G22.1170: Fundamental Algorithms I
Problem Set 1
(Due Thursday, February, 22 2007)

Problems from Cormen, Leiserson and Rivest:
2-4 *Algebra with big-Oh* & 2-5 *Variations on O and Ω*. Also, 7.5-3 *Data Structure*.

2-4 Asymptotic notation properties

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c. $f(n) = O(g(n))$ implies $\lg(f(n)) = O(lg(g(n)))$, where $lg(g(n)) > 0$ and $f(n) \geq 1$ for all sufficiently large n.

d. $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$.

e. $f(n) = O((f(n))^2)$.

f. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$.

h. $f(n) + o(f(n)) = \Theta(f(n))$.

2-5 Variations on O and Ω

Some authors define Ω in a slightly different way than we do; let's use $\overset{\infty}{\Omega}$ (read "omega infinity") for this alternative definition. We say that $f(n) = \overset{\infty}{\Omega}(g(n))$ if there exists a positive constant c such that $f(n) \geq cg(n) \geq 0$ for infinitely many integers n.

a. Show that for any two functions $f(n)$ and $g(n)$ that are asymptotically nonnegative, either $f(n) = O(g(n))$ or $f(n) = \overset{\infty}{\Omega}(g(n))$ or both, whereas this is not true if we use Ω in place of $\overset{\infty}{\Omega}$.

b. Describe the potential advantages and disadvantages of using $\overset{\infty}{\Omega}$ instead of Ω to characterize the running times of programs.

Some authors also define O in a slightly different manner; let's use O' for the alternative definition. We say that $f(n) = O'(g(n))$ if and only if $|f(n)| = O(g(n))$.

c. What happens to each direction of the "if and only if" in Theorem 2.1 under this new definition?

Some authors define $\tilde{O}$ (read "soft-oh") to mean O with logarithmic factors ignored:

$\tilde{O}(g(n)) = \{f(n) :$ there exist positive constants $c, k,$ and n_0 such that $0 \leq f(n) \leq cg(n)\lg^k(n)$ for all $n \geq n_0\}$.

d. Define $\tilde{\Omega}$ and $\tilde{\Theta}$ in a similar manner. Prove the corresponding analog to Theorem 2.1.

7.5-3 Show how to implement a first-in, first-out queue with a priority queue. Show how to implement a stack with a priority queue. (FIFO's and stacks are defined in Section 11.1.)

Problem 1.1 Order the following functions by their growth rate (the most slowly-growing function appearing first); if two functions are same, group them together.

(1)	1	(2)	7	(3)	$7^{\lg n}$
(4)	$(\lg n)^{\lg n}$	(5)	$\sqrt{n}\lg^2 n$	(6)	n
(7)	$n \lg n$	(8)	$n^{\frac{1}{\lg n}}$	(9)	$n^{\lg 7}$
(10)	$n^{1+\frac{\lg\lg n}{\lg n}}$	(11)	$n^{\lg\lg n}$	(12)	$\left(1-\frac{1}{n}\right)^n$
(13)	$\left(1-\frac{1}{7}\right)^n$	(14)	$\left(1+\frac{1}{n}\right)^n$	(15)	$\left(1+\frac{1}{7}\right)^n$

Problem 1.2 a. Suppose $T_1(n)$ is $\Omega(f(n))$ and $T_2(n)$ is $\Omega(g(n))$. Which of the following statements are true? Justify your answer.

1. $T_1(n) + T_2(n) = \Omega\left(\max(f(n), g(n))\right).$
2. $T_1(n)\, T_2(n) = \Omega\left(f(n)\, g(n)\right).$

b. Now answer the previous question for this definition of $\overset{\infty}{\Omega}$.

(See problem 2-5 in CLR pp. 39: $T(n) = \overset{\infty}{\Omega}(f(n))$, if there is a positive constant C such that

$T(n) \geq C \cdot f(n) \geq 0,$ infinitely often, *i.e.*, for infinitely many values of n.)

Problem 1.3 Solve the following recurrence equation

$$T(n) = T(n-1) + 4\,n^3.$$