

# İTÜ



## BLG335E, Analysis of Algorithms 1

<b>Name:</b>	İbrahim KARAHAN
<b>Student Number:</b>	150160550
<b>CRN:</b>	10983
<b>Project 1</b>	

# Contents

1) Introduction .....	3
2) Analyze .....	3
Part A .....	3
Insertion Sort .....	3
Merge Sort .....	3
Part B.....	3
Part C.....	6
Part D.....	7
3)Conclusion.....	7
4)Compilation.....	7

## 1. Introduction

Project aim is writing merge and insertion sort and analyze running time complexity of these two algorithm. Firstly, I read the data from the file which name is log\_inf.csv and keep this value in to the object and send each object to the vector. I wrote two insertion sort, two of them are totally same but one of them is last\_price and the other one is time\_stamp. Also I wrote two merge sort, two of them are same but one of them last\_price and the other one is for time\_stamp value. I wrote a function for writing to the file and file name is sorted.csv. When I write the merge and insertion sort I used our lecture slide as a references and I used the pseudo code of lecture slide. I try to write my own code according to the pseudo code. Also I used the formula and the other necessary thing from the lecture slide.

## 2. Analyze

A)

At this part of project, I try to analyze algorithm according the formula which is given in the lecture slide. Firstly, I try to analyze insertion sort. In the insertion sort firstly I try to understand the best case. At the best case inner loop is not working which mean that vector already is sorted so best case is  $\Theta(n)$ . In my code I kept the value in object and copying object when it needs. This is a constant value so it did not change the calculation. When we looked at the worst which mean is vector is reversed order or shuffled. At this part every part of algorithm is working so worst case is  $\Theta(n^2)$ . So asymptotic upper bound for insertion sort would be  $O(n^2)$ .

Merge sort is a divide and conquer algorithm. In the merge sort I divide the vector two halves. And sort this halves. Also divide part is working until there is one element rest. After dividing part for each part I call merge sort function. For one of them will take  $T(n/2)$  time. At total  $2 * T(n/2)$  time. Also every call time complexity is  $(\log n)$ . If vector size is  $n$  so we call  $(n * \log n)$  so asymptotic upper bound for insertion sort would be  $O(n * \log n)$ .

B)

At this part, I try to analyze running time of insertion and merge sort algorithm. I used clock function for getting the time. I prepare 4 table and 2 of them for insertion and two of them for merge sort. Each insertion and merge sort separated in two, one of them for last\_price and the other one is time\_stamp.

I used ITU SSH server for merge and insertion sort price. Getting the insertion sort running time, I faced with a problem which is time problem. At SSH server time is not enough to get time value of 900.000 data. SSH gives killed result because of not enough time.

I used my own computer for the other part. Comparing with SSH server my own computer is slower. Because computer has 1.5 GHz.

	1000	10000	100000	900000
1)	0	0,06	0,98	12,03
2)	0	0,06	0,95	11,85
3)	0	0,07	0,89	11,93
4)	0	0,06	0,94	11,97
5)	0	0,08	0,96	11,9
6)	0	0,07	0,89	11,8
7)	0	0,07	0,97	11,87
8)	0	0,08	0,95	11,92
9)	0	0,06	0,96	11,9
10)	0	0,06	0,96	11,87

Figure-1

Merge Sort for last\_price.

Log\_inf.csv file is used for input.

It is compiled at ITU SSH server.

Average time for 1K=0. Average time for 10K=0,07.

Average time for 100K=0,94. Average time for 900K=11,90.

	1000	10000	100000	900000
1)	0.02	2,11	108,4	Killed
2)	0.02	2,09	109,044	Killed
3)	0.02	2,04	107,301	Killed
4)	0.02	2,05	108,301	Killed
5)	0.02	2,06	109,307	Killed
6)	0.02	2,07	108,401	Killed
7)	0.02	2,08	108,55	Killed
8)	0.02	2,07	109,667	Killed
9)	0.02	2,08	108,457	Killed
10)	0.02	2,03	108,741	Killed

Figure-2

Insertion sort for last\_price.

Log\_inf.csv file is used for input.

It is compiled at ITU SSH server.

Average time for 1K=0,02. Average time for 10K=2,06.

Average time for 100K=108,616.

	1000	10000	100000	900000
1)	1,977	21,96	317,556	817,572
2)	2,16	16,904	301,056	779,816
3)	1,629	15,812	311,241	776,254
4)	1,723	17,367	308,254	779,254
5)	2,462	18,35	309,447	774,974
6)	1,903	23,77	307,654	811,456
7)	1,585	16,472	303,147	810,658
8)	1,853	17,308	305,027	787,321
9)	1,501	18,501	308,214	780,547
10)	1,617	19,201	309,521	795,231

Figure-3

Merge sort for time\_stamp.

sorted.csv file is used for input because of log.inf.csv is already sorted.

It is compiled at my own computer.

