|  |  |
|---|---|
| **Hacettepe University** |  |
| **Computer Science and Engineering Department** |  |
| **Name and Surname** | : İdil CAN |
| **Identity Number** | : 21727011 |
| **Course** | : BBM204 |
| **Experiment** | : Assignment 1 |

**INDEX**

**Topic** ........................................ **Page No**

## 1. **PROBLEM DEFİNİTİON**

In this assignment, we needed to measure the time it needs to execute. In order these data, we plot a graph and get the approximate complexities of the algorithms. We examined five algorithms: Selection sort, insertion sort, merge sort, radix sort and binary search.

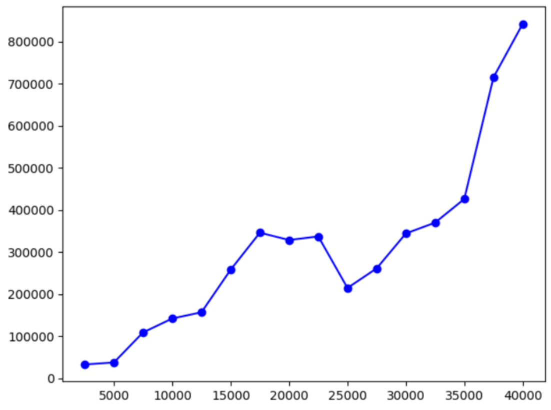
The main idea was that there are different complexities for one algorithm. Best case, average case and worst case. Each one happens in different situations.

## 2. **FINDINGS**

For the graphs that shown below: the values on the left are time in microseconds and the values on the bottom are the size of the arrays.

The best case for *sorting algorithms* is when input array is sorted. The average case is when input array is random and the worst case is when the input array is reversed.

### 2.1. Selection Sort

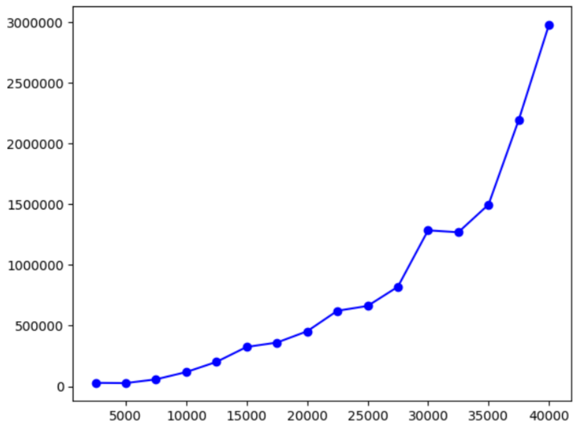


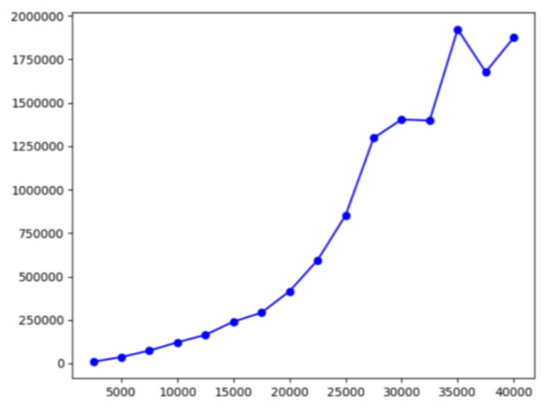
BEST CASE:

The graph on the left is the graph of the best case of the selection sort. Even, it's hard to see, we can observe that it's similar to $y = kx^2$ graph.

AVERAGE CASE:

The graph on the right is the graph of the average case of selection sort. It's easy to see that it's almost identical to $y = kx^2$ graph.





WORST CASE:

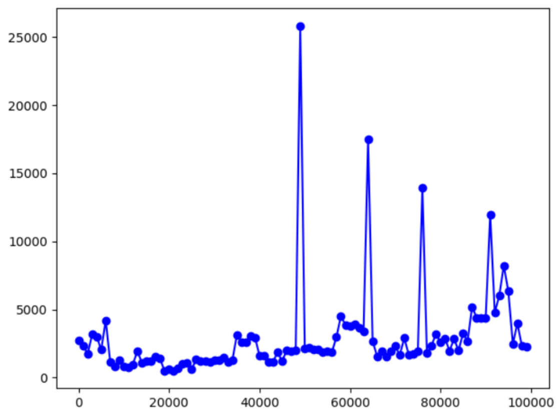
The graph on the left shows worst case of the selection sort. It’s obvious in smaller sets for inputs but it’s not easily seen in great sizes of inputs. The reason behind is probably about the rate of randomness. The graph’s formula is $y = kx^2$.

