COMS 4231: Analysis of Algorithms I, Fall 2021

Problem Set 2, due Friday October 15, 11:59pm on Gradescope

Please follow the homework submission guidelines posted on Courseworks.

- **In all problems that ask you to give an algorithm, include a justification of the correctness of the algorithm and of its running time.**
- **All times bounds are worst-case, unless specified otherwise.**

Problem 1. [10 points] Do Exercise 5.4-6 in the CLRS book, page 142. Use indicator random variables.

Problem 2. [20 points] An array $A[1\ldots n]$ is said to have a *majority element* if the same element appears in (strictly) more than $n/2$ positions. We want to design an efficient algorithm to determine whether a given array has a majority element, and if so, to find that element. The elements of the array are not necessarily from an ordered domain like the integers, and so we cannot perform comparisons of the form "is $A[i] > A[j]$?". The only operations on the elements that we can perform are equality tests of the form "is $A[i] = A[j]$?", and this test takes constant time.

1. [7 points] Show that if the array A contains a majority element, then that element is also a majority element of either the first half of the array (i.e. $A[1...\lfloor n/2 \rfloor]$) or the second half of the array (or both).

Use this property to design a divide and conquer algorithm that solves the majority element problem in time $O(n\log n)$.

2. [13 points] A different divide and conquer algorithm is based on the following idea. Pair up the elements of A arbitrarily to get $\lfloor n/2 \rfloor$ pairs (for example, pair $A[1]$ with $A[2]$, $A[3]$ with $A[4]$ etc). For each pair, if the two elements of the pair are equal then keep only one copy, and if the two elements are not equal then discard both of them.

a. Show that if n is even and A has a majority element x, then x is also a majority element among the elements that are kept.

b. If n is odd, then give a rule regarding whether or not to keep the last element that was not paired, so that the property of part (a) holds in this case too, i.e. if A has a majority element x then x is also a majority element among the elements that are kept. Show that the property holds.

c. Use parts (a) and (b) to design a $O(n)$-time algorithm for the majority element problem.

Problem 3. [13 points] We are given an unsorted array A of n numbers, and a positive integer $k < n$. We want to find the k elements of A that are closest in value to the first element of A; these elements can be output in any order, and in case of a tie you can choose arbitrarily. For example,

if A is the array [7, 2, 5, 15, 6, 12, 10, 18, 0, -4] and k =3 then the algorithm should return 5, 6, 10 (in some order). If k=4 then the algorithm should return 2, 5, 6, 10 or 5, 6, 12, 10. Give an O(n)-time algorithm for this problem.
(Note that k is not a constant, it is part of the input; your algorithm should run in O(n) time for any value of k, for example for k=log$n$ or k=n/5.)

Problem 4. [17 points] Given an unsorted array with n elements, and a positive integer $k < n$, we wish to find the k-1 elements of rank $\left\lceil \frac{n}{k} \right\rceil$, $\left\lceil \frac{2n}{k} \right\rceil$, …, $\left\lceil \frac{(k-1)n}{k} \right\rceil$. Give an O($nlogk$)-time algorithm for this problem.

Problem 5. [20 points]
1. What is the expected running time of RANDOMIZED-QUICKSORT as given in the book (and in class) when applied to an array all of whose elements are equal? Justify your answer.

2. If we expect many equal elements in the input array, it is better for QUICKSORT to use a modified PARTITION routine that partitions subarrays with respect to the pivot x into 3 parts consisting of elements $<x$, $=x$, $>x$ respectively. Give such a modified PARTITION routine so that it partitions *in place* a subarray $A[p...r]$ with respect to a pivot element $x=A[r]$ into three parts as suggested above. The routine should permute the elements of the subarray and return a pair of indices (q_1, q_2) such that, in the final subarray all elements of $A[p...q_1]$ are smaller than the pivot x, all elements of $A[q_1+1...q_2]$ are equal to x, and all elements of $A[q_2+1...r]$ are greater than x.

3. Modify RANDOMIZED-QUICKSORT so that it runs in expected time O($n$log$n$) for every (arbitrary) input array. Justify briefly your bound on the expected running time. (You don't need to give a complete proof; adjust appropriately the analysis given in the book and the class for the general case where the elements may not be distinct.)

Problem 6. [20 points] Do Problem 8.6 ("Lower bound on merging sorted lists") in CLRS, page 208.