Istanbul Technical University

Department of Computer Engineering

BLG 335E Analysis of Algorithms
Assignment 1 Report

Ece Naz Sefercioğlu
150130140

A. Asymptotic Upper Bound on Running Time For:

Insertion Sort Algoritm: $\Theta(n^2)$ [1]

Merge Sort Algoritm: $\Theta(n.\log n)$ [2]

Linear Search Algoritm: $\Theta(n)$ [3]

B. Test Results

Merge Sort

N\K	1	2	10	N/2
10	3.6e-05	5.8e-05	4.6e-05	2.9e-05
100	0.000483	0.000469	0.000467	0.000465
1000	0.002274	0.002136	0.002017	0.002021
1000000	2.96077	2.9811	2.96401	2.95534

Linear Search

N\K	1	2	10	N/2
10	2e-06	2.3e-05	2e-06	7e-06
100	4.5e-05	6.9e-05	8.7e-05	0.000119
1000	0.000405	0.00053	0.000912	0.00536
1000000	0.104623	0.135346	0.274439	1257.11

Insertion Sort

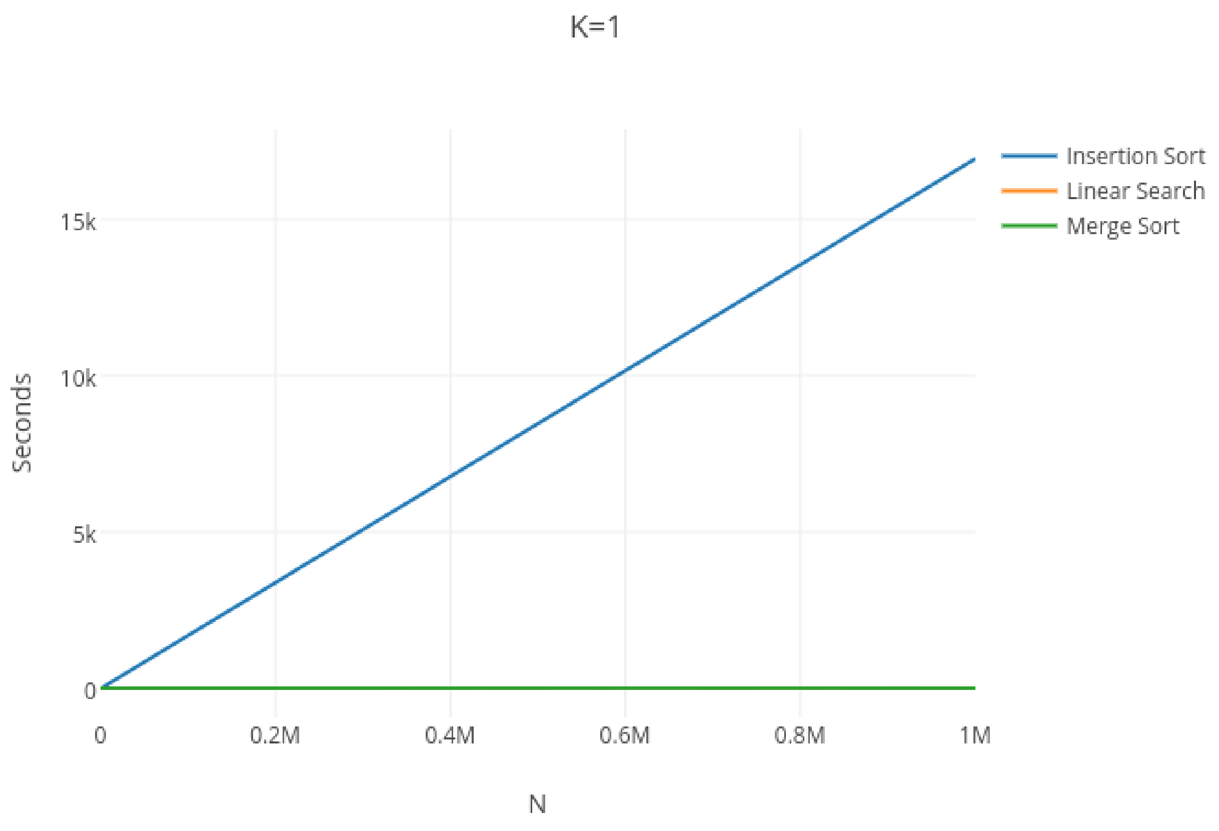
N\K	1	2	10	N/2
10	4e-06	1.6e-05	1.2e-05	1.2e-05
100	0.000753	0.000712	0.000712	0.000712
1000	0.044569	0.019336	0.017668	0.018655
1000000	16940.5	16850.6	16973.3	16744.9

