

Marmara University Faculty of Engineering

CSE 2046 Analysis of Algorithms, Spring 2022

Homework 2

Instructure: Assoc. Prof. ÖMER KORCAK Due Date:11.05.2022

|  |  |  |  |
| --- | --- | --- | --- |
|  | Department | Student Id Number | Name & Surname |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |

**Purpose**

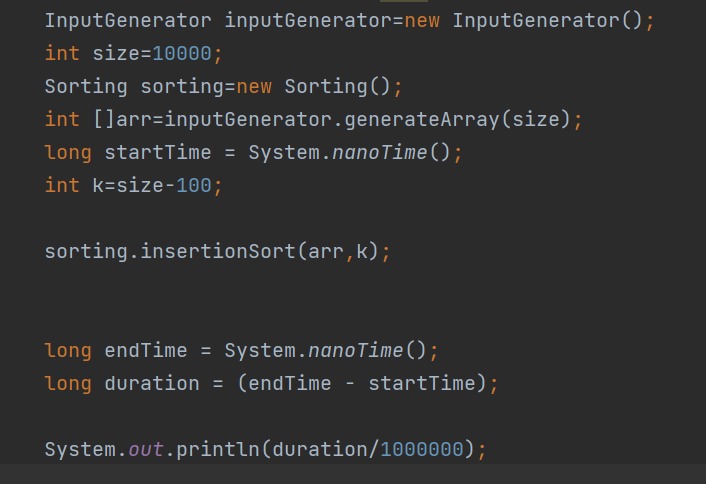
To investigate and compare some basic algorithms and custom algorithms in terms of time, space and input size relationship.

**Designing Experiment**

During designing the experiment we decided to use millisecond(**ms**) as a metric parameter. To calculate execution timing, we used java libraries that measure the running time of an algorithm. and as input, we perform measurements multiple times and the same input size then, we calculate the average of them. For example, we perform measurements for insertion sort same input size 10 times and we dived the result by 10. In each measurement, we perform different combinations to deeply investigate algorithms such as that random number generator, unique order, etc.

**Imported Codes**

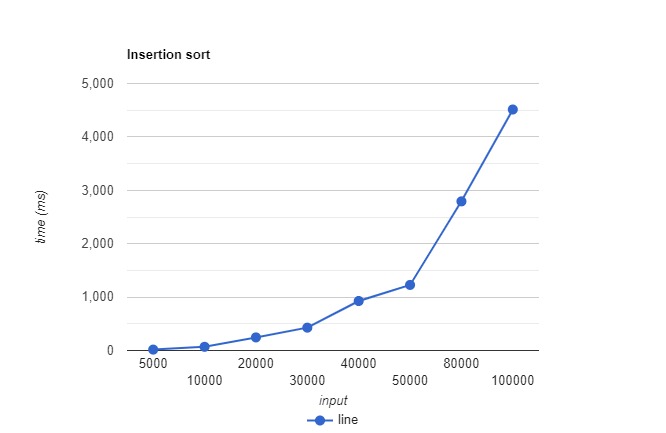




**A)** **Algorithms**

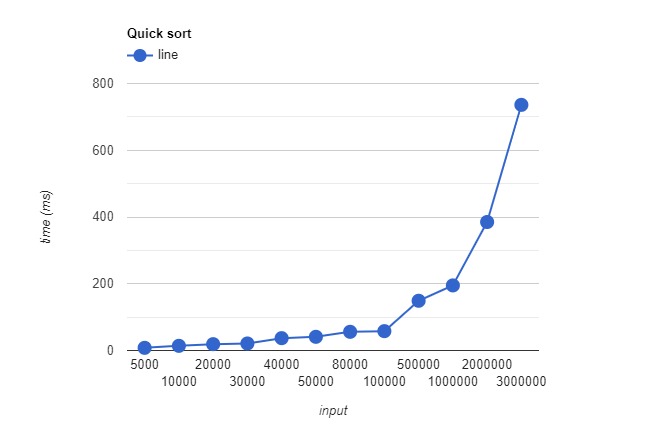
1)Insertion Sort

Insertion sort is one of the worst algorithms among these algorithms in terms of time complexity. And we observed how this algorithm is slow. But in terms of space complexity, it may be the most efficient algorithm. During our measurement, we saw that in the same input size with different cases, the algorithm execution time difference is high. And we observed that when input size increasing time complexity increasing upwardly.



