Characterizing Running Times

Devere Anthony Weaver

Introduction

When analyzing algorithm running times, we could compute the precise running time; however, this is tedious and doesn't provide much in return for the amount of effort. Thus we often concern ourselves for the cases of large inputs.

When looking at input sizes large enough to make relevant only the order of growth of the running time, we are studying **asymptotic efficiency** of algorithms. We're concerned with how the running time of an algorithm increases with the size of the input in the limit, as the size of the input increases without bound. Often an algorithm that is asymptotically more efficient is the best choice for all by very small inputs.

O-notation, Ω-notation, Θ-notation

O-notation characterizes and upper bound on the asymptotic behavior of a function. A function grows no faster than a certain rate based on the highest-order term.

Ω-notation characterizes a lower bound on the asymptotic behavior of a function. In other words, it says that a function grows at least as fast as a certain rate, based on the highest-order term.

Θ-notation characterizes a tight bound on the asymptotic behavior of a function. It says that a function grows precisely at a certain rate, based on the highest-order term.

The characterization of Θ-notation means if we can show some function $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$, then we have shown that $f(n) = \Theta(g(n))$. In other words, we can find a tight bound on $f(n)$ by showing that is bounded below and above by some function $g(n)$.

If needed, review the analysis of insertion-sort which was used to illustrate these informal definitions.

Asymptotic Notation: Formal Definitions

As described above, O-notation describes an **asymptotic upper bound**. We use it to give an upper bound on a function to within a constant factor.

Definition (O-notation)**.** *For a given function* $g(n)$*,*

$$O(g(n)) = \{f(n) : \exists c, n_0 > 0 \ni 0 \leq f(n) \leq cg(n), \ \forall n \geq n_0\}.$$

A function $f(n) \in O(g(n))$ if there exists a positive constant c such that $f(n) \leq cg(n)$ for sufficiently large n. The above definition also requires every function $f(n)$ in this set to be **asymptotically nonnegative** (i.e. nonnegative when n is sufficiently large). This implies $g(n)$ must also be asymptotically nonnegative.

Ω-notation provides an **asymptotic lower bound**.

Definition (Ω-notation)**.** *For a given function $g(n)$,*

$$\Omega(g(n)) = \{f(n) : \exists\ c, n_0 > 0 \ni 0 \leq cg(n) \leq f(n),\ \ \forall n \geq n_0\}.$$

We use Θ-notation for an **asymptotically tight bound**.

Definition (Θ-notation)**.** *For a given function $g(n)$,*

$$\Theta(g(n)) = \{f(n) : \exists\ c_1, c_2, n_0 > 0 \ni 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n),\ \ \forall n \geq n_0\}.$$

In other words, $f(n) \in \Theta(g(n))$ iff it is bound above and below by some constant values of $g(n)$. Again, this provides a tight bound on our function $f(n)$ and is more precise than the asymptotic notations above.

Theorem. *For any two functions $f(n)$ and $g(n)$, we have that $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.*

In other words, we can show a function $f(n)$ is tightly bound by $g(n)$ if we can show that $cg(n)$ is an upper bound of $f(n)$ and is $cg(n)$ is a lower bound of $f(n)$ for some $c > 0$.

In addition to the above asymptotic notations, there are two others that are commonly used.

Definition (o-notation)**.** *For some function $g(n)$,*

$$o(g(n)) = \{f(n) : \exists c, n_0 > 0 \ni 0 \leq f(n) < cg(n),\ \ \forall n \geq n_0\}.$$

The difference between the two "Oh"-notations being that the little-o notation says $cg(n)$ is strictly greater than $f(n)$ for all constants $c > 0$, while big-O notation says $cg(n)$ is greater than or equal to $f(n)$ for some constant $c > 0$. This slight difference in definitions changes the manner in which you have to prove these asymptotic notations hold.

Definition (ω-notation)**.** *For some function $g(n)$,*

$$\omega(g(n)) = \{f(n) : \exists\ c, n_0 > 0 \ni 0 \leq cg(n) < f(n),\ \ \forall n \geq n_0\}.$$

This is analogous to the difference between o-notation and O-notation.

Standard Notations and Common Functions

Note, this section covered a lot of mathematical notation and common formulas. Simply reference "CLRS" Ch.3.3 - "Standard Notations and Common Functions" as needed during analysis of algorithms.