Data Structure & Algorithms
CS210 A

Programming Assignment IV

Both Quick sort and Merge sort are comparison based algorithms. Their time complexity is governed by the number of comparisons performed while sorting a given input instance. For an input of n numbers, the following theoretical facts were derived in the class.

Asymptotic no. of Comparisons	**Merge Sort**	**Quick Sort**
Average case	$n \log_2 n$	$1.39 n \log_2 n$
Best case	$n \log_2 n$	$n \log_2 n$
Worst case	$n \log_2 n$	n^2

In the light of these facts, one would prefer Merge sort to Quick sort. However, the test of efficiency of algorithms is incomplete until we carry out their implementations and compare their performances on real life data.

With totally unbiased and fresh mind, you need to compare performance of Quick sort and Merge sort. Does Merge sort outperform Quick sort in practice as well ? How often Quick sort takes $O(n^2)$ time ? Is it worth using Quick sort in real life ? Main goal of this assignment is to carry out programming experiments to answer these questions. This assignment has two modules described below.

1. **Quick Sort versus Merge Sort**

   The aim of this module is to compare Quick Sort and Merge Sort. For a given n, you have to compare their efficiency parameters on n randomly generated numbers. Fill up the following table based on your experimental results.

	$n = 10^2$	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$
Average running time of Quick Sort					
Average running time of Merge Sort					
Average number of comparisons during Quick Sort					
The value of $1.39 n \log_2 n$					
Average number of comparisons during Merge Sort					
The value of $n \log_2 n$					
Number of times Merge Sort outperformed Quick Sort					

   What do you infer ? (Your answer must not exceed 5 lines.)

2. **Distribution of the running time of Quick Sort**

   Find the distribution of the running time (or comparisons) of Quick sort for various values of n. You want this information to determine the reliability of quick sort.

   (a) You will run Quick Sort for **large number** of random instances of different sizes. You will enter your experimental results in the table given below.

	$n = 10^2$	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$
Average running time of Quick Sort					
$1.39n \log_2 n$					
Average no. of comparisons during Quick Sort					
# cases where run time exceeds average by 5%					
# cases where run time exceeds average by 10%					
# cases where run time exceeds average by 20%					
# cases where run time exceeds average by 30%					
# cases where run time exceeds average by 50%					
# cases where run time exceeds average by 100%					

Make intelligent inferences for Quick Sort based on this table (not exceeding 5 lines.)

(b) You have to draw the plots showing the distribution of the running time of Quick sort. In particular, you should draw the histograms of frequencies for running times of Quick Sort in different intervals. If possible, have a single picture which will have the plots for all n (after suitable scaling). Make intelligent inferences from these plots (not exceeding 5 lines.)

Important points requiring your attention:

- You must implement both the algorithms in the most efficient manner. In particular, for the Quick sort, you must implement the *Partition*() procedure that should make a single pass and use only $O(1)$ extra space.
- While carrying out the above experiment, especially for calculating average, you must make sure that you repeat the experiment sufficiently large number of times. In particular, for a given value of n, you must repeat your experiment at least 500 (and preferably few thousands) times. In the report, you must mention very explicitly the number of repetitions.
- Make sure you use a suitable random number generator. Ideally, you should generate sufficiently long floating points in the range [0,1]. Note that excess repetitions of random numbers may be catastrophic for your experiment (think over it).
- You should use sufficiently precise time (which may count microseconds). For large input size, you should mention time in milliseconds or seconds.
- While writing the codes of respective algorithms, follow good programming practices (indentation, comments, suitable names for variables).
- Do not underestimate this assignment. You may learn so much from this assignment. Moreover, you should use your creative skills while drawing right inferences and the plots. The best two reports (including viva performance) will be posted on the course website. The students of these reports will also get 20% extra marks for this assignment as a reward.

Evaluation: You will be assigned a slot of 15 minutes for demo/viva. You will have to bring a report of at most 2 pages at the time of demo/viva. Also bring your laptop on which you carried out the experimentation. You must be able to demonstrate that the running time of an algorithm for an input during the demo is nearly the same as the running time you reported in the report (plots).