CSC236 tutorial quiz #6*
week #10, Winter 2015

We would rather you be able to re-derive the Master Theorem than memorize it, even if rederiving it would take you a while. The tutorial this week is meant to strengthen your intuition about what's going on in the proof of the Master Theorem.

Recall that we are analyzing the asymptotic complexity of a recursive function ("recurrence") of the form

$$T(n) := \begin{cases} 1 & \text{if } n = 1 \\ aT(n/b) + f(n) & \text{otherwise} \end{cases}$$

where the asympototic complexity of f is $\Theta(n^k)$[1]. Throughout this handout we will assume n is a power of b, i.e. $n = b^e$ for some natural number e.

The proof of the Master Theorem goes by summing the quantity contributed to $T(n)$ by each "level" of recursion. In class we pictured the calculation of $T(n)$ as a tree of arity a and height $\log_b(n) + 1$, where the internal nodes correspond to recursive function calls, and a "level" of recursion corresponds to one of the $\log_b(n) + 1$ horizontal lines of nodes.

With the further simplifying assumption that $f(n) = n^k$ exactly, we arrived at the following non-recursive formula for T:

$$T(n) = \sum_{i=0}^{\log_b(n)} a^i \left(\frac{n}{b^i}\right)^k$$

Despite the fact that the contribution of f at each level of the recursion depends on a and b, it factors out nicely:

$$T(n) = n^k \sum_{i=0}^{\log_b(n)} a^i \left(\frac{1}{b^i}\right)^k$$

$$T(n) = n^k \sum_{i=0}^{\log_b(n)} \left(\frac{a}{b^k}\right)^i$$

Write a function in Python or any other language that, given $a, b, e \geq 1$ and $k \geq 0$, prints the $e+1$ terms of the previous summation for when $n = b^e$ (not including the factor n^k). Try it out for $e = 20$, in at least the following examples corresponding to the three cases of the Master Theorem:

1. $a = 3, b = 2, k = 1$. This describes the runtime of Karatsuba's trick for multiplying two $n/2$-digit integers.
2. $a = 2, b = 2, k = 1$. This describes the runtime of Mergesort. You can also try $a = 1, b = 2, k = 0$, which describes the runtime of Binary Search.
3. $a = 1, b = 2, k = 1$. *Simple* examples of algorithms whose runtime falls in case 3 of the Master Theorem are hard to come by. These parameters would fit a recursive algorithm on arrays that does a recursive call on half of the array, and a linear ($k = 1$) amount of work before and after the recursive call (e.g. something involving accessing each array element 1 to 3 times).

*Counting the easy quiz from two weeks ago.

[1] Which implies f is in $O(n^k)$, and not in $O(n^{k-.001})$