Average time for 1K=1,84. Average time for 10K=18,564.

Average time for 100K=308,111. Average time for 900K=791,303.

	1000	10000	100000	900000
1)	0,059	1,50	10,635	128,229
2)	0,043	1,47	14	150,738
3)	0,051	1,43	13,507	131,437
4)	0,056	1,43	12,408	145,653
5)	0,064	1,48	11,206	151,81
6)	0,046	1,47	13,447	142,379
7)	0,048	1,46	15,271	144,379
8)	0,047	1,45	13,22	137,647
9)	0,042	1,47	14,334	138,618
10)	0,051	1,46	15,386	145,321

Figure-4

Insertion sort for time\_stamp.

sorted.csv file is used for input because of log.inf.csv is already sorted.

It is compiled at my own computer.

Average time for 1K=0,05. Average time for 10K=1,42.

Average time for 100K=13,386. Average time for 900K=141,621

C)

At this part of analyze, I try to prepare graphic for visualizing the result of average running time. According to knowledge at part we can see easily that in insertion sort if we compare rate of data range and average running time we can see rate is  $n^2$ . We face same result at merge sort.

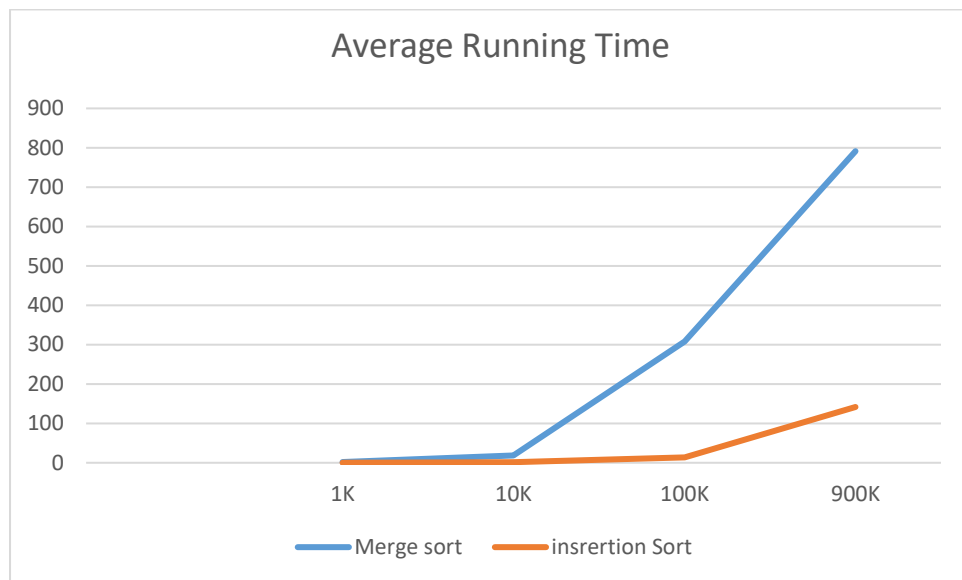


Figure-5

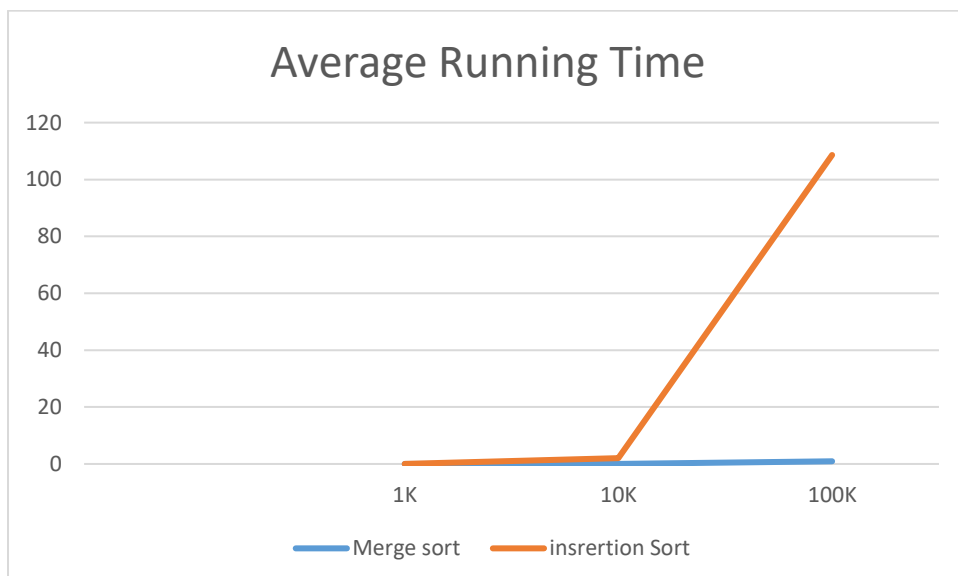


Figure-6

D)

