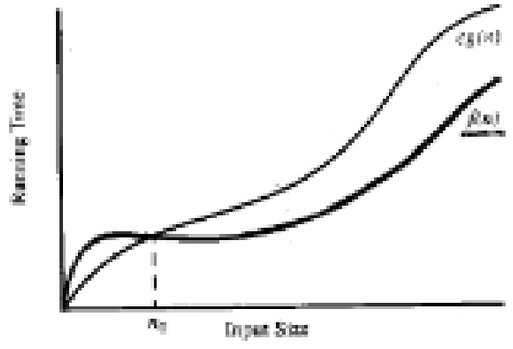
eecs 281 DATA STRUCTURES AND ALGORITHMS

Lecture 3: Algorithm Analysis
Foundational Data Structures
(Review of Some 280 Material)

Asymptotic Algorithm Analysis

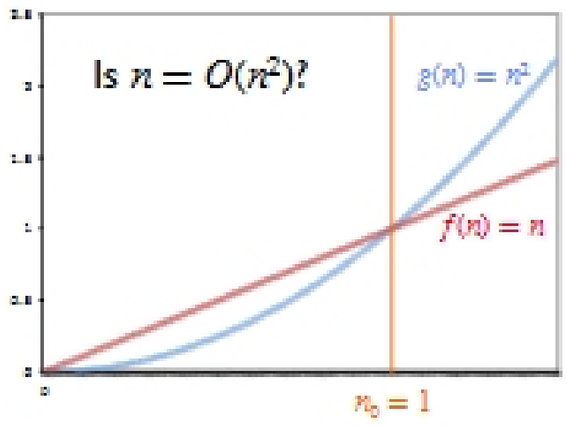
An algorithm with complexity $f(n)$ is said to be not slower than another algorithm with complexity $g(n)$ if $f(n)$ is bounded by $g(n)$ for large n



Commonly written as
$f(n) = O(g(n))$
(read: $f(n)$ is big-Oh $g(n)$),
a.k.a. the asymptotic (or big-Oh) notation

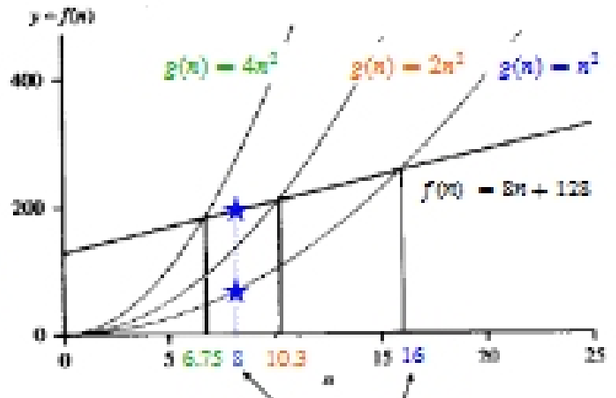
Big-Oh – Definition

$f(n) = O(g(n))$ if and only if there are constants
$c > 0$, $n_0 \geq 0$ such that $f(n) \leq c\ g(n)$ whenever $n \geq n_0$



Big-Oh – Example

Let
$f(n) = 8n + 128$
$g(n) = n^2$
Is $f(n) = O(g(n))$?
Is $8n + 128 \leq c\ n^2$?



Let $c = 1$, clearly, for $n = 8$, $f(n) > g(n)$
At what value of n_0 is $g(n) > f(n), \forall n \geq n_0$?

How about for $c = 2$ and $c = 4$?

Big-Oh – Definition

As long as there is a $c > 0$, and $n_0 \geq 0$ such that $c \cdot g(n) \geq f(n)$ for all $n \geq n_0$, we say that $f(n) = O(g(n))$
In this example, $8n + 128 = O(n^2)$

Mathematically:
$f(n) = O(g(n))$ iff $\exists\, c > 0, n_0 \geq 0 \mid \forall n, n \geq n_0, f(n) \leq c\, g(n)$
$O(g(n)) = \{f(n): \exists\, c > 0, n_0 \geq 0 \mid \forall n, n \geq n_0, 0 \leq f(n) \leq c\, g(n)\}$

So more accurately, $f(n) \in O(g(n))$
but conveniently people write $f(n) = O(g(n))$,
though NOT $f(n) \leq O(g(n))$

Big-Oh – Definition

In other words, we only care about LARGE n, it doesn't matter what c is

- obviously, c cannot be 10^{100} (one googol, the conjectured upper bound on the number of atoms in the observable universe)!

Also, asymptotically, $n^2 + k = O(n^2)$, k constant (Why?)

Big-Oh: Sufficient (but not necessary) Condition

If $\left[\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = c < \infty\right]$ then $f(n)$ is $O(g(n))$

$\log_2 n = O(2n)$?
$f(n) = \log_2 n$
$g(n) = 2n$

$\lim_{n\to\infty}\left(\frac{\log n}{2n}\right)$
$= \lim_{n\to\infty}\left(\frac{1}{2n}\right)$
$= 0 = c < \infty$

∞ / ∞
Use L'Hôpital's Rule
$\Rightarrow \log_2 n = O(2n)$

$\sin\left(\frac{n}{100}\right) = O(100)$?
$f(n) = \sin\left(\frac{n}{100}\right)$
$g(n) = 100$

