

# Complexidade Assintótica

## ACH2002 - Introdução à Ciência da Computação II

Delano M. Beder

Escola de Artes, Ciências e Humanidades (EACH)  
Universidade de São Paulo  
dbeder@usp.br

08/2008

Material baseado em slides dos professores Marcos Chaim, Cid de Souza e Cândida da Silva

# Crescimento Assintótico de Funções

- Custo da solução aumenta com o tamanho  $n$  do problema
  - O tamanho  $n$  fornece uma medida da dificuldade para resolver o problema
    - Tempo necessário para resolver o problema aumenta quando  $n$  cresce
  - Exemplo: número de comparações para achar o maior elemento de um vetor(array) ou para ordená-lo aumenta com o tamanho da entrada  $n$ .
- Escolha do algoritmo não é um problema crítico quando  $n$  é pequeno.
  - O problema é quando  $n$  cresce.
- Por isso, é usual analisar o comportamento das funções de custo quando  $n$  é bastante grande.

# Comportamento Assintótico

- Vamos comparar funções assintoticamente, ou seja, para valores grandes, desprezando constantes multiplicativas e termos de menor ordem.

	$n = 100$	$n = 1000$	$n = 10^4$	$n = 10^6$	$n = 10^9$
$\log n$	2	3	4	6	9
$n$	100	1000	$10^4$	$10^6$	$10^9$
$n \log n$	200	3000	$4 \cdot 10^4$	$6 \cdot 10^6$	$9 \cdot 10^9$
$n^2$	$10^4$	$10^6$	$10^8$	$10^{12}$	$10^{18}$
$100n^2 + 15n$	$1,0015 \cdot 10^6$	$1,00015 \cdot 10^8$	$\approx 10^{10}$	$\approx 10^{14}$	$\approx 10^{20}$
$2^n$	$\approx 1,26 \cdot 10^{30}$	$\approx 1,07 \cdot 10^{301}$	?	?	?

# Comportamento Assintótico

1 milhão ( $10^6$ ) de operações por segundo

Função de custo	10	20	30	40	50	60
$n$	0,00001s	0,00002s	0,00003s	0,00004s	0,00005s	0,00006s
$n^2$	0,0001s	0,0004s	0,0009s	0,0016s	0,0025s	0,0036s
$n^3$	0,001s	0,008s	0,027s	0,064s	0,125s	0,216s
$n^5$	0,1s	3,2s	24,3s	1,7min	5,2min	12,96min
$2^n$	0,001s	1,04s	17,9min	12,7dias	35,7 anos	366 séc.
$3^n$	0,059s	58min	6,5anos	3855séc.	$10^8$ séc.	$10^{13}$ séc.

# Comportamento Assintótico

Influência do aumento de velocidade dos computadores no tamanho  $x$  do problema

Função de custo	Computador Atual (C)	Computador 100C	Computador 1000C
$n$	$x$	$100x$	$1000x$
$n^2$	$x$	$10x$	$31.6x$
$n^3$	$x$	$4,6x$	$10x$
$2^n$	$x$	$x + 6,6$	$x + 10$

(Tabela 1.4 Página 18) Nívio Ziviani. *Projeto de Algoritmos com implementações em C e Pascal*. Editora Thomson, 2a. Edição, 2004.

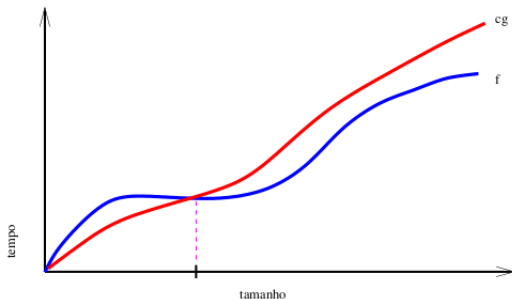
# Comportamento Assintótico

- Se  $f(n)$  é a função de complexidade de um algoritmo  $A$ 
  - O *comportamento assintótico* de  $f(n)$  representa o *limite* do comportamento do custo (complexidade) de  $A$  quando  $n$  cresce.
- A análise de um algoritmo (função de complexidade)
  - Geralmente considera apenas algumas operações elementares ou mesmo uma operação elementar (e.g., o número de comparações).
- A complexidade assintótica relata crescimento assintótico das operações elementares.

# Relacionamento assintótico

## Definição

*Uma função  $g(n)$  domina assintoticamente outra função  $f(n)$  se existem duas constantes positivas  $c$  e  $m$  tais que, para  $n \geq m$ , tem-se  $|f(n)| \leq c|g(n)|$ .*



## Exemplo:

$$g(n) = n \text{ e } f(n) = n^2$$

$$|n| \leq |n^2| \text{ para todo } n \in \mathbb{N}.$$

$$\text{Para } c = 1 \text{ e } m = 0 \Rightarrow |g(n)| \leq |f(n)|.$$

Portanto,  $f(n)$  domina assintoticamente  $g(n)$ .