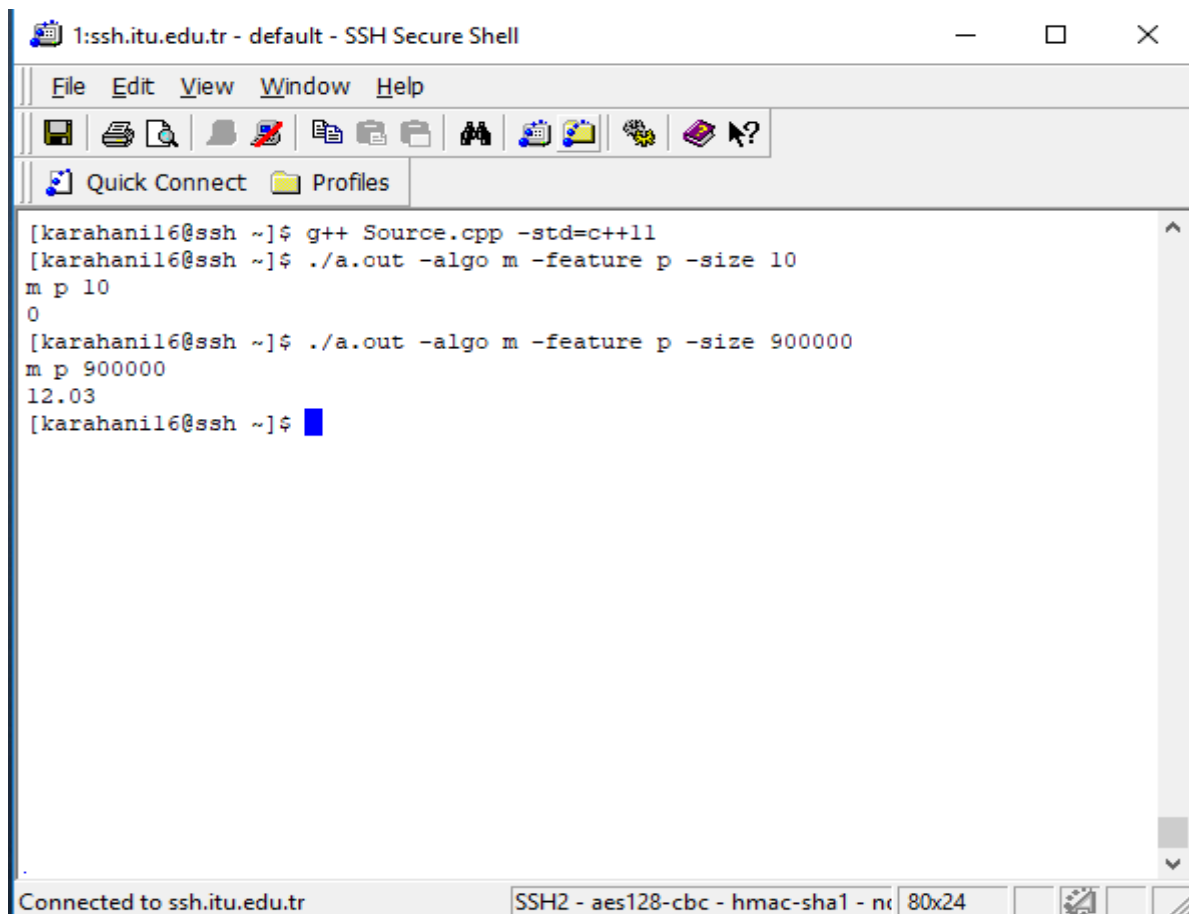
At this part, I make a comment about insertion and merge sort after sorted this file if we add a new line to our sorted file then sort again. According to my knowledge merge sort is a proper algorithm and rapid algorithm for doing this sort comparing with insertion sort. Because in the insertion sort changing the last element we traverse the all vector or array so this takes more time than merge sort so I choose merge sort for doing this sort.

### 3. Conclusion

To sum up, according to result which I get average running time and graphic of result, I can easily say that merge sort is a stable and faster algorithm comparing with insertion sort. But I get a different result when I compiled at my own computer as you see in the part b. I commend this result when I complied algorithm I try to use this program so this effect my result but the meaningful result I get with using ITU SSH server.

### 4. Compilation

When you compiled my program at ITU SSH server. You should add `-std=c++11` command.



```
1:ssh.itu.edu.tr - default - SSH Secure Shell
File Edit View Window Help
[Icons]
Quick Connect Profiles

[karahani16@ssh ~]$ g++ Source.cpp -std=c++11
[karahani16@ssh ~]$ ./a.out -algo m -feature p -size 10
m p 10
0
[karahani16@ssh ~]$ ./a.out -algo m -feature p -size 900000
m p 900000
12.03
[karahani16@ssh ~]$
```

Connected to ssh.itu.edu.tr | SSH2 - aes128-cbc - hmac-sha1 - n | 80x24

Figure-7