CSE 202 Calibration Homework
Fall 2004
Due Monday, Oct. 4

Recurrence Let $T(n)$ be the function given by the recursion: $T(n) = nT(\lfloor\sqrt{n}\rfloor)$ for $n > 1$ and $T(1) = 1$. Is $T(n) \in O(n^k)$ for some constant k, i.e. is T bounded by a polynomial in n? Prove your answer either way. (Note: logic and definition of O notation are more important than exact calculations for this problem.)

Reasoning about order Let $f(n)$ be a positive, integer-valued function on the natural numbers that is non-decreasing. Show that if $f(2n) \in O(f(n))$, then $f(n) \in O(n^k)$ for some constant k. Is the converse also always true?

Binary Tree Isomorphism Consider the following recursive algorithm, which makes the following assumptions. x, y are the roots of two binary trees, T_x and T_y. $Left(z)$ is a pointer to the left child of node z in either tree, and $Right(z)$ points to the right child. If the node doesn't have a left or right child, the pointer returns "NIL". Each node z also has a field $Size(z)$ which returns the number of nodes in the sub-tree rooted at z. $Size(NIL)$ is defined to be 0.

The algorithm $SameTree(x, y)$ returns a boolean answer that says whether or not the trees rooted at x and y are isomorphic, i.e., the same if you ignore the difference between left and right pointers.

1. Program: SameTree(x,y: Nodes): Boolean;
2. IF $Size(x) \neq Size(y)$ THEN return $False$; halt.
3. IF $x = NIL$ THEN return $True$; halt.
4. IF ($SameTree(Left(x), Left(y))$ AND $SameTree(Right(x), Right(y))$) OR ($SameTree(Right(x), Left(y))$ AND $SameTree(Left(x), Right(y))$) THEN return $True$; halt.
5. Return $False$; halt.

Give a time analysis (up to order) for this algorithm, giving the worst-case time complexity $T(n)$ when both trees are of size n. (Hint: consider the case when the trees rooted at x and y are both complete balanced trees with n nodes. This gives the intuition, but not a proof. To get a complete proof, you then need either a clever insight or to use an inductive argument. If you use a proof by induction, be careful about O notation. The constants involved need to be the same at all levels of the induction,i.e., they cannot change between induction hypothesis and conclusion. Again, logic is more important than calculation.)

Base Conversion Present and analyze an $O(n^2)$ time algorithm that inputs an array of n base 10 digits representing a positive integer in base 10 and outputs an array of base 2 bits representing the same integer in base 2. Count each operation on a single digit as a step, e.g., adding two n bit binary strings takes time $O(n)$ since one addition involves $O(n)$ bit operations.

Implementing Base Conversion . Implement the above algorithm, and test it on many random n bit strings for $n = 128$, $n = 256$, $n = 512$, $n = 1024$, $n = 2048$, $n = 4096$, $n = 8192$, $n = 16384$, and $n = 32768$. Plot time vs. input size on a log vs. log curve. Does the algorithm's observed time fit the analysis? Why or why not?