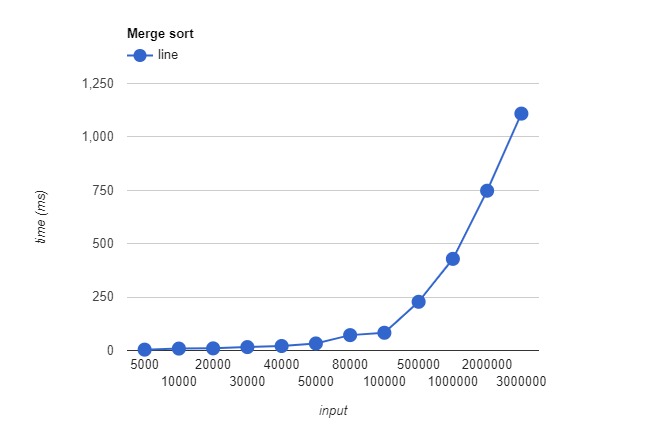
2)Quick Sort

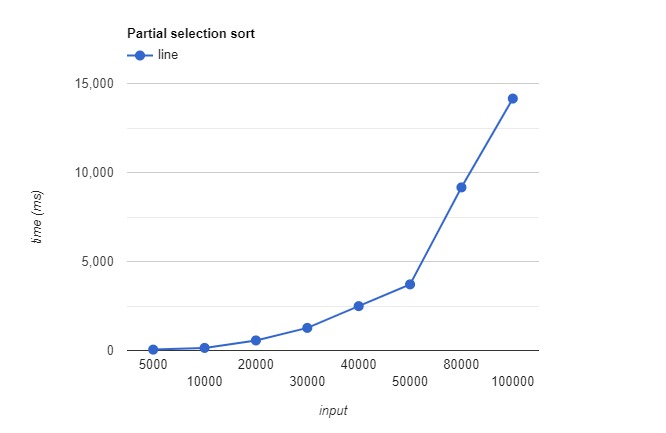
Quicksort is one of the fastest algorithms among these algorithms in terms of time complexity. We clearly say that this algorithm is efficient according to the time. the algorithm execution time difference is low. Seen that graph is increasing more smooth



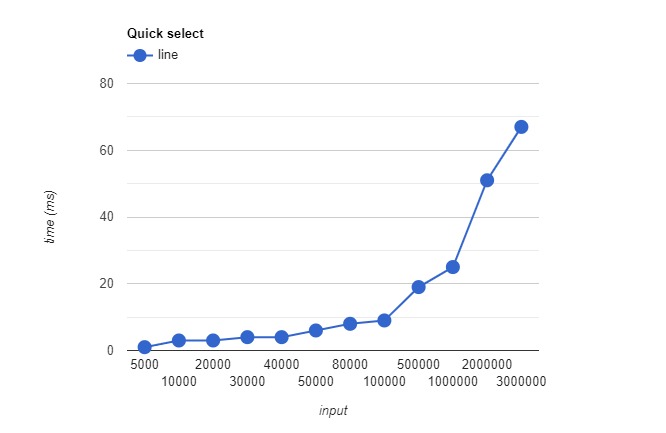
3)Merge Sort

Merge sort most efficient algorithm we have seen so far in the lecture. It based on divide-and-conquer. In our experiment observed that it is the fastest algorithm.

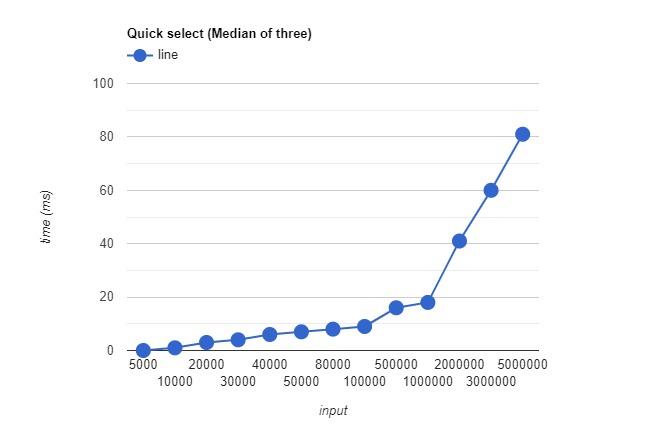


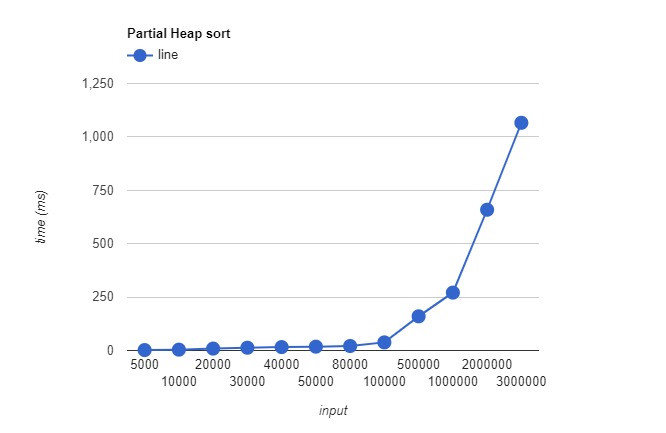
4)Partial Selection Sort 

5)Quick Selection



6)Quick Selection(median of three)



