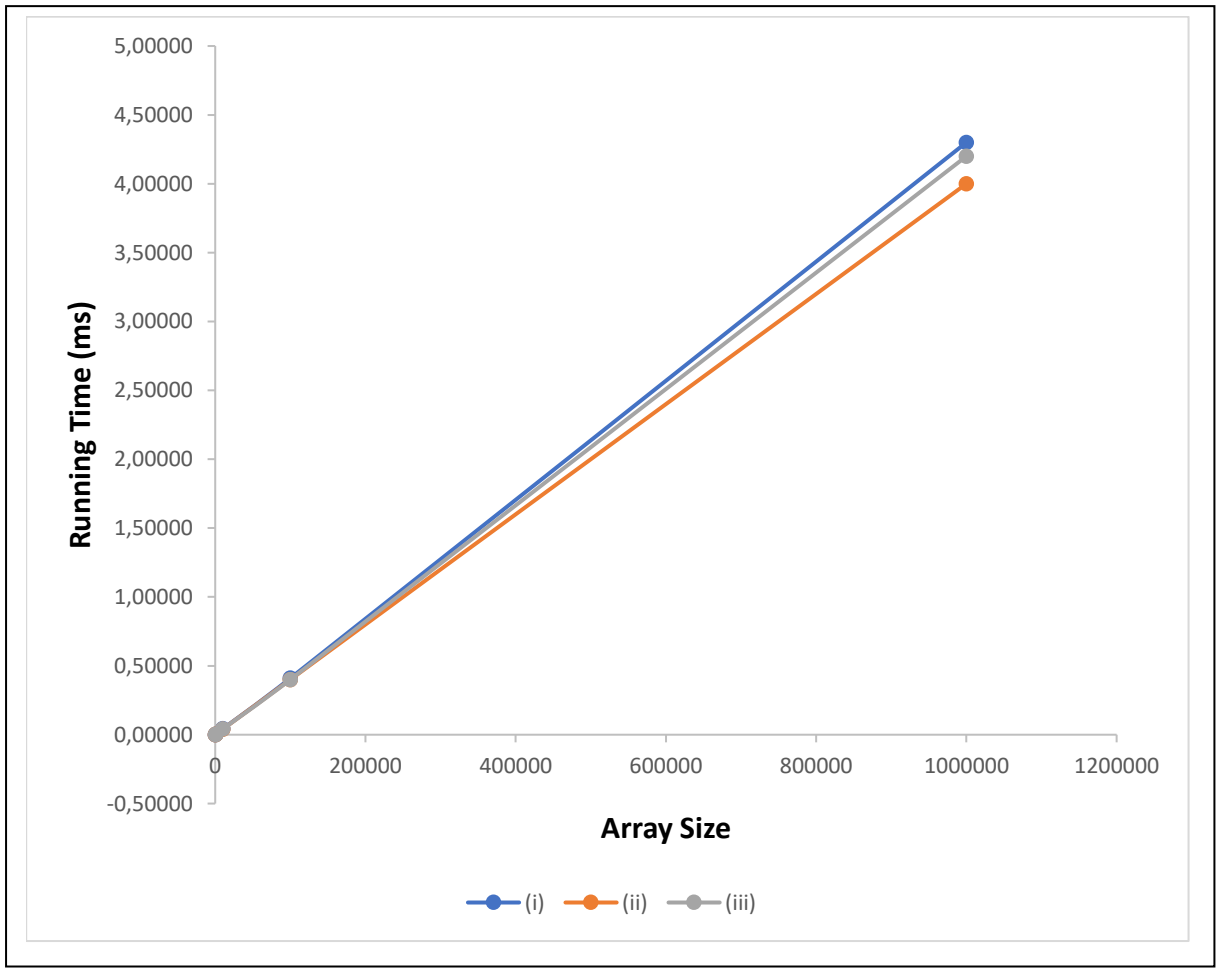
- Table 1: Time values in milliseconds to perform **algorithm 1** in different array sizes depending on the relation between arr1 and arr2. (in (i) all numbers in arr1 are smaller than arr2, in (ii) all numbers in arr2 are smaller than arr1, and (iii) there is no such ordering between these arrays)



