ASSIGNMENT 1

Subject : Analyzing the complexity of sorting algorithms
TAs: Selim Yilmaz, Levent Karacan, Merve Ozdes
Due Date: 15.03.2018 23:59:59

1 Introduction

Analysis of algorithms is the area of computer science that provides tools to analyze the efficiency of different methods of solutions. Efficiency of an algorithm depends on these parameters; i) how much time, ii) memory space, iii) disk space it requires. Analysis of algorithms is mainly used to predict performance and compare algorithms that are developed for the same task. Also it provides guarantees for performance and helps to understand theoretical basis.

A complete analysis of the running time of an algorithm involves the following steps:

- Implement the algorithm completely.
- Determine the time required for each basic operation.
- Identify unknown quantities that can be used to describe the frequency of execution of the basic operations.
- Develop a realistic model for the input to the program.
- Analyze the unknown quantities, assuming the modeled input.
- Calculate the total running time by multiplying the time by the frequency for each operation, then adding all the products.

In this experiment, you will analyze different sorting algorithms and compare their running times on a number of dataset with changing sizes.

2 Problem Definition

The strength of a search algorithm reveals on a data comprising a great deal of instances. For such a data set, it is expected to observe that the sorting algorithms having worse performance in terms of complexity like *selection sort*, *insertion sort*, or *bubble sort* require higher amount of time whereas those having better performance like *merge sort* or *quick sort* finish the process faster, which is an expected case within the scope of this assignment. In Table 1, you are provided comparative complexity/runtime comparison of well-known sorting algorithms.

Table 1: Complexity comparison of sorting algorithms

	Best	Average	Worst
Selection Sort	$\Omega(n^2)$	$\theta(n^2)$	$O(n^2)$
Bubble Sort	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$
Insertion Sort	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$
Heap Sort	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$
Quick Sort	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$
Merge Sort	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$
Radix Sort	$\Omega(nk)$	$\theta(nk)$	$O(nk)$

The dataset you are to employ in this assignment is a real dataset and contains a great deal amount of traffic flows of a test network. Traffic flow is a flow that records all communication packets, including header and payload, sent bidirectional between sender and receiver within a certain period of time. The flow you are provided is generated from a real network trace [1] through *FlowMeter*, a network traffic flow generator written in Java [2]. Click here to download the dataset. In the flow data you are provided, there are about 250,000 captures and in order for you to comparatively compare the performance of sorting algorithms with a changing size of data, we partitioned the whole data into smaller sizes. The partitioned and actual datasets have 100; 1,000; 50,000; 100,000; 250,000 captures, respectively.

FlowMeter generates more than 80 features including *Flow ID*, *source and destionation IPs*, and the like. For details regarding the feature set, refer to [3]. As it is out of scope of this assignment, actually you do not need to know what type of information is recorded in these features. So do not get confused with the details of these features.

3 Assignment Task

As stated before, the main objective of this assignment is to analyze the runtimes of different sorting algorithms whose proven mathematical complexities are given in Table 1. To do so, you are provided a series of files as a testbed containing different number of flow captures.

In this assignment, you are expected to *i*) choose three sorting algorithms, *ii*) implement them in *Java*, *iii*) sort all datasets, *iv*) measure the time (in second) that each algorithm requires. There is no restriction on your algorithm choice; however you are strongly encouraged to choose them such that each has different complexity (refer to Table 1) so that you are able to observe the variation in performance especially on the big data.

While sorting a dataset, you should consider the consistency within a record. To be more precise, let's suppose you are sorting dataset D, according to a feature (sorting key) j. If the value of jth key of ith flow record is greater than that of kth record, you should replace whole feature values of i with those of k. See the example below:

	Feature		
Record	(j-1)	j	(j+1)
i	7	10	8
k	4	6	1

	Feature		
Record	(j-1)	j	(j+1)
k	4	6	1
i	7	10	8

Design your implementation such that it takes three arguments;

i the path of dataset with the name of file on which sorting process is applied.

ii index of the feature used as a key for sorting. The key j used in sorting process is actually feature index ($7 < j \leq 84$).

iii an option that indicates whether sorted data will be saved, or not. It should be given T (True) to save, otherwise it should be F (False).

Your implementation should run with following command:

```
java assignment1 <dataset path> <feature index> <save T/F>
java assignment1 ../TrafficFlow1000.csv 56 T
```

4 Reporting

You report should clearly cover the following points:

- *Problem definition.* You should clearly express the idea behind this assignment.
- *Findings.* You should demonstrate the results with table(s) (as Table 2) and plots.
- *Discussion* about the time complexity and memory requirements of every algorithm you choose.

Note: You are not restricted to use a particular text editor that you will need to use while preparing your report; however we encourage you to use `LaTeX`.

Table 2: My caption

Algorithm & Data Set	TrafficFlow100	TrafficFlow1000	TrafficFlow50000	TrafficFlow100000	TrafficFlowAll
1st Algorithm					
2nd Algorithm					
3rd Algorithm					

Notes

- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/spring2018/bbm204) and you are supposed to be aware of everything discussed in Piazza.
- You will submit your work from https://submit.cs.hacettepe.edu.tr/index.php with the file hierarchy as below:

  <student id.zip>
  → report <DIR>
  → report.pdf <FILE>
  → src <DIR>

- The class name in which main method belongs should be **`Assignment1.java`**. All classes should be placed in `src` directory. Feel free to create subdirectories, corresponding the package(s), but each should be in `src` directory.
- This file hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

Policy

All work on assignments must be done **individually** unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an **abstract** way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) **will not be tolerated**. In short, turning in someone else's work (from internet), in whole or in part, as your own will be considered **as a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.

[2] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of tor traffic using time based features. In *ICISSP*, pages 253–262, 2017.

[3] Cicflowmeter. http://www.unb.ca/cic/datasets/flowmeter.html. (Accessed: 25.02.2018).