CSci 4041: Algorithms and Data Structures (Spring'15)

Homework 3, Due 03/05/15 (in class)

Answer all of the following questions, and always explain your answer. You are expected to do the work on your own.

Any proofs you do have to be rigorous, covering all cases, to get credit. Just giving an example does not constitute a correct proof, and will not receive credit. For complexity analysis, just stating the complexity is not sufficient. Clearly explain how you get the complexity.

The homework is due in class on March 05. We want a paper copy submission.

Good Luck!

1. (30 points) Consider a sorting problem in which we do not know the numbers exactly. Instead, for each number, we know an interval on the real line to which it belongs. That is, we are given n closed intervals of the form $[a_i, b_i]$, where $a_i \leq b_i$. We wish to sort these intervals, i.e., to produce a permutation $\langle i_1, \ldots, i_n \rangle$ of the intervals such that for $j = 1, 2, \ldots, n$, there exist $c_j \in [a_{ij}, b_{ij}]$ satisfying $c_1 \leq c_2 \leq \cdots \leq c_n$.

   (a) (10 points) Design a randomized algorithm for sorting n intervals. Your algorithm should have the general structure of an algorithm that quicksorts the left endpoints (the a_i values), but it should take advantage of overlapping intervals to improve the running time. (As the intervals overlap more and more, the problem of sorting the intervals becomes progressively easier. Your algorithm should take advantage of such overlapping, to the extent that it exists.) Please provide the pseudo-code, and a brief description of the algorithm.

   (b) (10 points) Prove that your algorithm correctly sorts any given n intervals.

   (c) (10 points) Argue that your algorithm runs in expected time $\Theta(n \log n)$ in general, but runs in expected time $\Theta(n)$ when all of the intervals overlap (i.e., when there exists a value x such that $x \in [a_i, b_i]$ for all $i = 1, \ldots, n$). Your algorithm should not be checking for this case explicitly; rather, its performance should naturally improve as the amount of overlap increases.

2. (25 points) Suppose that you are given n red and n blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same amount of water, and vice versa.

   Your task is to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the water into the

blue jug. This operation will tell you whether the red or the blue jug can hold more water, or that they have the same volume. Assume that such a comparison takes one time unit. Your goal is to find an algorithm that makes a minimum number of comparisons to determine the grouping. Remember that you may not directly compare two red jugs or two blue jugs.

(a) (5 points) Describe a deterministic algorithm that uses $\Theta(n^2)$ comparisons to group the jugs into pairs. Please provide pseudo-code, and a brief description of the algorithm.

(b) (10 points) Prove a lower bound of $\Omega(n \log n)$ for the number of comparisons that an algorithm solving this problem must make.

(c) (10 points) Describe a randomized algorithm whose expected number of comparisons is $O(n \log n)$. Please provide pseudo-code, a brief description of the algorithm, a brief argument why the algorithm is correct, and a proof for the $O(n \log n)$ bound for the expected number of comparisons. What is the worst-case number of comparisons for your algorithm?

3. (30 points) Suppose that, instead of sorting an array, we just require that the elements increase on average. More precisely, we call an n-element array A k-sorted if, for all $i = 1, 2, \ldots, n-k$, the following holds:

$$\frac{1}{k} \sum_{j=i}^{i+k-1} A[j] \leq \frac{1}{k} \sum_{j=i+1}^{i+k} A[j] \ .$$

(a) (6 points) Prove that an n element array is k-sorted if and only if $A[i] \leq A[i+k]$ for all $i = 1, 2, \ldots, n-k$.

(b) (12 points) Give an algorithm that k-sorts an n-element array in $O(n \log(n/k))$ time. Please provide pseudocode, a brief description of the algorithm, an argument for its correctness, and an analysis of the running time.

(c) (12 points) Give an algorithm that sorts a k-sorted array of length n in $O(n \log k)$ time. Please provide pseudocode, a brief description of the algorithm, an argument for its correctness, and an analysis of the running time.

4. (25 points) The problem considers the analysis of RANDOMIZED-SELECT using indicator random variables in a manner akin to our analysis of RANDOMIZED-QUICKSORT. As in the quicksort analysis, we assume that all elements are distinct, and we rename the elements of the input array A as $z_1, z_2, \ldots, z_n$, where z_i is the ith smallest element. Thus, the call RANDOMIZED-SELECT$(A, 1, n, k)$ returns z_k.

For $1 \leq i < j \leq n$, let

$X_{ijk} = \mathbb{1}\{z_i \text{ is compared with } z_j \text{ sometime during the execution of the algorithm to find } z_k\}$.

(a) (9 points) Give an exact expression for $E[X_{ijk}]$. The expression may take different values, depending on i, j, k.

(b) (8 points) Let X_k denote the total number of comparisons between elements of array A when finding z_k. Show that

$$E[X_k] \leq 2 \left(\sum_{i=1}^{k} \sum_{j=k}^{n} \frac{1}{j-i+1} + \sum_{j=k+1}^{n} \frac{j-k-1}{j-k+1} + \sum_{i=1}^{k-2} \frac{k-i-1}{k-i+1} \right) \ .$$

(c) (8 points) Show that $E[X_k] \leq 4n$. Further, assuming that all elements of A are distinct, conclude that RANDOMIZED-SELECT runs in expected time $O(n)$.

Extra Credit Problem:

EC1 (10 points) Consider a set of n intervals $D_i = [a_i, b_i], i = 1, \ldots, n$, similar to problem 1. A subset of q intervals $D_{i_1}, \ldots, D_{i_q}$ are said to overlap if there is a x such that $x \in D_{i_j}, j = 1, \ldots, q$, i.e., x is in each of the q intervals, where $1 \leq q \leq n$. Given a set of intervals $D_1, \ldots, D_n$, design an algorithm with running time $O(n \log n)$ which finds out the largest q such that q of the given intervals overlap.