

Asymptotic notation

Last modified: 2026-10-12 02:31:22 (PDT)

The cost of an algorithm depends on the size n of its input. Asymptotic notation compares how two costs grow as n grows, ignoring constant factors and small inputs.

Upper, lower and tight bounds

Definition 0.1 (Big-O notation). Let f and g be functions from the positive integers to $[0, \infty)$. We write $f(n) = O(g(n))$ if there are a constant $c > 0$ and an integer n_0 such that

$$f(n) \leq c g(n) \quad \text{for every } n \geq n_0.$$

That is, f grows no faster than a constant multiple of g once n is large enough. This is the definition of Arora and Barak (2009, sec. 0.3).

Definition 0.2 (Big-Omega notation). For f and g as in Definition 0.1, we write $f(n) = \Omega(g(n))$ if there are a constant $c > 0$ and an integer n_0 such that

$$f(n) \geq c g(n) \quad \text{for every } n \geq n_0.$$

Dividing by c shows that this holds exactly when $g(n) = O(f(n))$, with constant $1/c$.

Definition 0.3 (Big-Theta notation). For f and g as in Definition 0.1, we write $f(n) = \Theta(g(n))$ if both $f(n) = O(g(n))$ (Definition 0.1) and $f(n) = \Omega(g(n))$ (Definition 0.2). Then f and g grow at the same rate, up to constant factors.

Remark 0.1 (The equals sign is one-way). $O(g(n))$ stands for a set of functions, and $f(n) = O(g(n))$ says that f belongs to it. So $n = O(n^2)$ is true, but $n^2 = O(n)$ is false, and the statement cannot be read from right to left. Cormen et al. (2009, sec. 3.1) reads the notation this way.

Examples

Example 0.1 (A quadratic is $O(n^2)$ but not $O(n)$). Let $f(n) = 3n^2 + 5n$.

For $n \geq 5$, we have $5n \leq n \cdot n = n^2$, so

$$\begin{aligned} f(n) &= 3n^2 + 5n \\ &\leq 3n^2 + n^2 \\ &= 4n^2. \end{aligned}$$

So $f(n) = O(n^2)$, with $c = 4$ and $n_0 = 5$ in Definition 0.1. Since $f(n) \geq 3n^2$ for every n , also $f(n) = \Omega(n^2)$ (Definition 0.2), and so $f(n) = \Theta(n^2)$ (Definition 0.3).

But $f(n)$ is not $O(n)$. If it were, there would be $c > 0$ and n_0 with $3n^2 + 5n \leq cn$ for every $n \geq n_0$. Dividing both sides by $n > 0$ gives $3n + 5 \leq c$ for every $n \geq n_0$, which fails for any n larger than both n_0 and $(c - 5)/3$.

Example 0.2 (Counting the operations in an inner product). The inner product of two vectors $\tilde{x}, \tilde{z} \in \mathbb{R}^D$ is

$$\tilde{x}^\top \tilde{z} = x_1 z_1 + x_2 z_2 + \cdots + x_D z_D.$$

Computing it takes D multiplications, one for each product $x_d z_d$, and $D - 1$ additions to sum the D products: $2D - 1$ arithmetic operations in all. Since $2D - 1 \leq 2D$ for every $D \geq 1$, the cost is $O(D)$, with D in the role of n , $c = 2$ and $n_0 = 1$ in Definition 0.1. Since $2D - 1 \geq D$ for every $D \geq 1$, it is also $\Omega(D)$ (Definition 0.2), so it is $\Theta(D)$ (Definition 0.3). When a machine learning text says an inner product “costs $O(D)$ ”, it means this count of operations, up to a constant factor.

References

- Arora, Sanjeev, and Boaz Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511804090>.
- Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. 3rd ed. MIT Press.