15-251: Great Theoretical Ideas In Computer Science

Recitation 5

- The running time of an algorithm A is a function $T_A : \mathbb{N} \to \mathbb{N}$ defined by $T_A(n) = \max_{I \in S}\{\text{number of steps } A \text{ takes on } I\}$, where S is the set of instances I of size n.
- 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 will be at 5:00PM - 6:00PM today and 2:00PM - 3:00PM on Saturday in GHC 4301.
- If you want more practice for the course material, try the small group session!

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) Give an input positive integer N, output $N!$.

(b) Given as input a positive integer N, output True if $N = M!$ for some positive integer M.

(c) 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).

Where is the Median?

You are given two sorted arrays of integers with equal length. You want to determine the median of all the elements (if there are two elements in the middle, take the smallest one to be the median). Obtain a running time bound in terms of n, which we define to be the total number of elements in the arrays. We want you to measure the running time of the algorithms in terms of **the number of comparisons** (i.e. comparison of two numbers) they make. So 1 comparison corresponds to 1 step, and every other operation is free.

(Extra) $\mathcal{O}$, I Think I Understand Asymptotics Now

Let f, g, h be functions from $\mathbb{N}$ to $\mathbb{N}$. Prove or disprove the following:

(a) If $f \in \mathcal{O}(g)$ and $g \in \mathcal{O}(h)$, then $f \in \mathcal{O}(h)$

(b) If $f \in \mathcal{O}(g)$, then $g \in \mathcal{O}(f)$

(c) $f \in \mathcal{O}(g)$ or $f \in \Omega(g)$

(Bonus) What the, your Guesses are Two High!

Suppose I am thinking of a number between 1 and n, and will tell you if your guess is too high, too low, or correct. However, I only allow you to guess too high once, or you lose. How quickly can you guess my number? One possible solution is to just guess incrementally from 1 to my number, which takes $\mathcal{O}(n)$ time. Can you do better?