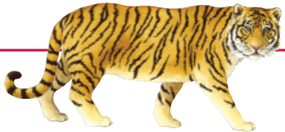

Special Topic 14.2

Oh, Omega, and Theta

We have used the big-Oh notation somewhat casually in this chapter, to describe the growth behavior of a function. Strictly speaking, $f(n) = O(g(n))$ means that f grows *no faster* than g.

But it is permissible for f to grow much more slowly. Thus, it is technically correct to state that $f(n) = n^2 + 5n - 3$ is $O(n^3)$ or even $O(n^{10})$.

Computer scientists have invented additional notation to describe the growth behavior of functions more accurately. The expression

$$f(n) = \Omega(g(n))$$

means that f grows at least as fast as g, or, formally, that for all n larger than some threshold, the ratio $f(n)/g(n) \geq C$ for some constant value C. (The Ω symbol is the capital Greek letter omega.) For example, $f(n) = n^2 + 5n - 3$ is $\Omega(n^2)$ or even $\Omega(n)$.

The expression

$$f(n) = \Theta(g(n))$$

means that f and g grow at the same rate—that is, both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ hold. (The Θ symbol is the capital Greek letter theta.)

The Θ notation gives the most precise description of growth behavior. For example, $f(n) = n^2 + 5n - 3$ is $\Theta(n^2)$ but not $\Theta(n)$ or $\Theta(n^3)$.

The Ω and Θ notation is very important for the precise analysis of algorithms. However, in casual conversation it is common to stick with big-Oh, while still giving as good an estimate as one can.