C. Plots

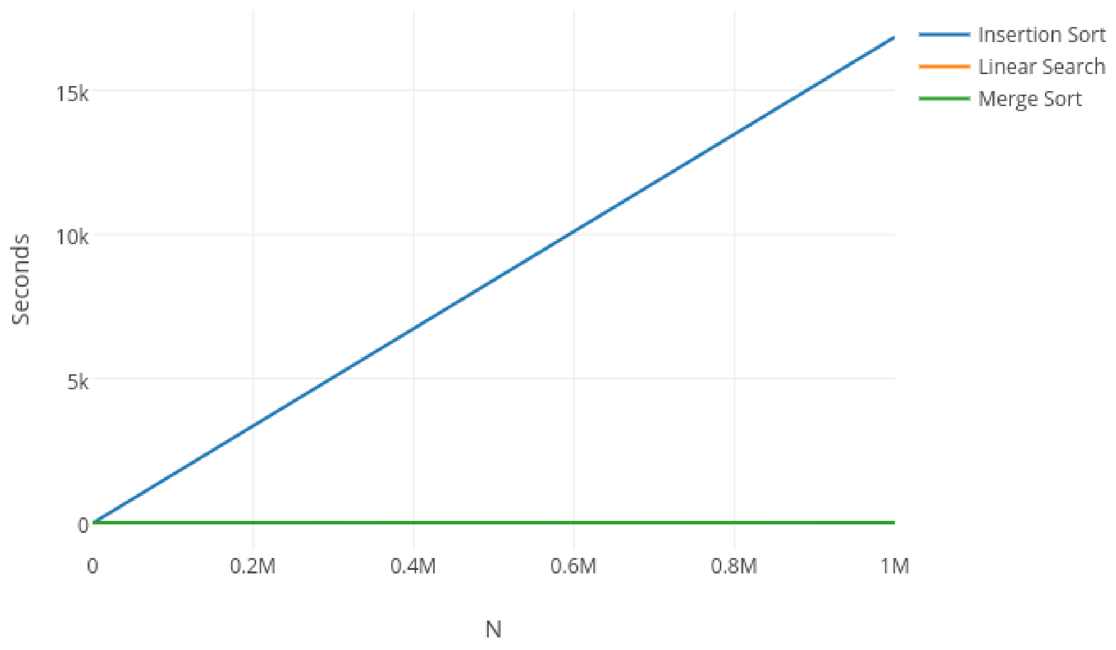
Running time of linear search and merge search is close, and running time of insertion sort is way larger especially on 1M data set, lines of merge sort and linear sort overlaps on cases, K=1, K=2, K=10. On these cases merge sort takes longer to finish than linear search.

When it is zoomed in, it is seen that while the time consumption of linear search is increased at the degree of its plot, merge sort carried more stable degree on its plot line.

Shortly, the larger the K value the larger the degree linear search line has.

