

Recurrences, Divide and Conquer, Randomization

October 8, 2009

Homework 2

Due Date: Thursday, 15 October 2009 by end of lecture.

Problem 2-1. (30pts) Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 2T(n/4) + \sqrt{n}$

2. $T(n) = 7T(n/3) + n^2$

3. $T(n) = T(n-2) + \log n$

4. $T(n) = 3T(\sqrt{n}) + \log n$

5. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

6. $T(n) = T(n/3) + T(n/5) + \Theta(n)$

Problem 2-2. (20pts) You are given n points on a plane by their x-y coordinates. Point i is specified by (x_i, y_i) . The goal is to find 3 points a, b , and c that minimize sum of the three distances (distance from a to b plus distance from b to c plus distance from a to c). Design an algorithm, prove correctness and analyze running time. Brute-force algorithms will not get any points. [Hint: this is similar to the problem considered in-class.]

Problem 2-3. (20pts) Show that if in the second case of Master Theorem (i.e. when $f(n) = \Theta(n^{\lg_b a} \lg^k n)$) we have $k < 0$, then the proof shown during the lecture breaks down.

- Point out exactly where does the proof break down.
- Using the same proof approach, show upper and lower bounds on $T(n)$ that are within a logarithmic factor of each other.
- Bound the summation in question more carefully to show that for $k \leq -2$ we have $T(n) = \Theta(n^{\lg_b a})$.
- Extra credit: prove a tight result for $k = -1$.

Problem 2-4. (10pts) You are given an array on n integers $a_1 < a_2 < \dots < a_n$. Give an $O(\log n)$ algorithm that finds index i where $a_i = i$ or proves that such i does not exist.

Problem 2-5. (20pts) The input consists of $n - 1$ integers in the range 0 through $n - 1$. In other words, all integers in this range are present except one. The goal is to find the missing integer.

The complication is that the integers are presented "one bit at a time". In other words, n is so large that input integers have more bits than can be fit into a single memory cell. Instead, you are given a method to extract (i.e. read or query) j th bit from i th input integer in constant time. For example, this means that in $O(n \log n)$ time you can construct a 2-dimensional array where position i, j will hold bit j from i th input integer.

Problem 2-6. Extra credit: Do Exercise 7-4, page 188 (page 162 in 2nd edition).