# Crescimento de funções. Notação assimptótica.

Fernando Lobo

Algoritmos e Estrutura de Dados



## Motivação

- Ter um modo de descrever a escalabilidade de algoritmos.
- Exemplo: Quando duplicamos o tamanho do input, o que é que acontece ao tempo de execução?
- Estamos interessados na ordem (ou taxa) de crescimento do tempo de execução de algoritmos.
- Usamos a notação assimptótica para descrever essa taxa de crescimento.
  - $O \rightarrow$ em inglês, big oh
  - $\Omega \rightarrow$ big omega
  - $\Theta \rightarrow$ big theta

# Notação O

- Seja $T(n)$ uma função (ex: tempo que INSERTION-SORT demora a ordenar $n$ elementos.)
- Dizemos que $T(n) = O(f(n))$ se para valores de $n$ suficientemente grandes,

$$T(n) \leq c \cdot f(n), \text{ em que } c \text{ é uma constante positiva}$$

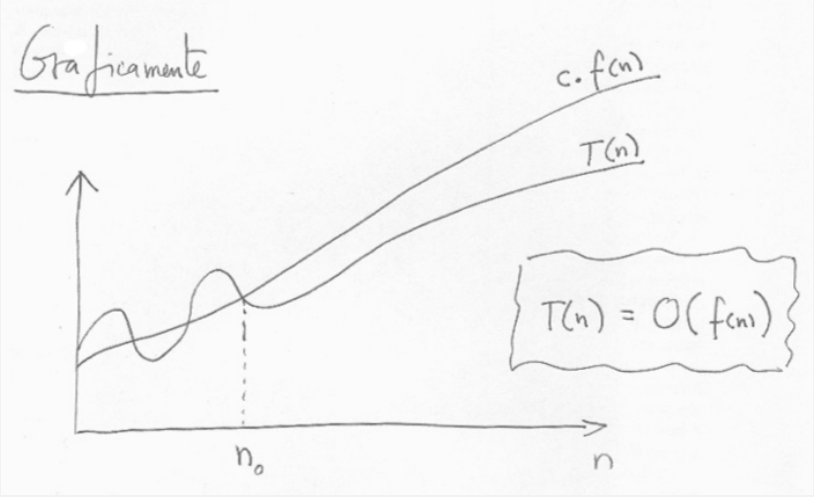
- O sinal '=' é um abuso de notação (significa $\in$)
- Lê-se $T(n)$ é de ordem $f(n)$
- O inverso não é verdadeiro.



# Notação O

- Formalmente $O(f(n))$ é um conjunto de funções.

$$O(f(n)) = \{T(n) : \exists_{c>0}\ \exists_{n_0>0}\ \forall_{n>n_0}\ 0 \leq T(n) \leq c \cdot f(n)\}$$




- $f(n)$ é assimptoticamente um limite superior para $T(n)$, a menos de um factor constante.

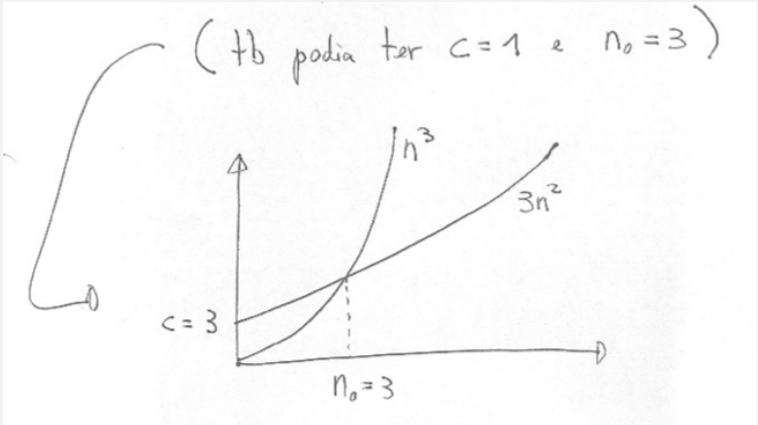
# Exemplo

- $3n^2 = O(n^2)$
- Porquê? Porque conseguimos "arranjar" um $c$ e um $n_0$ tal que: $3n^2 \leq c \cdot n^2$ , $\forall_{n>n_0}$
- Basta colocar $c = 3$, $n_0 = 1$
  (também podemos colocar $c = 5$, $n_0 = 83$)
- O importante é encontrarmos um $\underline{c}$ e um $\underline{n_0}$.



