Comparing time complexity of algorithms

From Chapter 3, standard notation and common functions:

- When the base of a log is not mentioned, it is assumed to be base 2.

- Analogy between comparisons of functions f(n) and g(n) and comparisons of real numbers a and b:

$f(n) = O(g(n))$	is like	$a \leq b$
$f(n) = \Omega(g(n))$	is like	$a \geq b$
$f(n) = \Theta(g(n))$	is like	$a = b$
$f(n) = o(g(n))$	is like	$a < b$
$f(n) = \omega(g(n))$	is like	$a > b$

- A polynomial of degree d is $\Theta(n^d)$.

- For all real constants a and b such that $a > 1, b > 0$,

  $$\lim_{x \to \infty} \frac{n^b}{a^n} = 0$$

  so $n^b = o(a^n)$. Any exponential function with a base > 1 grows faster than any polynomial function.

- **Notation used for common logarithms:**
  $lgn = log_2 n$ (binary logarithm)
  $lnn = log_e n$ (natural logarithm)

- **More logarithmic facts:**
  For all real $a > 0, b > 0, c > 0$, and n,

  $a = b^{(log_b a)}$ // Ex: $2^{(lgn)} = n^{(lg2)} = n$
  $log_b a^n = n log_b a$
  $log_b x = y$ iff $x = b^y$
  $log_b a = (log_c a)/(log_c b)$ // the base of the log doesn't matter asymptotically
  $a^{(log_b c)} = c^{(log_b a)}$

  $lg^b n = o(n^a)$ // any polynomial grows faster than any polylogarithm
  $n! = o(n^n)$ // factorial grows slower than n^n
  $n! = \omega(2^n)$ // factorial grows faster than exponential with base ≥ 2
  $lg(n!) = \Theta(nlgn)$ // Stirling's rule

- Iterated logarithm function:
  $lg^* n$ (log star of n)
  $lg^{(i)} n$ is the log function applied i times in succession.
  $lg^* n = \min(i \geq 0 \text{ such that } lg^{(i)} n \leq 1)$

x	$lg^* x$
$(-\infty, 1]$	0
$(1, 2]$	1
$(2, 4]$	2
$(4, 16]$	3
$(16, 65536]$	4
$(65536, 2^{65536}]$	5

  $lg^* 2 = 1$
  $lg^* 4 = 2$
  $lg^* 16 = 3$
  $lg^* 65536 = 4$

  Very slow-growing function.

COMPARING TIME COMPLEXITY OF FUNCTIONS:

Given 2 functions, which one grows faster (i.e. which one grows faster)?

Tech 1: Factor sides by common terms
n^2 and n^3 // *divide both sides by* n^2 *to get 1 and* n
clearly n grows faster than 1, so $n^2 = \mathcal{O}(n^3)$

Tech 2: Take log of both sides, then substitute very large values for n
2^n and n^2
$lg 2^n = n lg 2 = n(1) = n$
$lg n^2 = 2 lg n$
substitute 2^{100} for n in checking n and $2lgn$,
we have $2^{100} > 2 * lg 2^{100} = 200$
So $2^n = \Omega(n^2)$
Exponentials dominate polynomials.

Tech 3: Take the limit as n goes to ∞.

- (in class example); Rank the functions given below by decreasing order of growth; that is, find an arrangement g_1, g_2, g_3, g_4 of the functions satisfying $g_1 = \Omega(g_2) = \Omega(g_3) = \Omega(g_4)$.

$$g_1 = 2^n \quad g_2 = n^{\frac{3}{2}} \quad g_3 = nlgn \quad g_4 = n^{lgn}$$

In the space below, list the functions given above in terms of decreasing running time (highest to lowest, left to right), as n increases to ∞ (justify your answers):

$$g_1 = \quad \Omega(g_4) = \quad \Omega(g_2) = \quad \Omega(g_3)$$

Note: The explanations below are not proofs because we can't prove anything by example. But they do give us an idea of the relative values of each function as n gets very large. Also, the substitution of 2^{128} for n is an arbitrary choice that allows us to compare functions with a very large value of n.

1. Explanation for $g_1 = \Omega(g_2)$:
   Show $2^n = \Omega(n^{\frac{3}{2}})$:
   Take lg of both sides to get $lg2^n = nlg2 = n$ and $lgn^{\frac{3}{2}} = \frac{3}{2}lgn$.
   Substitute large value for n: let $n = 2^{128}$. Then we are asking, which is bigger, 2^{128} or $\frac{3}{2}lg2^{128}$? We get $2^{128} > \frac{3}{2}lg2^{128}$ because $2^{128} \approx 3.4 * 10^{38} > \frac{3}{2}128 = 192$.
   Alternately, we could just use the observation that exponentials dominate polynomials for any base > 1.

2. Explanation for $g_1 = \Omega(g_3)$:
   Show $2^n = \Omega(nlgn)$:
   Take lg of both sides to get $lg2^n = nlg2 = n$ and $lg(nlgn) = lgn + lglgn$.
   Substitute large value for n: let $n = 2^{128}$. Then we are asking, which is bigger, 2^{128} or $lg2^{128} + lglg2^{128}$? We get $2^{128} > 128 + lglg2^{128} = 128 + lg128 = 128 + 7$. So $g_1 = \Omega(g_3)$ because $2^{128} \approx 3.4 * 10^{38} > 135$.

3. Explanation for $g_1 = \Omega(g_4)$:
   Show $2^n = \Omega(n^{lgn})$:
   Take lg of both sides to get $lg2^n = nlg2 = n$ and $lg(n^{lgn}) = lgn * lgn$.
   Substitute large value for n: let $n = 2^{128}$. Then we are asking, which is bigger, 2^{128} or $lg2^{128} * lg2^{128}$? We get $2^{128} \geq 128 * 128 = 16384$ because $2^{128} \approx 3.4 * 10^{38} > 16384$.

4. Explanation for $g_4 = \Omega(g_2)$:
   Show $n^{lgn} = \Omega(n^{\frac{3}{2}})$:
   Take lg of both sides to get $lg(n^{lgn}) = lgn * lgn$ and $lg(n^{\frac{3}{2}}) = \frac{3}{2} * lgn$.
   We can cancel a factor of lgn on each side to get $lgn > \frac{3}{2}$, which is true because $\frac{3}{2}$

is a constant.

5. Explanation for $g_4 = \Omega(g_3)$:
   Show $n^{lgn} = \Omega(nlgn)$:
   Take lg of both sides to get $lg(n^{lgn}) = lgn * lgn$ and $lg(nlgn) = lgn + lglgn$.
   Substitute large value for n: let $n = 2^{128}$. Then $128 * 128 = 16384 > lg2^{128} + lglg2^{128} = 128 + 7 = 135$.

6. Explanation for $g_2 = \Omega(g_3)$:
   Show $(n^{\frac{3}{2}}) = \Omega(nlgn)$:
   Take lg of both sides to get $lgn^{\frac{3}{2}} = \frac{3}{2}lgn$ and $lg(nlgn) = lgn + lglgn$.
   Substitute large value for n: let $n = 2^{128}$. Then $\frac{3}{2}lg2^{128} = 192 > lg2^{128} + lglg2^{128} = 135$. Since 192 is not that much larger than 135, choose n^{1024}. Then $\frac{3}{2}lg2^{1024} = 1536 > 1024 + 10$.