CSE 431/531 Homework Assignment 1

Due in class on Thursday, Feb 01.

February 1, 2007

There are totally 7 problems, 10 points each. You should do them all. We will grade only 5 problems chosen at my discretion. If it so happens that you don't do one of the problems we don't grade, then no points will be deducted.

To "disprove" a statement, you must find a counter example to show that the statement is wrong. In general, your answers should be short but concise. (This will come with experience.)

Problem 1. Prove or disprove each of the following statements. You can assume that $f, g : \mathbb{N}^+ \to \mathbb{R}^+$.

1. $f(n) = \Theta(f(n/2))$
2. $2^n = \Theta(2^{n+1})$
3. $f(n) = O(g(n))$ implies $g(n) \lg f(n) = O(f(n) \lg g(n))$

Problem 2. Suppose both $f(n)$ and $g(n)$ are functions from $\mathbb{N}^+$ to $\mathbb{R}^+$, and they tend to ∞ as $n \to \infty$. Prove or disprove each of the following statements.

1. $\lg(f(n)) = O(\lg(g(n)))$ implies $f(n) = O(g(n))$
2. $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$
3. $\lg(f(n)) = \Theta(\lg(g(n)))$ implies $f(n) = \Theta(g(n))$
4. $f(n) = o(g(n))$ implies $\lg(f(n)) = o(\lg(g(n)))$
5. $\lg(f(n)) = o(\lg(g(n)))$ implies $f(n) = o(g(n))$

Problem 3. Arrange the following functions in ascending order of growth rate. That is, if $f(n)$ immediately follows $g(n)$ in the list, then $f(n) = O(g(n))$.

$$\begin{array}{lllll} n! & 2^{\lg^* n} & n^{\lg \lg n} & \lg^*(\lg n) & (\lg n)^{\lg(\lg n)} \\ \ln \ln n & 2^{\sqrt{n}} & 2^{2^n} & \sqrt{\lg n} & n^2 \end{array}$$

Explain your ordering. (To avoid unnecessary explanation, order the functions first, like a, b, c, d, then explain why $a = O(b)$, $b = O(c)$, and $c = O(d)$.)

Problem 4. Find a function f satisfying case 3 of the Master theorem but does not satisfy the regularity condition. Justify your answer.

Problem 5. Find asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant and positive for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 6T(n/3) + n^2$

2. $T(n) = T(7n/8) + n$
3. $T(n) = T(n-1) + n$
4. $T(n) = T(\sqrt{n}) + 1$
5. $T(n) = 3T(n/2) + n \lg n$
6. $T(n) = 2T(n/2) + \frac{n}{\lg n}$
7. $T(n) = T(n-1) + \lg n$
8. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

Note: $g(n)$ is an asymptotic upper bound for $T(n)$ if $T(n) = O(g(n))$. Additionally, if you found $T(n) = \Theta(g(n))$, then that's enough for both upper and lower bounds.

Problem 6. Give an example of functions $f, g : \mathbb{N}^+ \to \mathbb{R}^+$ such that none of the three relations $f(n) = O(g(n))$, $g(n) = O(f(n))$, and $f(n) = \Theta(g(n))$ is valid, although $f(n)$ and $g(n)$ both increase monotonically to ∞.

Problem 7 (Textbook, Problem 8, Chapter 2). You're doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The setup for this experiment, on a particular type of jar, is as follows. You have a ladder with n rungs, and you want to find the highest rung from which you can drop a copy of the jar and not have it break. We call this the *highest safe rung*.

It might be natural to try binary search: drop a jar from the middle rung, see if it breaks, and then recursively try from rung $n/4$ or $3n/4$ depending on the outcome. But this has the drawback that you could break a lot of jars in finding the answer.

If your primary goal were to conserve jars, on the other hand, you could try the following strategy. Start by dropping a jar from the first rung, then the second rung, and so forth, climbing one higher each time until the jar breaks. In this way, you only need a single jar – at the moment it breaks, you have the correct answer – but you may have to drop it n times (rather than $\lg n$ as in the binary search solution).

So here is the trade-off: it seems you can perform fewer drops if you're willing to break more jars. To understand better how this trade-off works at a quantitative level, let's consider how to run this experiment given a fixed "budget" of $k \geq 1$ jars. In other words, you have to determine the correct answer – the highest safe rung – and can use at most k jars in doing so.

(a) Suppose you are given a budget of $k = 2$ jars. Describe a strategy for finding the highest safe rung that requires you to drop a jar at most $f(n)$ times, for some function $f(n)$ that grows slower than linearly (i.e. sub-linear).

(b) Now suppose you have a budget of $k > 2$ jars, for some given k. Describe a strategy for finding the highest safe rung using at most k jars. If $f_k(n)$ denotes the number of times you need to drop a jar according to your strategy, then the functions $f_1, f_2, f_3, \ldots$ should have the property that each grows asymptotically slower than the previous one: $\lim_{n\to\infty} f_k(n)/f_{k-1}(n) = 0$, for each k.