CS 112: Notes on Asymptotic Notation

by Leo Reyzin

Computer scientists usually measure resources consumed by algorithms as a function of input size. (Typically we measure time, but we may be interested in other resources, such as memory, power, network bandwidth, etc.) Thus, to compare resource consumption, we need to learn to compare functions. Comparing functions is trickier than comparing numbers. The following observation helps: it is typically the case that we are interested in what happens as input size gets bigger.

Therefore, computer science uses notation for comparing functions as their input goes to infinity. The notation allows for fairly rough comparisons: with one exception, it ignores constant factors (this is justified by the fact that, for example, the number of instructions executed or wall clock time on different platforms will typically be within a constant factor of each other, so this notation makes life easier by not requiring you to specify which exact notion of time and what platform you are measuring)

Here the notation:

relation	rough meaning	$\lim_{n\to\infty} \frac{f(n)}{g(n)}$
$f = o(g)$	"$<$"	0
$f = O(g)$	"$\leq$"	$< \infty$
$f = \Theta(g)$	"$\approx$"	> 0 and $< \infty$
$f \sim g$	$=$	1
$f = \Omega(g)$	"$\geq$"	> 0
$f = \omega(g)$	"$>$"	∞

O means "at most" but is overused (people will make incorrect statements like "at least $O(n^2)$" when they really mean "at least $\Omega(n^2)$"). In fact, O is so popular that this notation is often called Big-O notation. Usually, what you need is Θ, because it gives you the right answer within a constant factor. The tilde ($\sim$) is favored by our textbook; it gives you the constant factor, as well, and only leaves out parts that go to zero as n goes to infinity.

It is often easy to take the limit of $f(n)/g(n)$ by simple algebraic manipulation. If that doesn't help, L'Hôpital's rule usually does.

There are pathological cases when the limit does not exist because $f(n)/g(n)$ keeps oscillating; however, I have not encountered such cases in real algorithms, so we will not worry about them here.