# Outro exemplo

- $3n^2 = O(n^3)$
- Também conseguimos "arranjar" um $c$ e um $n_0$ tal que: $3n^2 \leq c \cdot n^3$ , $\forall_{n>n_0}$
- Por exemplo $c = 3$, $n_0 = 1$



## Outro exemplo

- $1000n^2 + 1000n = O(n^2)$
- Temos de mostrar que $1000n^2 + 1000n \leq c \cdot n^2$, para um $c > 0$ , $\forall_{n>n_0}$
- Ex: $c = 1200$

$$1000n^2 + 1000n \leq 1200n^2$$
$$1000n \leq 200n^2$$
$$n \geq 5$$

- Em resumo: Para $c = 1200$ e $n_0 > 5$,

$$1000n^2 + 1000n \leq c \cdot n^2$$



## Outro exemplo

- Mostrar que $T_{\text{INSERTION-SORT}}(n) = an^2 + bn + c$, com $a$, $b$, $c$ constantes, é $O(n^2)$
- $\exists_{k>0}\ \exists_{n_0>0}\ :\ an^2 + bn + c \leq kn^2\ ,\ \forall_{n>n_0}$
- Basta ver que para $n \geq 1$: ($\rightsquigarrow$ 1 é o nosso $n_0$)
  - $bn \leq bn^2$
  - $c \leq cn^2$
- Logo:

$$\begin{aligned} an^2 + bn + c &\leq an^2 + bn^2 + cn^2 \\ &= (a + b + c)n^2 \\ &= kn^2 \end{aligned}$$

Ou seja: $n_0 = 1$, $k = (a + b + c)$ $\square$

## Notação O dá-nos um limite superior

- $T_{\textsc{Insertion-Sort}}(n) = O(n^2)$
- mas também é $O(n^3)$
- e também é $O(n^7)$
- e também é $O(2^n)$
- etc.



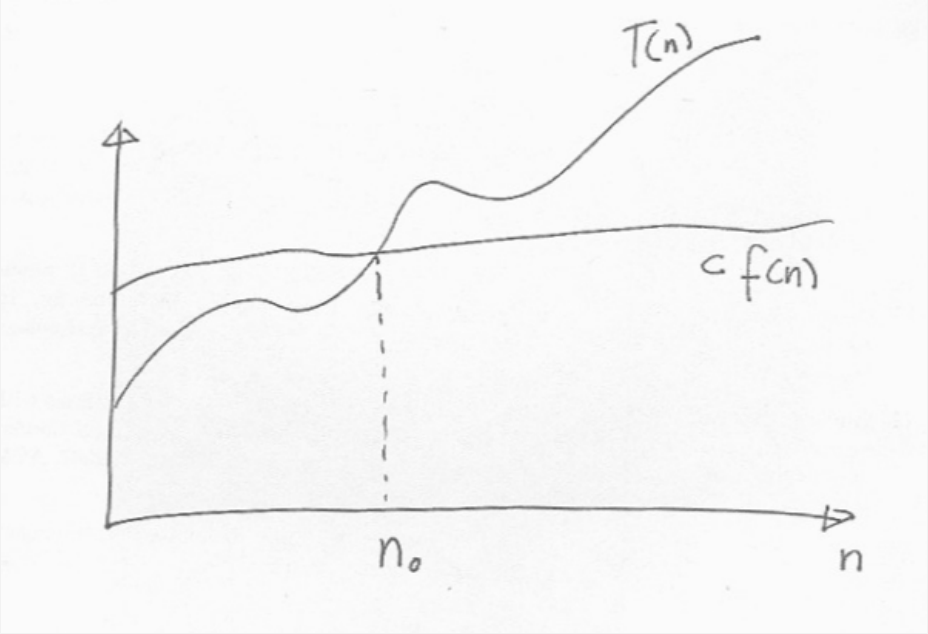
## Notação Ω

- limite assimptotico inferior
- $T(n) = \Omega(f(n))$ se para valores de $n$ suficientemente grandes,

