Big O Big Idea

The big idea for Big O is the following:

- Ignore constant coefficients.

  Constant multiples are roughly equivalent to hardware changes. Besides, this makes analysis much easier. (Note: sometimes the constants do matter; then a finer analysis is required.)

- Consider asymptotic behavior (large inputs)

  Most algorithms run reasonably well on small enough inputs. But as the inputs get larger and larger, performance worsens. Big O will be a measure of how performance degrades with input size.

Worst Case Upper Bounds

Big O provides upper bounds (says our algorithms will be no slower than something). Usually we will get upper bounds on *worst case* performance. This will then mean we have an upper bound (performance guarantee) on *all* cases. Big O can, however, be used to bound average case, best case, etc., if that is what we are interested in.

The Definition

We say that $f = O(g)$ (read "f is Big O of g"), if there is some constant k such that for large input sizes n:

$$f(n) \leq kg(n)$$

That is, if we ran an algorithm with performance f on a suitably fast machine, it would perform better than an algorithm with performance g on a slower machine (k times as slow).

Just the Facts

Here are the most important rules for working with Big O (assume that f and g are positive and increasing, like running times for most algorithms). In Big O expressions we can

replace	with	provided
$kf(n)$	$f(n)$	k is a constant
$f(n) + g(n)$	$\max(f(n), g(n))$	no restrictions
f	g	$f = O(g)$ in sums and products
a polynomial	x^n	n is the degree of the polynomial

Get in Line

The most important functions, in order from smallest to biggest are:

$$1, \log(n), n, n\log n, n^2, n^3, 2^n, 3^n, n!$$

The Rest of the Family

Big O is used for giving upper bounds ($f = O(g)$ means f is no worse than g, at least as far as big O measures). For lower bounds, and "roughly equal" we use "big Omega" and "big Theta" (Ω and Θ):

$$\begin{array}{lll} f = \Omega(g) & \text{means} & g = O(f). \\ f = \Theta(g) & \text{means} & f = O(g) \text{ and } g = O(f). \end{array}$$

So big O is a lot like $\leq$, big Omega is a lot like $\geq$, and big Theta is a lot like $=$.