CIS 575. Introduction to Algorithm Analysis
Material for January 22, 2024

Reasoning About Big O

The topic of this note in covered in *Cormen*'s Section 3.2.

1 Rules for Big O

We shall now study some of the properties of the Big O relation.

1.1 Order Relation

It is easy to verify that *Big O* induces what mathematicians call a "preorder" in that it is

Reflexive $f \in O(f)$ for all f

Transitive if $f \in O(g)$ and $g \in O(h)$ then $f \in O(h)$.

Big-O "respects" the degree of a polynomial, in that

$$n^p \in O(n^q) \text{ if and only if } p \leq q$$

This property, which we shall use very frequently, might have caused us to *conjecture* that the preorder is *total* in that for given f, g, either $f \in O(g)$ or $g \in O(f)$ (or both). But that conjecture is *false*, as demonstrated by a simple counterexample involving a (rather contrived) function g that oscillates between 1 and n^2: $f(n) = n$, $g(n) = n^{1+\sin(n\frac{\pi}{2})}$.

1.2 Algebraic Rules

Assume that $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$. Then

- $f_1 \cdot f_2 \in O(g_1 \cdot g_2)$
- $f_1 + f_2 \in O(\mathsf{max}(g_1, g_2))$

The second property justifies that we can analyze the various phases of an algorithm separately, and then get the final answer as the *maximum* of the intermediate results. For example: if initialization takes time in $O(n)$ and processing takes time in $O(n^2)$ and wrapping up takes time in $O(n)$, then the running time is in $O(n^2)$.

To illustrate how to reason about big O, let us prove the second property. Our assumptions are that there exists $c_1, c_2 > 0$, and $n_1, n_2 \geq 0$, such that

$$\begin{array}{rcl} f_1(n) & \leq & c_1 g_1(n) \text{ for all } n \geq n_1 \\ f_2(n) & \leq & c_2 g_2(n) \text{ for all } n \geq n_2 \end{array}$$

With $\mathbf{n_0} = \mathsf{max}(\mathbf{n_1}, \mathbf{n_2})$, and with $g = \mathsf{max}(g_1, g_2)$, for $n \geq n_0$ we may now calculate

$$\begin{array}{rcl} f_1(n) + f_2(n) & \leq & c_1 g_1(n) + c_2 g_2(n) \\ & \leq & c_1 g(n) + c_2 g(n) \\ & = & (\mathbf{c_1} + \mathbf{c_2}) g(n) \end{array}$$

which does indeed establish $f_1 + f_2 \in O(g)$.

1.3 Limit Property

A *sufficient condition* for $f \in O(g)$ is that there exists c with $0 \leq c < \infty$ such that

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

For if that's the case, we know from calculus that there exists n_0 such that for $n \geq n_0$,

$$\frac{f(n)}{g(n)} \leq c + 1$$

and hence $f(n) \leq (c + 1) g(n)$.

From this result it is immediate to verify for example that for any non-negative a, and any b and c, we have $an^2 + bn + c \in O(n^2)$.

But the existence of a limit is *not* a necessary condition. To see this, let

$$f(n) = 2^{\lfloor \lg n \rfloor}$$

where lg denotes the *binary* logarithm (where for example $\lg 8 = 3$), and where $\lfloor x \rfloor$ denotes the *floor* of x (where for example $\lfloor 4.62 \rfloor = 4$.)

Then we have $f(n) \in O(n)$ (since $f(n) \leq 2^{\lg n} = n$) but the sequence

$$\frac{f(n)}{n}$$

does not have any limit (it fluctuates between $\frac{1}{2}$ and 1).