Projeto e Análise de Algoritmos I
Notação Assintótica

Antonio Luiz Basile

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie

February 27, 2018

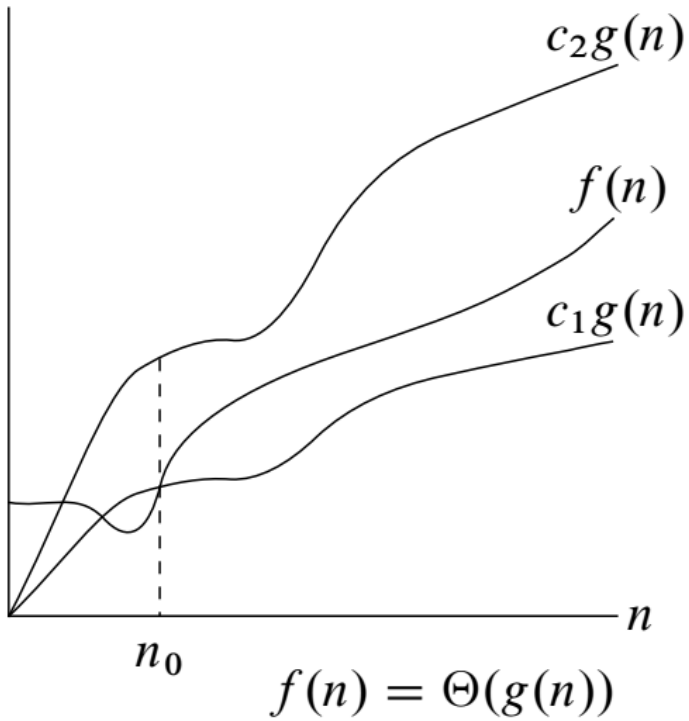
Crescimento de Funções (CLR)

- Quando olhamos para entradas com tamanhos suficientemente grandes para tornar apenas a ordem de crescimento do tempo de execução relevante, estamos estudando a eficiência **assintótica** dos algoritmos.
- As notações utilizadas para descrever o tempo de execução assintótico de um algoritmo são definidas em termos das funções cujos domínios são o conjunto dos **números naturais** $\mathbb{N} = \{0, 1, 2, \dots\}$.
- Em geral usaremos a notação assintótica para caracterizar o **tempo** de execução dos algoritmos. A notação assintótica, no entanto, pode ser aplicada a funções que caracterizam outros aspectos dos algoritmos, como por exemplo, o **espaço** utilizado.

Notação Θ

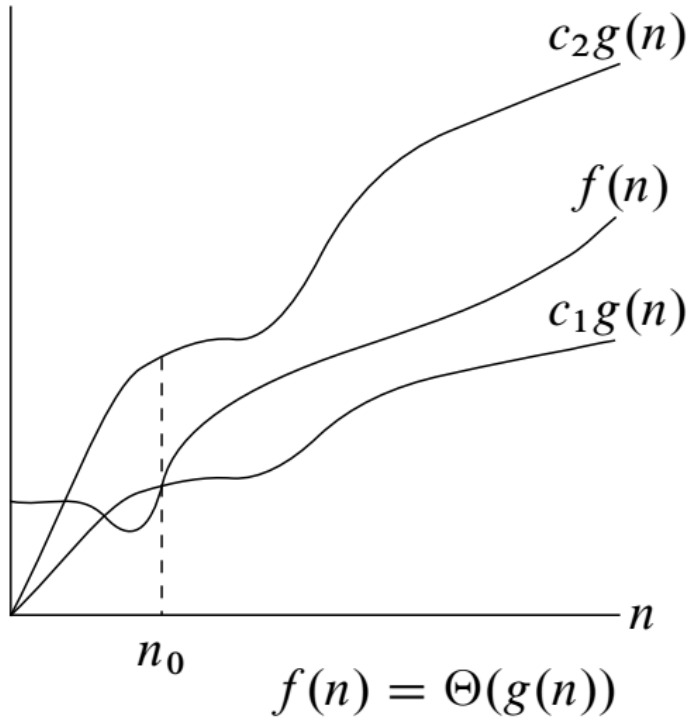
Para uma dada função $g(n)$, denotamos por $\Theta(g(n))$ o conjunto de funções

$$\Theta(g(n)) = \{f(n) : \text{existem constantes positivas } c_1,\ c_2 \text{ e } n_0 \text{ tal que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para todo } n \geq n_0\}.$$



Notação Θ

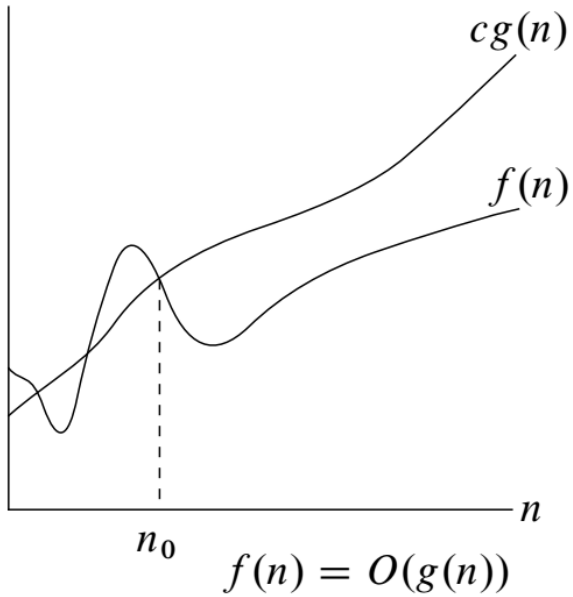
Uma função $f(n)$ pertence ao conjunto $\Theta(g(n))$ se existem constantes positivas c_1 e c_2 tal que ela possa ser sanduichada entre $c_1 g(n)$ e $c_2 g(n)$ para n suficientemente grande. Diz-se que $g(n)$ é um limite assintoticamente justo para $f(n)$.



