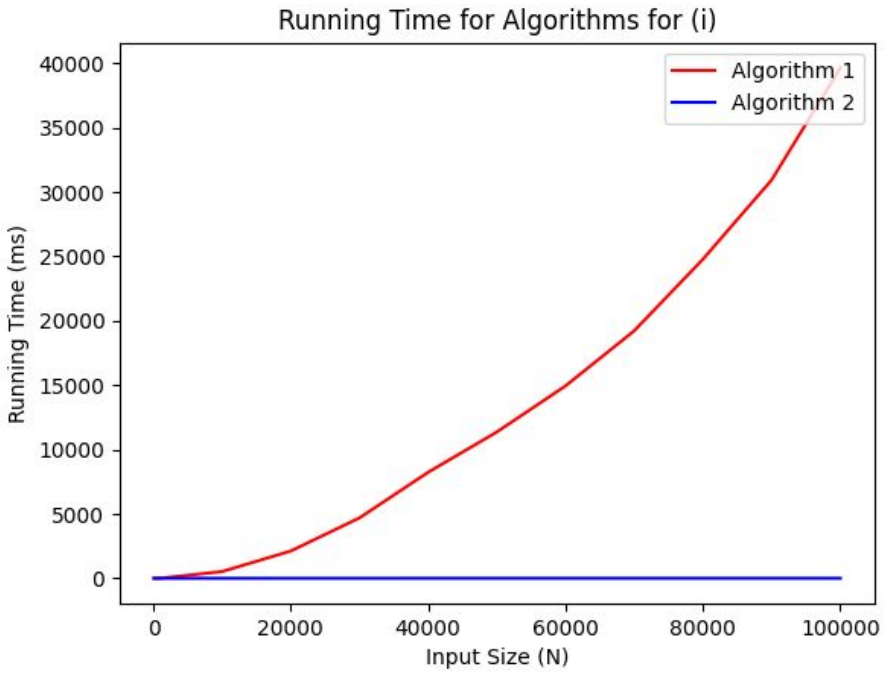
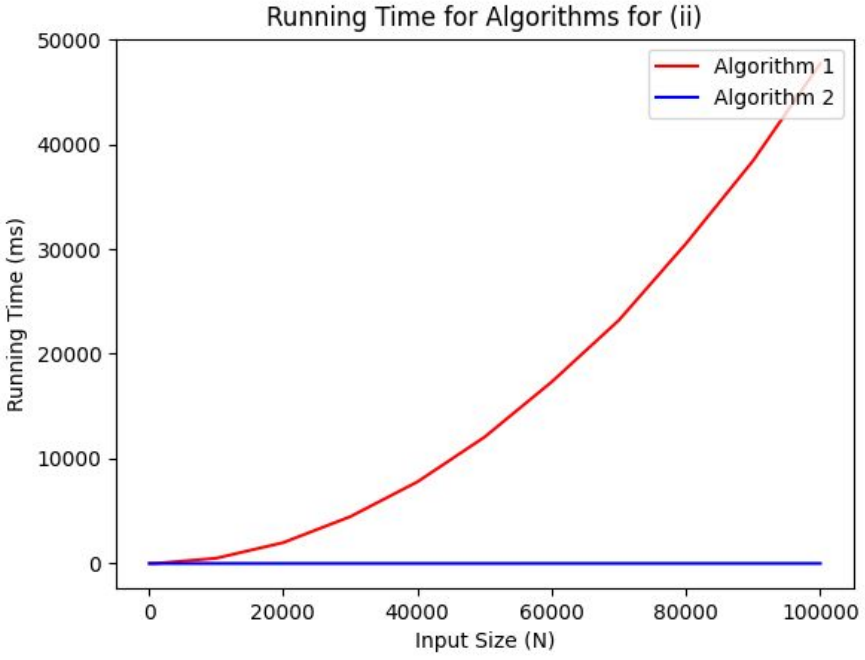
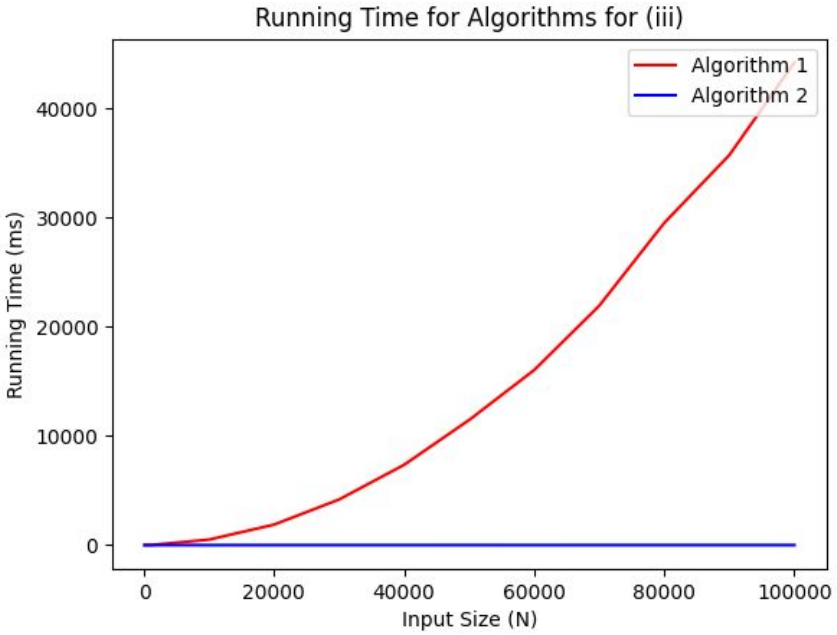
Berk Saltuk Yılmaz

