

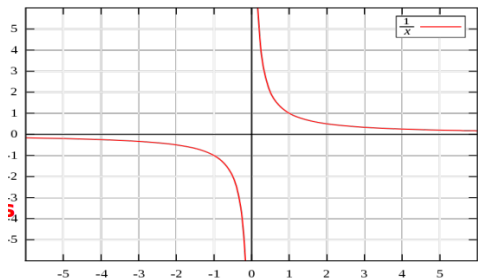
ACH2002

Aula 5

Análise assintótica (parte 2)

Aula passada

- Comportamento assintótico de funções (quando n cresce)

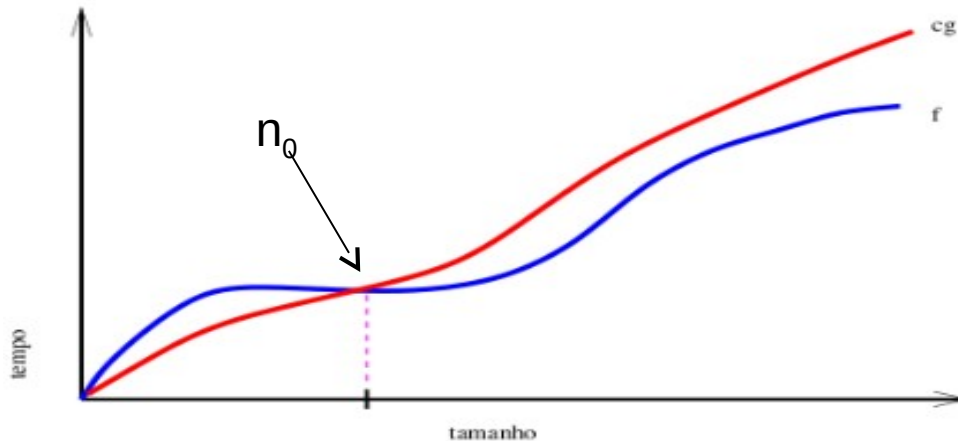


- Notação O: quando uma função domina outra assintoticamente

Notação O

- **Definição:** → Conjunto de funções dominadas por $g(n)$
 - $O(g(n)) = \{f(n): \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq f(n) \leq cg(n), \text{ para todo } n \geq n_0\}$
→ funções assintoticamente não negativas
- Informalmente, dizemos que, se $f(n) \in O(g(n))$, então $f(n)$ cresce no máximo tão rapidamente quanto $g(n)$.

cg é um limite superior de f



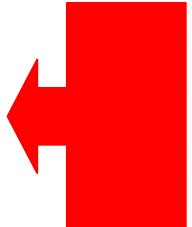
Ex:

- Quem aqui pertence a $O(n^2)$?

	$n = 100$	$n = 1000$	$n = 10^4$	$n = 10^6$	$n = 10^9$
$\log n$	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	$\approx 1,26 \cdot 10^{30}$	$\approx 1,07 \cdot 10^{301}$?	?	?

Ex:

- Quem aqui pertence a $O(n^2)$?



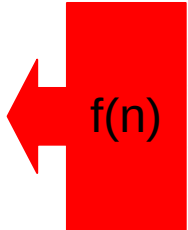
	$n = 100$	$n = 1000$	$n = 10^4$	$n = 10^6$	$n = 10^9$
$\log n$	2	3	4	6	9
n	100	1000	10^4	10^6	10^9
$n \log n$	200	3000	$4 \cdot 10^4$	$6 \cdot 10^6$	$9 \cdot 10^9$
n^2	10^4	10^6	10^8	10^{12}	10^{18}
$100n^2 + 15n$	$1,0015 \cdot 10^6$	$1,00015 \cdot 10^8$	$\approx 10^{10}$	$\approx 10^{14}$	$\approx 10^{20}$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

Ex:

- Quem aqui pertence a $O(n^2)$?

O certo é escrever $f(n) \in O(n^2)$

Mas é comum o “abuso de notação” $f(n) = O(n^2)$ (lemos $f(n)$ é $O(n^2)$)



	$n = 100$	$n = 1000$	$n = 10^4$	$n = 10^6$	$n = 10^9$
$\log n$	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	$\approx 1,26 \cdot 10^{30}$	$\approx 1,07 \cdot 10^{301}$?	?	?

