Introduction to Randomized Algorithms: Assignment

Instructor: Kshitiz Verma

March 31, 2017

Instructions

1. Submission deadline: April 10, 2017, 23:59:59 (Firm deadline. You have exactly 10 days.)
2. Plagiarism, in any form, will result in 0.
3. You have to submit a report written only in Latex using the IEEEtran.cls document class. The report should not be more than 3 pages. It is recommended to have as many figures as you can to explain the ideas clearly. Apart from the files clearly marked for checking, you have to submit the zipped folder where you have carried out all the work.
4. April 7 is the last date to seek any advice by meeting me personally. There is **no other** mode for seeking clarifications.

Assignment

1. In this part, you will apply the knowledge of randomized algorithms in implementing Randomized Quick Sort. As a first step you are supposed to compare running times of your RQS implementation with the deterministic quick sort and merge sort.
   (a) You are essentially supposed to write a program that implements the three as a function and runs all the three on the same input. The best way to compare running times of a sorting algorithm is to count the number of comparisons made between the two elements. So you should count the number of comparisons made by all the three algorithms. Plot the number of comparisons made by the three algorithms as a function of the input size. Basically you have to plot a graph with X-axis as the size of the input and Y-axis as the number of comparisons. Your program should be able to handle large size of array (millions or more). Don't hesitate to go up to billions either. Report your findings.
   (b) You have to compare the theoretical probability of comparing two elements with the outputs from your program. In this case, you can show multiple graphs. For example, you choose to show 4 graphs for probability comparison. You may choose to fix elements i and j randomly, whatever comes to your mind. You have to to do this repeatedly for various permutations of the same data. Note that in this part, you should NOT change data in each iteration. You should also keep the elements same. So that the probability of comparison is a constant in each iteration and is equal to $\frac{2}{j-i+1}$ for fixed elements i and j.
   (c) You have to also show that the deviation of running time of RQS from its expected running time is low. In this case, keep the size of the array fixed. At least three good graphs are possible. In one you fix the elements of the array and only permute them many times. This keeps the array size n

and elements fixed. They are only permuted randomly uniformly. In the other you may vary the elements but keep the size fixed so each iteration has input of the same size but elements may be different. In the third one, you may change both the size of the array and elements.

In a nutshell, you will reconfirm our theoretical results on RQS using computer programs. I am expecting only line graphs. No surfaces, unless you know what you are doing. Also note that you have to deal with as large input size as you can. In general, it is good to have around 1 million elements in the array.

2. Implement Frievald's Technique to test whether AB=C. The problem statement is same as we have discussed in the class. You are supposed to implement this algorithm on YOUR OWN. However, you are free to take help of the Internet as long as you finally write the code on your own. You have to draw a graph of how the probability of the error behaves as the matrix size gets larger.