

To us, the function f will usually be the running time of an algorithm we analyze

O -notation[†]

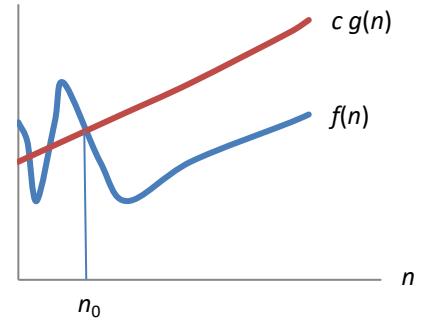
O , big O (upper limit)

Notation : $f(n) = O(g(n))$.

Intuition : f is smaller than g .

When $n \rightarrow \infty$: $f(n) \leq c \cdot g(n)$.

Definition : $\exists(c > 0), n_0 : \forall(n > n_0) f(n) \leq c \cdot g(n)$.



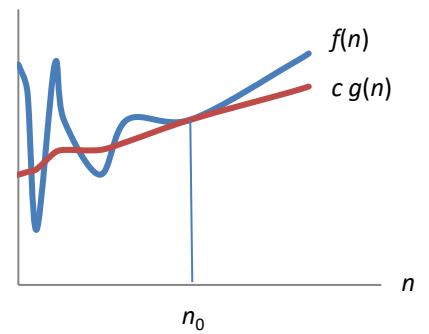
Ω (lower limit)

Notation : $f(n) = \Omega(g(n))$.

Intuition : f is larger than g .

When $n \rightarrow \infty$: $f(n) \geq c \cdot g(n)$.

Definition : $\exists(c > 0), n_0 : \forall(n > n_0) f(n) \geq c \cdot g(n)$.



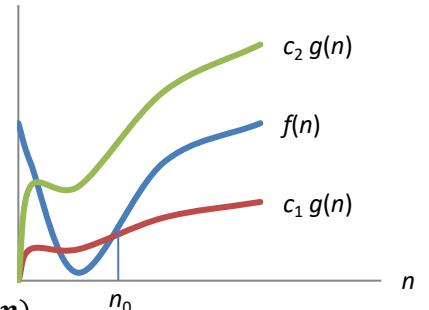
Θ (“as”)

Notation : $f(n) = \Theta(g(n))$.

Intuition : f grows like g , i.e. between $c_1 \cdot g$ and $c_2 \cdot g$.

When $n \rightarrow \infty$: $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$.

Definition : $\exists(c_1, c_2 > 0), n_0 : \forall(n > n_0) c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$.



o , little o [‡]

Notation : $f(n) = o(g(n))$.

Intuition : f is a lot smaller than g .

When $n \rightarrow \infty$: $f(n) \ll g(n)$.

Definition : $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$.

[†] More correctly, asymptotic notation.

[‡] O -notation might not be *asymptotically tight*: $2n^2 = O(n^2)$ is asymptotically tight, but $2n = O(n^2)$ is not. We use o -notation (little o) to indicate that our analysis is un-tight.