CS648A : Randomized Algorithms
Semester II, 2022-23, CSE, IIT Kanpur

Programming Assignment 1

Deadline : 11:55 PM, 16 January 2023.

Most Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. Remember - **Before cheating the instructor, you are cheating yourself**. The onus of learning from a course lies first on you. So act wisely while working on this assignment.
- Refrain from collaborating with the students of other groups. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.

General guidelines

1. This assignment is to be done in groups of 2 students. You have to form groups on your own. You are strongly advised not to work alone.
2. **Naming the file**:
   The submission file has to be given a name that reflects the information about the type of the assignment, the number of the assignment, and the roll numbers of the 2 students of the group. If you are submitting the solution of Programming Assignment x, you should name the file as **Prog_x_Rollnumber1_Rollnumber2.pdf**.
3. **Each student of a group** has to upload the same submission file separately. Be careful during the submission of an assignment. Once submitted, it can not be re-submitted.
4. Deadline is strict. Make sure you upload the assignment well in time to avoid last minute rush.

1. Randomized Quick Sort versus Quick Sort

Convince yourself that the behavior of the randomized quick sort on any arbitrary permutation of n distinct numbers is identical to the behavior of the (deterministic) quick sort on a random permutation of n distinct numbers. In particular, the only extra *overhead* in the randomized quick sort is that of selecting a uniformly random pivot element in each recursive call. The aim of this part of the assignment is to carry out experiments to justify this fact empirically.

You will repeat the following task for $n = 10^2, 10^3, 10^4, 10^5, 10^6$. For a given value of n you will generate a set of n random numbers uniformly distributed in [0, 1]. You will run quick sort on this set and repeat it sufficiently large number of times. On the other hand, Randomized Quick Sort will be run multiple times on any one **fixed** instance of the set. You will compare these algorithms on the basis of running time, number of comparisons, and the value $2n \log_e n$. In addition, you will compare them on *double sorting time* which is explained as follows. For a given input, first execute a sorting algorithm to get a sorted output, now once again execute the same sorting algorithm on this sorted output. The time taken is called the double sorting time.

You will have to fill up the following table. You also have to comment on the overhead of selecting uniformly random pivot elements during the randomized quick sort.

$n \rightarrow$	10^2	10^3	10^4	10^5	10^6
$2n \log_e n$					
Average running time of Quick Sort					
Average running time of Randomized Quick Sort					
Average no. of comparisons during Quick Sort					
Average no. of comparisons during Randomized Quick Sort					
Average value of double sort time by Quick Sort					
Average value of double sort time by Randomized Quick Sort					

Important Note: The implication of this part of the assignment is that for the remaining 2 parts of this assignment, you need to execute the (deterministic) quick sort on a sequence of n randomly generated numbers instead of executing the randomized quick sort on a given permutation of n numbers.

2. Randomized Quick Sort versus Merge Sort

The aim of this part is to compare the randomized quick sort and the merge sort. For this purpose, you have to empirically compare the efficiency parameters of the quick sort (not randomized quick sort) and merge sort on a sequence of n uniformly randomly generated numbers for various values of n. Fill up the following table on the basis of your experimental results. This has to be followed by a concise inference (not exceeding 5 sentences)

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

3. Reliability of Randomized Quick Sort

The aim of this part is to empirically find out the deviation from the average running time of the randomized quick sort for various values of n. You will have to enter your experimental results in the table given below. This has to be followed by a concise inference (not exceeding 5 sentences)

$n \rightarrow$	10^2	10^3	10^4	10^5	10^6
Average running time of randomized quick sort					
$2n \log_e n$					
Average no. of comparisons during randomized quick sort					
No. of cases where run time exceeds average by 5%					
No. of cases where run time exceeds average by 10%					
No. of cases where run time exceeds average by 20%					
No. of cases where run time exceeds average by 30%					
No. of cases where run time exceeds average by 50%					
No. of cases where run time exceeds average by 100%					

Optional

This part is totally optional and won't fetch you any extra marks.

Neatly draw the plots showing the distribution of the running time of the randomized quick sort (for the same experiment you carried out in the 3rd part described above). In particular, you should draw the histograms of frequencies for running times in different intervals. If possible, have a single picture which will have all the plots scaled properly. (You may try using $T(n)/(2n \log_e n)$ for the x-axis). Make intelligent inferences from these plots (not exceeding 5 lines.). I sincerely believe that at least one group of the entire class will do this optional part.

Important point you should consider in this assignment

1. You must make sure that for any given n, you repeat the corresponding experiment sufficiently large number of times. In particular, the number of repetitions should be at least 500 (and preferably few thousands). In the report, you must mention very explicitly the number of repetitions.

2. Make sure you use a suitable random number generator. Ideally, you should generate sufficiently long floating point number in the range [0,1]. Note that repetitions of random numbers can be catastrophic for a casually written code of quick sort. Think over it.

3. This point is about the first part of the assignment. For average value of double sort time, you will realize that Quick sort takes huge time even for $n = 10^4$. As a result, if you are not able to get the average value of double sort time for Quick sort for $n = 10^5$ or $n = 10^6$, please extrapolate the results of smaller values of n to get an estimate of the average value of double sort time for 10^5 and 10^6 (and fill these values in the table).

4. You should use sufficiently precise time (which may count microseconds). For large input size, you should mention time in milliseconds or seconds.

5. [Do not miss this point]
While writing the codes of respective algorithms, you must strive to design the most efficient implementation. In particular, for quick sort, the partition method must be one of the ideal ones – single pass over the subarray, and just a couple of variables. Remember – if your code of quick sort and/or merge sort is not efficient, you will not get the correct results.

6. Follow good programming practice(indentation, comments, suitable names for variables).

7. If each group does this assignment very sincerely and meticulously, the outcome will be truly memorable for the years to come.