

# Análise de Algoritmos

Parte destes slides são adaptações de slides dos Profs.  
Paulo Feofiloff, José Coelho de Pina e Cristina G. Fernandes

7 de agosto de 2025

# Introdução

CLRS 1.1, 1.2, 2.1 e 2.2

AU 3.3, 3.4 e 3.6

# Ordenação

$A[1 \dots n]$  é **crescente** se  $A[1] \leq \dots \leq A[n]$ .

**Problema:** Rearranjar um dado vetor  $A[1 \dots n]$  de modo que ele fique crescente.

Entra:

1										$n$
33	55	33	44	33	22	11	99	22	55	77

# Ordenação

$A[1 \dots n]$  é **crescente** se  $A[1] \leq \dots \leq A[n]$ .

**Problema:** Rearranjar um dado vetor  $A[1 \dots n]$  de modo que ele fique crescente.

Entra:

1										$n$
33	55	33	44	33	22	11	99	22	55	77

Sai:

1										$n$
11	22	22	33	33	33	44	55	55	77	99

# Ordenação por inserção

*chave* = 38

1						<i>j</i>				<i>n</i>
20	25	35	40	44	55	38	99	10	65	50

# Ordenação por inserção

*chave* = 38

1						<i>i</i>	<i>j</i>				<i>n</i>
20	25	35	40	44	55	38	99	10	65	50	

# Ordenação por inserção

*chave* = 38

1					<i>i</i>	<i>j</i>				<i>n</i>
20	25	35	40	44	55	38	99	10	65	50

1				<i>i</i>		<i>j</i>				<i>n</i>
20	25	35	40	44		55	99	10	65	50

# Ordenação por inserção

*chave* = 38

1					<i>i</i>	<i>j</i>				<i>n</i>
20	25	35	40	44	55	38	99	10	65	50

1				<i>i</i>		<i>j</i>				<i>n</i>
20	25	35	40	44		55	99	10	65	50

1			<i>i</i>			<i>j</i>				<i>n</i>
20	25	35	40		44	55	99	10	65	50



# Ordenação por inserção

*chave* = 38

1					<i>i</i>	<i>j</i>				<i>n</i>
20	25	35	40	44	55	38	99	10	65	50

1				<i>i</i>		<i>j</i>				<i>n</i>
20	25	35	40	44		55	99	10	65	50

1			<i>i</i>			<i>j</i>				<i>n</i>
20	25	35	40		44	55	99	10	65	50

1		<i>i</i>				<i>j</i>				<i>n</i>
20	25	35		40	44	55	99	10	65	50

# Ordenação por inserção

*chave* = 38

1					<i>i</i>	<i>j</i>				<i>n</i>
20	25	35	40	44	55	38	99	10	65	50

1				<i>i</i>		<i>j</i>				<i>n</i>
20	25	35	40	44		55	99	10	65	50

1			<i>i</i>			<i>j</i>				<i>n</i>
20	25	35	40		44	55	99	10	65	50

1		<i>i</i>				<i>j</i>				<i>n</i>
20	25	35		40	44	55	99	10	65	50

1		<i>i</i>				<i>j</i>				<i>n</i>
20	25	35	38	40	44	55	99	10	65	50

# Ordenação por inserção

<i>chave</i>	1							<i>j</i>			<i>n</i>
99	20	25	35	38	40	44	55	99	10	65	50

