15-251: Great Theoretical Ideas In Computer Science

Recitation 4

- For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = O(g(n))$ if there exist constants $c, n_0 > 0$ such that $\forall n \geq n_0$, we have $f(n) \leq cg(n)$.
- For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = \Omega(g(n))$ if there exist constants $c, n_0 > 0$ such that $\forall n \geq n_0$, we have $f(n) \geq cg(n)$.
- For both of the above, your choice of c and n_0 cannot depend on n.
- For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
- Homework solution sessions have been moved to Fridays, 3:30-4:30 in DH 1212!
- This week we will be starting to hold conceptual office hours, Fridays from 4:30-6:30 in GHC 4301.
- Both of these will be happening today! Take advantage of all of the resources available to you!

$\mathcal{O}$, I Get It!

(a) Prove or disprove: If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.

(b) Prove or disprove: If $f = O(g)$, then $g = O(f)$.

(c) Let $f(n) = ((n/2)!)^2$. Prove either $f(n) = O(n!)$ or $f(n) = \Omega(n!)$, your choice.

(d) Prove that $n^{3.1} - n^2 + \log_2 n - 10$ is <u>not</u> $O(n^3)$.

Bits and Pieces

Determine which of the following problems can be computed in worst-case polynomial-time, i.e. $O(n^k)$ time for some constant k, where n denotes the number of bits in the binary representation of the input. If you think the problem can be solved in polynomial time, give an algorithm in pseudo-code, explain briefly why it gives the correct answer, and argue carefully why the running time is polynomial. If you think the problem cannot be solved in polynomial time, then provide a proof.

(a) Given as input a positive integer N, output $N!$.

(b) Given as input a positive integer N, output True iff $N = M^2$ for some positive integer M.

I Thought This Was 251, Not 210

Recall the matrix multiplication problem in which we are given two n by n matrices and we want to output their product. We are interested in the number of integer multiplications that we need to do to compute this problem. We saw in class that there is an algorithm with running time satisfying

$$T(n) = 7 \cdot T(n/2) + O(n^2)$$

Determine a tight big-O bound for this recurrence. (You may assume that n is a power of 2.)