Homework 2: Analysis Report

2. Plots of running time (y-axis) versus the input size N (x-axis) for (i), (ii), and (iii):







3. <u>Best, average, and worst cases for each algorithm and the corresponding worst-case time complexity:</u>

<u>Algorithm 1:</u>

The best case is where there is no need for shifting, in this situation all elements of the first array will be located into the third, and then the second array's elements will be located directly after the elements of the first array. This corresponds to the (i). 2 out of 3 nested loops will be executed in this situation.

In the average case, sometimes shifting will be required and other times it will not be required. This is possible when there is no order between the elements of the first and the second arrays. This corresponds (iii).

The worst case is where all elements of the second array are lesser than the first array because this will cause shifting for all elements. This might correspond (ii). However, in the nested for loops, the second and the third for loops will iterate the total size 2*N because if the second for loop iterates until a random number a, the third for loop iterate 2N - a times and break statement will be executed. This means 2*N iterations will be done.

The time scale of graphs also supports these ideas since the maximum running times can be ordered as (ii) > (iii) > (i).

As mentioned above in the worst-case firstly N iterations will be done, then, the nested loops will be executed. However, these 3 for loops will be executed N * 2N at most. Thus, it can be said that the worst-case time complexity will be $O(N + N \cdot 2N) = O(N^2)$. In other words, it can be said that it will have N * N time complexity.

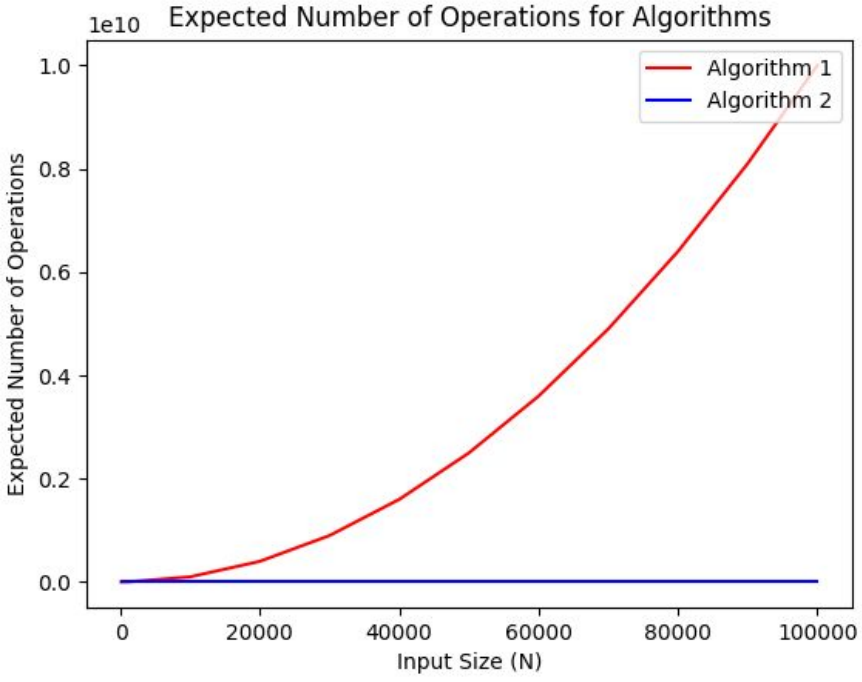
Algorithm 2:

In this algorithm, there are no strict differences for the best, average, and worst cases. In each situation, the smallest elements will be located first and others will be added. In other words, the first while loop will be executed N + N - 1 time at most. Then, one of the other while loops will be executed N times at most. In all cases, it will be N + N. This can be obtained from the graphs since there are no strict differences between (i), (ii), and (iii). Thus, it can be said that the worst-case time complexity of this algorithm is $O(N)$.

4. Specifications of the system:

I have obtained the results on a computer that has Windows 10, an Intel Core i7-9750H processor, 256GB SSD, 16GB RAM, and Nvidia Geforce GTX1650 GPU.

5. Using the theoretical analysis in (3), for each N value plot the expected number of operations:



6. Comparison:

It can clearly be said that the graph I have obtained in (5) is coherent with the worst-case time complexity results I have found in (3). Moreover, the expected number of operations graph and the graphs I have plotted in (2) are similar. I can conclude that the running time of algorithms increases at the same rate as the expected growth rate of the number of operations. In (2) I have observed that as the input size increases the running time of Algorithm 1 increases as the square of the change in output size. For example, if the input size is doubled, the running time increases as the square of it, i.e. 4 times. Moreover, Algorithm 2 is always linear and the graph in (5) and the graphs in (2) support the results I have found in (3).