# Ordenação por inserção

<i>chave</i>	1						<i>j</i>			<i>n</i>	
99	20	25	35	38	40	44	55	99	10	65	50

<i>chave</i>	1								<i>j</i>		<i>n</i>
10	20	25	35	38	40	44	55	99	10	65	50

# Ordenação por inserção

<i>chave</i>	1							<i>j</i>			<i>n</i>
99	20	25	35	38	40	44	55	99	10	65	50

<i>chave</i>	1								<i>j</i>		<i>n</i>
10	10	20	25	35	38	40	44	55	99	65	50

# Ordenação por inserção

<i>chave</i>	1						<i>j</i>			<i>n</i>	
99	20	25	35	38	40	44	55	99	10	65	50

<i>chave</i>	1								<i>j</i>		<i>n</i>
10	10	20	25	35	38	40	44	55	99	65	50

<i>chave</i>	1									<i>j</i>	<i>n</i>
65	10	20	25	35	38	40	44	55	99	65	50

# Ordenação por inserção

<i>chave</i>	1							<i>j</i>			<i>n</i>
99	20	25	35	38	40	44	55	99	10	65	50

<i>chave</i>	1								<i>j</i>		<i>n</i>
10	10	20	25	35	38	40	44	55	99	65	50

<i>chave</i>	1									<i>j</i>	<i>n</i>
65	10	20	25	35	38	40	44	55	65	99	50

# Ordenação por inserção

<i>chave</i>	1						<i>j</i>			<i>n</i>	
99	20	25	35	38	40	44	55	99	10	65	50

<i>chave</i>	1								<i>j</i>		<i>n</i>
10	10	20	25	35	38	40	44	55	99	65	50

<i>chave</i>	1									<i>j</i>	<i>n</i>
65	10	20	25	35	38	40	44	55	65	99	50

<i>chave</i>	1										<i>j</i>
50	10	20	25	35	38	40	44	55	65	99	50



# Ordenação por inserção

*chave* 1 *j* *n*

99

20	25	35	38	40	44	55	99	10	65	50
----	----	----	----	----	----	----	----	----	----	----

*chave* 1 *j* *n*

10

10	20	25	35	38	40	44	55	99	65	50
----	----	----	----	----	----	----	----	----	----	----

*chave* 1 *j* *n*

65

10	20	25	35	38	40	44	55	65	99	50
----	----	----	----	----	----	----	----	----	----	----

*chave* 1 *j*

50

10	20	25	35	38	40	44	50	55	65	99
----	----	----	----	----	----	----	----	----	----	----

# Ordenação por inserção

Algoritmo rearranja  $A[1..n]$  em ordem crescente.

**ORDENA-POR-INSERÇÃO** ( $A, n$ )

1   **para**  $j \leftarrow 2$  até  $n$  **faça**

2        $chave \leftarrow A[j]$

3        $i \leftarrow j - 1$

4       **enquanto**  $i \geq 1$  e  $A[i] > chave$  **faça**

5            $A[i + 1] \leftarrow A[i]$    ▷ desloca

6            $i \leftarrow i - 1$

7        $A[i + 1] \leftarrow chave$    ▷ insere

# Ordenação por inserção

Algoritmo rearranja  $A[1..n]$  em ordem crescente

ORDENA-POR-INSERÇÃO ( $A, n$ )

