A Short Note on O-Notation

Naman Kumar*
CMI

Abstract

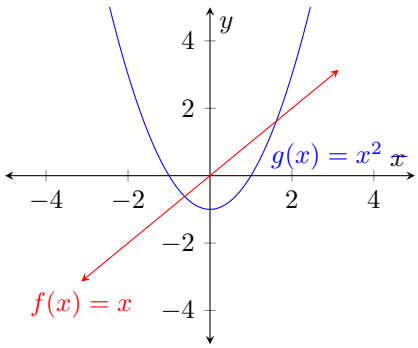
I refuse to believe that I am the only individual on Earth who has a consistent problem with understanding various O-notations coherently. This is a short compendium that illustrates what each of these notations refer to; I hope it serves as a simple lookup that makes it clear what each term represents.

1 Warmup

O-notation is used to represent certain relationships between *functions*. This relationship (informally) has to do with the growth rate of a function – it quantifies the idea that 'some functions grow faster than others.' At its core, therefore, O-notation is a way to represent asymptotic inequalities. What makes it confusing (to me) is this precise mapping: what exact inequality each notation represents.

We consider first a simple example.

Example 1.1. *The function $g(x) = x^2 - 1$ grows faster than $f(x) = x$.*



It can be easily seen that $f(x)$ 'overtakes' $g(x)$ at some point, from which point onwards each x satisfies the inequality $f(x) > g(x)$. This is the notion that is being captured by O-notation.

Definition 1.1 (Big-O Notation)**.** *Let $f, g : \mathbb{R} \to \mathbb{R}$ be two functions. We say that $f = O(g)$ iff there exists some $M \in \mathbb{R}^+$ and $x_0 \in \mathbb{R}$ such that*

$$|f(x)| \leq Mg(x)\ \forall x \geq x_0.$$

In other words, g **grows faster** than f, or **at the same rate**. Consider the examples:

- $x^2 = O(x^2)$.
- $x^2 = O(e^x)$.

*namankumar@cmi.ac.in.

- $x^2 \neq O(x)$.

Note that if the runtime of an algorithm is $O(f)$, then the algorithm runs **faster than or equal to** f.

- An $O(x^2)$ algorithm can run in time x^2.
- An $O(e^x)$ algorithm can run in time x^2.
- An $O(x)$ algorithm can never take time x^2 on any input.

2 List of Symbols

O-notation comprises of several symbols. Here we make note of a few of them.

2.1 Big-O

If $f(x) = O(g(x))$, then

- $g(x)$ grows faster than or equal to $f(x)$.
- An $O(g(x))$ time algorithm eventually runs in time less than or equal to $g(x)$.
- A protocol with communication $O(g(x))$ communicates less than or equal to $g(x)$ bits.

2.2 Little-o

If $f(x) = o(g(x))$, then

- $g(x)$ grows strictly faster than $f(x)$.
- An $o(g(x))$ time algorithm eventually runs in time strictly less than $g(x)$.
- A protocol with communication $o(g(x))$ communicates strictly less than $g(x)$ bits.

2.3 Big-Ω

If $f(x) = \Omega(g(x))$, then

- $g(x)$ grows slower than or equal to $f(x)$.
- An $\Omega(g(x))$ time algorithm eventually runs in time more than or equal to $g(x)$.
- A protocol with communication $\Omega(g(x))$ communicates more than or equal to $g(x)$ bits.

2.4 Little-ω

If $f(x) = \omega(g(x))$, then

- $g(x)$ grows strictly slower than $f(x)$.
- An $\omega(g(x))$ time algorithm eventually runs in time strictly more than $g(x)$.
- A protocol with communication $\omega(g(x))$ communicates strictly more than $g(x)$ bits.

2.5 Theta

If $f(x) = \Theta(g(x))$, then

- $g(x)$ grows at the same rate as $f(x)$.
- A $\Theta(g(x))$ time algorithm eventually runs in time a multiple of $g(x)$.

- A protocol with communication $\Theta(g(x))$ communicates a multiple of $g(x)$ bits.