BLG 335E
Analysis of Algorithms I
Fall 2017
Project 1
Report

(11.10.2017)

Recep Can BABAOĞLU
150130112

1. Compiling the Code

```
g++ -std=c++11 -c operasyon.cpp
g++ -std=c++11 -c main.cpp
g++ operasyon.o main.o -o sample
```

Preventing the warnings about chromo library: "-std=c++11" is requried.

```
Can-MacBook-Pro:algo 1.001 canbaba$ g++ -std=c++11 -c operasyon.cpp
Can-MacBook-Pro:algo 1.001 canbaba$ g++ -std=c++11 -c main.cpp
Can-MacBook-Pro:algo 1.001 canbaba$ g++ operasyon.o main.o -o sample
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -full -m canbaba.txt baba.txt
Elapsed time is :  79 ms
```

2.Questions

2.1) I could not get all values from the code. Because of the compiling time of insertion sort is too much for 100k and 1m sets. When I have need, I used complexity formula of sorting algorithm for providing graphs and tables. I mentioned it on the given table.

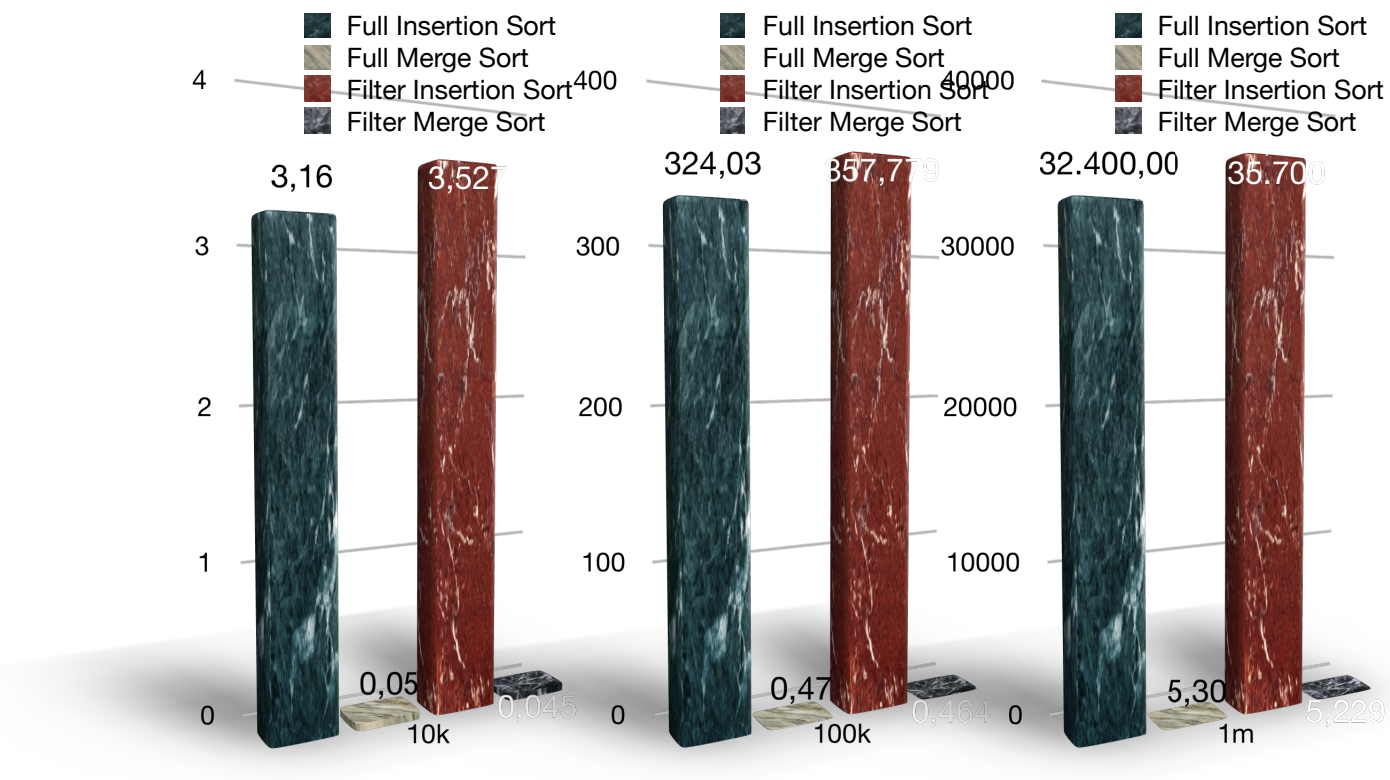
```
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -full -i hs-set-10k.txt baba.txt
Elapsed time is :  3155629 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -full -m hs-set-10k.txt baba.txt
Elapsed time is :  46129 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -filter -i hs-set-10k.txt baba.txt
Elapsed time is :  3527405 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -filter -m hs-set-10k.txt baba.txt
Elapsed time is :  45537 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -full -i hs-set-100k.txt baba.txt
^[^[^[^[
^[
Elapsed time is :  324025111 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -filter -i hs-set-100k.txt baba.txt

Elapsed time is :  357779769 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -filter -m hs-set-100k.txt baba.txt

Elapsed time is :  464731 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -filter -m hs-set-1m.txt baba.txt
Elapsed time is :  5229320 ms
```

Given screenshots shows calculated elapsed time from computer.

On the given table calculated elapsed times converted to seconds.

	10k set	100k set	1m set
Full Insertion Sort	3.155629 s	324.025111 s	~32400 s
Full Merge Sort	0.046129 s	~0.47 s	~5.3 s
Filter Insertion Sort	3.527405 s	357.779769 s	~35700 s
Filter Merge Sort	0.045537 s	0.464731 s	5.229329 s

~ Could not calculated from computer.



2.2) I have tested insertion sort and merge sort on 10k, 100k and 1m sets. On small sets insertion sort should be more faster than merge sort but given sets is not small enough for observing the speed difference. As seen in the tables merge sort is faster than insertion sort on the whole project. But almost sorted sets accelerate the insertion sort. On the merge sort side this acceleration is not too much.

```
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -full -i hs-set-10k-sorted.txt baba.txt
Elapsed time is :  1230978 ms
Can-MacBook-Pro:algo 1.001 canbaba$ ./sample -full -m hs-set-10k-sorted.txt baba.txt
Elapsed time is :  37724 ms
```

2.3) Rarity or Set are have same elements. When comparing 2 element if they are same, sorting algorithm do not change their place. It reduces time consumption. As a result of sorting by Rarity or Set instead of Name, elapsed time will be reduced.

2.4) Stable sorting sorts the identical elements in their same order as they appears in the input. Insertion sort and merge sort are stable. Disadvantages of stable sorting algorithms are time consumption and higher CPU and memory usage. If I use unstable sorting algorithm I could not get sorted vector respectively Class, Cost, Name. When I try to sort the set which was already sorted by Class, our sorted set will be unsorted. I get just sorted set by Cost. Full sort does not work.