0  $j \leftarrow 2$

1 enquanto  $j \leq n$  faça

2      $chave \leftarrow A[j]$

3      $i \leftarrow j - 1$

4     enquanto  $i \geq 1$  e  $A[i] > chave$  faça

5          $A[i + 1] \leftarrow A[i]$      ▷ desloca

6          $i \leftarrow i - 1$

7      $A[i + 1] \leftarrow chave$      ▷ insere

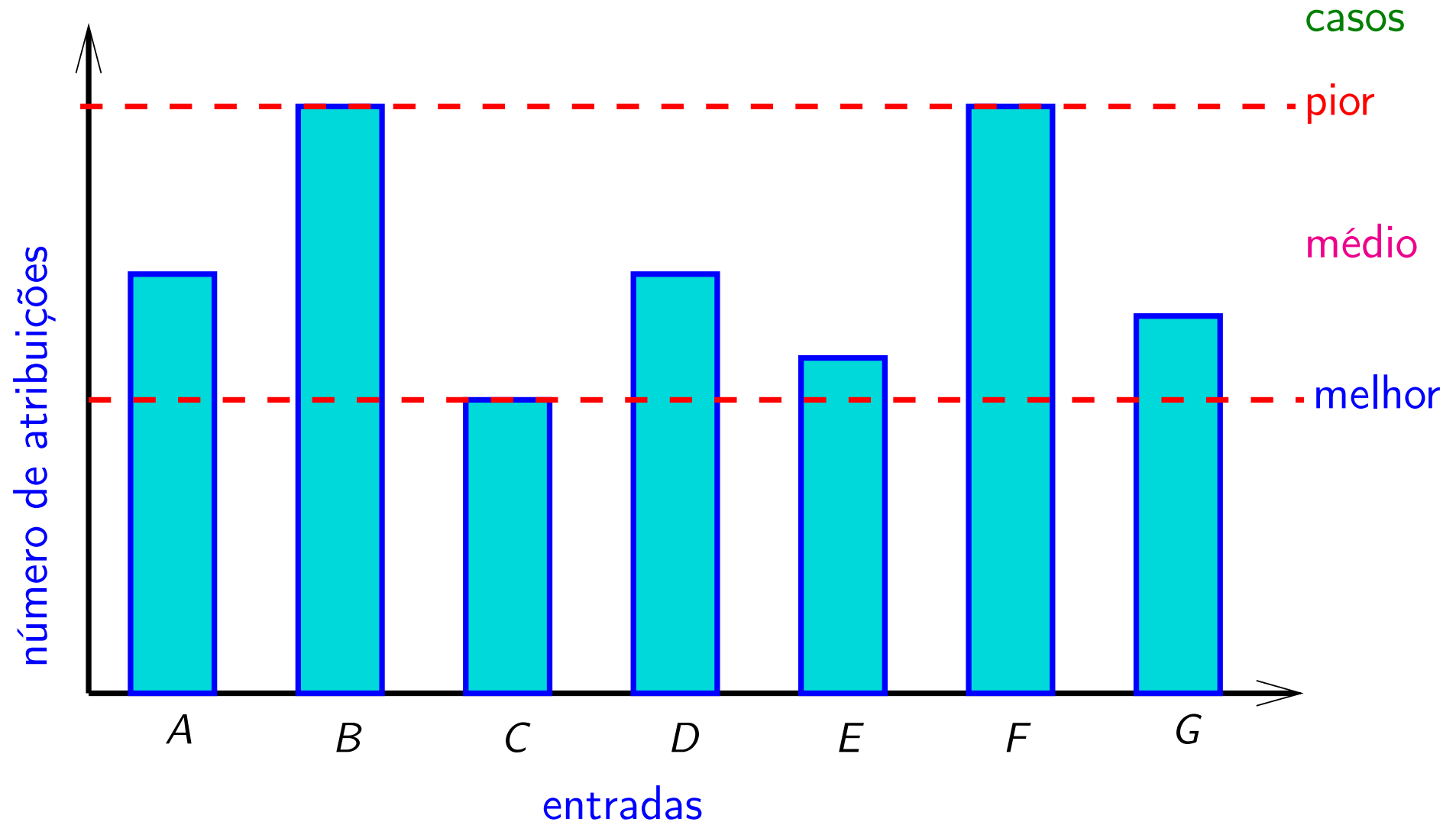
8      $j \leftarrow j + 1$

Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

Número mínimo, médio ou máximo?

Melhor caso, caso médio, pior caso?



# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

LINHAS 3–6 ( $A, j, chave$ )

3      $i \leftarrow j - 1 \quad \triangleright \quad 2 \leq j \leq n$

4     enquanto  $i \geq 1$  e  $A[i] > chave$  faça

5          $A[i + 1] \leftarrow A[i]$

6          $i \leftarrow i - 1$

linha	atribuições (número máximo)
3	?
4	?
5	?
6	?
total	?

# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

LINHAS 3–6 ( $A, j, chave$ )

3      $i \leftarrow j - 1 \quad \triangleright \quad 2 \leq j \leq n$

4     enquanto  $i \geq 1$  e  $A[i] > chave$  faça

5          $A[i + 1] \leftarrow A[i]$

6          $i \leftarrow i - 1$

linha	atribuições (número máximo)
3	= 1
4	= 0
5	?
6	?
total	?

# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

LINHAS 3–6 ( $A, j, chave$ )

3      $i \leftarrow j - 1 \quad \triangleright \quad 2 \leq j \leq n$

4     enquanto  $i \geq 1$  e  $A[i] > chave$  faça

5          $A[i + 1] \leftarrow A[i]$

6          $i \leftarrow i - 1$

linha	atribuições (número máximo)
3	$= 1$
4	$= 0$
5	$\leq j - 1$
6	$?$
<hr/>	
total	$?$



# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

LINHAS 3–6 ( $A, j, chave$ )

3      $i \leftarrow j - 1 \quad \triangleright \quad 2 \leq j \leq n$

4     enquanto  $i \geq 1$  e  $A[i] > chave$  faça

5          $A[i + 1] \leftarrow A[i]$

6          $i \leftarrow i - 1$

linha	atribuições (número máximo)
3	$= 1$
4	$= 0$
5	$\leq j - 1$
6	$\leq j - 1$

$$\text{total} \leq 2j - 1 \leq 2n - 1$$

# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

## ORDENA-POR-INSERTÃO ( $A, n$ )

```
1  para  $j \leftarrow 2$  até  $n$  faça    ▷  $j \leftarrow j + 1$  escondido
2       $chave \leftarrow A[j]$ 
3      LINHAS 3–6 ( $A, j, chave$ )
7       $A[j + 1] \leftarrow chave$ 
```

linha	atribuições (número máximo)
1	?
2	?
3–6	?
7	?
<hr/>	
total	?

# Quantas atribuições ( $\leftarrow$ ) algoritmo faz?

## ORDENA-POR-INSERÇÃO ( $A, n$ )

```
1  para  $j \leftarrow 2$  até  $n$  faça    ▷  $j \leftarrow j + 1$  escondido
2       $chave \leftarrow A[j]$ 
3      LINHAS 3–6 ( $A, j, chave$ )
7       $A[j + 1] \leftarrow chave$ 
```

linha	atribuições (número máximo)
1	$= n - 1 + 1$
2	$= n - 1$
3–6	$\leq (n - 1)(2n - 1)$
7	$= n - 1$

$$\text{total} \leq 2n^2 - 1$$

# Análise mais fina

linha	atribuições (número máximo)
1	$= n - 1 + 1$
2	$= n - 1$
3-6	$\leq 3 + 5 + \dots + (2n-1) = (n+1)(n-1) = n^2 - 1$
7	$= n - 1$

$$\text{total} \leq n^2 + 3n - 3$$

$n^2 + 3n - 3$  versus  $n^2$

$n$	$n^2 + 3n - 3$	$n^2$
1	1	1
2	7	4
3	15	9
10	127	100
100	10297	10000
1000	1002997	1000000
10000	100029997	100000000
100000	10000299997	10000000000

$n^2$  domina os outros termos

## Exercício 1.B

Se a execução de cada linha de código consome **1 unidade** de tempo, qual o consumo total?

ORDENA-POR-INSERÇÃO ( $A, n$ )

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 

3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$   $\triangleright$  desloca
6           $i \leftarrow i - 1$ 

7       $A[i + 1] \leftarrow chave$   $\triangleright$  insere
```

# Solução

linha	todas as execuções da linha
1	$= n$
2	$= n - 1$
3	$= n - 1$
4	$\leq 2 + 3 + \dots + n = (n - 1)(n + 2)/2$
5	$\leq 1 + 2 + \dots + (n - 1) = n(n - 1)/2$
6	$\leq 1 + 2 + \dots + (n - 1) = n(n - 1)/2$
7	$= n - 1$
total	$\leq (3/2)n^2 + (7/2)n - 4$

## Exercício 1.C

Se a execução da linha  $i$  consome  $t_i$  unidades de tempo, para  $i = 1, \dots, 7$ , qual o consumo total?

ORDENA-POR-INSERÇÃO ( $A, n$ )

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 

3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$   $\triangleright$  desloca
6           $i \leftarrow i - 1$ 

7       $A[i + 1] \leftarrow chave$   $\triangleright$  insere
```



