Chapter 13

Asymptotic Notation

13.1 Definitions

Analysis of algorithms is concerned with estimating how many steps various algorithms make while solving problems of various sizes. In particular, given an algorithm, we want to make statements like "For input of size n, the algorithm will terminate in at most $f(n)$ steps." If we try to accurately estimate the number of steps, a cumbersome bound like

$$f(n) = \frac{1}{11}n^3 + 12n^2 + 15\frac{1}{2}n + \log_3 n + 17$$

might arise. Such precision only complicates matters and does not add to our understanding of the algorithm's efficiency. The following notational convention allows to simplify bounds by concentrating on their "main terms."

Definition 13.1.1. *For two functions $f, g : \mathbb{N}^+ \to \mathbb{R}$,*

- *$f(n) = O(g(n))$ if and only if there exists a positive constant $c \in \mathbb{R}$ and a constant $n_0 \in \mathbb{N}$, such that $|f(n)| \leq c|g(n)|$ for all $n \geq n_0$.*
- *$f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$.*
- *$f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.*

Asymptotic notation does wonders to the above ugly bound: We can now say that $f(n) = \Theta(n^3)$, which makes it easier to see how the number of steps performed by the algorithm grows as n gets larger and larger. Notice how the asymptotic notation swallowed all the constants and lower-order terms! To prove that $f(n) = \Theta(n^3)$ we need to show that there exist positive constants $c_1, c_2 \in \mathbb{R}$ and a constant $n_0 \in \mathbb{N}$, such that $c_1 n^3 \leq f(n) \leq c_2 n^3$ for all $n \geq n_0$. (We dropped the absolute values that come from Definition 13.1.1 since $f(n)$ and n^3 are nonnegative for $n \in \mathbb{N}^+$.) We can take $n_0 = 1$, $c_1 = \frac{1}{11}$, and $c_2 = 45.6$. For the lower bound, clearly $f(n) \geq \frac{1}{11}n^3$ when $n \in \mathbb{N}^+$. For the upper bound, note that in this range $n^3 \geq n^2 \geq n \geq \log_3 n$, and $n^3 \geq 1$. All these inequalities can be proved by elementary algebraic manipulation. Thus we get

$$f(n) \leq \frac{1}{11}n^3 + 12n^3 + 15\frac{1}{2}n^3 + n^3 + 17n^3 \leq 45.6n^3.$$

We can also perfectly well say that $f(n) = O(n^4)$ or that $f(n) = O(n^{25})$; these bounds are considerably less informative but correct. On the other hand, the bound $f(n) = O(n^2)$ (or even $f(n) = O(n^{2.99})$) is *not* correct. Indeed, we have seen that $f(n) \geq \frac{1}{11}n^3$. On the other hand, for any positive constant $c \in \mathbb{R}$, $\frac{1}{11}n^3 \geq cn^2$ for all $n \geq 11c$. Thus there is no positive constant $c \in \mathbb{R}$ and a constant $n_0 \in \mathbb{N}$ so that $f(n) \leq cn^2$ for all $n \geq n_0$.

Asymptotic notation is asymmetric, so we never write a statement like $O(g(n)) = f(n)$; the O, Ω, and Θ are always present on the right side of the equality sign. (However, we can write $n^2 + O(n) = \Theta(n^2)$, for example.) The right way to think of statements like $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ is as inequalities; always remember what the notation means according to Definition 13.1.1.

13.2 Examples and properties

The following asymptotic inequalities can all be easily proved and are very useful. Do the proofs as an exercise. You might find induction or tools from elementary calculus helpful for some of these. You'll also need simple properties of logarithms, like the identity

$$\log_a n = \frac{\log_b n}{\log_b a}.$$

- For two constants $u, v \in \mathbb{R}$, if $u < v$ then $n^u = O(n^v)$. ("A bigger power swallows a smaller one.")
- If $f(n)$ is a degree-d polynomial in n then $f(n) = O(n^d)$. If the coefficient of n^d in $f(n)$ is nonzero then $f(n) = \Theta(n^d)$.
- For any real constants $b > 1$ and p, $n^p = O(b^n)$. ("An exponential swallows a power.")
- For any real constants $q > 0$ and p, $(\ln n)^p = O(n^q)$. ("A power swallows a logarithm.")
- For any real constants $a, b > 1$, $\log_a n = \Theta(\log_b n)$. This implies that we can write bounds like $O(\log n)$, $O(n \log n)$, etc., without specifying the base of the logarithm. ("Asymptotic notation swallows bases of logarithms.")

We conclude this lecture by demonstrating how new asymptotic inequalities can be derived from existing ones. These are often used in the analysis of algorithms, although they are so much a part of the folklore that they are rarely referred to explicitly.

Proposition 13.2.1. *The following hold:*

(a) If $f(n) = O(g(n))$ and $p \in \mathbb{N}$ is a constant then $p \cdot f(n) = O(g(n))$.

(b) If $f(n) = O(h(n))$ and $g(n) = O(w(n))$ then $f(n)+g(n) = O(\max(|h(n)|, |w(n)|))$.

(c) If $f(n) = O(h(n))$ and $g(n) = O(w(n))$ then $f(n) \cdot g(n) = O(h(n) \cdot w(n))$.

Proof. We prove each claim individually.

(a) If $f(n) = O(g(n))$ then there exists a positive constant $c \in \mathbb{R}$ and a constant $n_0 \in \mathbb{N}$, such that $|f(n)| \leq c|g(n)|$ for all $n \geq n_0$. Thus for $p \in \mathbb{N}$, $|p \cdot f(n)| = p|f(n)| \leq (pc)|g(n)|$ for all $n \geq n_0$, and by Definition 13.1.1, $p \cdot f(n) = O(g(n))$.

(b) If $f(n) = O(h(n))$ and $g(n) = O(w(n))$ then there exist two positive constants $c_1, c_2 \in \mathbb{R}$ and constants $n_1, n_2 \in \mathbb{N}$, such that $|f(n)| \leq c_1|h(n)|$ for all $n \geq n_1$ and $|g(n)| \leq c_2|w(n)|$ for all $n \geq n_2$. Then

$$|f(n)+g(n)| \leq |f(n)|+|g(n)| \leq c_1|h(n)|+c_2|w(n)| = (c_1+c_2)\max(|h(n)|, |w(n)|)$$

for all $n \geq \max(n_1, n_2)$, and by Definition 13.1.1, $f(n)+g(n) = O(\max(|h(n)|, |w(n)|))$.

(c) If $f(n) = O(h(n))$ and $g(n) = O(w(n))$ then there exist two positive constants $c_1, c_2 \in \mathbb{R}$ and constants $n_1, n_2 \in \mathbb{N}$, such that $|f(n)| \leq c_1|h(n)|$ for all $n \geq n_1$ and $|g(n)| \leq c_2|w(n)|$ for all $n \geq n_2$. Then

$$|f(n) \cdot g(n)| = |f(n)| \cdot |g(n)| \leq (c_1|h(n)|) \cdot (c_2|w(n)|) = (c_1c_2)|h(n) \cdot w(n)|$$

for all $n \geq \max(n_1, n_2)$, and by Definition 13.1.1, $f(n) \cdot g(n) = O(h(n) \cdot w(n))$.

□