Basics of the Big Oh Notation

Suppose that g is a function from the positive integers to the real numbers. Recall that

$$O(g) = \left\{ f \colon \mathbf{Z}_+ \to \mathbf{C} \,\middle|\, \begin{array}{l} \text{there exists a positive integer } n_0 \text{ and a constant } C \\ \text{such that } |f(n)| \leq C|g(n)| \text{ for all integers } n \geq n_0 \end{array} \right\}.$$

$$\begin{array}{cl}
O(1) & \text{constant complexity} \\
\cap & \\
O(\log n) & \text{logarithmic complexity} \\
\cap & \\
O(n) & \text{linear complexity} \\
\cap & \\
O(n \log n) & \\
\cap & \\
O(n^2) & \text{quadratic complexity} \\
\cap & \\
O(n^3) & \text{cubic complexity} \\
\cap & \\
O(2^n) & \text{exponential complexity} \\
\cap & \\
O(e^n) & \text{exponential complexity}
\end{array}$$

Figure 1: A hierarchy of common complexity classes.

Lemma 1. *If $f \in O(g)$, then $O(f) \subseteq O(g)$.*

Proof. By definition, $f \in O(g)$ means that there exists an integer $n_0 > 0$ and a real number C such that $|f(n)| \leq D|g(n)|$ for all $n \geq n_0$. Similarly, if $f_0 \in O(f)$, then there exists an integer $m_0 > 0$ and a real number D such that $|f_0(m)| \leq C|f(m)|$ for all $m \geq m_0$. Therefore, $|f_0(n)| \leq CD|g(n)|$ for all $n \geq \max(n_0, m_0)$; thus, $f_0 \in O(g)$. $\square$

Recall the following simple fact from the lecture notes on Asymptotics.

Lemma 2. *If f and g are functions from the positive integers to the complex numbers such that $g(n) \neq 0$ for all large n, then $\lim_{n\to\infty} |f(n)/g(n)| = 0$ implies $f = O(g)$.*

Theorem 3. *The inclusions given in Figure 1 hold.*

Proof. We have

$$\lim_{n\to\infty} \frac{1}{\log n} = \lim_{n\to\infty} \frac{\log n}{n} = \lim_{n\to\infty} \frac{n}{n \log n} = \lim_{n\to\infty} \frac{n \log n}{n^2} = 0$$

and

$$\lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{n^3}{2^n} = \lim_{n\to\infty} \frac{2^n}{e^n} = 0.$$

By Lemma 2, this implies $1 = O(\log n)$, $\log n = O(n)$, $n = O(n \log n)$, $n \log n = O(n^2)$, $n^2 = O(n^3)$, $n^3 = O(2^n)$, $2^n = O(e^n)$. The claim follows from Lemma 1. $\square$

We will encounter many examples of algorithms that have a running time that belongs to one of the classes given in Figure 1. The standard template library of C++ provides the following examples.

Example 1. The stack container class in STL (which happens to be a special case of an double-ended queue) supports `push`, `pop`, `top` operations that take constant time $O(1)$.

Example 2. The set container class in the STL supports insert, find, erase operations that take $O(\log N)$ operations when the set is of size N.

Example 3. The STL support the data structure of a heap, which is a particular organization of a sequence for instance used in sorting. The `push_heap` and `pop_heap` operations take $O(\log N)$ time, `make_heap` takes $O(N)$ time, and `sort_heap` takes $O(N \log N)$ time.

Example 4. The `min` and `max` operations on container classes take $O(N)$ time.

For more examples, see [D.R. Musser, G.J. Derge, and A. Saini, *STL Tutorial and Reference Guide*, Addison-Wesley, 2001]. For more details about the implementation of the standard template library, see [P.J. Plauger,A.A. Stepanov, M. Lee, and D.R. Musser, *The C++ Standard Template Library*, Prentice Hall, 2001].