## Solução para $t_i = 1$

linha	todas as execuções da linha
1	$= n$
2	$= n - 1$
3	$= n - 1$
4	$\leq 2 + 3 + \dots + n = (n - 1)(n + 2)/2$
5	$\leq 1 + 2 + \dots + (n - 1) = n(n - 1)/2$
6	$\leq 1 + 2 + \dots + (n - 1) = n(n - 1)/2$
7	$= n - 1$
<hr/>	
total	$\leq (3/2)n^2 + (7/2)n - 4$

# Solução

linha	todas as execuções da linha		
1	=	$n$	$\times t_1$
2	=	$n - 1$	$\times t_2$
3	=	$n - 1$	$\times t_3$
4	$\leq$	$2 + 3 + \dots + n = (n - 1)(n + 2)/2$	$\times t_4$
5	$\leq$	$1 + 2 + \dots + (n - 1) = n(n - 1)/2$	$\times t_5$
6	$\leq$	$1 + 2 + \dots + (n - 1) = n(n - 1)/2$	$\times t_6$
7	=	$n - 1$	$\times t_7$
<hr/>			
total	$\leq$	?	

# Solução

linha	todas as execuções da linha		
1	=	$n$	$\times t_1$
2	=	$n - 1$	$\times t_2$
3	=	$n - 1$	$\times t_3$
4	$\leq$	$2 + 3 + \dots + n = (n - 1)(n + 2)/2$	$\times t_4$
5	$\leq$	$1 + 2 + \dots + (n - 1) = n(n - 1)/2$	$\times t_5$
6	$\leq$	$1 + 2 + \dots + (n - 1) = n(n - 1)/2$	$\times t_6$
7	=	$n - 1$	$\times t_7$
<hr/>			
total	$\leq$	$((t_4 + t_5 + t_6)/2) \times n^2$	
	+	$(t_1 + t_2 + t_3 + t_4/2 - t_5/2 - t_6/2 + t_7) \times n$	
	-	$(t_2 + t_3 + t_4 + t_7)$	

# Solução

linha	todas as execuções da linha		
1	=	$n$	$\times t_1$
2	=	$n - 1$	$\times t_2$
3	=	$n - 1$	$\times t_3$
4	$\leq$	$2 + 3 + \dots + n = (n - 1)(n + 2)/2$	$\times t_4$
5	$\leq$	$1 + 2 + \dots + (n - 1) = n(n - 1)/2$	$\times t_5$
6	$\leq$	$1 + 2 + \dots + (n - 1) = n(n - 1)/2$	$\times t_6$
7	=	$n - 1$	$\times t_7$
<hr/>			
total	$\leq$	$c_2 \times n^2 + c_1 \times n + c_0$	

$c_2, c_1, c_0$  são constantes que dependem da máquina.

$n^2$  é para sempre! Está nas entranhas do algoritmo!

# Notação O

Intuitivamente...

$O(f(n)) \approx$  funções que não crescem mais rápido que  $f(n)$   
 $\approx$  funções assintoticamente menores ou iguais a um múltiplo de  $f(n)$

$n^2$        $(3/2)n^2$        $9999n^2$        $n^2/1000$       etc.

crescem todas com a mesma velocidade

# Notação O

Intuitivamente...