Notação O

Para uma dada função $g(n)$, denotamos por $O(g(n))$ o conjunto de funções

$$O(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tal que } 0 \leq f(n) \leq cg(n) \text{ para todo } n \geq n_0\}.$$

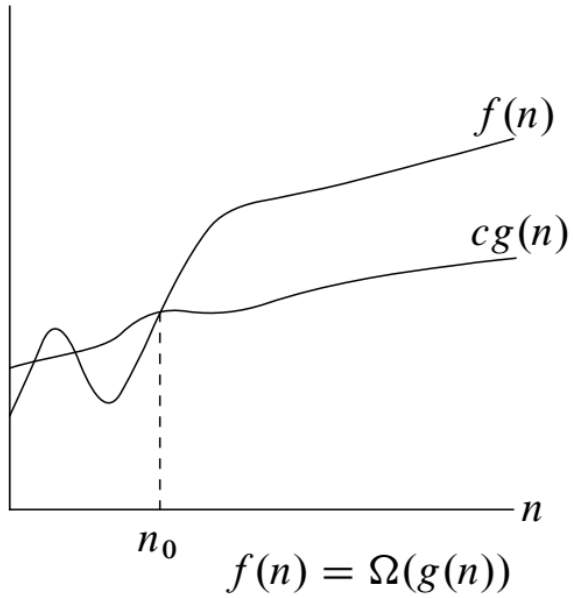


Usamos a notação O (lê-se "ó grande") como um limite superior para uma função, dentro de um fator constante.

Notação Ω

Para uma dada função $g(n)$, denotamos por $\Omega(g(n))$ o conjunto de funções

$$\Omega(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tal que } 0 \leq cg(n) \leq f(n) \text{ para todo } n \geq n_0\}.$$



Usamos a notação Ω (lê-se "ômega grande") como um limite assintótico inferior para uma função, dentro de um fator constante.

Comparando Funções

$$f(n) = \Theta(g(n)) \text{ e } g(n) = \Theta(h(n)) \text{ implica } f(n) = \Theta(h(n)) \quad (1)$$

$$f(n) = O(g(n)) \text{ e } g(n) = O(h(n)) \text{ implica } f(n) = O(h(n)) \quad (2)$$

$$f(n) = \Omega(g(n)) \text{ e } g(n) = \Omega(h(n)) \text{ implica } f(n) = \Omega(h(n)) \quad (3)$$

$$f(n) = \Theta(f(n)) \text{ e } f(n) = O(f(n)) \text{ e } f(n) = \Omega(f(n)) \quad (4)$$

$$f(n) = \Theta(g(n)) \text{ se e somente se } g(n) = \Theta(f(n)) \quad (5)$$

$$f(n) = O(g(n)) \text{ se e somente se } g(n) = \Omega(f(n)) \quad (6)$$

Exemplos

1. $2n + 10$ é $O(n)$?
2. n^2 é $O(n)$?
3. $3n^3 + 20n^2 + 5$ é $O(n^3)$?
4. $3 \log n + 5$ é $O(\log n)$?
5. 2^{n+2} é $O(2^n)$?

Exemplo 1: $2n + 10$ é $O(n)$?

Podemos realizar uma manipulação para encontrar c e n_0:

$$2n + 10 \leq c.n$$

$$c.n - 2n \geq 10$$

$$(c - 2)n \geq 10$$

$$n \geq \frac{10}{c - 2}$$

A afirmação é válida para $c = 3$ e $n_0 = 10$.

Exemplo 2: n^2 é $O(n)$?

É preciso encontrar c que seja maior ou igual a n para todo valor de n_0:

$$n^2 \leq c.n$$

$$n \leq c$$

É impossível, pois c deve ser constante.

Exemplo 3: $3n^3 + 20n^2 + 5$ é $O(n^3)$?

É preciso encontrar $c > 0$ e $n_0 \geq 1$, tais que $3n^3 + 20n^2 + 5 \leq c.n^3$, para $n \geq n_0$. Como

$$3n^3 + 20n^2 + 5 \leq (3 + 20 + 5).n^3$$

podemos tomar $c = 28$ e qualquer $n_0 > 1$

Exemplo 4: $3 \log n + 5$ é $O(\log n)$?

É preciso encontrar $c > 0$ e $n_0 \geq 1$, tais que $3 \log n + 5 \leq c . \log n$, para $n \geq n_0$. Note que

$$3 \log n + 5 \leq (3 + 5) . \log n \text{ se } n > 1 \ (\log 1 = 0)$$

basta tomar, por exemplo, $c = 8$ e qualquer $n_0 > 2$

Exemplo 5: 2^{n+2} é $O(2^n)$?

É preciso $c > 0$ e $n_0 \geq 1$, tais que $2^{n+2} \leq c.2^n$, para todo $n \geq n_0$. Note que

$$2^{n+2} = 2^2.2^n = 4.2^n$$

Assim, basta tomar, por exemplo, $c = 4$ e qualquer n_0