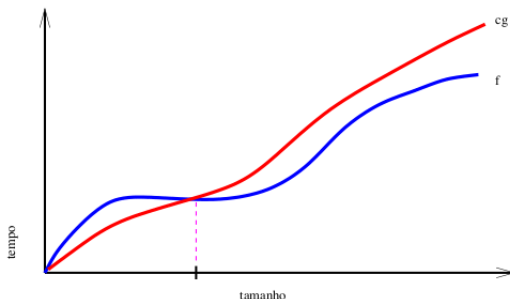
- Knuth criou a notação  $O$  (O grande) para expressar que  $g(n)$  domina assintoticamente  $f(n)$ , escreve-se  $f(n) = O(g(n))$  e lê-se: " $f(n)$  é da ordem no máximo  $g(n)$ ".
- Para que serve isto para o bacharel em Sistemas de Informação?
  - Muitas vezes calcular a função de complexidade  $g(n)$  de um algoritmo  $A$  é complicado.
  - É mais fácil determinar que  $f(n)$  é  $O(g(n))$ , isto é, que assintoticamente  $f(n)$  cresce no máximo como  $g(n)$ .

# Notação $O$

## Definição

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

*Informalmente, dizemos que, se  $f(n) \in O(g(n))$ , então  $f(n)$  cresce no máximo tão rapidamente quanto  $g(n)$ .*



## Definição

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

*Informalmente, dizemos que, se  $f(n) \in O(g(n))$ , então  $f(n)$  cresce no máximo tão rapidamente quanto  $g(n)$ .*

## Exemplo:

$$\frac{3}{2}n^2 - 2n \in O(n^2)$$

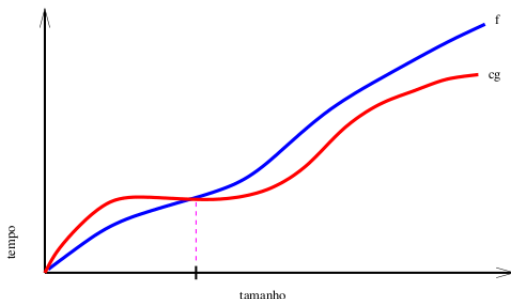
*Valores de  $c$  e  $n_0$  que satisfazem a definição são*

$$c = \frac{3}{2} \text{ e } n_0 = 2$$

## Definição

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

*Informalmente, dizemos que, se  $f(n) \in \Omega(g(n))$ , então  $f(n)$  cresce no mínimo tão lentamente quanto  $g(n)$ .*



## Definição

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

*Informalmente, dizemos que, se  $f(n) \in \Omega(g(n))$ , então  $f(n)$  cresce no mínimo tão lentamente quanto  $g(n)$ .*

## Exemplo:

$$\frac{3}{2}n^2 - 2n \in \Omega(n^2)$$

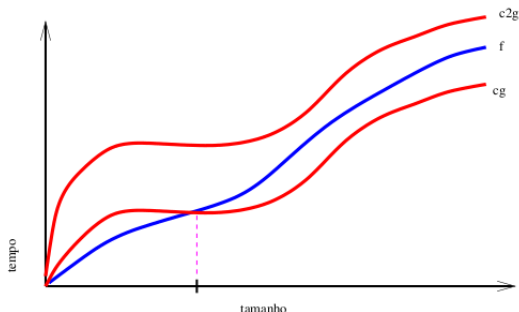
*Valores de  $c$  e  $n_0$  que satisfazem a definição são*

$$c = \frac{1}{2} \text{ e } n_0 = 2$$

## Definição

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

*Informalmente, dizemos que, se  $f(n) \in \Theta(g(n))$ , então  $f(n)$  cresce tão rapidamente quanto  $g(n)$ .*



## Definição

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

*Informalmente, dizemos que, se  $f(n) \in \Theta(g(n))$ , então  $f(n)$  cresce tão rapidamente quanto  $g(n)$ .*

## Exemplo:

$$\frac{3}{2}n^2 - 2n \in \Theta(n^2)$$

*Valores de  $c_1$ ,  $c_2$  e  $n_0$  que satisfazem a definição são*

$$c_1 = \frac{1}{2}, c_2 = \frac{3}{2} \text{ e } n_0 = 2$$

## Definição

$o(g(n)) = \{ f(n) : \text{para toda constante positiva } c, \text{ existe uma constante } n_0 > 0 \text{ tal que } 0 \leq f(n) < cg(n), \text{ para todo } n \geq n_0 \}.$