$O(f(n)) \approx$  funções que não crescem mais rápido que  $f(n)$   
 $\approx$  funções assintoticamente menores ou iguais a um múltiplo de  $f(n)$

$n^2$        $(3/2)n^2$        $9999n^2$        $n^2/1000$       etc.

crescem todas com a mesma velocidade

- ▶  $n^2 + 99n$  é  $O(n^2)$
- ▶  $33n^2$  é  $O(n^2)$
- ▶  $9n + 2$  é  $O(n^2)$
- ▶  $0,00001n^3 - 200n^2$  não é  $O(n^2)$

# Definição

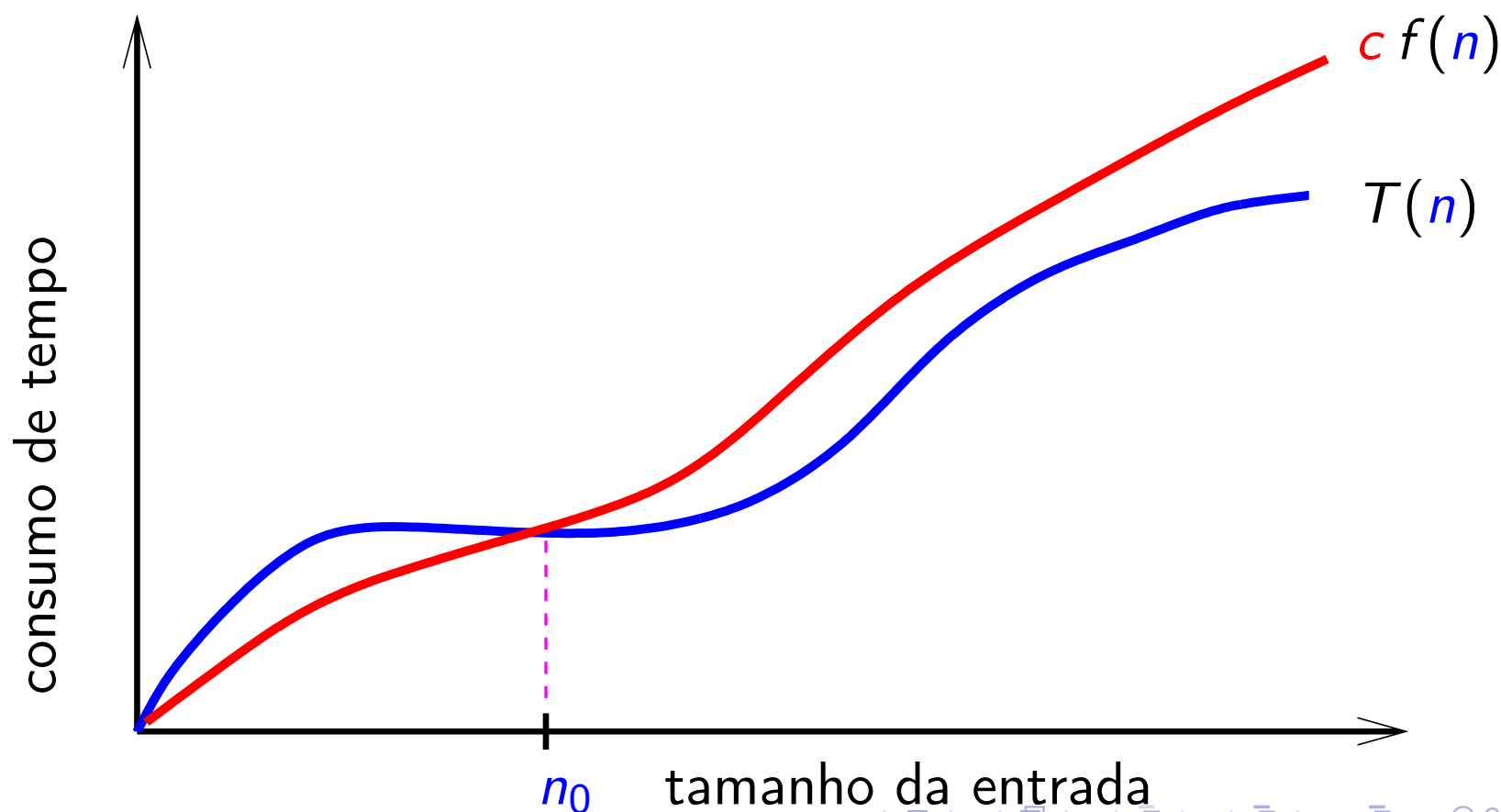
Sejam  $T(n)$  e  $f(n)$  funções dos inteiros nos reais.

