Jacob Alspaw
jaa134
EECS 340 - Algorithms
Assignment 2

3-3A

If we ignore functions that include a lg^* (log-star), then the ranking by order of growth can be seen below.

1. $2^{2^{n+1}}$	2. 2^{2^n}	3. $(n+1)!$	4. $n!$
5. e^n	6. $n * 2^n$	7. 2^n	8. $(\frac{3}{2})^n$
9. $(lg\ n)^{lg\ n}$	10. $n^{lg\ lg\ n}$	11. $(lg\ n)!$	12. n^3
13. n^2	14. $4^{lg\ n}$	15. $n\ lg\ n$	16. $lg(n!)$
17. n	18. $2^{lg\ n}$	19. $(\sqrt{2})^{lg\ n}$	20. $2^{\sqrt{2lg\ n}}$
21. $lg^2\ n$	22. $ln\ n$	23. $\sqrt{lg\ n}$	24. $ln\ ln\ n$
25. $n^{\frac{1}{lg\ n}}$	26. 1		

Exponential functions will grow much faster than any polynomial function. Function 1 beats function 2 because its uppermost power is always one greater. Factorials that start at greater numbers are larger. Number 4 is greater than number 5 because there is the same amount of numbers multiplied, but the average number multiplied is larger in function 4. Function 5 beats function 6 because the dominating term is the exponential case, where function 5 has the greater base. Function 6 is greater than function 7 because it is multiplied by a constant. Function 8 has a smaller base than function 7. Function 9 is also exponential, but the exponential case grows much slower than previous examples. The base may grow faster in function 10 than function 9, but the exponential value grows much much slower. Function 12 is greater than function 11 for a decent amount of numbers, but at substantially high numbers, factorials increase amazingly well. Function 12 is a higher order polynomial than function 13. Function 13 is just slightly less as we approach infinity than function 13. The exponential case of function 14 will increase faster than the logarithmic case of function 15. Function sixteens factorial does not cancel the hindered growth it recieves from the log. Function 18 is exponential, but the exponential case grows slow enough that it is slower than the linear case of function 17. Function 19 has a lower base than function 18. Function 20 has an extremely slow growing exponential case. Log base 2 will grow faster than natural log, especially when the log is squared like in function 21. Root log in function 23 is

extremely slow growing, but still faster than a double natural log. Function 25 will tend a value of 1 at substantially high values, but will asymptotically remain above 1.
Functions 25 and 26 are in the same class such that $g_{25}(n) = \Theta(g_{26}(n))$.

3-3B

An example of a non-negative function $f(n)$ such that for all functions $g_i(n)$ in part (a), $f(n)$ is neither $O(g_i(n))$ nor $\Omega(g_i(n))$:

$f(n) = 0$ when n is odd

$f(n) = (n^{n!})!$ when n is even

•$f(n)$ is NOT $\leq c * g_i(n)$ for all $n \geq n_0 \rightarrow f(n) \neq O(g_i(n))$
•$f(n)$ is NOT $\geq c * g_i(n)$ for all $n \geq n_0 \rightarrow f(n) \neq \Omega(g_i(n))$

3-4A

False, if $f(n) = n$ and $g(n) = n^2$, then $f(n) = O(g(n))$ holds true, but $g(n) = O(f(n))$ does not hold true.

3-4B

False, if $f(n) = n^2$ and $g(n) = n$, then $g(n)$ would be considered the minimum. The statement $n^2 = \Theta(n)$ is not true.

3-4C

True, $f(n) = O(g(n))$ suggests that there exists some positive constant, c, by which $f(n) \leq c * g(n)$ for all $n \geq n_o$. If g(n) is greater than f(n) for all $n \geq n_o$, then the comparison of function f's and g's logs will hold the same relationship, which will imply that the statement is true.

$$0 \leq lg(f(n)) \leq lg(c * g(n)) = lg(c) + lg(g(n)) = (lg(c) + 1)lg(g(n))$$

If we say $lg(c)+1 = constant$, then we obtain $lg(f(n)) \leq constant * lg(g(n))$, which is by the definition of big-O.

3-4D

False, if $f(n) = 2n$ and $g(n) = n$, then the statement is false. $2^{f(n)} = 2^{2n} = 4^n$ while $2^{g(n)} = 2^n$. $2^{f(n)} = \omega(2^{g(n)}) which is not the same as big-O by definition.$

3-4E

False, if $f(n) = 1/n$, then $f(n)^2 = \frac{1}{n^4}$. The definition of big-O will not hold true, because $f(n)$ is a greater polynomial than $f(n)^2$. $f(n) = \omega(f^2(n))$ because $\lim_{x\to\infty} \frac{f(x)}{f^2(x)} = \lim_{x\to\infty} n = \infty$ which is the definition of little-Omega

and not big-O.

3-3F

True. If $f(n) = O(g(n))$, there exists a non-negative constant, c_o, such that $f(n) \leq c_o * g(n)$ for all $n \geq n_o$. This equates to $\frac{1}{c_o} f(n) \leq g(n)$ or $c_1 * f(n) \leq g(n)$ where $c_1 = \frac{1}{c_o}$. Because c_o originally made function $g(n)$ greater than $f(n)$ by definition of big-O, the inverse of c_o will make function $f(n)$ smaller than $g(n)$ which is definition of big-Omega.

3-3G

False, if $f(n) = n!$ then the definition of Θ notation does not hold true. There exists no positive constant, c, such that $n! \leq c * (\frac{n}{2})!$ for all $n \geq n_o$.

3-3H

True. A property of little-o notation is such: for anonymous function g(n), $\lim_{x \to \infty} \frac{g(n)}{f(n)} = 0$. We obtain the new equation $f(n) + g(n) = \Theta(f(n))$, where $g(n) \leq c * f(n)$ for all constants $c \geq 0$ and all $n \geq n_o$. Therefore, we can say that function $f(n)$ has greater growth and we can ignore function $g(n)$ for the Theta notation. We obtain the final equation $f(n) = \Theta(f(n))$, which is true. There exists constants k_1 and k_2 such that $k_1 * f(n) \leq f(n) \leq k_2 * f(n)$ for all $n \geq n_o$.