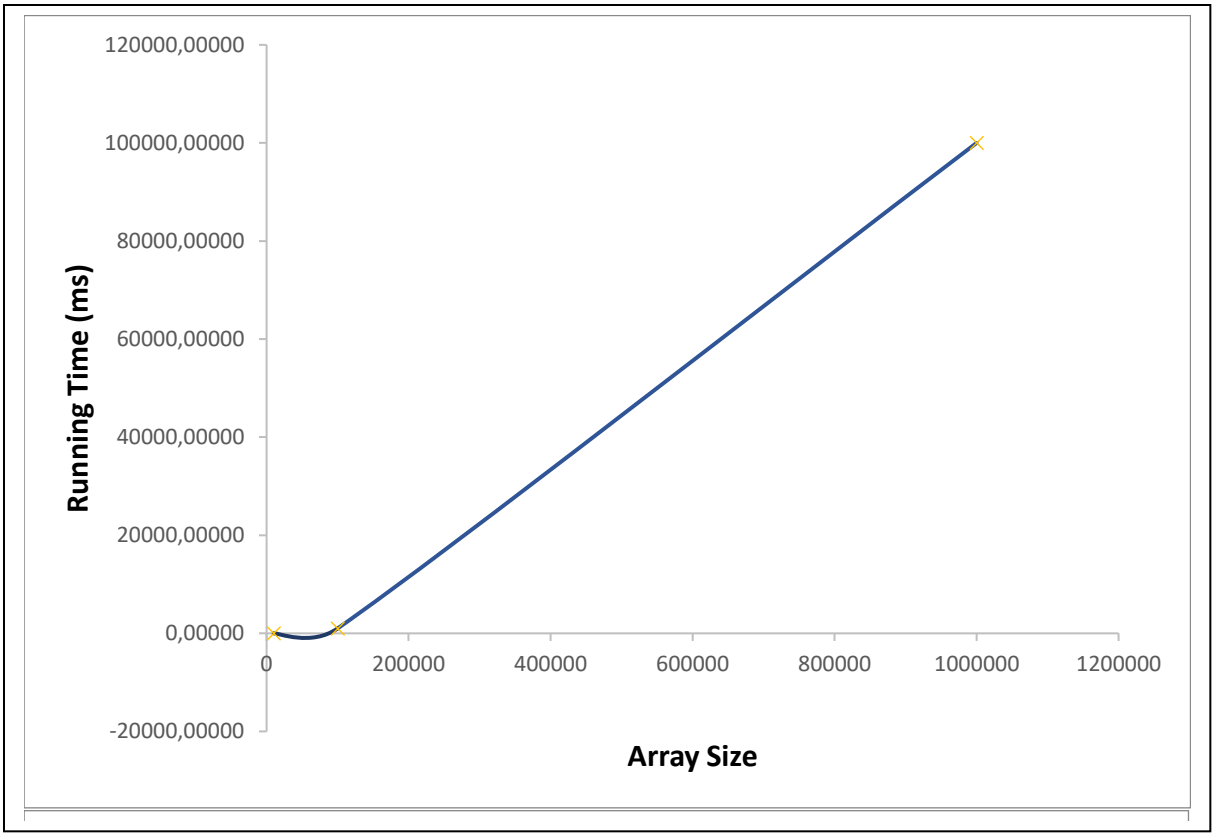

- Figure 1: Graph of experimental time values for **algorithm 1** according to the relation between arr1 and arr2.

| Array Size | (i) | (ii) | (iii) |
|---|---|---|---|
| 10 | 0,00020 | 0,00020 | 0,00020 |
| 1000 | 0,00500 | 0,00400 | 0,00400 |
| 10000 | 0,04300 | 0,04000 | 0,04200 |
| 100000 | 0,41000 | 0,40000 | 0,40000 |
| 1000000 | 4,30000 | 4,00000 | 4,20000 |

