Introduction to Algorithms February 13, 2004
Massachusetts Institute of Technology 6.046J/18.410J
Professors Erik Demaine and Shafi Goldwasser Handout 5

Problem Set 2

This problem set is due **in class** on **Tuesday, February 24**.

Reading: Chapters §5.1-5.3, 7, 9

There are **four** problems. Each problem is to be done on a **separate sheet** (or sheets) of paper. Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.

You will often be called upon to "give an algorithm" to solve a certain problem. Giving an algorithm entails:

1. A description of the algorithm in English and, if helpful, pseudocode.
2. A proof (or argument) of the correctness of the algorithm.
3. An analysis of the running time of the algorithm.

It is also suggested that you include at least one worked example or diagram to show more precisely how your algorithm works. Remember, your goal is to communicate. Graders will be instructed to take off points for convoluted and obtuse descriptions. If you cannot solve a problem, give a brief summary of any partial results.

Problem 2-1. Fuzzy Sorting of Intervals **(Extra Credit)**

Consider the sorting problem where all the numbers are not known exactly. Instead, for each number, you know an interval on the real line to which it belongs. That is, you are given n closed intervals of the form $[a_i, b_i]$ where $a_i \leq b_i$. Assume that no interval contains any other interval. That is, if $a_i \leq a_j$ then $b_i \leq b_j$. You are asked to ***fuzzy-sort*** these intervals, i.e., produce a permutation $\langle i_1, i_2, \ldots, i_n \rangle$ of the intervals, such that for all j, there exists a $c_j \in [a_{i_j}, b_{i_j}]$ satisfying $c_1 \leq c_2 \leq \cdots \leq c_n$.

Suppose that each interval is guaranteed to overlap at least $d-1$ other intervals. Give an $O(n \log \frac{n}{d})$ algorithm to fuzzy-sort n intervals with d degrees of overlap. Thus, if $d = \Theta(1)$, the algorithm runs in $O(n \log n)$ time, and if $d = \Theta(n)$, the algorithm runs in $O(n)$ time.

Problem 2-2. Randomized Quicksort Variants

Regularly, RANDOMIZED QUICKSORT selects a single pivot element uniformly at random. In this problem you will analyze two possible modifications to this choice of pivot.

(a) For an integer parameter $k \geq 1$, the HYBRID RANDOMIZED QUICKSORT algorithm uses regular RANDOMIZED QUICKSORT whenever $n > k$, but uses INSERTION SORT whenever $n \leq k$. Analyze the expected running time of HYBRID RANDOMIZED QUICKSORT in terms of n and k. For what values of k does the algorithm run in $O(n \lg n)$ time?

In the CAUTIOUS RANDOMIZED QUICKSORT algorithm, the pivot is chosen by repeatedly selecting random candidate pivots and stopping only once a "good" pivot is found. A pivot is *good* if it partitions an array of n elements into two subarrays each with at least $n/3$ elements. To determine whether a pivot is good, CAUTIOUS RANDOMIZED QUICKSORT runs the PARTITION subroutine, computes the split, and if the split is not good it undoes the actions made by PARTITION in linear time.

(b) What is the probability of selecting a good pivot after a single trial?

(c) What is the maximum recursion depth of CAUTIOUS RANDOMIZED QUICKSORT?

(d) What is the expected running time of CAUTIOUS RANDOMIZED QUICKSORT?

Problem 2-3. In-Place Median

You are given a DVD-ROM storing n values and wish to find the median. Since the disk is read-only, you cannot swap or move elements. Your computer has $O(\log n)$ read/write memory, so you can't simply copy the DVD into memory. Give an $O(n \log n)$ expected-time algorithm to find the median without writing to disk.

Problem 2-4. Clothing Store

You decide to open a "Short'n'Tall" clothing store, catering only to very short and very tall people. As part of market research, you measure n people who arrive at times $a_1, \ldots, a_n$. Their heights are the positive numbers $h_1, \ldots, h_n$. (Person i arrives at time a_i and has height h_i.) Neither the arrival times nor the heights are sorted.

(a) You decide that your store will only sell clothing to the top and bottom $(1/k)$th of the population. You wish to separate out the tallest n/kth people and the shortest n/kth people. Give an $O(n)$-time algorithm to find both of these groups.

After a lawsuit from the Association of Average People, you decide to make custom clothes for people of all heights. Suppose that the amount of material to clothe a person is proportional to their height. If $\sum_{i=0}^{n} h_i = C$, then C units of material can clothe all n people. Unfortunately you have only $C/2$ units of material on hand.

You decide to clothe as many people as possible, but give priority to those who arrived first. As a result, everyone under the *weighted median* will get their clothes. A weighted median is an index k such that:

$$\sum_{i:a_i<a_k} h_i \leq \frac{C}{2} \quad and \quad \sum_{j:a_k<a_j} h_j \leq \frac{C}{2}$$

(b) Give an algorithm to compute the weighted median of n elements in $O(n \lg n)$ worst-case time.

(c) Give an algorithm to compute the weighted median in $\Theta(n)$ worst-case time.