COSC 455
Algorithm Analysis Notation Handout
Spring 2014

Here's a quick and somewhat hand-wavy look at the various notations we use when we are analyzing the time or space complexity of an algorithm. One thing to note is that in all cases we are looking at the "real-world" CS definitions, which mean that all functions are functions of whatever we are measuring (time, space) and that they will never be negative.

Big O

Also known as Landau's symbol, this is the most common complexity descriptor we use. By definition:

$$O(g(n)) = \exists C, n \geq 0, f(n) \leq Cg(n)$$

Intuitively, big O indicates that the algorithm ($f(n)$) grows at the same rate or slower than $g(n)$ multiplied by some constant C. So, an algorithm that has a time growth function:

$$f(n) = 4n^3 + 5n^2 + 6n + 12$$

is $O(n^3)$. Why? The dominant factor here is the n^3 factor. For most $n, 5n^3$ will be larger than $f(n)$, so n^3 is our $g(n)$. Basically, big O represents $\leq$.

Other notations

Common additional notations include:

big omega - $f(n) = \Omega(g(n)) : f(n) \geq C(g(n))$
theta/big theta - $f(n) = \Theta(g(n)) : C_1 g(n) \leq f(n) \leq C_2 g(n)$

More rare additional notations include:

little o/small o - $f(n) = o(g(n)) : f(n) < Cg(n)$
little omega/small omega - $f(n) = \omega(g(n)) : f(n) > C(g(n))$