PROJECT-1 REPORT
Analysis of Algorithms

Emre KÖSE – 150130037

11.10.2017

Introduction

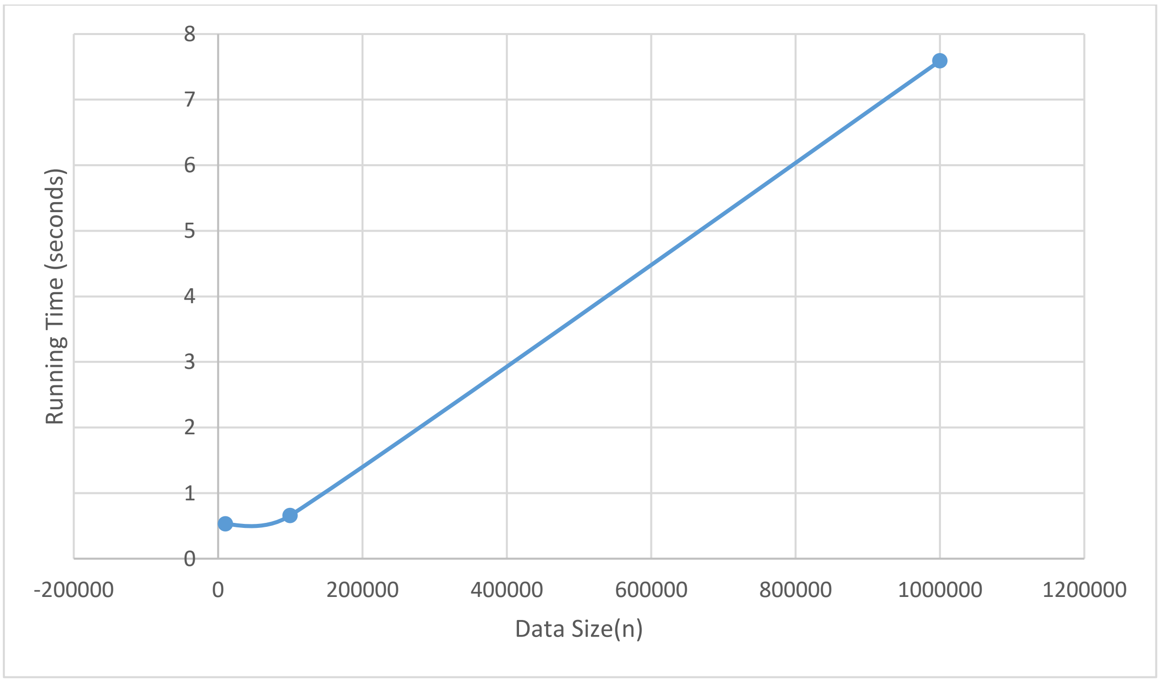
In this project, in the basis of merge sort and insertion sort algorithms, some features of algorithms were covered. In addition, these algorithms were implemented and compared according to each other.

Part A

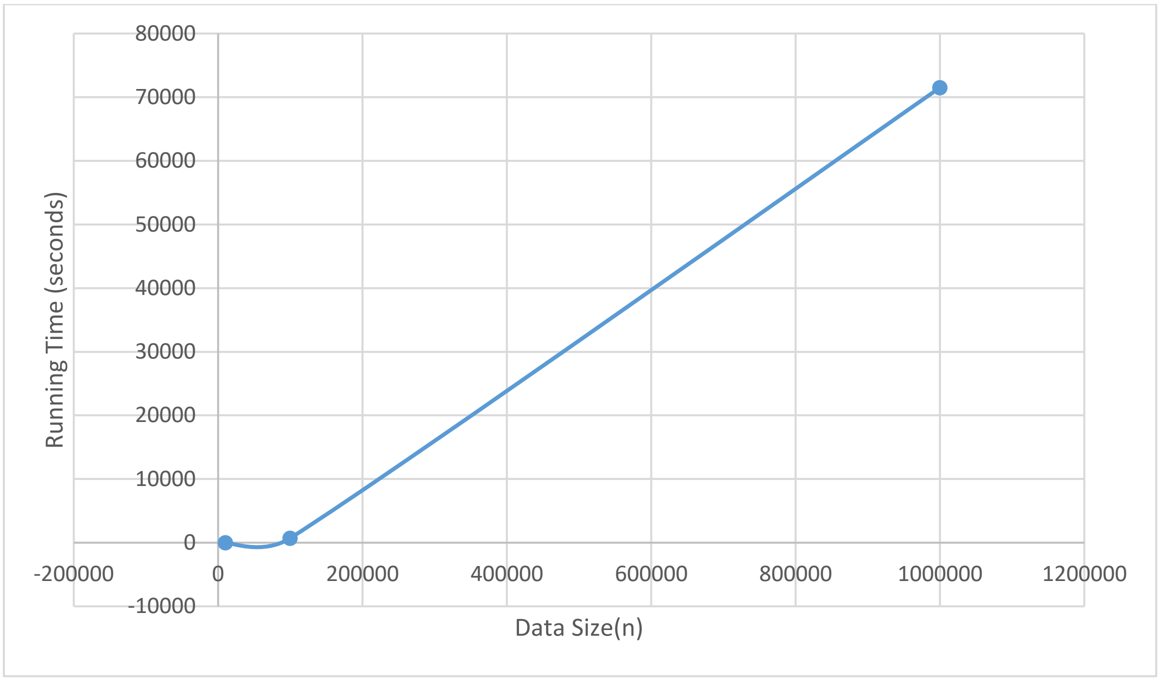
In this part, the program which is implemented with two algorithms was ran in different size-input files which have 10k, 100k and 1M elements. This is done with different sorting procedures and finally running times of sorting algorithms were recorded as in below: (Insertion sort was not experienced for input of dataset that has 1M element with fully sort procedure, because of the running time will be so long, but there is no problem about implementation of algorithm)