*Informalmente, dizemos que, se  $f(n) \in o(g(n))$ , então  $f(n)$  cresce mais lentamente que  $g(n)$ .*

## Exemplo:

$$1000n^2 \in o(n^3)$$

*Para todo valor de  $c$ , um  $n_0$  que satisfaz a definição é:*

$$n_0 = \left\lceil \frac{1000}{c} \right\rceil + 1$$



## Definição

$\omega(g(n)) = \{ f(n) : \text{para toda constante positiva } c, \text{ existe uma constante } n_0 > 0 \text{ tal que } 0 \leq cg(n) < f(n), \text{ para todo } n \geq n_0 \}.$

*Informalmente, dizemos que, se  $f(n) \in \omega(g(n))$ , então  $f(n)$  cresce mais rapidamente que  $g(n)$ .*

## Exemplo:

$$\frac{1}{1000}n^2 \in \omega(n)$$

*Para todo valor de  $c$ , um  $n_0$  que satisfaz a definição é:*

$$n_0 = \lceil 1000c \rceil + 1$$

# Definições equivalentes

$$f(n) \in o(g(n)) \quad \text{se} \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) \in O(g(n)) \quad \text{se} \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) \in \Theta(g(n)) \quad \text{se} \quad 0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) \in \Omega(g(n)) \quad \text{se} \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0$$

$$f(n) \in \omega(g(n)) \quad \text{se} \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

## Reflexividade:

$$f(n) \in O(f(n)).$$

$$f(n) \in \Omega(f(n)).$$

$$f(n) \in \Theta(f(n)).$$

## Simetria:

$$f(n) \in \Theta(g(n)) \text{ se, e somente se, } g(n) \in \Theta(f(n)).$$

## Simetria Transposta:

$$f(n) \in O(g(n)) \text{ se, e somente se, } g(n) \in \Omega(f(n)).$$

$$f(n) \in o(g(n)) \text{ se, e somente se, } g(n) \in \omega(f(n)).$$

## Transitividade:

Se  $f(n) \in O(g(n))$  e  $g(n) \in O(h(n))$ , então  $f(n) \in O(h(n))$ .

Se  $f(n) \in \Omega(g(n))$  e  $g(n) \in \Omega(h(n))$ , então  $f(n) \in \Omega(h(n))$ .

Se  $f(n) \in \Theta(g(n))$  e  $g(n) \in \Theta(h(n))$ , então  $f(n) \in \Theta(h(n))$ .

Se  $f(n) \in o(g(n))$  e  $g(n) \in o(h(n))$ , então  $f(n) \in o(h(n))$ .

Se  $f(n) \in \omega(g(n))$  e  $g(n) \in \omega(h(n))$ , então  $f(n) \in \omega(h(n))$ .

# Operações com a notação $O$

$$f(n) = O(f(n))$$

$$c \times f(n) = O(f(n)), c \text{ é uma constante}$$

$$O(f(n)) + O(f(n)) = O(f(n))$$

$$O(O(f(n))) = O(f(n))$$

$$O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$$

$$O(f(n))O(g(n)) = O(f(n)g(n))$$

$$f(n)O(g(n)) = O(f(n)g(n))$$

# Exercícios

Quais as relações de comparação assintótica ( $O$ ,  $\Omega$ ,  $\Theta$ ) das funções:

$$f_1(n) = 2^\pi$$

$$f_2(n) = 2^n$$

$$f_3(n) = n \log n$$

$$f_4(n) = \log n$$

$$f_5(n) = 100n^2 + 150000n$$

$$f_6(n) = n + \log n$$

$$f_7(n) = n^2$$

$$f_8(n) = n$$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$	$f_8$
$f_1$	$\Theta$							
$f_2$		$\Theta$						
$f_3$			$\Theta$					
$f_4$				$\Theta$				
$f_5$					$\Theta$			
$f_6$						$\Theta$		
$f_7$							$\Theta$	
$f_8$								$\Theta$

- [1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. *Algoritmos - Tradução da 2a. Edição Americana*. Editora Campus, 2002 (Capítulo 3).
- [2] Michael T. Goodrich & Roberto Tamassia. *Estruturas de Dados e Algoritmos em Java*. Editora Bookman, 4a. Ed. 2007 (Capítulo 4).
- [3] Nívio Ziviani. *Projeto de Algoritmos com implementações em C e Pascal*. Editora Thomson, 2a. Edição, 2004 (Seção 1.3).