Operações com a notação O

$$\begin{aligned} f(n) &= O(f(n)) \\ c \times f(n) &= O(f(n)), c \text{ é uma constante} \\ O(f(n)) + O(f(n)) &= O(f(n)) \\ O(O(f(n))) &= O(f(n)) \\ \rightarrow O(f(n)) + O(g(n)) &= O(\max(f(n), g(n))) \\ O(f(n))O(g(n)) &= O(f(n)g(n)) \\ f(n)O(g(n)) &= O(f(n)g(n)) \end{aligned}$$

Particularmente útil para analisar algoritmos (sequências de trechos de código)

Ex: InsertionSort é $O(n^2)$

- Tempo de execução do algoritmo = soma dos tempos de execução para cada instrução
- $T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n-1)$

- Pior caso:** vetor em ordem inversa (deve comparar cada elemento $A[j]$ com cada elemento do subarranjo ordenado $A[j \dots j-1]$ → $t_j = j$ para $j=2, 3, \dots, n$)

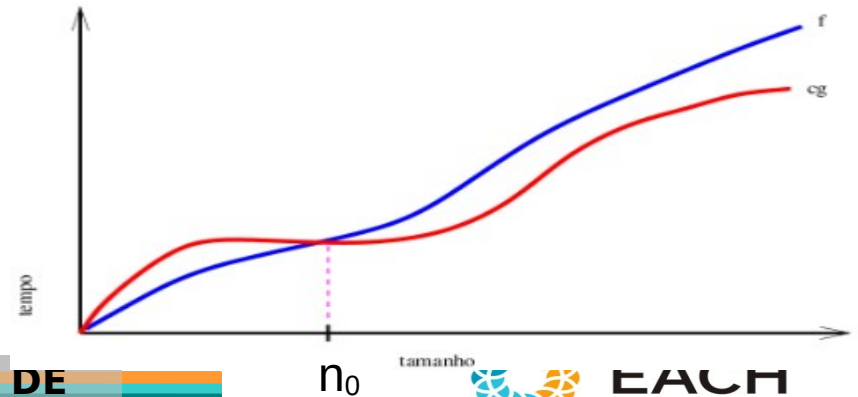
$$\sum_{j=2}^n (j) = \frac{n(n-1)}{2} + 1 \quad \sum_{j=2}^n (j-1) = \frac{n(n-1)}{2}$$
$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \frac{n(n-1)}{2} + 1 + c_6 \frac{n(n-1)}{2} + c_7 \frac{n(n-1)}{2} + c_8(n-1) =$$
$$\left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n - (c_2 + c_4 + c_5 + c_8)$$

Também é $O(n^k)$
para $k > 2 \dots$

Tempo de execução, neste caso, pode ser expresso como $an^2 + bn + c$ para constantes **a**, **b** e **c** que dependem dos custos de instrução c_i → **função quadrática** de n

Definição: Notação Ω

- $\Omega(g(n)) = \{f(n): \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq cg(n) \leq f(n), \text{ para todo } n \geq n_0\}$
→ funções assintoticamente não negativas
- Informalmente, dizemos que, se $f(n) \in \Omega(g(n))$, então $f(n)$ cresce no mínimo tão lentamente quanto $g(n)$.
- Note que se $f(n) \in O(g(n))$ define um limite superior para $f(n)$, $\Omega(g(n))$ define um limite inferior

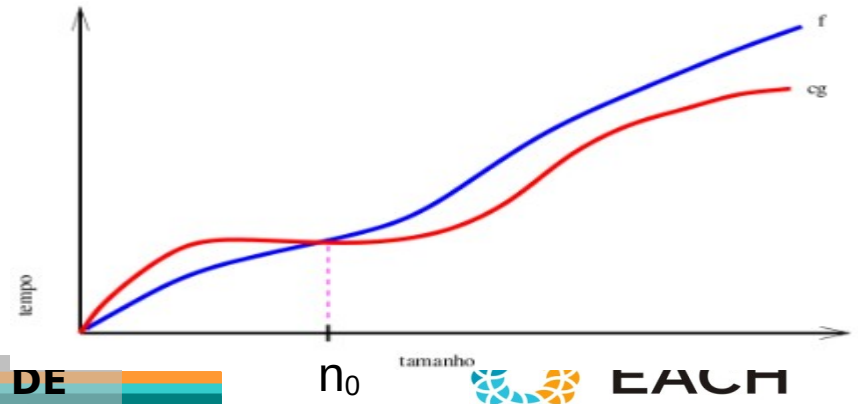


Notação Ω

Ex: Qualquer algoritmo de ordenação é $\Omega(n)$.

Porque no mínimo tem que conferir cada posição do array...

Mas é sempre interessante dar uma avaliação mais precisa



Ex: InsertionSort é $\Omega(n^2)$

➤ Tempo de execução do algoritmo = soma dos tempos de execução para cada instrução

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n-1)$$

(por conta do pior caso)

➤ **Pior caso:** vetor em ordem inversa (deve comparar cada elemento $A[j]$ com cada elemento do subarranjo ordenado $A[j \dots j-1]$ → $t_j = j$ para $j=2, 3, \dots, n$)

$$\sum_{j=2}^n (j) = \frac{n(n-1)}{2} + 1 \quad \sum_{j=2}^n (j-1) = \frac{n(n-1)}{2}$$
$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \frac{n(n-1)}{2} + 1 + c_6 \frac{n(n-1)}{2} + c_7 \frac{n(n-1)}{2} + c_8(n-1) =$$
$$\left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n - (c_2 + c_4 + c_5 + c_8)$$

Também é $\Omega(n)$

...

