Notación asintótica

La notación asintótica surge de la necesidad de representar de forma *sencilla* la complejidad en tiempo (y en espacio) de un algoritmo.

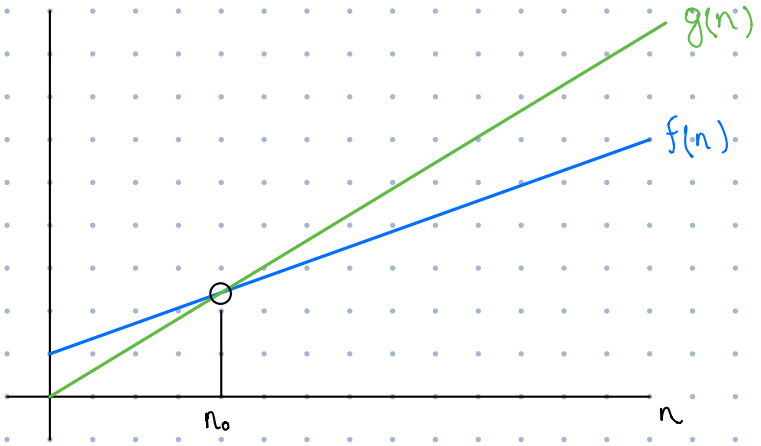
Notaciones

Supongamos que la función de tiempo de un algoritmo es $f(n)$

- *Big-oh* → O (nos indica una límite superior de $f(n)$)
- *Big-omega* → Ω (nos indica una límite inferior de $f(n)$)
- Theta → θ (nos indica un «límite» promedio de $f(n)$)

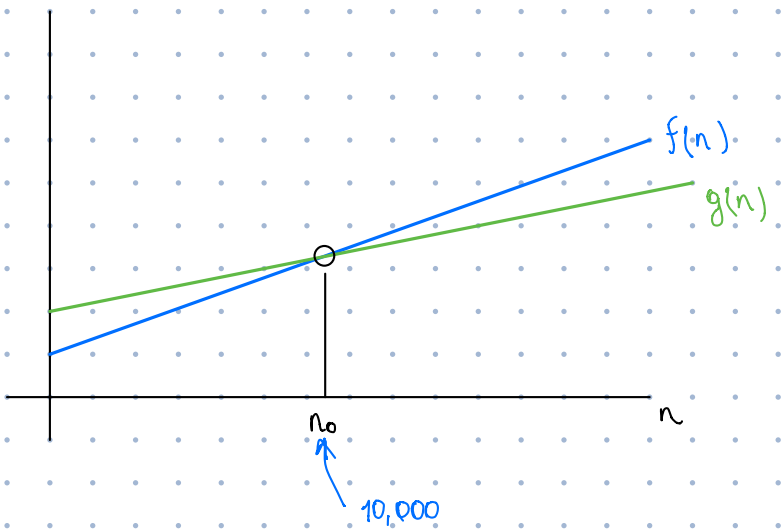
Notación big-oh

Una función $f(n) = O(g(n))$ ssi existen constantes positivas c y n_0 tales que $f(n) \leq c \cdot g(n)$ para todo $n \geq n_0$.



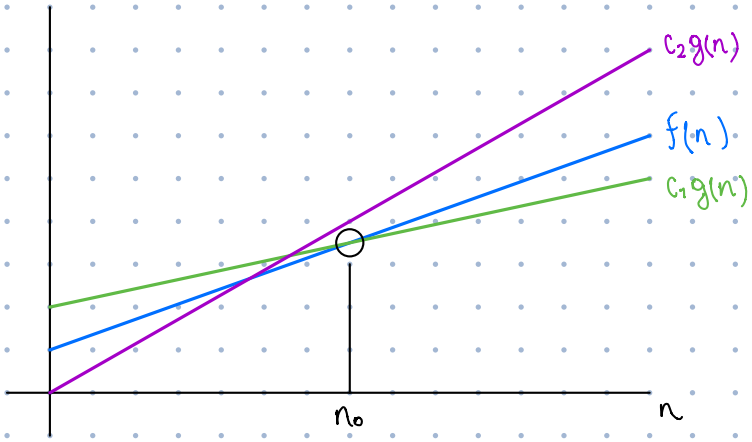
Notación big-omega

Una función $f(n) = \Omega(g(n))$ ssi existen constantes positivas c y n_0 tales que $f(n) \geq c \cdot g(n)$ para todo $n \geq n_0$.



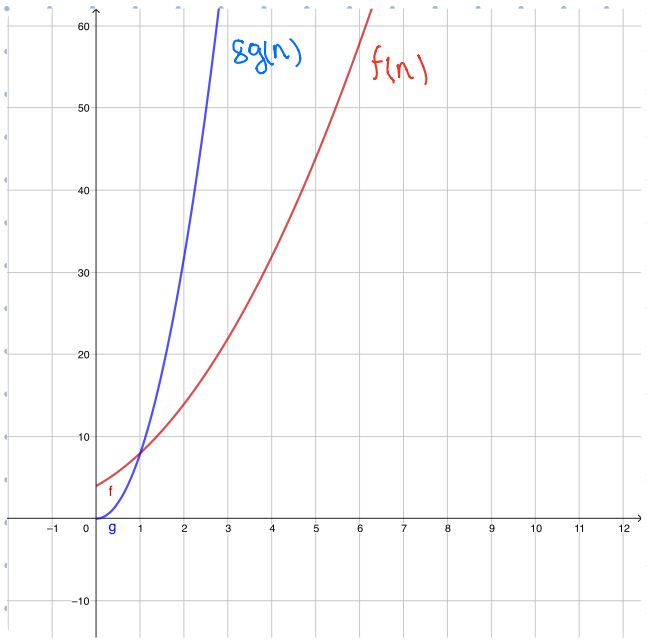
Notación theta

Una función $f(n) = \theta(g(n))$ ssi existen constantes positivas c_1, c_2 y n_0 tales que $c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq n_0$.

