COMS 4231: Analysis of Algorithms I, Fall 2018

Problem Set 2, due Friday October 5, 11:59pm on Courseworks

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 are worst-case times unless specified otherwise.

Problem 1. Solve the following recurrences asymptotically. For each recurrence show $T(n)=\Theta(f(n))$ for an appropriate function f. Assume that $T(n)$ is constant for sufficiently small n. Justify your answers. You may use the master theorem (if applicable) or any of the other methods.

a. $T(n) = 4T(n/2) + n^2$
b. $T(n) = 3T(n/4) + n$
c. $T(n) = 3T(n/2) + n\log^2 n$
d. $T(n) = 2T(n-3) + 1$
e. $T(n) = T(\sqrt{n}) + 1$
f. $T(n) = T(n/2) + T(n/3) + n$

Hints. For part (e), one way to solve this is by expanding (unfolding) the recurrence; how many iterations does it take until the argument of the function becomes ≤ 2? Another way is to use a change of variables as in an example we did in class.
For part (f) you may want to use the recursion tree method to bound $T(n)$ asymptotically from above and below, or you can guess the form of the solution and use the substitution method to verify it.

Problem 2. An array A[1…n] contains a random permutation of the numbers 1,…,n, where all the permutations are equally likely.
1. What is the probability that A[1] > A[2]? Justify your answer.
2. Calculate the expected number of pairs of indices i, j such that $1 \leq i < j \leq n$ and A[i] > A[j].

Problem 3. 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. 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. 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?

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, and you may invoke the analysis given in the book and the class for RANDOMIZED-QUICKSORT for the case where all the elements are distinct.)

Problem 6. Do Problem 4.5 ("Chip testing") in CLRS, page 109.