Mas um limitante inferior mais preciso (próximo de $f(n)$) é mais interessante

➤ Tempo de execução, neste caso, pode ser expresso como $an^2 + bn + c$ para constantes **a**, **b** e **c** que dependem dos custos de instrução c_i → **função quadrática** de n

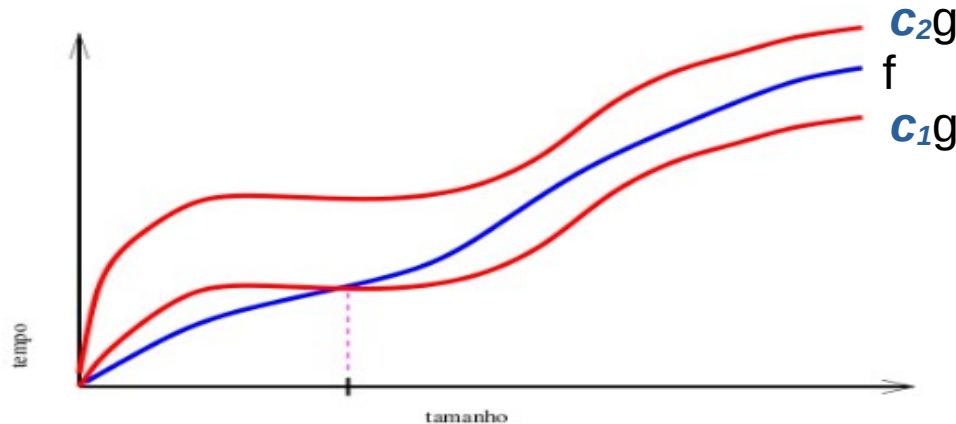
Notação Θ

Definição:

f é “ensanduichada” por g

- $\Theta(g(n)) = \{f(n): \text{existem constantes positivas } c_1, c_2 \text{ e } n_0 \text{ tais que } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n), \text{ para todo } n \geq n_0\}$

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



Aula de hoje

- “Revisão” da notação Θ
- Notações o e ω
- ...

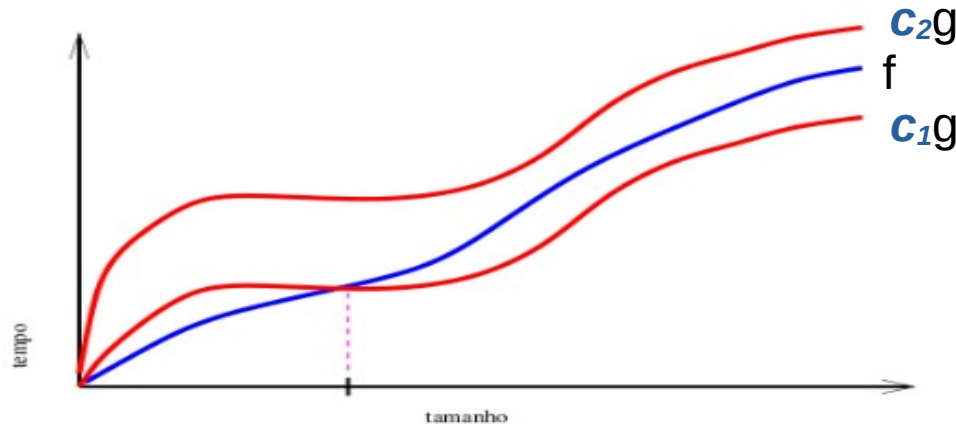
Notação Θ

Definição:

f é “ensanduichada” por g

- $\Theta(g(n)) = \{f(n): \text{existem constantes positivas } c_1, c_2 \text{ e } n_0 \text{ tais que } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n), \text{ para todo } n \geq n_0\}$

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



Notação Θ

序 Definição:

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

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

鋤 ???

Notação Θ

Definição:

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

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

Fazendo $c_1 = 1/2$ e $c_2 = 3/2$ teremos $\left| \frac{1}{2}n^2 \right| \leq \left| \frac{3}{2}n^2 - 2n \right| \leq \left| \frac{3}{2}n^2 \right|$

para $n \geq 2$

Note que o n_0 tem que valer para as duas desigualdades!!!

Notação Θ

💡 Mas, já vimos que:

$$\text{💡 } \frac{3}{2}n^2 - 2n \in O(n^2) \rightarrow \left| \frac{3}{2}n^2 - 2n \right| \leq \left| \frac{3}{2}n^2 \right|$$

$$\text{💡 } \frac{3}{2}n^2 - 2n \in \Omega(n^2) \rightarrow \left| \frac{1}{2}n^2 \right| \leq \left| \frac{3}{2}n^2 - 2n \right| \quad \text{e ...}$$

$$\text{💡 } \frac{3}{2}n^2 - 2n \in \Theta(n^2) \rightarrow \left| \frac{1}{2}n^2 \right| \leq \left| \frac{3}{2}n^2 - 2n \right| \leq \left| \frac{3}{2}n^2 \right|$$

💡 Será coincidência?