Dizemos que  $T(n)$  é  $O(f(n))$  se

existem constantes positivas  $c$  e  $n_0$  tais que

$$T(n) \leq c f(n)$$

para todo  $n \geq n_0$ .

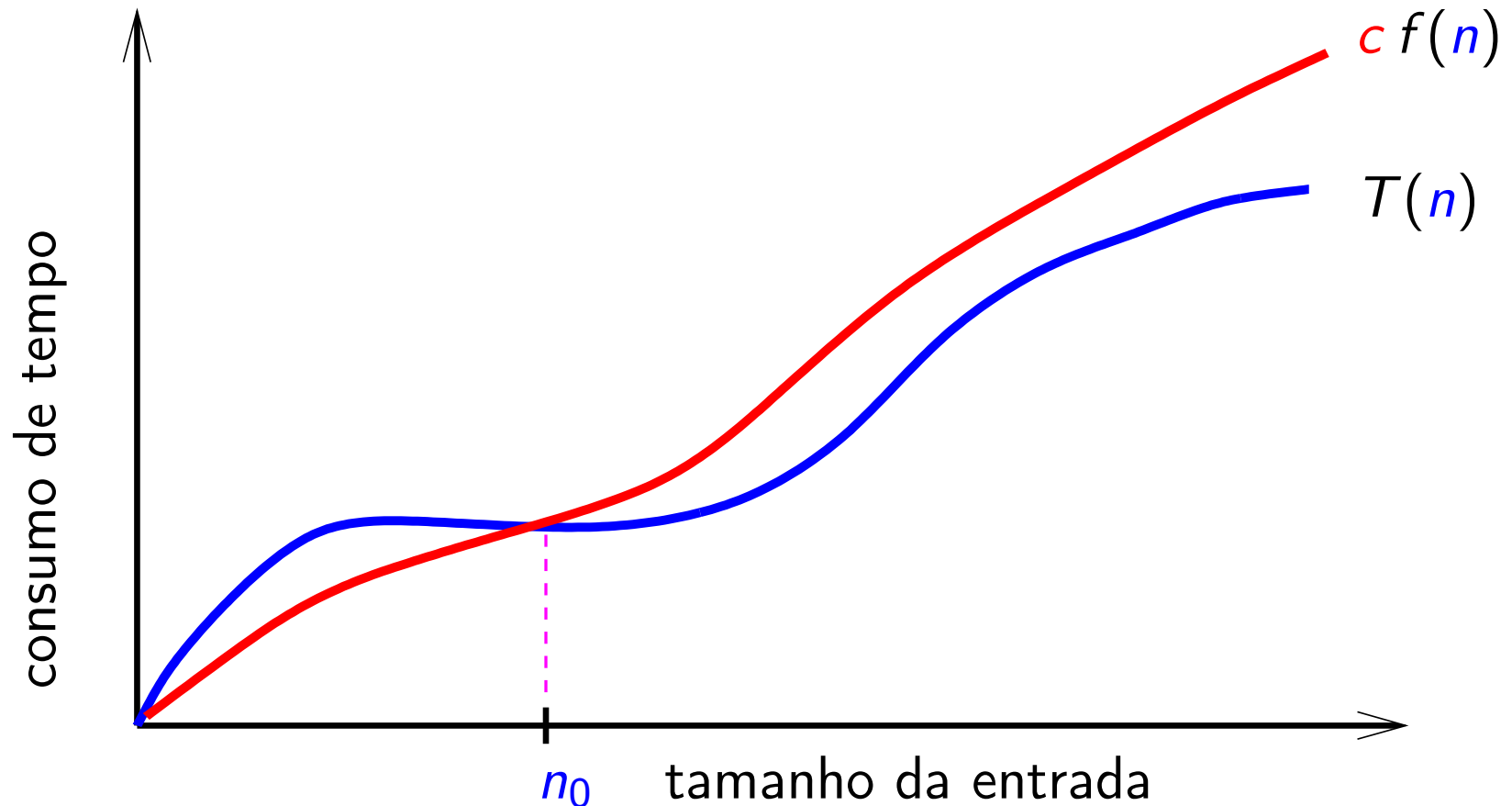


## Mais informal

$T(n)$  é  $O(f(n))$  se existe  $c > 0$  tal que

$$T(n) \leq c f(n)$$

para todo  $n$  suficientemente GRANDE.





# Exemplos

$T(n)$  é  $O(f(n))$  lê-se “ $T(n)$  é O de  $f(n)$ ”

# Exemplos

$T(n)$  é  $O(f(n))$  lê-se “ $T(n)$  é O de  $f(n)$ ”

## Exemplo 1

$10n^2$  é  $O(n^3)$ .

# Exemplos

$T(n)$  é  $O(f(n))$  lê-se “ $T(n)$  é O de  $f(n)$ ”

## Exemplo 1

$10n^2$  é  $O(n^3)$ .

**Prova:** Para todo inteiro  $n \geq 0$ , temos que  $0 \leq 10n^2 \leq 10n^3$ .

# Exemplos

$T(n)$  é  $O(f(n))$  lê-se “ $T(n)$  é O de  $f(n)$ ”

## Exemplo 1

$10n^2$  é  $O(n^3)$ .

**Prova:** Para todo inteiro  $n \geq 0$ , temos que  $0 \leq 10n^2 \leq 10n^3$ .

**Outra prova:** Para todo  $n \geq 10$ , temos  $0 \leq 10n^2 \leq n \times n^2 = 1n^3$ .

# Exemplos

$T(n)$  é  $O(f(n))$  lê-se “ $T(n)$  é O de  $f(n)$ ”

## Exemplo 1

$10n^2$  é  $O(n^3)$ .

