Big O notation (Landau's symbol)

* describes the asymptotic behavior of functions.
  (How faster a function grows or declines)

* Rate of growth of a function ⟶ order
  (Hence, the "O")

* Definition ⇒ ← pervasive in computer science since natural numbers are considered.

Suppose $f(x)$ & $g(x)$ are defined on some subset of real numbers. Then,

$$f(x) = O(g(x)) \text{ for } x \to \infty$$

if and only there exist constants N & C such that

$$|f(x)| \leq C\,|g(x)| \text{ for all } x > N$$

* Generalized definition ⇒

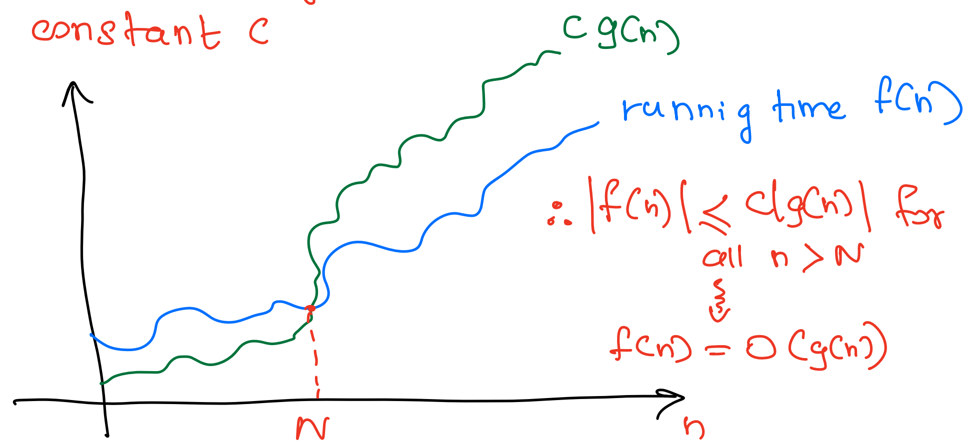
If a is a real number,

$$f(x) = O(g(x)) \text{ for } x \to a$$

iff there exists constants $d > 0$ and C s.t.

$$|f(x)| \leq C\,|g(x)| \text{ for all } x \text{ with } |x-a| < d$$

If a running time $f(n)$ is $O(g(n))$, then for larger enough n, the running time is at most for some constant c



$\therefore |f(n)| \leq c|g(n)|$ for all $n > N$

$\Downarrow$

$f(n) = O(g(n))$

Functions commonly encountered when analyzing algorithms:

* check wikipedia for a more comprehensive table.

notation	name.
$O(1)$	constant
$O(\log n)$	logarithmic
$O((\log(n))^c)$	polylogarithmic
$O(n)$	linear
$O(n^2)$	quadratic
$O(n^c)$	polynomial
$O(c^n)$	exponential

increasing complexity

Properties of Big O

* if the function f can be written as a finite sum of other functions, then the fastest growing one determines the order of f(n)

eg: $f(n) = 9\log(n) + 5(\log n)^4 + 3n^2 + 2n^3 = O(n^3)$
as $n \to \infty$

* $O(n^c) \neq O(c^n)$. if $c > 1$, c^n grows much faster.
* functions that grows faster than n^c for any c are
  superpolynomial
* functions that grows slower than c^n are
  Subexponential.
* $O(\log n) = O(\log n^c)$
  $\{$ because $O(c \log n) = O(\log n)$ $\}$
  Big Oh ignores the constant factors.
* $O(\log_a n) = O(\log_b n)$ where $a \neq b$
  log's with different constant bases are equivalent.
* $O(2^n) \neq O(3^n)$
  exponentials with different bases are not equivalent
* $O(n^2) = O((cn)^2) = O(c^2 n^2)$
* $O(2^n) \neq O(2^{nc}) = O((2^c)^n)$
* Product Rule:
  $f_1 = O(g_1)$ and $f_2 = O(g_2) \Rightarrow f_1 f_2 = O(g_1 g_2)$
  $f \cdot O(g) = O(fg)$
* Sum Rule:
  If $f_1 = O(g_1)$ & $f_2 = O(g_2)$ then
  $f_1 + f_2 = O(\max(g_1, g_2))$
* Multiplication by a constant:

Let k be a non-zero constant. then

$$O(|k|\, g) = O(g)$$

in other words $\Rightarrow$

$$f = O(g) \Rightarrow k.f = O(g)$$