💡 ???



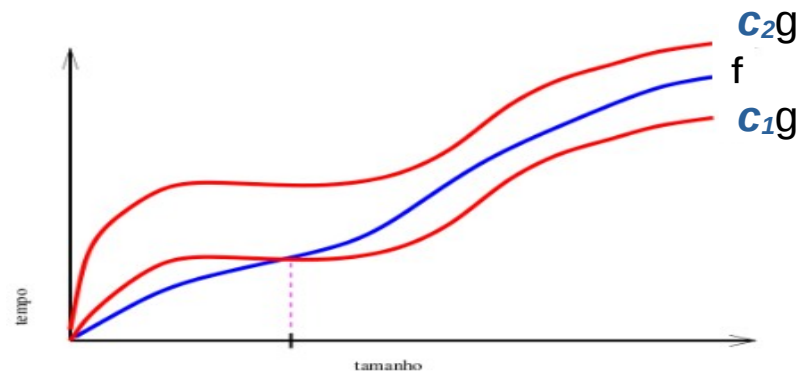
Notação Θ

Mas, já vimos que:

$$\frac{3}{2}n^2 - 2n \in O(n^2) \rightarrow \left| \frac{3}{2}n^2 - 2n \right| \leq \left| \frac{3}{2}n^2 \right|$$

$$\frac{3}{2}n^2 - 2n \in \Omega(n^2) \rightarrow \left| \frac{1}{2}n^2 \right| \leq \left| \frac{3}{2}n^2 - 2n \right|$$

$$\frac{3}{2}n^2 - 2n \in \Theta(n^2) \rightarrow \left| \frac{1}{2}n^2 \right| \leq \left| \frac{3}{2}n^2 - 2n \right| \leq \left| \frac{3}{2}n^2 \right|$$



Será coincidência?

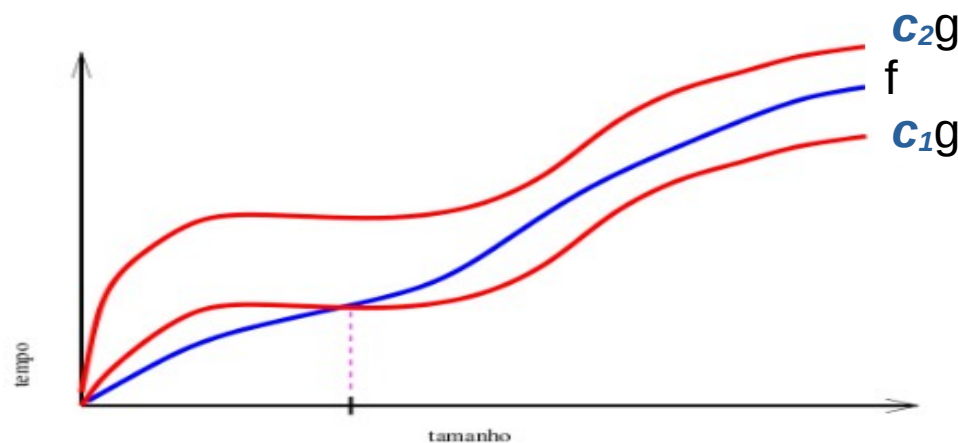
Não!

Se $f(n) \in O(g(n))$ e $f(n) \in \Omega(g(n))$, então $f(n) \in \Theta(g(n))$



Notação Θ

- Mas:
 - Será coincidência?
 - Não!
 - Se $f(n) \in O(g(n))$ e $f(n) \in \Omega(g(n))$, então $f(n) \in \Theta(g(n))$



Ex: InsertionSort é $\Theta(n^2)$

➤ Tempo de execução do algoritmo = soma dos tempos de execução para cada instrução

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n-1)$$

(por conta do pior caso)

➤ **Pior caso:** vetor em ordem inversa (deve comparar cada elemento $A[j]$ com cada elemento do subarranjo ordenado $A[j \dots j-1]$ → $t_j = j$ para $j=2, 3, \dots, n$)

$$\sum_{j=2}^n (j) = \frac{n(n-1)}{2} + 1 \quad \sum_{j=2}^n (j-1) = \frac{n(n-1)}{2}$$
$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \frac{n(n-1)}{2} + 1 + c_6 \frac{n(n-1)}{2} + c_7 \frac{n(n-1)}{2} + c_8(n-1) =$$
$$\left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n - (c_2 + c_4 + c_5 + c_8)$$

➤ Tempo de execução, neste caso, pode ser expresso como $an^2 + bn + c$ para constantes **a**, **b** e **c** que dependem dos custos de instrução c_i → **função quadrática** de n

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) 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	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) 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	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ



Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$



	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Prove que $2^\pi = \Theta(2^\pi)$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$



