Symbol	Notation	Bound Description	Limit theorem	Definition with constants	Example
Θ (==) Theta	$f(n) = \Theta(g(n))$	Asymptotic **tight** bound (g(n) is an asymptotic tight bound for f(n))	$\lim_{n\to\infty}\frac{f(n)}{g(n)} = c \neq 0$ (non-zero constant) It implies that: $\lim_{n\to\infty}\frac{g(n)}{f(n)} = \frac{1}{c} \neq 0$	There exist positive constants c_0, c_1 and n_0 s.t.: $c_0\, g(n) \le \boldsymbol{f(n)} \le c_1\, g(n)$ for all $n \ge n_0$	$\underbrace{25n^2 + 100n}_{f(n)} = \Theta(\underbrace{n^2}_{g(n)})$
O (≤) Big-Oh	$f(n) = O(g(n))$	Asymptotic **upper** bound (can be tight)	$\lim_{n\to\infty}\frac{f(n)}{g(n)} = 0 \text{ or } c$	There exist positive constants c_1 and n_0 such that: $\boldsymbol{f(n)} \le c_1\, g(n)$ for all $n \ge n_0$	$n^2 + 100n = O(n^3)$ $25n^2 + 100n = O(n^2)$
Ω (≥) Omega	$f(n) = \Omega(g(n))$	Asymptotic **lower** bound (can be tight)	$\lim_{n\to\infty}\frac{g(n)}{f(n)} = 0 \text{ or } c$	There exist positive constants c_0 and n_0 such that: $c_0\, g(n) \le \boldsymbol{f(n)}$ for all $n \ge n_0$	$n^2 + 100n = \Omega(n\sqrt{n})$ $25n^2 + 100n = \Omega(n^2)$ $\frac{n^2}{1000} - 300n = \Omega(n^2)$
o (<) Little-oh	$f(n) = o(g(n))$	Asymptotic **upper** bound but NOT tight	$\lim_{n\to\infty}\frac{f(n)}{g(n)} = 0$ Cannot be a constant	For any positive constant c_1, there exists n_0 s.t.: $\boldsymbol{f(n)} < c_1\, g(n)$ for all $n \ge n_0$	$n^2 + 100n = o(n^3)$ $25n^2 + 100n \neq o(n^2)$
ω (>) Little-omega	$f(n) = \omega(g(n))$	Asymptotic **lower** bound but NOT tight	$\lim_{n\to\infty}\frac{g(n)}{f(n)} = 0$ Cannot be a constant	For any positive constant c_0, there exist n_0 s.t.: $c_0\, g(n) < \boldsymbol{f(n)}$ for all $n \ge n_0$	$n^2 + 100n = \omega(n\sqrt{n})$ $25n^2 + 100n \neq \omega(n^2)$

Properties

1. *f(n) = **O**(g(n)) => g(n) = **Ω**(f(n))*
2. *f(n) = **Ω**(g(n)) => g(n) = **O**(f(n))*
3. *f(n) = **Θ**(g(n)) => g(n) = **Θ**(f(n))*
4. *If f(n) = O(g(n)) **and** f(n) = Ω(g(n)) => f(n) = Θ(g(n))*
5. *If f(n) = Θ(g(n)) => f(n) = O(g(n)) **and** f(n) = Ω(g(n))*

Transitivity (proved in slides):

6. *If* $f(n) = O(g(n))$ *and* $g(n) = O(h(n))$, *then* $f(n) = O(h(n))$.
7. *If* $f(n) = \Omega(g(n))$ *and* $g(n) = \Omega(h(n))$, *then* $f(n) = \Omega(h(n))$.

Substitution method:

If $\lim_{x\to\infty} h(x) = \infty$, and *h(x)* is monotonically increasing

then: $f(\boldsymbol{x}) = O(g(\boldsymbol{x})) \Rightarrow f(\boldsymbol{h}(\boldsymbol{x})) = O(g(\boldsymbol{h}(\boldsymbol{x})))$.

Notation abuse:

Instead of $f(n) \in \Theta(g(n))$

we use: $f(n) = \Theta(g(n))$

$a^{\log_b(n)} = n^{\log_b(a)}$ but $(a^n \neq n^a)$

If $0 \le c < d$, then $n^c = o(n^d)$.

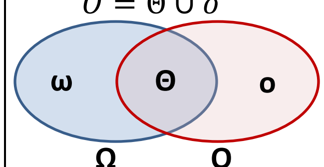
(Higher-order polynomials grow faster than lower-order ones.)

For any d, if $c > 1$, $n^d = o(c^n)$

(Exponential functions grow faster than polynomial ones.)

$\Omega = \Theta \cup \omega$

$O = \Theta \cup o$



e.g. for g(n) =n^2

Typically, *f(n)* is the running time of an algorithm. (*f(n)* can be a complicated function.)

We try to find a *g(n)* that is **simple** (e.g. n^2), and bounds *f(n). E.g. f(n) = Θ(g(n))* .