

Symbol	“Think”	Value of L^\dagger	Formal Meaning [‡]
o	“ $<$ ”	$L = 0$	$f(n) = O(g(n))$ but not $f(n) = \Omega(g(n))$
O	“ \leq ”	$L < \infty$	$\exists c, n_0[\forall n \geq n_0[f(n) \leq cg(n)]]$
Ω	“ \geq ”	$L > 0$	$\exists c, n_0[c \neq 0 \ \& \ \forall n \geq n_0[f(n) \geq cg(n)]]$
Θ	“ $=$ ”.	$0 < L < \infty$	$f(n) = O(g(n))$ and also $f(n) = \Omega(g(n))$

[†] Value of $L = \lim_{n \rightarrow \infty} f(n)/g(n)$ when it exists.

[‡] Formal meaning of $f(n) = _ (g(n))$.

Figure 1: Greek Notation

Asymptotic Formulæ

$$f(n) = o(g(n))$$

Casual: f is *strictly* smaller (faster) than g .

Definition: $f(n) = O(g(n))$ but not $f(n) = \Omega(g(n))$

Calculus: $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$

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

Casual: f is smaller (faster) than or equal to g .

Informal: f is eventually smaller than g (up to a constant).

Definition: $\exists c, n_0$ such that $\forall n \geq n_0 \quad f(n) \leq cg(n)$

Calculus: $\lim_{n \rightarrow \infty} f(n)/g(n) < \infty$

$$f(n) = \Omega(g(n))$$

Casual: f is greater (slower) than or equal to g .

Informal: f is eventually greater than g (up to a constant).

Definition: $\exists c \neq 0, n_0$ such that $\forall n \geq n_0 \quad f(n) \geq cg(n)$

Calculus: $\lim_{n \rightarrow \infty} f(n)/g(n) > 0$

$$f(n) = \Theta(g(n))$$

Casual: f is approximately the same as g .

Definition: $\exists c_1 \neq 0, c_2, n_0$ such that $\forall n \geq n_0 \quad c_1g(n) \leq f(n) \leq c_2g(n)$

Calculus: $\lim_{n \rightarrow \infty} f(n)/g(n) \in (0, \infty)$

Usage the symbols O , o , Ω , and Θ are *not* function symbols. They should only be used “at the top level” on the right hand side of an equation. Always use parentheses with these functions.

Generic $\Theta(f(n))$ formulæ.

- If $f(n) = c \cdot g(n)$ then $f(n) = \Theta(g(n))$. *Constants don't matter.*
- If $f(n) = a_0 + a_1n + a_2n^2 + \dots + a_{k-1}n^{k-1} + a_kn^k$, where $a_k \neq 0$, then $f(n) = \Theta(n^k)$. *Don't sweat the small stuff (polynomials).*
- If $g(n) = o(f(n))$ then $f(n) + g(n) = \Theta(f(n))$. *Don't sweat the small stuff (general).*
- $f(n) + g(n) = \Theta(\max(f(n), g(n)))$. [This one comes up over and over in algorithm analysis.]
- If $f(n) = \Theta(\log n)$ and $g(n) = \Theta(n^\epsilon)$ (where $\epsilon > 0$) then $f(n) = o(g(n))$. *Logarithms are smaller than polynomials.*
- If $f(n) = \Theta(n^k)$ and $g(n) = \Theta(c^n)$ (where $c > 1$) then $f(n) = o(g(n))$. *Polynomials are smaller than exponential functions.*

Logarithms.

- If $f(n) \rightarrow +\infty$ and $g(n) \rightarrow +\infty$ and $f(n) = \Theta(g(n))$ then $\log f(n) = \Theta(\log g(n))$. (*N.B. the converse does not hold!*) *logs preserve Θ .*
- [Charlie's Rule] If $f(n) \rightarrow +\infty$ and $g(n) \rightarrow +\infty$ and $\lim_{n \rightarrow \infty} \frac{\log f(n)}{\log g(n)} < 1$ then $f(n) = o(g(n))$.¹ This rule helps a lot in dealing with odd functions like $n^{1/(\log \log n)}$.
- $\log_a n = \Theta(\log_b n)$. *The base doesn't matter.*

Factorials.

Stirling's formula $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + \frac{1}{288n^2} + o(n^{-2})\right)$

where $g(n) = o(n^{-2})$.

Consequently $\log(n!) = \Theta(n \log n)$.

¹I know about this nice result from Charlie Obimbo, who presented it at WCCCE '03.