Hacettepe University
Computer Science and Engineering Department

Name and Surname : Meltem Tokgöz
Identity Number : 21527381

Course : BBM 204 Software Laboratoy2
Experiment : Running Time Analyze
Language : Java
Data Due : 15.03.2018 23:59
Advisors : Selim Yılmaz

e-mail : b21527381@cs.hacettepe.edu.tr

Main Program : Assignment1.java

1.Software Design Note

What is theproblem?

We are required to perform performance analysis according to the running time of the algorithms we selected in this assignment. Even though all sorting algorithms perform the same task, their working time is different. Because their complexity is different. Different data sets have been given so that we can see this more clearly. (As the situation becomes much clearer in data sets where the data is much more data.) We are asked to measure and analyze the run times of all datasets in each sorting algorithm.

The algorithms and complexities we can consider are as follows.

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	$\Omega(nk)$	$\theta(nk)$	$O(nk)$

Solution

I learned how to analyze the algorithms in this experiment. First, I decide choose which algorithm, and then I try to understand algorithms and I implement and coded them. I try different data set run them and I measure the time all data set of the algorithms.

These I add table in report and I reviews.

2.Design

I used 4 class in my assignment.

The main class is Assignment1.java in the assignment. I get arguments from in this class and then I read to necessary file. I created 2 ArrayList, it keeps feature index sortArray and keeps file rows rFile.

After that, I used the System.*currentTimeMillis*() function to measure the running time and I invoke the function I want.

The other three classes are selectionSort, heapSort, quickSort. In these classes I encode the algorithm as required. Finally I have the array indexed according to the feature index.
If the last argument is T, I replace the file with the sorted version. If F is not making a change.

3.Table

The data I get when I run the algorithms is as follows. Calculated in seconds.

Selection Sort Complexity O(n^2)
Heap Sort Complexity O(nlog(n))
Quick Sort Complexity O(nlog(n))-O(n^2)

Data Set Algorithm	TrafficFlow100	TrafficFlow1000	TrafficFlow50000	TrafficFlow100000	TrafficFlowAll
Selection Sort	0,003 s	0,036 s	3,182 s	21,577 s	207,945 s
Quick Sort	0,002 s	0,006 s	0,093 s	0,124 s	0,392 s
Heap Sort	0,003 s	0,008 s	0,072 s	0,109 s	0,363 s

4.Karşılaştırma

The strength of a search algorithm reveals on a data comprising a great deal of instances. I see for such a data set, it is sorting algorithms having worse performance in terms of complex it insertion sort, higher amount of time whereas those having better performance like merge sort or heap sort finish the process faster.
At the same time running times have changed according to the state of the data. When the data sets are sorted according to some indexes, the best case is sorted according to some indexes and worst case formed.

Below are some examples of sorting and running time.

j=56 “T” Execution time

Data Set Algorithm	TrafficFlow100	TrafficFlow1000	TrafficFlow50000	TrafficFlow100000	TrafficFlowAll
Selection Sort	0,002	0,036 s	1,909 s	9,228 s	148,024 s
Quick Sort	0,001 s	0,004 s	0,134 s	0,201 s	0.347 s
Heap Sort	0,0 s	0,001 s	0,021 s	0,022 s	0.080 s

j=84 “F” Execution time

Data Set Algorithm	TrafficFlow100	TrafficFlow1000	TrafficFlow50000	TrafficFlow100000	TrafficFlowAll
Selection Sort	0,001	0,018 s	3,162 s	19,615 s	169,499 s
Quick Sort	0,002 s	0,006 s	0,052 s	0,155 s	0,218 s
Heap Sort	0.0 s	0,001 s	0,022 s	0,042 s	0,065 s

j=10 “T” Execution time

Data Set Algorithm	TrafficFlow100	TrafficFlow1000	TrafficFlow50000	TrafficFlow100000	TrafficFlowAll
Selection Sort	0,002 s	0,004 s	2,891 s	15,206 s	159,228 s
Quick Sort	0,002 s	0,009 s	0,052 s	0,184 s	0,292 s
Heap Sort	0,0 s	0,011 s	0, 109 s	0,123 s	0,273 s