Growth of Functions

The analysis of algorithms often requires a body of mathematical tools. In this lecture, we introduce some important tools and standards of notation.

- Asymptotic notation
- Standard notation and common functions

1 Asymptotic Notation

The notation used to describe the asymptotic running time of an algorithm is defined in terms of functions whose domains are the set of natural numbers.

This notation refers to how the problem scales as the problem gets larger. In some cases it is important to look at how fast the problem is for small amounts of input, particularly when the same task is being done over and over again. However, we are mainly concerned with algorithms for large problems, hence how the performance scales as the problem size approaches infinity is what we will look at.

1.1 O-notation

O-notation, pronounced "big-oh notation", is used to describe the asymptotic upper bound of an algorithm. Formally, it is defined as:

> For a given function, $g(n)$, we denote by $O(g(n))$ the set of functions, $O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.

Less formally, this means that for all sufficiently big n, the running time (space requirements etc) of the algorithm is less than $g(n)$ multiplied by some constant. Even less formally, if $f(n) = O(n^2)$ then we can say that $f(n)$ runs no slower than n^2.

For example:

- Given $f(n) = 3n + 5$, then $f(n) = O(n)$.
- Given $f(n) = n^3$, then $f(n) \neq O(n^2)$.
- Given $f(n) = n^2$, then $f(n) = O(n^3 - 7n^2 + 3)$.

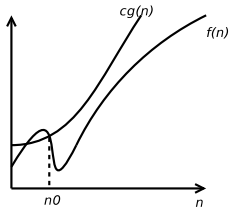

Figure 1: "There exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$." $f(n)$ is thus $O(g(n))$.

1.2 Ω-notation

Ω-notation, pronounced "big-omega notation", is used to describe the asymptotic lower bound of an algorithm. Formally, it is defined as:

> For a given function, $g(n)$, we denote by $\Omega(g(n))$ the set of functions, $\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.

Less formally, this means that for all sufficiently big n, the running time (space requirements etc) of the algorithm is greater than $g(n)$ multiplied by some constant. Even less formally, if $f(n) = \Omega(n^2)$ then we can say that $f(n)$ runs no faster than n^2.

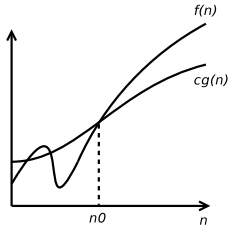

Figure 2: "There exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$." $f(n)$ is thus $\Omega(g(n))$.

For example:

- Given $f(n) = 3n + 5$, then $f(n) = \Omega(n)$.
- Given $f(n) = n^2$, then $f(n) \neq \Omega(n^3)$.
- Given $f(n) = n^4$, then $f(n) = \Omega(n^3 - 7n^2 + 3)$.

1.3 Θ-notation

We can now talk about algorithms in terms of a limit to how fast or how slow they run. However, our notation is not *asymptotically tight*. If we say that an algorithm runs in $f(n) = O(n^2)$, then that algorithm could actually run in n time. Usually we want to have a tight bound, we want to be able to say that an algorithm runs no slower and no faster than a particular function. To do this, we use Θ-notation.

> For a given function, $g(n)$, we denote by $\Theta(g(n))$ the set of functions $\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$.

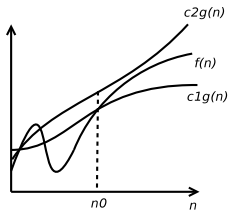

Figure 3: "There exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$". $f(n)$ is thus $\Theta(g(n))$.

For example:

- Given $f(n) = \frac{1}{2}n^2 - 3n + 7$, then $f(n) = \Theta(n^2)$.
- Given $f(n) = \sum_{i=1}^{n} i$, then $f(n) = \Theta(n^2)$.

1.4 *o*-notation

Sometimes we want to identify bounds that are not asymptotically tight, *o*-notation, pronounced little-oh notation. is used to denote an upper bound that is not asymptotically tight.

> For a given function, $g(n)$, we denote by $o(g(n))$ the set of functions $o(g(n)) = \{f(n)$: for *any* positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.

The implication of this, is that $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

For example, $2n = o(n^2)$ but $2n^2 \neq o(n^2)$.

1.5 ω-notation

ω-notation, pronounced little-omega notation, is used to denote a lower bound that is not asymptotically tight.

> For a given function, $g(n)$, we denote by $\omega(g(n))$ the set of functions $\omega(g(n)) = \{f(n)$: for *any* positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.

The implication of this is that $g(n)$ becomes insignificant relative to $f(n)$ as n approaches infinity:

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

For example $\frac{n^2}{2} = \omega(n)$ but $\frac{n^2}{2} \neq \omega(n^2)$.