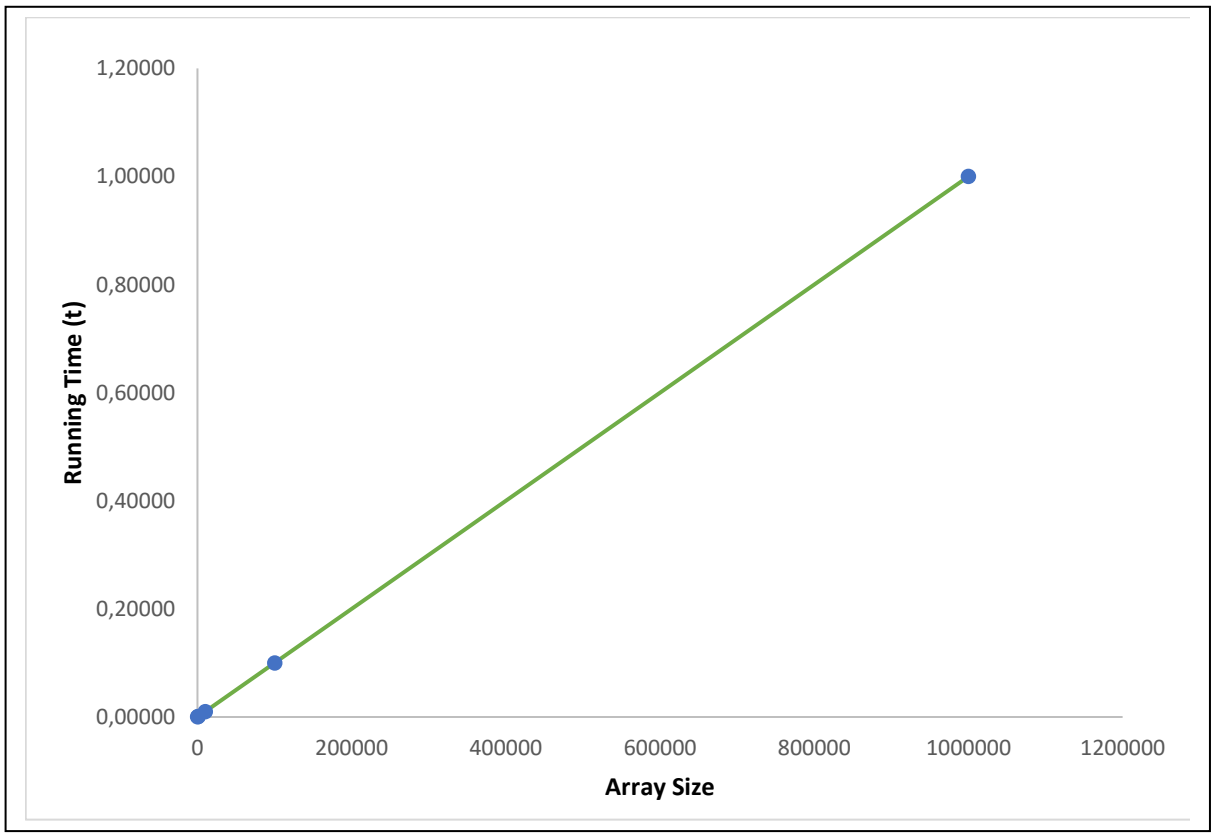
- Table 2: Time values in milliseconds to perform **algorithm 2** in different array sizes depending on the relation between arr1 and arr2. (in (i) all numbers in arr1 are smaller than arr2, in (ii) all numbers in arr2 are smaller than arr1, and (iii) there is no such ordering between these arrays)




- Figure 2: Graph of experimental time values for **algorithm 2** according to the relation between arr1 and arr2.




➢ Figure 3: Graph of theoretical growth rate for **algorithm 1 (**with expected worst-case growth rates).




➢ Figure 4: Graph of theoretical growth rate for **algorithm 2 (**with expected worst-case growth rates).