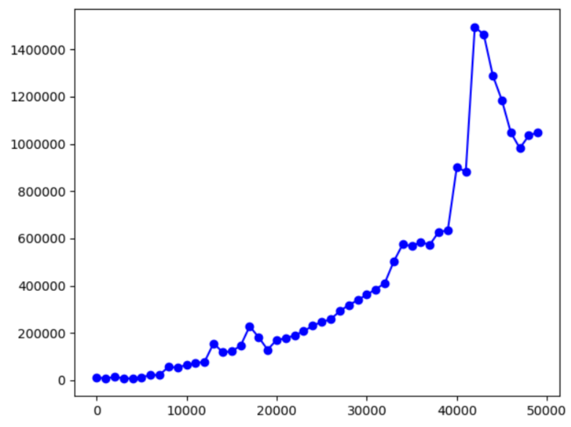
We may say that for selection sorts complexity is $N^2$, no matter case: Best, average or worst. So, we can say that selection sort works at $O(N^2)$.

### 2.2. Insertion Sort

BEST CASE:

The graph that shown left is for the best case of the insertion sort. The reason of those which are separate is probably because of my computer’s task management. If we discard those divergent, we see a graph that looks like a linear graph. We can assume the graph $y = kx$.



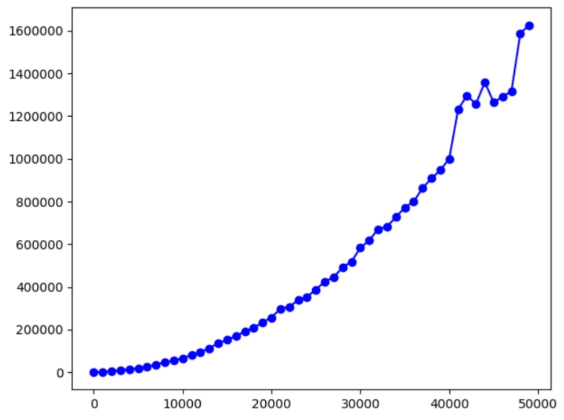


AVERAGE CASE:

The graph on the left shows the average case scenario for insertion sort. As we can see in the graph there is a quadratic equation. The graph converges to $y = kx^2$.

WORST CASE:

The graph on the left shows the worst-case scenario for insertion sort. This is obviously a quadratic equation like y = $kx^2$.

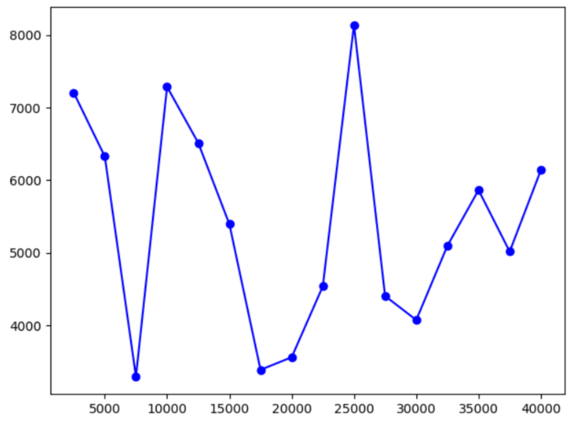


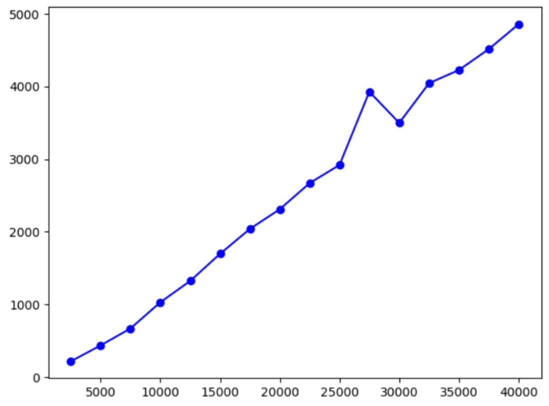
In the light of these graphs we may say that Insertion sort works at O(N) at best case. In other cases, insertion sort works at $O(N^2)$.

