Growth of Functions, Continued

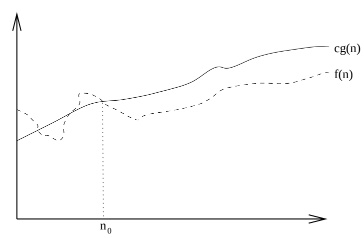
CLRS 3

Last time we looked at the problem of comparing functions (running times).

$$3n^2 \lg n + 2n + 1 \text{ vs. } 1000n \lg^{10} n + n \lg n + 5$$

Basically, we want to quantify how fast a function grows when $n \longrightarrow \infty$.
$\Downarrow$
asymptotic analysis of algorithms

More precisely, we want to compare 2 functions (running times) and tell which one is larger (grows faster) than the other. We defined O, Ω, Θ:



- f is below $g \Leftrightarrow f \in O(g) \Leftrightarrow f \leq g$
- f is above $g \Leftrightarrow f \in \Omega(g) \Leftrightarrow f \geq g$
- f is both above and below $g \Leftrightarrow f \in \Theta(g) \Leftrightarrow f = g$

Example: Show that $2n^2 + 3n + 7 \in O(n^2)$

Upper and lower bounds are symmetrical: If f is upper-bounded by g then g is lower-bounded by f and we have:

$$f \in O(g) \Leftrightarrow g \in \Omega(f)$$

(Proof: $f \leq c \cdot g \Leftrightarrow g \geq \frac{1}{c} \cdot f$). Example: $n \in O(n^2)$ and $n^2 \in \Omega(n)$

An $O()$ upper bound is not a tight bound. Example:
$2n^2 + 3n + 5 \in O(n^{100})$
$2n^2 + 3n + 5 \in O(n^{50})$
$2n^2 + 3n + 5 \in O(n^3)$
$2n^2 + 3n + 5 \in O(n^2)$

Similarly, an $\Omega()$ lower bound is not a tight bound. Example:
$2n^2 + 3n + 5 \in \Omega(n^2)$
$2n^2 + 3n + 5 \in \Omega(n \log n)$
$2n^2 + 3n + 5 \in \Omega(n)$
$2n^2 + 3n + 5 \in \Omega(\lg n)$

An asymptotically **tight** bound for f is a function g that is equal to f up to a constant factor: $c_1 g \leq f \leq c_2 g, \forall n \geq n_0$. That is, $f \in O(g)$ and $f \in \Omega(g)$.

Some properties:

- $f = O(g) \Leftrightarrow g = \Omega(f)$
- $f = \Theta(g) \Leftrightarrow g = \Theta(f)$
- reflexivity: $f = O(f), f = \Omega(f), f = \Theta(f)$
- transitivity: $f = O(g), g = O(h) \longrightarrow f = O(h)$

The growth of two functions f and g can be found by computing the limit $lim_{n \longrightarrow \infty} \frac{f(n)}{g(n)}$. Using the definition of O, Ω, Θ it can be shown that :

- if $lim_{n \longrightarrow \infty} \frac{f(n)}{g(n)} = 0$: then intuitively $f < g \Longrightarrow f = O(g)$ and $f \neq \Theta(g)$.
- if $lim_{n \longrightarrow \infty} \frac{f(n)}{g(n)} = \infty$: then intuitively $f > g \Longrightarrow f = \Omega(g)$ and $f \neq \Theta(g)$.
- if $lim_{n \longrightarrow \infty} \frac{f(n)}{g(n)} = c, c > 0$: then intuitively $f = c \cdot g \Longrightarrow f = \Theta(g)$.

This property will be very useful when doing exercises.

Comments

- The correct way to say is that $f(n) \in O(g(n))$. Abusing notation, people normally write $f(n) = O(g(n))$.

$$3n^2 + 2n + 10 = O(n^2), n = O(n^2), n^2 = \Omega(n), n \log n = \Omega(n), 2n^2 + 3n = \Theta(n^2)$$

- When we say "the running time is $O(n^2)$" we mean that the worst-case running time is $O(n^2)$ — best case might be better.
- When we say "the running time is $\Omega(n^2)$", we mean that the *best case* running time is $\Omega(n^2)$ — the worst case might be worse.
- Insertion-sort:
  - Best case: $\Omega(n)$
  - Worst case: $O(n^2)$
  - We can also say that worst case is $\Theta(n^2)$ because there exists an input for which insertion sort takes $\Omega(n^2)$. Same for best case.

- Therefore the running time is $\Omega(n)$ and $O(n^2)$.
- But, we cannot say that the running time of insertion sort is $\Theta(n^2)$!!!

- Use of O-notation makes it much easier to analyze algorithms; we can easily prove the $O(n^2)$ insertion-sort time bound by saying that both loops run in $O(n)$ time.
- We often use $O(n)$ in equations and recurrences: e.g. $2n^2 + 3n + 1 = 2n^2 + O(n)$ (meaning that $2n^2 + 3n + 1 = 2n^2 + f(n)$ where $f(n)$ is some function in $O(n)$).
- We use $O(1)$ to denote constant time.
- One can also define o and ω (little-oh and little-omega):
  - $f(n) = o(g(n))$ corresponds to $f(n) < g(n)$
  - $f(n) = \omega(g(n))$ corresponds to $f(n) > g(n)$
  - we will not use them; we'll aim for tight bounds Θ.
- Not all functions are asymptotically comparable! There exist functions f, g such that f is not $O(g)$, f is not $\Omega(g)$ (and f is not $\Theta(g)$).

Growth of Standard Functions

- Polynomial of degree d:
  $$a_0 + a_1 n + \ldots a_d n^d = \Theta(n^d)$$
  where $a_1, a_2, \ldots, a_d$ are constants (and $a_d > 0$).
- Any polylog grows slower than any polynomial:
  $$\log^a n = O(n^b), \forall a > 0$$
  Exercise: prove it!
- Any polynomial grows slower than any exponential with base $c > 1$:
  $$n^b = O(c^n), \forall b > 0, c > 1$$
  Exercise: prove it!

Review of Log and Exp

- Base 2 logarithm comes up all the time (from now on we will always mean $\log_2 n$ when we write $\log n$ or $\lg n$).
- Note: $\log n << \sqrt{n} << n$
- Log Properties:
  - $\lg^k n = (\lg n)^k$
  - $\lg \lg n = \lg(\lg n)$
  - $a^{\log_b c} = c^{\log_b a}$
  - $a^{\log_a b} = b$
  - $\log_a n = \frac{\log_b n}{\log_b a}$
  - $\lg b^n = n \lg b$
  - $\lg xy = \lg x + \lg y$
  - $\log_a b = \frac{1}{\log_b a}$
- Exp properties:
  - $a^0 = 1$
  - $a^{-1} = 1/a$
  - $(a^m)^n = a^{mn}$
  - $a^m \cdot a^n = a^{m+n}$