CIS 575. Introduction to Algorithm Analysis
Material for January 24, 2024

Big Omega and Big Theta

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

1 Big-Omega, Big-Theta

We have introduced the Big O notation which is used to express upper bounds. We shall now introduce two additional notations: Big Omega to express lower bounds, and Big Theta to express tight bounds. Both can be easily defined in terms of Big O.

Big-Omega Big Omega is dual to Big O:

$$g \in \Omega(f) \text{ holds exactly when } f \in O(g).$$

Thus, $n^p \in \Omega(n^q)$ iff $p \geq q$.

Big-Theta Big Theta combines Big O and Big Omega:

$$f \in \Theta(g) \text{ holds exactly when } f \in O(g) \text{ and } f \in \Omega(g)$$

Thus, for each g, $\Theta(g) = O(g) \cap \Omega(g)$.

We see that if $f \in \Theta(g)$ then there exists $c_1, c_2 > 0$, and $n_0 \geq 0$, such that for $n \geq n_0$, $f(n)$ is "sandwiched" between $c_1 g(n)$ and $c_2 g(n)$. In that case, f and g grow very much at the same rate.

$f \in \Theta(g)$ is obviously an equivalence relation[1] that induces equivalence classes among the functions. In particular,

$$n^p \in \Theta(n^q) \text{ iff } p = q$$

and if f is a polynomial of degree q, that is of the form $a_q \cdot n^q + a_{q-1} \cdot n^{q-1} + \ldots + a_1 n + a_0$ with $a_q > 0$, then $f \in \Theta(n^q)$.

With $\log_b$ the logarithm with base b, for $a, b > 1$ we have $\log_a(n) = \log_a(b) \log_b(n)$ and hence $\log_a(n) \in \Theta(\log_b(n))$. When describing asymptotic behavior, it thus does not matter which logarithm we use; we shall often use the binary $\log_2(n)$ which we often simply write as $\lg(n)$.

[1]That is, it is reflexive, transitive, and symmetric.