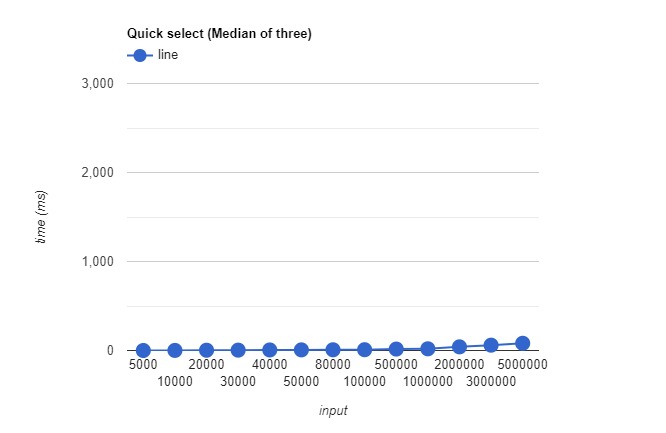
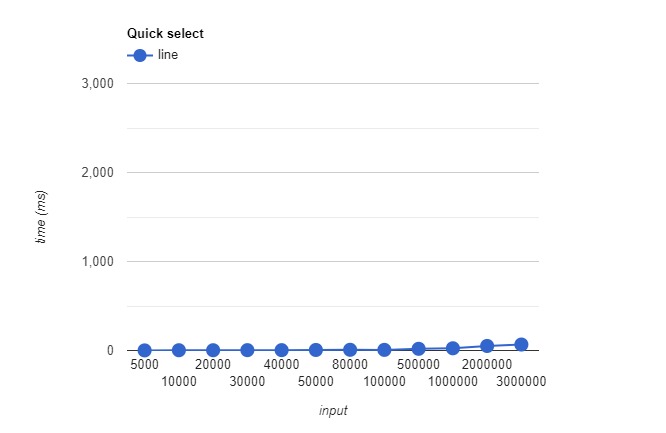
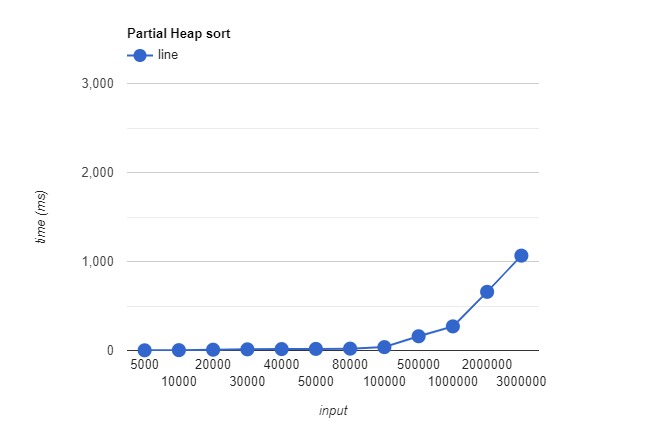
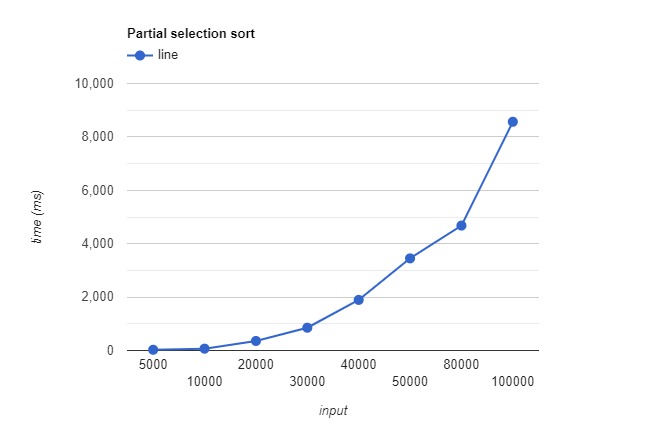
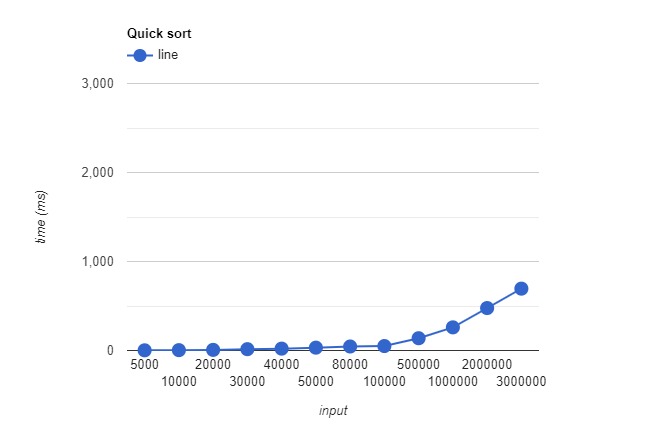
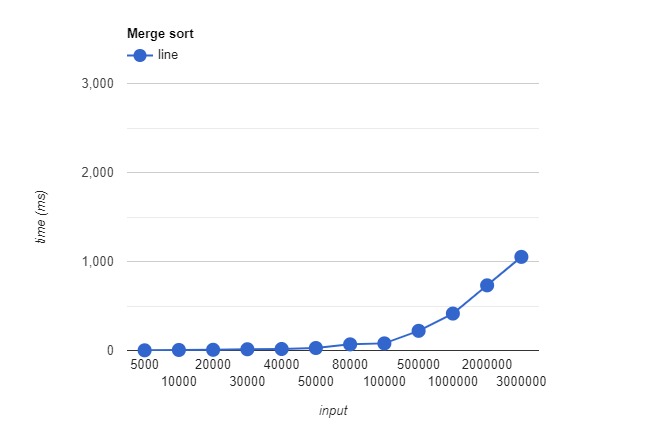
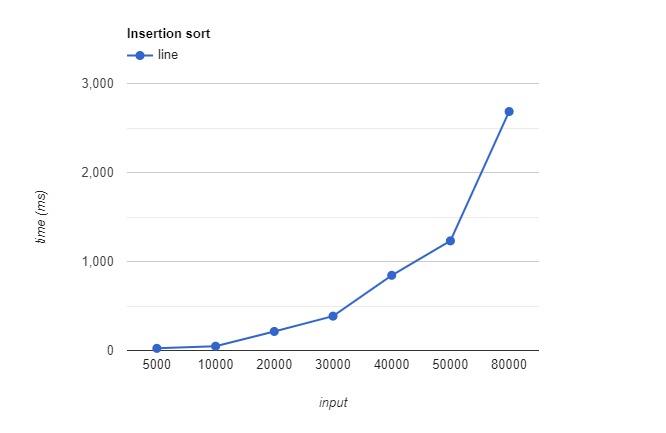
7)Partial Heap Sort