### 2.3. Radix Sort

BEST CASE:

This graph is the best case of radix sort. It may look like a random graph, when you take average, it is close to y = kx equation. The reason behind this random vision is the task management of the computer.



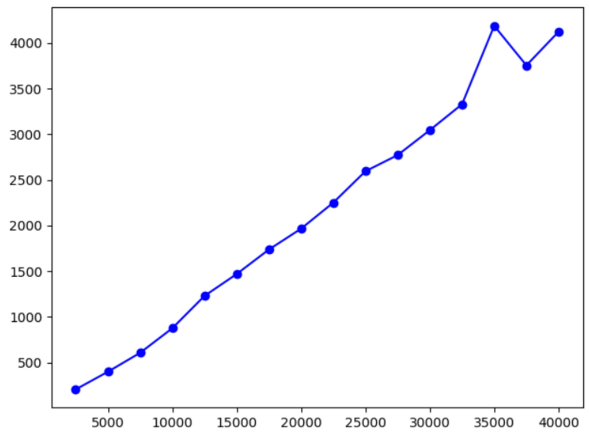


AVERAGE CASE:

This graph shows average case of the radix sort. As it can be seen this is also close to y = kx equation.

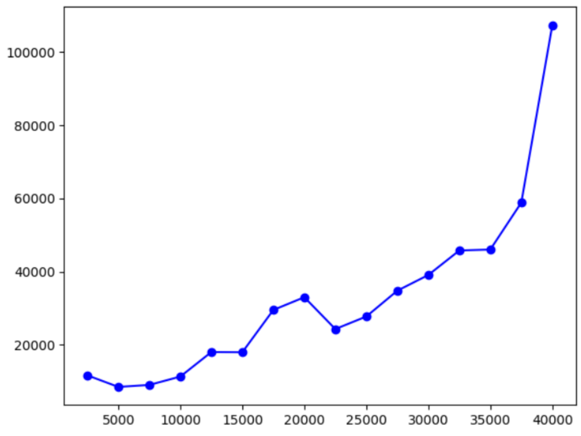
WORST CASE:

This is the graph of the worst-case scenario of the radix sort. As it can be seen, it's close to the y = kx graph.



In radix sort, it doesn't matter if the given case. It's always works on O(N).

### 2.4. Merge Sort

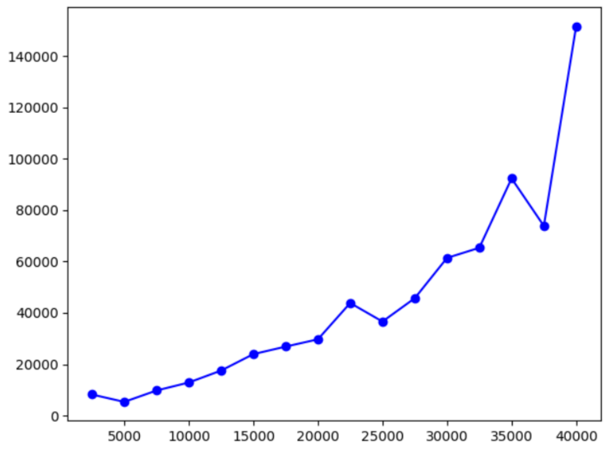


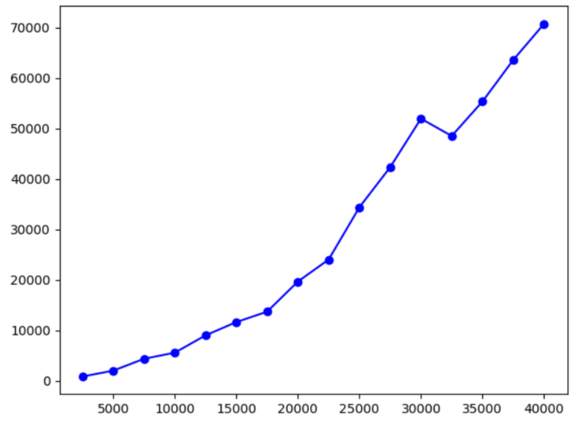
BEST CASE:

As it can be seen from the graph. It might look like an quadratic equation but it should look like y = k.logx equation. The difference is from the flaw in accuracy and the computers task management.