## *Discussion:*

I have made three experiments, and observed the runtime in which:

i. all numbers in arr1 are smaller than arr2
ii. all numbers in arr2 are smaller than arr1
iii. there is no such ordering between these arrays

for each of the two algorithms, those were:

1. Append all items in arr1 in the same order to arr3. Then, for all items in arr2 starting from the beginning, find the right place to insert in arr3 by shifting the items in arr3 if needed.

2. Compare pairs of numbers across arr1 and arr2 and insert the smallest to arr3. In this algorithm, it is allowed to visit every item only once.

I have ended up with results for these two algorithms which are shown in tables 1 & 2 on the previous pages. I have also plotted graphs for the data in the tables to visualize the data (figure 1 & 2) and observe the structure, then make implications about the growth rates of these two algorithms. Additionally, each of the experiments (i, ii, iii) conducted for the data arrangements in arrays made it possible to determine the best, average, and worst-case scenarios for each of the two algorithms.

### *Algorithm 1:*

To evaluate algorithm 1 in terms of its growth rate, I have evaluated the graph that I have ended up with, constructed with the experimental data, and contrasted it with the theoretical graph (figure 3). A special note that I should make here is that these graphs (figure 1 & 3) has the values for the given sizes of the arrays, and not for each size of the possible arrays. Therefore, the graphs are only estimations (the best fit), therefore the graphs do not show the shape of the ideal (theoretical) graph.

After my evaluation, I have seen that the graph (figure 1), and values of the experiment fit the shape of the mathematical graph of $y = x^2$ best, therefore the growth rate of the algorithm is $O(N^2)$, where N is the size of the array. As for the best, average, worst-case scenarios, it can be interpreted from the graph, that the best case for this algorithm is the (ii) case, all numbers in arr2 are smaller than arr1. This is surprising, as when I have analyzed the algorithm, the (i) case, where the numbers in arr1 are smaller than arr2, should be the best case, as because the items in arr1 are smaller than of those in arr2, there would be no shifts required, but what I have seen in the algorithm is that in any case the items must be traversed, so that might be the reason of this part of the experiment. The average case is the (i) case according to the experiment, where, as explained, where the numbers in arr1 are smaller than arr2. The worst-case is the (iii) case, where there is no ordering between the arrays, this is not very surprising, as there are many shifts in this case, but in my analysis, the worst-case should have been the case (ii), in which all the items in arr3 must be shifted.

### *Algorithm 2:*

Similar to what I have done in evaluation in algorithm 1, for algorithm 2 in terms of its growth rate, I have evaluated the graph that I have ended up with, constructed with the experimental data, and contrasted it with the theoretical graph (figure 4). Similar to algorithm 1, the graphs (figure 2 & 4) have the values for the given sizes of the arrays, and not for each size of the possible arrays. Therefore, the graphs are only estimations (the best fit), therefore the graphs do not show the shape of the ideal (theoretical) graph. But the graph for algorithm 2 appears to be linear, so the predicted shape of the graph should not be a concern this time.

After my evaluation, I have seen that the graph (figure 2), and values of the experiment fits the shape of the mathematical graph of $y = x$ best, therefore the growth rate of the algorithm is $O(N)$, where N is the

size of the array. For the best, average, and worst-case scenarios, what the graph tells is that the case (i), where all numbers in arr1 are smaller than arr2, is the worst case, the case (iii) there is no ordering between the arrays is the average case, and the case, where (ii) all numbers in arr2 are smaller than arr1 is the best case. But in my theoretical evaluation, I have seen that in this algorithm, actually there should be no best, average, and worst-case scenarios, as the algorithm looks at the items in two arrays simultaneously. Furthermore, when we take a look at the data in table 2, it can be seen that the differences between time values are very small and negligible, also some values are the same. We can conclude that the differences causing a distinction for the graph to have best, average, and worst cases are caused by these small differences, which may have been caused by some operations like input, output, etc. which we are considering to have an O(1) time complexity, which in reality take more time.