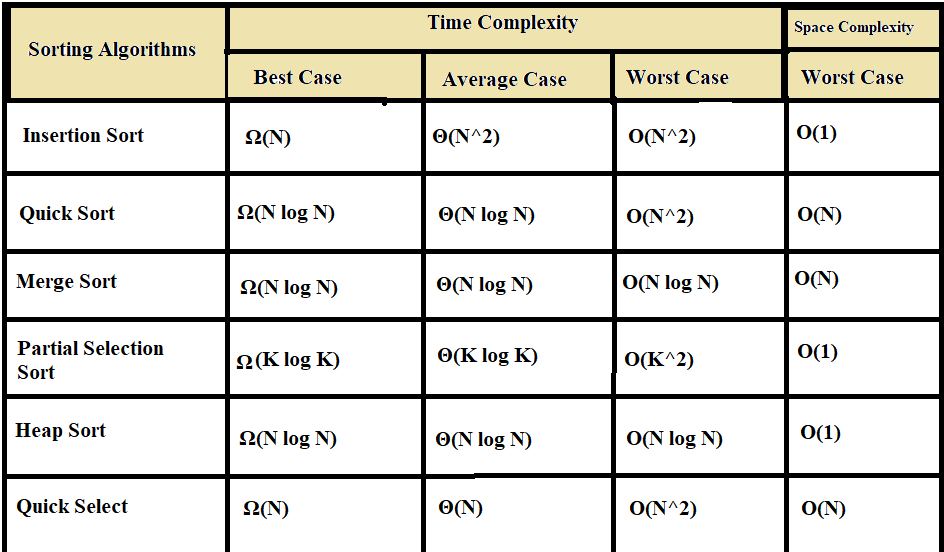
B) Comparison

First, we calculated half of the input as k value and we measured time. After that, we calculated input/4 for k value end input -100 to calculate some k value near beginning, middle and last values as k value. We aimed to calculate how k value affects the result as time efficiency. There is no big difference between

k values but when k values decrease also compiling time decreases accordingly and vice versa. Because of the small differences between different k values we only put n/2 and n/4 plots.

For k =s/4 time complexity graph changes like;





For us, theoretical expectations are mostly satisfied but in some measurements, we observed that they are doesn't match theoretical expectations. For example, the same input size and same kth value quicksort executing time are better than merge sort and occurred some time gap between them.

In terms of quick selection in the big size input, when we choose pivot as a three-element of median it is more efficient than normal quick selection. But in the lower input size, we had near results between them.

Mostly we don't expect such results. We should not blame theory for its wrong, running time affecting most of the things in the computer. The place where we are running this experiment is not completely insulated. We did an experiment for this. We try the same algorithm and inputs and the result was different.