	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Prove que $2^\pi = \Theta(2^\pi)$

Existem c_1 , c_2 e n_0 constantes positivas tal que

$$0 \leq c_1 2^\pi \leq 2^\pi \leq c_2 2^\pi \text{ para } n \geq n_0$$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$



	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Prove que $2^\pi = \Theta(2^\pi)$

Existem c_1 , c_2 e n_0 constantes positivas tal que

$$0 \leq c_1 2^\pi \leq 2^\pi \leq c_2 2^\pi \text{ para } n \geq n_0$$

$$\text{Ex: } c_1 = c_2 = n_0 = 1$$

$$c_1 = 0.5, c_2 = 2, n_0 = 0.1 \text{ (não podem ser 0...)}$$



Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

$n \log n = ? (n^2) - \text{Prove}$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

$n \log n = ? (n^2)$ – Prove

$n \log n = O(n^2)$, pois existem constantes positivas c e n_0 tais que:

$0 \leq n \log n \leq cn^2$, para todo $n \geq n_0$

Isso vale para $c = ?$, $n_0 = ?$

$0 \leq \log n \leq cn$, para todo $n \geq n_0$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

$n \log n = ? (n^2)$ – Prove

$n \log n = O(n^2)$, pois existem constantes positivas c e n_0 tais que:

$0 \leq n \log n \leq cn^2$, para todo $n \geq n_0$

Isso vale para $c = 1$, $n_0 = 1$

$0 \leq \log n \leq cn$, para todo $n \geq n_0$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Será que $n \log n$ é $\Omega(n^2)$?

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Será que $n \log n$ é $\Omega(n^2)$?

Se fosse, então existiriam constantes positivas c e n_0 tais que:

$$0 \leq cn^2 \leq n \log n, \text{ para todo } n \geq n_0$$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Será que $n \log n$ é $\Omega(n^2)$?

Se fosse, então existiriam constantes positivas c e n_0 tais que:

$0 \leq cn^2 \leq n \log n$, para todo $n \geq n_0$

Se fosse verdade, quem poderia ser esse c e n_0 ?

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Será que $n \log n$ é $\Omega(n^2)$?

Se fosse, então existiriam constantes positivas c e n_0 tais que:

$$0 \leq cn^2 \leq n \log n, \text{ para todo } n \geq n_0$$

Se fosse verdade, $0 \leq cn \leq \log n \Rightarrow c \leq \log n / n$ para todo $n \geq n_0$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Será que $n \log n$ é $\Omega(n^2)$?

Se fosse, então existiriam constantes positivas c e n_0 tais que:

$$0 \leq cn^2 \leq n \log n, \text{ para todo } n \geq n_0$$

Se fosse verdade, $0 \leq cn \leq \log n \Rightarrow c \leq \log n / n$ para todo $n \geq n_0$

Mas $\lim_{n \rightarrow \infty} \log n / n = 0$, MAS c deve ser positiva! CONTRADIÇÃO



Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Será que $n \log n$ é $\Omega(n^2)$?

Então $n \log n$ também NÃO é $\Theta(n^2)$, então aqui é só o mesmo....

Se fosse, então existiriam constantes positivas c e n_0 tais que:

$$0 \leq cn^2 \leq n \log n, \text{ para todo } n \geq n_0$$

Se fosse verdade, $0 \leq cn \leq \log n \Rightarrow c \leq \log n / n$ para todo $n \geq n_0$

Mas $\lim_{n \rightarrow \infty} \log n / n = 0$, MAS c deve ser positiva! CONTRADIÇÃO



Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

$n^2 = ? (n \log n)$ – Prove

$n^2 = \Omega(n \log n)$, pois

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

$n^2 = ? (n \log n)$ – Prove

$n^2 = \Omega(n \log n)$, pois existem constantes positivas c e n_0 tais que:

$0 \leq c n \log n \leq n^2$, para todo $n \geq n_0$

Isso vale para $c = ?$, $n_0 = ?$

$0 \leq c \log n \leq n$, para todo $n \geq n_0$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

$n^2 = ? (n \log n)$ – Prove

$n^2 = \Omega(n \log n)$, pois existem constantes positivas c e n_0 tais que:

$0 \leq c n \log n \leq n^2$, para todo $n \geq n_0$

Isso vale para $c = 1$, $n_0 = 1$

$0 \leq c \log n \leq n$, para todo $n \geq n_0$

Exercício

Quais as relações de comparação assintótica (O , Ω , Θ) das funções:

$$f_1(n) = 2^\pi = \Theta(1) \rightarrow \text{constante!!!}$$

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

FAÇAM AS PROVAS FORMAIS PARA OS DEMAIS!!!!
(Inclusive para o que NÃO É)