AVERAGE CASE:

The graph may not be looking like a proper y = k.logx equation but actually if we draw an imaginary line from the average place, it's looking like y = k.logx.





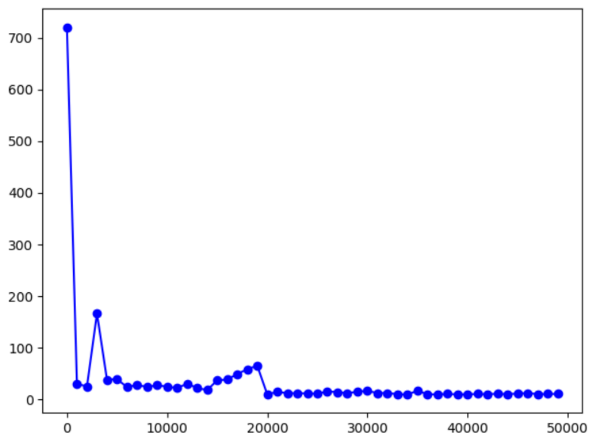
WORST CASE:

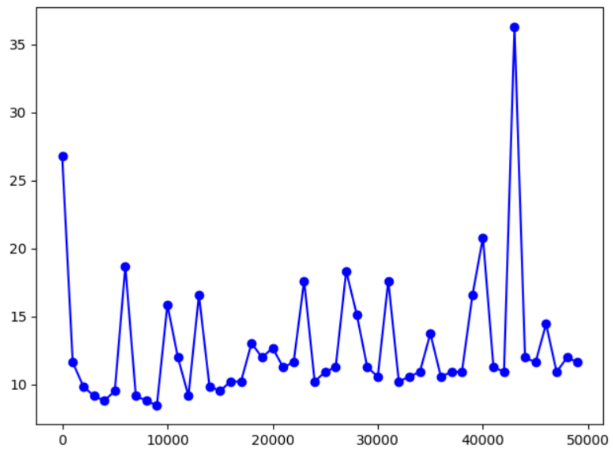
The worst case scenario is the most approximate graph of all cases. It’s close to y = k.logx curve. It’s not same because of the tasks my computer does at the time.

### 2.6. Binary Search

BEST CASE:

This graph might be looking a bit off but actually the best case scenario for binary search is the one we’re looking for the middle element and it’s y = 1 equation. Since my computer has to do task managements there is ups and downs in this graph.



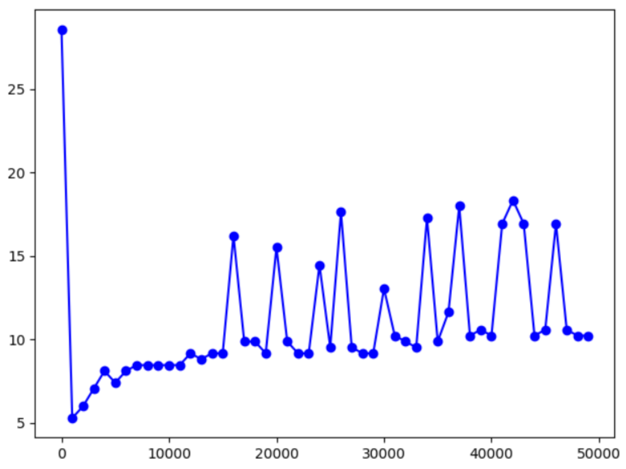


AVERAGE CASE:

The average case for the binary search is the one that given random number. The answer should be y = x.logx. But the computer’s task management and the searched element’s position may change the execution time a lot.

WORST CASE:

Worst case scenario for the binary search algorithm is looking for an element that does not in the array. In other words, it doesn’t exist. When we discard the divergent values that is based task management, this graph is close to y=x.logx.



For binary search, in best case, it works at O(1). In other cases, it works on O(N.logN).