	N=10k (Full)	N=100k (Full)	N=1M (Full)
Merge Sort	0.053 s	0.659 s	7.59 s
Insertion Sort	7.22 s	719.476 s	-

Running times for Full Sort Procedure



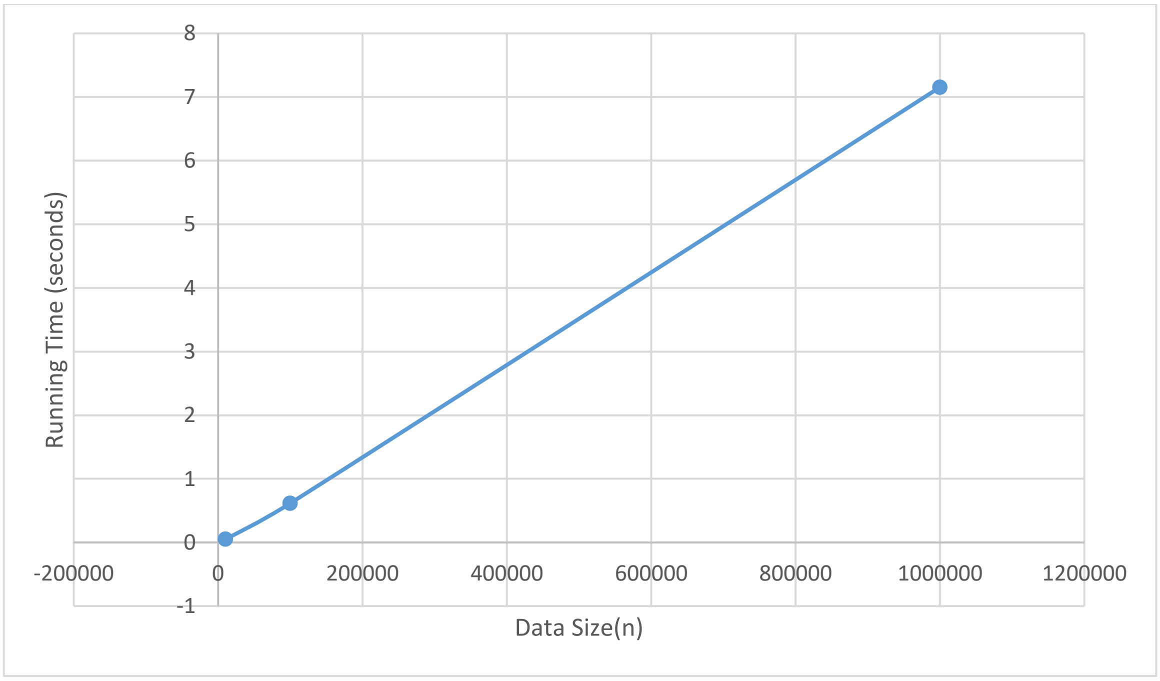
Merge Sort (Full Sort Procedure)