Material extra sobre O , Ω , Θ

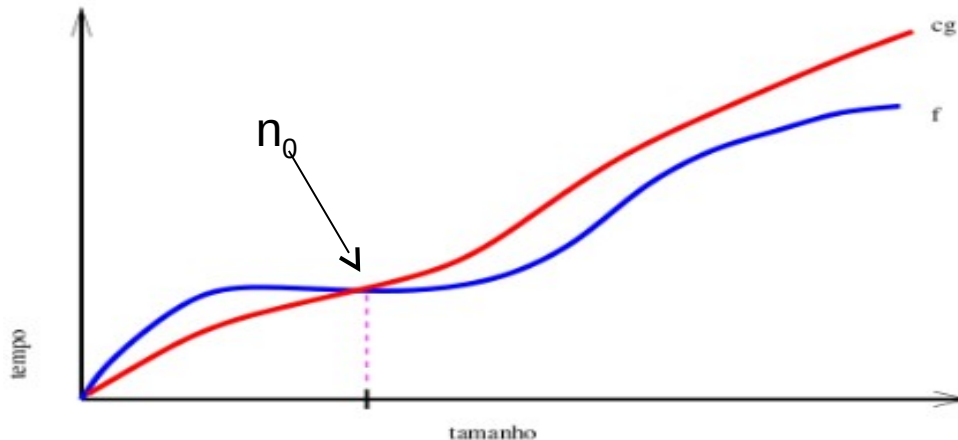
https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/Oh.html

Além de explicações, há 3 listas de exercícios. FAÇAM! (Cai na prova...)

Relembrando notação O

- **Definição:** → Conjunto de funções dominadas por $g(n)$
 - $O(g(n)) = \{f(n): \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq f(n) \leq cg(n), \text{ para todo } n \geq n_0\}$
→ funções assintoticamente não negativas
- Informalmente, dizemos que, se $f(n) \in O(g(n))$, então $f(n)$ cresce no máximo tão rapidamente quanto $g(n)$.

cg é um limite superior de f



Notação o

- 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 $g(n)$, ou $f(n)$ é **assintoticamente menor** que $g(n)$.
 - Intuitivamente, na notação o , a função $f(n)$ tem crescimento muito menor que $g(n)$ quando n tende para o infinito

Notação o

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

- ???

Quem será o n_0 ?

É de se esperar que ele dependa do c ...

O que aconteceria se $n_0 = 1/c$?

$$0 \leq f(n) < cn$$

$$0 \leq |1000n^2| < |cn^3|$$

Notação o

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

- ???

Quem será o n_0 ?

É de se esperar que ele dependa do c ...

O que aconteceria se $n_0 = 1/c$?

O que aconteceria se $n_0 = 1000/c$?

$$0 \leq f(n) < cg(n)$$

$$0 \leq |1000n^2| < |cn^3|$$

Notação o

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

Quem será o n_0 ?

É de se esperar que ele dependa do c ...

- Para todo valor de c (positivo), um n_0 que satisfaz a definição é:

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

Valor que igualaria
a inequação

$$0 \leq f(n) < cg(n)$$

$$0 \leq |1000n^2| < |cn^3|$$

Notação o

- Qual a diferença entre O e o ?
 - O : existem constantes positivas c e n_0 tais que $0 \leq f(n) \leq cg(n)$, para todo $n \geq n_0$
 - A expressão $0 \leq f(n) \leq cg(n)$ é válida para alguma constante $c > 0$
 - o : para toda constante positiva c , existe uma constante $n_0 > 0$ tal que $0 \leq f(n) < cg(n)$, para todo $n \geq n_0$
 - A expressão $0 \leq f(n) < cg(n)$ é válida para toda constante $c > 0$

Notação ω

- 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 $g(n)$, ou $f(n)$ é **assintoticamente maior** que $g(n)$.
 - Intuitivamente, na notação ω , a função $f(n)$ tem crescimento muito maior que $g(n)$ quando n tende para o infinito

Notação ω

- ω está para Ω , da mesma forma que \mathfrak{o} está para \mathcal{O}

$$0 \leq cg(\hat{n}) < f(n).$$

- $\left| \frac{1}{1000} n^2 \right| \in \omega(n) \text{ ?}$

■ ???

Notação ω

- ω está para Ω , da mesma forma que \mathfrak{o} está para O

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

$$0 \leq cg(\hat{n}) < f(n).$$

$$0 \leq cn < |1/1000 n^2|$$

- ???

Notação ω

- ω está para Ω , da mesma forma que \mathfrak{o} está para \mathcal{O}

- $|\frac{1}{1000}n^2| \in \omega(n) \text{ ?}$

$$0 \leq cg(\hat{n}) < f(n).$$

$$0 \leq cn < |1/1000 n^2|$$

$$0 \leq c < |1/1000 n|$$

- ???

Notação ω

- ω está para Ω , da mesma forma que o está para O

- $\left| \frac{1}{1000} n^2 \right| \in \omega(n) ?$

Procuro o n que iguala...

$$0 \leq cg(\hat{n}) < f(n).$$

$$0 \leq cn < |1/1000 n^2|$$

$$0 \leq \underline{c} < \underline{|1/1000 n|}$$

- ???