$\lim_{n\to\infty}\left(\frac{\sin\left(\frac{n}{100}\right)}{100}\right)$

Condition does not hold but nevertheless it is true that $f(n) = O(g(n))$

L'Hôpital's Rule

If $\lim_{x\to c} f(x) = \lim_{x\to c} g(x) = 0$ or $\pm\infty$
and $\lim_{x\to c} f'(x)/g'(x)$ exists then

$$\lim_{x\to c}\frac{f(x)}{g(x)} = \lim_{x\to c}\frac{f'(x)}{g'(x)}$$

Also useful, derivative of log:

$$\frac{d}{dx}\log_b(x) = \frac{1}{x\ln(b)}$$

$$\frac{d}{dx}\ln(f(x)) = \frac{f'(x)}{f(x)}$$

wikipedia

Log Identities

Identity	Example
$\log_a(xy) = \log_a x + \log_a y$	$\log_2(12) =$
$\log_a(x/y) = \log_a x - \log_a y$	$\log_2(4/3) =$
$\log_a(x^r) = r \log_a x$	$\log_2 8 =$
$\log_a(1/x) = -\log_a x$	$\log_2 1/4 =$
$\log_a x = \frac{\log x}{\log a} = \frac{\ln x}{\ln a}$	$\log_7 9 =$
$k = \log_2 n$ iff $2^k = n$	$\log_a a = ?$ $\log_a 1 = ?$

Power Identities

Identity	Example
$a^{(n+m)} = a^n a^m$	$2^3 =$
$a^{(n-m)} = a^n / a^m$	$2^{3-2} =$
$(a^{(n)})^m = a^{nm}$	$(2^2)^2 =$
$a^{-n} = \frac{1}{a^n}$	$2^{-4} =$
$a^{-1} = ?$ $a^0 = ?$ $a^1 = ?$	

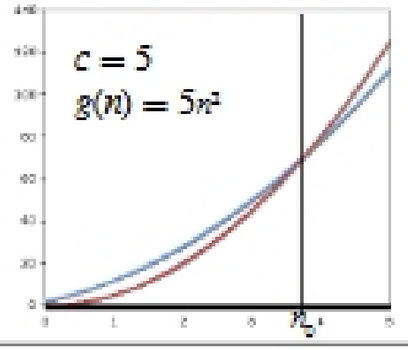
Big-Oh: We Can Drop Constants

$3n^2 + 7n + 42 = O(n^2)$?

$f(n) = 3n^2 + 7n + 42$
$g(n) = n^2$

Definition

$c > 0, n_0 \geq 0$ such that
$f(n) \leq c \cdot g(n), \forall n \geq n_0$



Sufficient Condition

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c < \infty$$

$$= \lim_{n\to\infty} \left(\frac{3n^2 + 7n + 42}{n^2} \right)$$

$$= \lim_{n\to\infty} \left(\frac{6n + 7}{2n} \right)$$

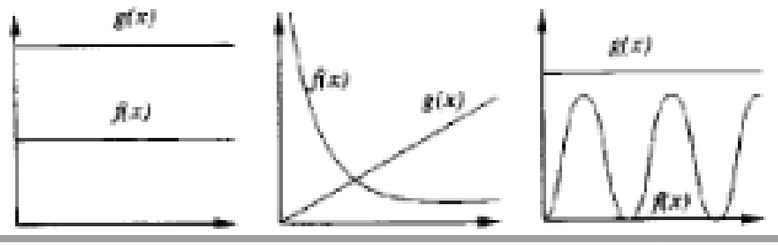
$$= \lim_{n\to\infty} \left(\frac{6}{2} \right)$$

Big-Oh – Common Mistakes

Mistake #0: $f(n) = O(g(n)) \Rightarrow f(n) = g(n)$ (NOT)

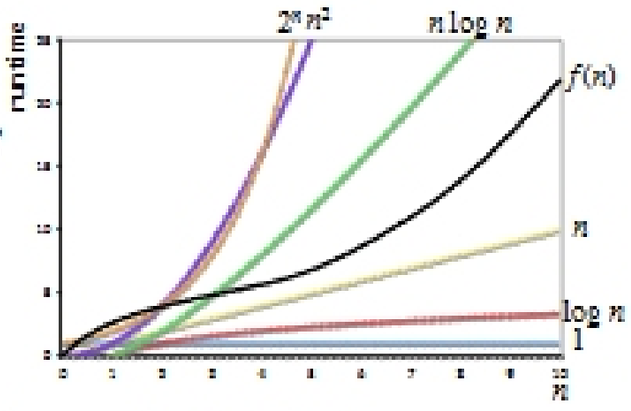
Mistake #1: If $f_1(n) = h(n)$ and $f_2(n) = h(n)$ then $f_1(n) = f_2(n)$; it follows that if $f_1(n) = O(g(n))$ and $f_2(n) = O(g(n))$ means $f_1(n) = f_2(n)$ (NOT)

Mistake #2: $f(n) = O(g(n)) \Rightarrow g(n) = O^{-1}(f(n))$ (NOT)
(There's no $O^{-1}()$!)



How Fast is Your Code?

Is $f(n) = O(2^n)$?
Is $f(n) = O(n^2)$?
Is $f(n) = O(n \log n)$?
Is $f(n) = O(n)$?
Is $f(n) = O(\log n)$?



Let $f(n)$ be the complexity of your code, how fast would you advertise it as?

While $f(n) = O(g(n)) \nRightarrow f(n) = g(n)$, you want to pick a $g(n)$ that is as close to $f(n)$ as possible (a "tight" bound)