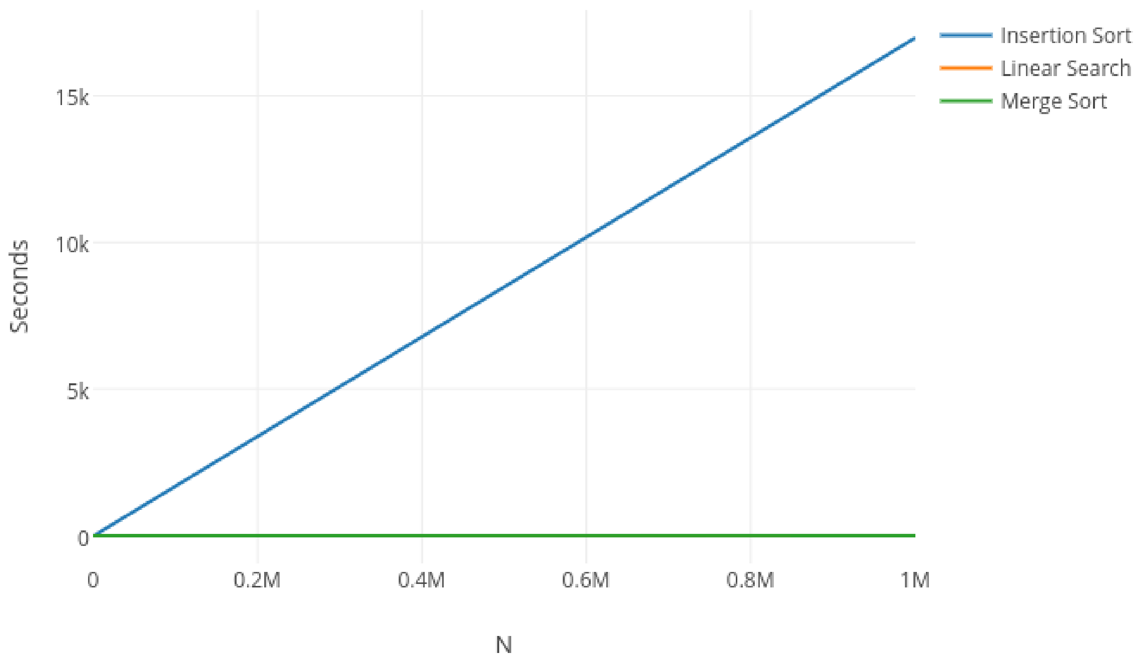


K=2

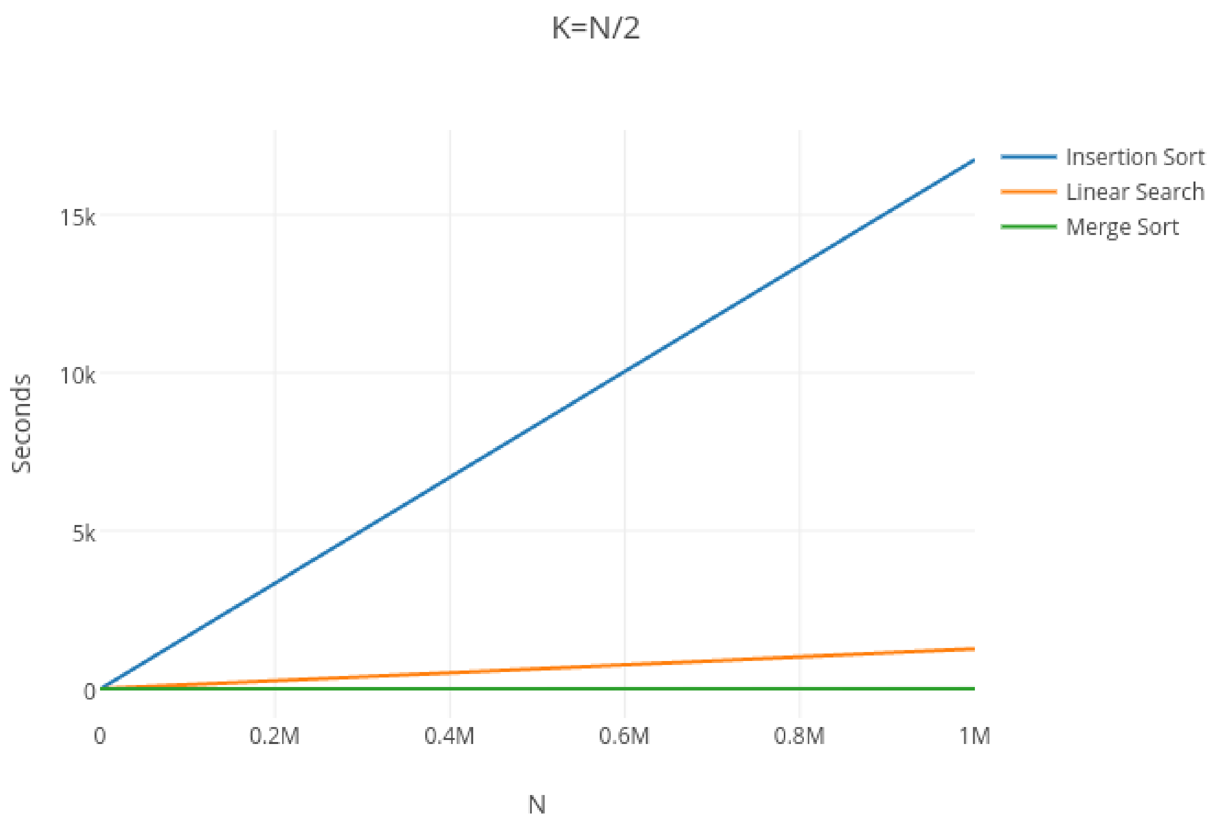


Insertion Sort
Linear Search
Merge Sort
15k
10k
5k
0
Seconds
0
0.2M
0.4M
0.6M
0.8M
1M
N

K=10



Insertion Sort
Linear Search
Merge Sort
15k
10k
5k
0
Seconds
0
0.2M
0.4M
0.6M
0.8M
1M
N



Plots were drawn on Plot.ly [4]

After the mathematical comparison of results, optimized outcome is shown on below table.

N\K	1	2	10	N/2
10	LS	IS	LS	LS
100	LS	LS	LS	LS
1000	LS	LS	LS	MS
1000000	LS	LS	LS	MS

From the table we can see linear search algorithm dominates most of the slots, having simple complexity. We also should take into consideration that with small values of N the results are closer.

When K gets larger, linear search gets slower, because the time spent on exchanges and search gets larger, leaving merge sort algoritm to be an optimum algorithm for big N and K values among others.

For smaller values we should chose linear search. Only when we want to find smallest values in an array, with no need to sort.

This table takes only sorting algorithms into account:

N\K	1	2	10	N/2
10	IS	IS	IS	IS
100	MS	MS	MS	MS
1000	MS	MS	MS	MS
1000000	MS	MS	MS	MS

If we want to have a sorted list with smallest values, we can not use linear search algorithm. Only the sorting algorithms should be implemented.

The table shows us insertion sort is better suited for small number of data sets to sort. While it seems merge sort has a wider data range to be more time efficient.

References

[1]:https://www.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/analysis-of-insertion-sort
[2]:https://www.cs.princeton.edu/courses/archive/spr10/cos226/lectures/05-22Mergesort-2x2.pdf
[3]:http://www.inf.ed.ac.uk/teaching/courses/dmmr/slides/13-14/Ch3.pdf
[4]:https://plot.ly/