Notação ω

- ω está para Ω , da mesma forma que \mathfrak{o} está para \mathcal{O}

- $\left| \frac{1}{1000} n^2 \right| \in \omega(n) \text{ ?}$
Procuro o n que iguala...
E somo 1

$$0 \leq cg(\hat{n}) < f(n).$$

$$0 \leq cn < |1/1000 n^2|$$

$$0 \leq \underline{c} < |1/1000 n|$$

- Para todo valor de c , um n_0 que satisfaz a definição é:
$$n_0 = |1000 c| + 1$$

Definições equivalentes

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

Definições equivalentes

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

Pois o denominador é sempre maior que o numerador,
e aumenta cada vez mais em relação ao numerador

Definições equivalentes

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

Definições equivalentes

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

Para ser infinito, o numerador deveria ser crescer mais rápido (ser maior) que o denominador

O que não acontece porque g (denominador) é um limite superior de f ($f \leq g$)

Como $g \geq f$ (nas condições de c e n_0), a razão f/g pode ser:

- 0 (quando $g \gg f$ (g é muito maior que f))
- 1 (se $g = f$)
- algum valor positivo menor que o infinito (g parecida com f , ligeiramente menor)

Definições equivalentes

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

(complete...)

Definições equivalentes

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

(completeem...)

Definições equivalentes

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

(complete...)

Definições equivalentes

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

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

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

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

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

Propriedades das Classes

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:

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

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

Propriedades das Classes

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 $f(n) \in \omega(h(n))$.

Exercício

Inclua o (o-pequeno) e ω nas análises

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

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

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

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

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

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

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

$$f_8(n) = n \quad n \log n = ? (n^2) - \text{Prove}$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				$O, ?$	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

Exercício

Inclua o (o-pequeno) e ω nas análises

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

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				$O, ?$	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

$n \log n = ? (n^2)$ – Prove

$n \log n = o(n^2)$, pois para **toda** constante positiva c , existe $n_0 > 0$ tais que:

$0 \leq n \log n < cn^2 = cn n$ para todo $n \geq n_0$

Independente de c ,

Exercício

Inclua o (o-pequeno) e ω nas análises

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

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O, o	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

$n \log n = ? (n^2)$ – Prove

$n \log n = o(n^2)$, pois para **toda** constante positiva c , existe $n_0 > 0$ tais que:

$0 \leq n \log n < cn^2 = cn n$ para todo $n \geq n_0$

Independente de c , $\log n < n$ para todo $n_0 > 1$

Exercício

Inclua o (o-pequeno) e ω nas análises

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

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O, o	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

Obs: Prove que f_5 NÃO é $o(f_7)$

Exercício

Inclua o (o-pequeno) e ω nas análises

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

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O, o	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

Obs: Prove que f_5 NÃO é $o(f_7)$

Se fosse, para toda constante $c > 0$, existiria $n_0 > 0$ tal que:

$$0 \leq 100n^2 + 150000n < cn^2 \text{ para todo } n \geq n_0$$

Se fosse verdade, $100 + 150000/n < c$ para todo $n \geq n_0$, (mas por maior que seja n_0 , toda constante c positiva deveria ser pelo menos 100???)

ABSURDO... Logo, f_5 NÃO é $o(f_7)$



Exercício

Inclua o (o-pequeno) e nas análises

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

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

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

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

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

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

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

$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ				O	
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7			Ω				Θ	
f_8								Θ

FAÇAM AS PROVAS FORMAIS PARA OS DEMAIS!!!!
(Inclusive para o que NÃO É)

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$

–

- $\Omega(?)$

- $o(?):$

- $w(?)$

- $o(?):$

- Tempo de execução do algoritmo = soma dos tempos de execução para cada instrução
- $T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n-1)$
- **Pior caso:** vetor em ordem inversa (deve comparar cada elemento $A[j]$ com cada elemento do subarranjo ordenado $A[j \dots j-1]$ → $t_j = j$ para $j=2, 3, \dots, n$)

$$\sum_{j=2}^n (j) = \frac{n(n-1)}{2} + 1 \quad \sum_{j=2}^n (j-1) = \frac{n(n-1)}{2}$$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \frac{n(n-1)}{2} + 1 + c_6 \frac{n(n-1)}{2} + c_7 \frac{n(n-1)}{2} + c_8(n-1) =$$

$$\left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n - (c_2 + c_4 + c_5 + c_8)$$

➤ Tempo de execução, neste caso, pode ser expresso como $an^2 + bn + c$ para constantes **a**, **b** e **c** que dependem dos custos de instrução c_i → **função quadrática** de n

Para que serve tudo isso?

Vamos simplificar nossas análises !!!!

