Homework 2, due Thurs, Feb 2, 2006

January 19, 2006

Guidelines

In these problems, you are not required to find c and n_0 and prove the result by induction. It is sufficient to give a proof that appeals to results in the chapter. For example, you can say, without further comment, "$5n^2 + 7n \leq O(n^2)$" or "$\sum_{j=1}^{n} j^3 = \Theta(\int_0^n x^3\,dx) = \Theta(n^4)$" or "a k-ary tree with height h has $\Theta(k^h)$ leaves and nodes" ($k > 1$). In fact, I'd prefer that you get some practice in this "bigger picture" way of thinking, and not always get bogged down with induction details.

Unequal Divide and Conquer

1. Do CLRS 4.2-4.

2. Do CLRS 4.2-5.

3. The above exercises illustrate that, if we use a divide-and-conquer approach to a problem, it is typically better to divide the problem into nearly equal-sized subproblems. The above and the last example of CLRS Section 4.2 also illustrate that it is not necessary that the subproblems be *exactly* the same size.

   We now switch from *analyzing* to *designing* algorithms. Given a problem of size n, we will break it into two problems, of size $a(n)$ and $n - a(n)$, where $a(n)$ is a "common" function satisfying $1 \leq a(n) \leq n/2$. We assume the Divide and Recombine cost is linear, so we have recurrence

   $$T(n) = T(a(n)) + T(n - a(n)) + cn.$$

   All other things being equal, we'd want $a(n) = n/2$ to minimize $T(n)$. But sometimes there is a separate (direct or indirect) cost involved in making $a(n)$ exactly $n/2$ and it's easier to choose other $a()$'s. Below we investigate which $a()$'s are acceptable while still meeting certain overall cost requirements. Problem:

   - How slowly-growing can $a(n)$ be and still make $T(n) \leq O(n \log^2(n))$? (Restrict attention to functions of the form $n^r \log^s(n)$, where r and s are constant real numbers.)