Plan for 241 day 2: 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:

  $$\begin{array}{lll} f(n) = O(g(n)) & \text{is like} & a \leq b \\ f(n) = \Omega(g(n)) & \text{is like} & a \geq b \\ f(n) = \Theta(g(n)) & \text{is like} & a = b \\ f(n) = o(g(n)) & \text{is like} & a < b \\ f(n) = \omega(g(n)) & \text{is like} & a > b \end{array}$$

- 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.
  $lg^* n = \min(i \geq 0$ 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

$lg2^n = nlg2 = n(1) = n$

$lgn^2 = 2lgn$

substitute 2^{100} for n in checking n and $2lgn$,

we have $2^{100} > 2 * lg2^{100} = 200$

So $2^n = \Omega(n^2)$

Exponentials dominate polynomials.

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