- $O(?)$

```
1 para j = 2 até tamanho[A] faça
2   chave = A[j]
3   // ordenando elementos à esquerda
4   i = j - 1
5   enquanto i > 0 e A[i] > chave faça
6     A[i+1] = A[i]
7     i = i - 1
8   fim enquanto
9   A[i+1] = chave
10 fim para
```

custo vezes

C_1	n	$\rightarrow O(n)$
C_2	$n-1$	$\rightarrow O(n)$
0	$n-1$	$\rightarrow O(n)$
C_4	$n-1$	$\rightarrow O(n)$
C_5	$\sum_{j=2}^n t_j$	$\rightarrow O(n^2)$
C_6	$\sum_{j=2}^n (t_j - 1)$	$\rightarrow O(n^2)$
C_7	$\sum_{j=2}^n (t_j - 1)$	$\rightarrow O(n^2)$
C_8	$n-1$	$\rightarrow O(n)$

=> max de tudo isso
= $O(n^2)$



Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...

- Para que serve?
- E se eu falar que existe um outro algoritmo $O(n \log n)$?

Aí é importante saber se o InsertionSort é especificamente $O(n \log n)$ também, ou não... e é?

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...

- Para que serve?
- E se eu falar que existe um outro algoritmo $O(n \log n)$?

- $\Omega(?)$:

Aí é importante saber se o InsertionSort é especificamente $O(n \log n)$ também, ou não... e é?

Mas sabemos que não é, vejamos o próximo

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...

- Para que serve?
- E se eu falar que existe um outro algoritmo $O(n \log n)$?

- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...

Aí é importante saber se o InsertionSort é especificamente $O(n \log n)$ também, ou não...

Ou seja, InsertionSort já “perdeu” desse novo algoritmo...

De fato... n^2 é o melhor que o InsertionSort consegue fazer (lembrando que nos baseamos na análise de pior caso...)

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$:

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$

Em que situação essa informação ajuda?

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
 - Em que situação essa informação ajuda?
 - Se um outro algoritmo B for $\Omega(n^3)$?
 - Ou outro algoritmo C for $O(n \log n)$?

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
 - Em que situação essa informação ajuda?
 - Se um outro algoritmo B for $\Omega(n^3)$?
 - Ou outro algoritmo C for $O(n \log n)$?
 - InsertionSort será mais eficiente que B, e menos que C

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
- $\omega(?)$:

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
- $\omega(?)$: $\omega(n \log n)$, $\omega(n)$, $\omega(1)$, ...

O que significa?

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
- $\omega(?)$: $\omega(n \log n)$, $\omega(n)$, $\omega(1)$, ...

De fato... InsertionSort não consegue nem se **equiparar** a um algoritmo que roda em $O(n \log n)$

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
- $\omega(?)$: $\omega(n \log n)$, $\omega(n)$, $\omega(1)$, ...
- $o(?)$:

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
- $\omega(?)$: $\omega(n \log n)$, $\omega(n)$, $\omega(1)$, ...
- $o(?)$: $o(n^2 \log n)$, $o(n^3)$, $O(n^k)$ $k \geq 3$, $O(2^n)$, ...

O que significa?

Para que serve tudo isso?

Ex: O que podemos falar do InsertionSort?

- $O(?)$: $O(n^2)$, $O(n^k)$ $k \geq 2$, $O(2^n)$, ...
 - Para que serve?
 - E se eu falar que existe um outro algoritmo $O(n \log n)$?
- $\Omega(?)$: $\Omega(n^2)$, $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$, ...
- $\Theta(?)$: $\Theta(n^2)$
- $\omega(?)$: $\omega(n \log n)$, $\omega(n)$, $\omega(1)$, ...
- $o(?)$: $o(n^2 \log n)$, $o(n^3)$, $O(n^k)$ $k \geq 3$, $O(2^n)$, ...

Ele será mais rápido que quaisquer outros algoritmos que rodem em Ω de um desses tempos...

Observações

- $O(1)$: complexidade **constante**
- $O(\log n)$: complexidade **logarítmica**
- $O(n)$: complexidade **linear**
- $O(n^2)$: complexidade **quadrática**
- $O(n^3)$: complexidade **cúbica**
- $O(2^n)$: complexidade **exponencial**

Mais exercícios?

- Tem também no cap. 3 do livro do Cormen

Plantão de Dúvidas

- Todas as quintas das 13:30 às 14:15
 - Sala 210N do A1

Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. Algoritmos - Tradução da 2a. Edição Americana. Editora Campus, 2002 (Capítulos 2 e 3).
- Michael T. Goodrich & Roberto Tamassia. Estruturas de Dados e Algoritmos em Java. Editora Bookman, 4a. Ed. 2007 (Capítulo 4).
- Nívio Ziviani. Projeto de Algoritmos com implementações em C e Pascal. Editora Thomson, 2a. Edição, 2004 (Seção 1.3).
- Notas de aula dos professores Marcos Chaim, Cid de Souza, Cândida da Silva e Delano M. Beder.
- Notas de aula dos professores Fátima L. S. Nunes e Norton T. Roman