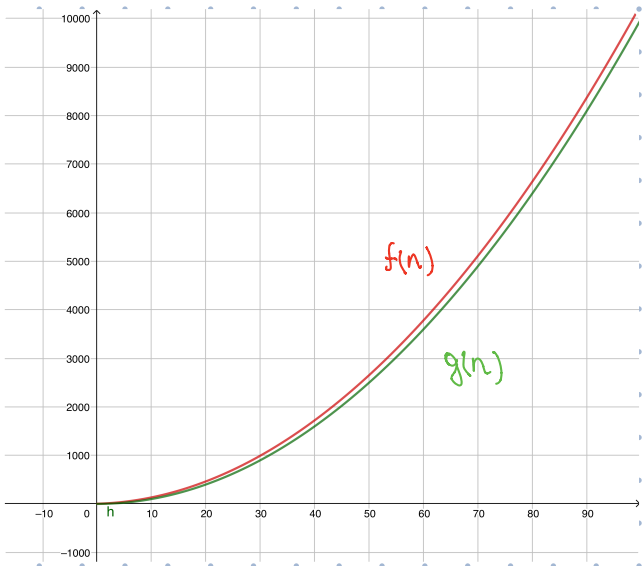


Ejemplo 1. Compruebe que $f(n) = n^2 + 3n + 4$ es $O(n^2)$, $\Omega(n^2)$ & $\theta(n^2)$.

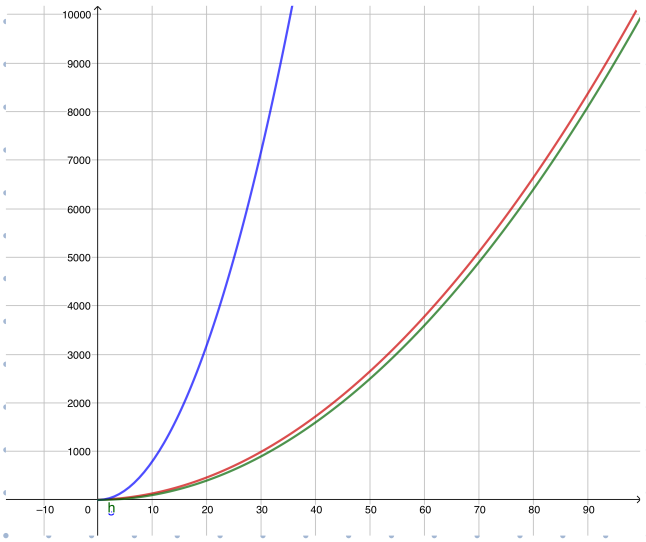
$n^2 + 3n + 4 \leq n^2 + 3n^2 + 4n^2 = 8n^2 \rightarrow c = 8$ & $f(n) = O(n^2)$



$n^2 \leq n^2 + 3n + 4 \rightarrow c = 1$ & $f(n) = \Omega(n^2)$



$n^2 \leq n^2 + 3n + 4 \leq 8n^2 \rightarrow c_1 = 1,\ c_2 = 8\ \&\ f(n) = \theta(n^2)$



⚠ Si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$, entonces $f(n) = \theta(g(n))$

Ejemplo 2. Compruebe que $f(n) = n!$ es $O(n^n)$ & $\Omega(1)$.

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n\text{-}1) \cdot n \leq n \cdot n \cdot n \cdot \ldots \cdot n \cdot n = n^n \rightarrow c = 1\ \&\ f(n) = O(n^n)$

$1 \cdot 1 \cdot 1 \cdot \ldots \cdot 1 \cdot 1 = 1 \leq 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n\text{-}1) \cdot n = n! \rightarrow c = 1\ \&\ f(n) = \Omega(1)$

Clases de funciones

Para la mayoría de los algoritmos, las funciones de tiempo pueden identificarse con alguna de las siguientes:

$O(1)$ — orden constante
$O(\log n)$ — orden logarítmico
$O(n)$ — orden lineal
$O(n \log n)$ — orden n logarítmo de n
$O(n^2)$ — orden cuadrático
$O(n^3)$ — orden cúbico
$O(2^n)$ — orden exponencial 2
$O(3^n)$ — orden exponencial 3
$O(n^n)$ — orden exponencial n

Podemos «ordenar» las funciones de tiempo según su comportamiento asintótico:

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < \ldots < O(2^n) < O(3^n)$$

⚠ Este es el objeto de estudio de la complejidad de algoritmos.

Ejemplo 3. Compare las funciones $f(n) = n^2 \log n$ & $g(n) = n(\log n)^{10}$.

$$n^2 \log n \sim n(\log n)^{10}$$

$$n^2 \log n - n(\log n)^{10} \sim 0$$

$$n \log n \,(n - (\log n)^9) \sim 0 \longrightarrow n \gtrsim (\log n)^9$$

$n \log n \geq 0$ (≥ 0 y ≥ 0)

$$n - (\log n)^9 \sim 0$$

$$n \sim (\log n)^9$$

* Aplicamos log de ambos lados

$$\log n \sim \log((\log n)^9)$$

$$\log n \sim 9 \log(\log n) \longrightarrow \log n \gtrsim 9 \log(\log n)$$

$\therefore f(n) \gtrsim g(n)$

⚠ El símbolo ≥ lo usamos en el estricto sentido del comportamiento asintótico. Esto quiere decir que puede ser que g sea «mayor» que f para algunos valores de n, pero que a partir de un cierto n_0 y en adelante, f es «mayor» que g.