  $T(n) \geq c \cdot f(n)$, em que $c$ é uma constante positiva
- Formalmente $\Omega(f(n))$ também é um conjunto de funções:

$$\Omega(f(n)) = \{T(n) : \exists_{c>0}\ \exists_{n_0>0}\ \forall_{n>n_0}\ 0 \leq c \cdot f(n) \leq T(n)\}$$

## Graficamente






## Exemplos

Exemplos de funções que pertencem a $\Omega(n^2)$

- $n^2$
- $1000n^2 + 100n$
- $n^3$
- $2^n$

## Exemplos

Exemplos de funções que pertencem a $O(n^2)$

- $n^2$
- $1000n^2 + 100n$
- $n \lg n$
- $70n$
- $5$



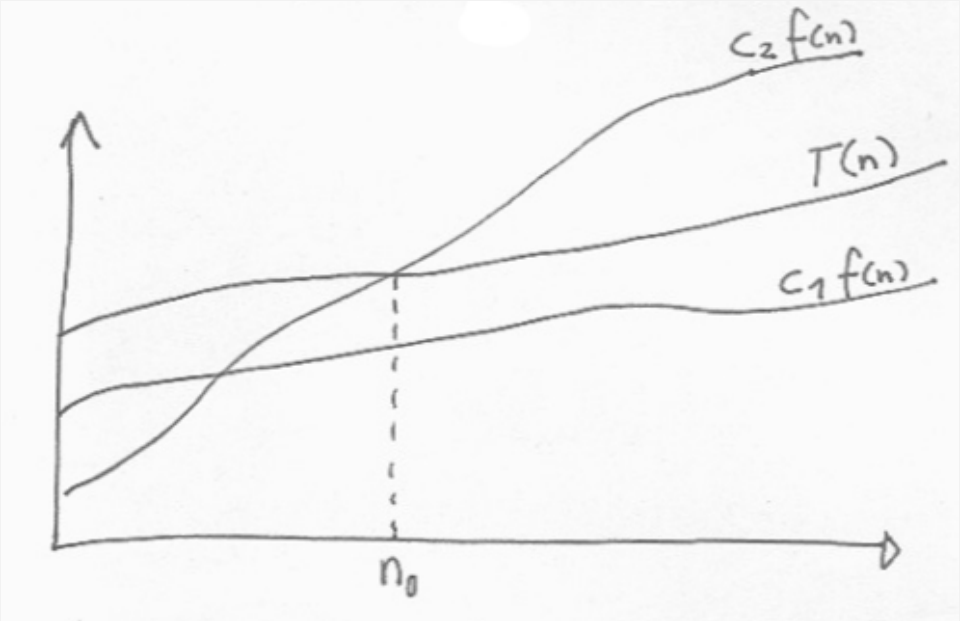
## Notação $\Theta$

- limite assimptotico "apertado"
  (asymptotic tight bound, em inglês)

$$\Theta(f(n)) = \{T(n) : \exists_{c_1>0}\ \exists_{c_2>0}\ \exists_{n_0>0} \forall_{n>n_0}\ : \\ 0 \leq c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)\}$$

- $T(n)$ está "ensanduichado"entre $c_1 \cdot f(n)$ e $c_2 \cdot f(n)$, para $n > n_0$.

## Graficamente






## Teorema

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

sse:

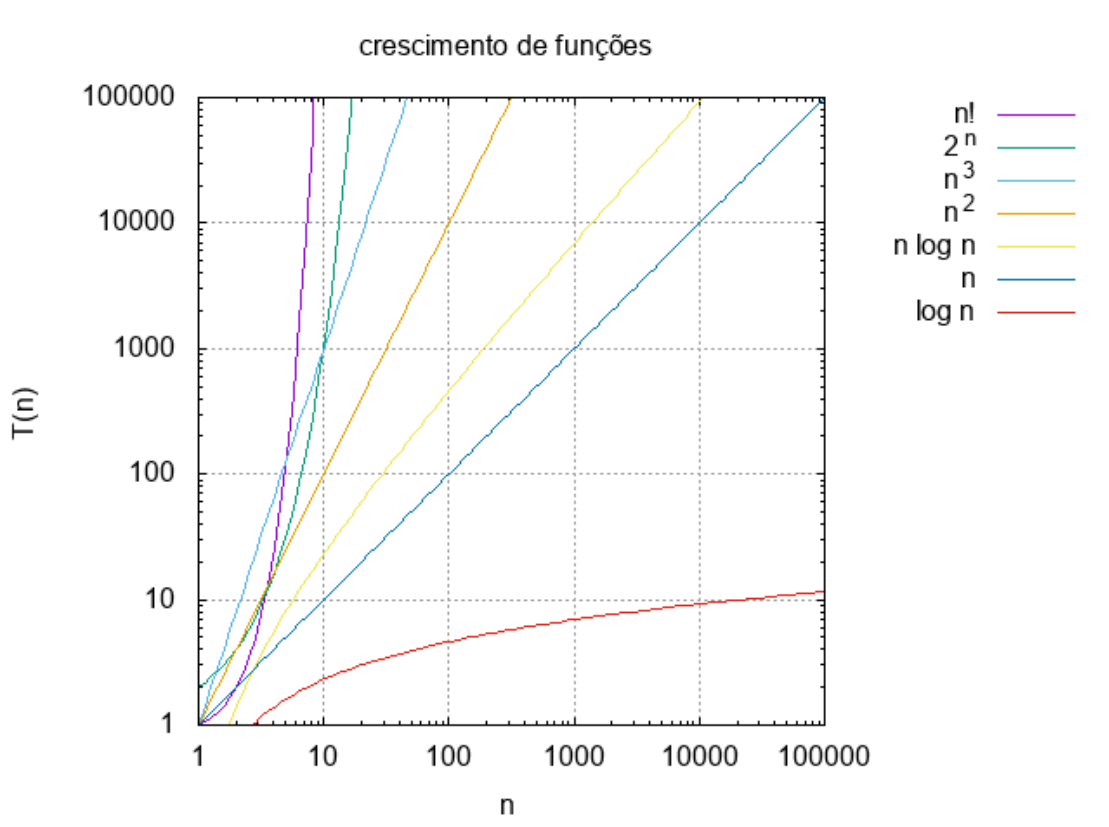
$f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$

# Tempos de execução típicos

| Complexidade | Exemplos |
|---|---|
| $\Theta(1)$ | Hashing |
| $\Theta(\lg n)$ | Pesquisa binária, pesquisa numa árvore balanceada |
| $\Theta(n)$ | Pesquisa linear |
| $\Theta(n \lg n)$ | MergeSort, QuickSort, HeapSort |
| $\Theta(n^2)$ | InsertionSort |
| $\Theta(n^3)$ | Algoritmo tradicional para multiplicação de matrizes |
| $\Theta(a^n)$ , $a > 1$ | Gerar todos os subconjuntos de um conjunto |
| $\Theta(n!)$ | Caixeiro viajante |

- Algoritmos com complexidade polinomial dizem-se tratáveis. Os outros são intratáveis.





# Revisões de matemática



## Revisão sobre exponenciais e logaritmos

- Exponenciais e logaritmos estão sempre a aparecer na análise de algoritmos.
- Exponenciais:
  - $a^m \cdot a^n = a^{m+n}$
  - $(a^m)^n = a^{m \cdot n}$
  - $a^{-1} = 1/a$
- Logaritmos:
  - $a = b^{\log_b a}$
  - $\log_c(a \cdot b) = \log_c a + \log_c b$
  - $\log_b a^n = n \cdot \log_b a$
  - $\log_b a = \frac{\log_c a}{\log_c b}$
  - $\log_b a = \frac{1}{\log_a b}$

## Polinómios crescem de modo mais lento que exponenciais

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0 \qquad \forall_{a,b \in \mathbb{R}\ ,\ a>1}$$

Logo:

$$n^b = O(a^n) \qquad \forall_{a,b \in \mathbb{R}\ ,\ a>1}$$

Exemplos:

$$\begin{aligned} 1000n^3 &= O(2^n) \\ 1000n^{500} &= O(1.000001^n) \end{aligned}$$

- Qualquer função exponencial de base maior que 1 cresce mais rápido que qualquer função polinomial (qualquer que seja o grau do polinómio).



## Logaritmos crescem de modo mais lento que polinómios

$$\lim_{n\to\infty} \frac{\lg^b n}{n^a} = 0 \qquad \forall_{a>0}$$

Logo:

$$\lg^b n = O(n^a) \qquad \forall_{a>0}$$

- Qualquer função polinomial de grau positivo cresce mais rápido que qualquer função polilogaritmica.