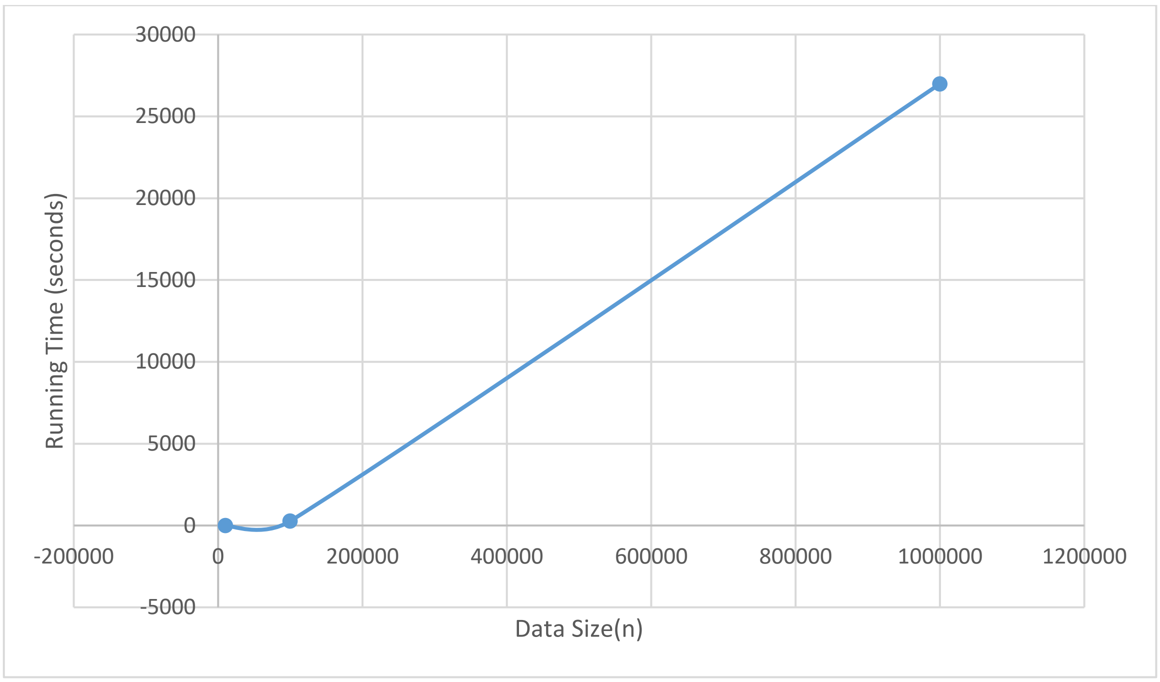
Insertion Sort (Full Sort Procedure)

	N=10k (Filter)	N=100k (Filter)	N=1M (Filter)
Merge Sort	0.0504 s	0.616s	7.153 s
Insertion Sort	2.660 s	273.21s	26988.702 s

Running times for Filter Sort Procedure



Merge Sort (Filter Sort Procedure)



Insertion Sort (Filter Sort Procedure)

PART B

As seen from tables which are given in part A, merge sort algorithm gives better results in terms of running time for all data sets and both sorting procedure and it is understood again from tables, running times of merge sort increase linearly because of having O(nLog(n)) time complexity and insertion sort algorithm's running times are increasing with quadratic values and that is because insertion sort has O(n^2) time complexity which n represent number of element for both algorithm. According to these results, in larger data sets merge sort is absolutely has more advantage than insertion sort algorithm. The advantage of insertion sort can be seen in more smaller data sets, if we try for our project with the element size like 1000 or smaller we can point out insertion sort can be more useful with these inputs.

PART C

For this part of project, I had some information from the game's official website and these are listed below:

- There are 5 different rarity features: Free, Common, Rare, Epic and Legendary.
- There are four main types of cards: spell cards, weapon cards, minion cards and hero cards.
- There are 13 different set types which are in standart and wild set clusters.
- There are many different card names for this game (>=30).

We implemented the filter algorithm to sort cards according to cards' types and there are 4 different types for cards. If we implement filter sort according to rarity or set, I think we can see similar results for filter sort with rarity because its feature size (5) is approximately same with the size of cards' types features which is 4. We can say these are different from the name parameter because name cluster has more distinct element number than others, this issue can lead algorithms to do more comparison, especially there will be less chance for existing of insertion sort's best case scenario because there will be more feature, the probability of the best case for insertion sort will be low, and this will affect the running time increase.

PART D

Firstly, stable sorting can be defined as sorting of data, which keeps same order as appear in the input after comparison of two elements with equal keys. For example, if we sort the numbers in array, we only consider the numbers, not any other thing will affect to our algorithm.

Merge and Insertion sort algorithms are comparison based stable algorithms. As we implemented in this project for filter sorting procedures we only check two different card's types, if one of them has smaller value, do something else do nothing and save the original input order.

If an unstable sorting algorithm was used in full sort, there are many incorrect sorting results because of not controlling of the input order, and protection of the input order will not be considered as an issue. I can say according to my researches about stability in sorting, if we used unstable sorting in the full sort, probably the element can be ordered correctly by name, but there will be some incorrect results in the sorting of class and cost parameters. Unstable sorting algorithms can be used for there are no equivalent elements in the input and probably will be faster than the stable sorting algorithms' results.

Reference:

- https://hearthstone.gamepedia.com/Card