CS303 (Spring 2008)- Solutions to Assignment 3

Problem 3-2

The point of this problem is of course to gain some familiarity with big-O notation, and the growth of some of the functions we use frequently. Here are the solutions:

A	B	O	Ω	Θ
$\log^k n$	n^ϵ	$\surd$		
n^k	c^n	$\surd$		
$\sqrt{n}$	$n^{\sin n}$			
2^n	$2^{n/2}$		$\surd$	
$n^{\log m}$	$m^{\log n}$	$\surd$	$\surd$	$\surd$
$\log(n!)$	$\log(n^n)$	$\surd$	$\surd$	$\surd$

The key observations here: any power of a logarithm grows more slowly than any polynomial, which in turn grows more slowly than any exponential. For exponentials, b^n grows more slowly than c^n whenever $b < c$ (that is, $b^n = O(c^n)$, but not vice versa). $n^{\log m} = b^{\log n \log m} = m^{\log n}$, so those two are actually exactly the same function (where b is the base of the log used). $n!$ can be approximated by Stirling's Formula, and grows roughly like n^n/e^n (with polynomial error functions). Once you take logs, that becomes $n \log n - n = \Theta(n \log n)$, which is also $\log(n^n)$. Finally, $n^{\sin n}$ will take on values larger than $\sqrt{n}$ by any constant factor whenever $\sin n > 1/2$ (which happens infinitely often), and values arbitrarily smaller than $\sqrt{n}$ whenever $\sin n < 1/2$, which also happens infinitely often. So neither function is $O()$ of the other.

Problem 3-4

(a) False. Take $f(n) = 1$ and $g(n) = n$.

(b) False. Same example as above, where $\min(f(n), g(n)) = 1$, but $f(n) + g(n) = n + 1$.

(c) True. To prove this, notice that $f(n) = O(g(n))$ implies that $f(n) \leq cg(n)$ for all n, for some constant c. Therefore, because lg is a monotone function, $\lg(f(n)) \leq \lg(cg(n)) = \lg c + \lg g(n) = O(\lg(g(n)))$, because the constant term $\lg c = O(\lg(g(n)))$.

(d) False. Take $f(n) = 2n, g(n) = n$. Then, $f(n) = O(g(n))$, but $2^{f(n)} = 4^n$, and $2^{g(n)} = 2^n$. By the same argument as in Problem 3-2, 4^n grows faster than 2^n. For any constant c, choosing $n = 1 + \log_2 c$, we have that $4^n = 2c \cdot 2^n > c \cdot 2^n$, so f cannot be bounded by $c \cdot g$.

(e) Depends. If we restrict ourselves to integer-valued functions, then $f(n) \leq f(n)^2$, so this holds. If we allow decreasing functions, such as $f(n) = 1/n$, then it is false.

(f) True. If $f(n) = O(g(n))$, then there is a constant c (and without loss of generality, $c > 0$) such that $f(n) \leq cg(n)$ for all n. Then, $g(n) \geq \frac{1}{c} f(n)$, so $g(n) = \Omega(f(n))$.

(g) False. Take $f(n) = 4^n$. Then, $f(n/2) = 4^{n/2} = 2^n$, and by case (d) above, we don't get that $f(n) = O(f(n/2))$; in particular, we don't get $f(n) = \Theta(f(n/2))$.

Exercise 3.1-3

The statement is a little sloppy, but once we make it precise, it will be immediately obvious why it is content-free. When we say that a function (such as the running time $T(n)$) is $O(n^2)$, it means that $T(n) \leq cn^2$. Thus, saying that the running time is "at least $O(n^2)$" means that the running time grows at least as fast as some function $T(n) \leq cn^2$. In particular, that includes the function $T(n) = 0$. Thus, the statement is true for any algorithm with running time at least 0, which is not all that informative.