**Prova:** Para todo inteiro  $n \geq 0$ , temos que  $0 \leq 10n^2 \leq 10n^3$ .

**Outra prova:** Para todo  $n \geq 10$ , temos  $0 \leq 10n^2 \leq n \times n^2 = 1n^3$ .

## Exemplo 2

$\lg n$  é  $O(n)$ .

# Exemplos

$T(n)$  é  $O(f(n))$  lê-se “ $T(n)$  é O de  $f(n)$ ”

## Exemplo 1

$10n^2$  é  $O(n^3)$ .

**Prova:** Para todo inteiro  $n \geq 0$ , temos que  $0 \leq 10n^2 \leq 10n^3$ .

**Outra prova:** Para todo  $n \geq 10$ , temos  $0 \leq 10n^2 \leq n \times n^2 = 1n^3$ .

## Exemplo 2

$\lg n$  é  $O(n)$ .

**Prova:** Seja  $n \geq 2$ . Como  $\frac{1}{t \ln 2} \leq 1$  para todo  $t \in [2, n]$ , temos

$$\lg n = 1 + \lg n - \lg 2 = 1 + \int_2^n \frac{1}{t \ln 2} dt \leq 1 + \int_2^n 1 dt = n - 2 + 1 \leq 1n$$

pelo Teorema Fundamental do Cálculo.

# Mais exemples

## Exemplo 3

$20n^3 + 10n \lg n + 5$  é  $O(n^3)$ .

# Mais exemplos

## Exemplo 3

$20n^3 + 10n \lg n + 5$  é  $O(n^3)$ .

**Prova:** Para todo  $n \geq 2$ , tem-se que

$$20n^3 + 10n \lg n + 5 \leq 20n^3 + 10n^3 + 5n^3 = 35n^3.$$



# Mais exemplos

## Exemplo 3

$20n^3 + 10n \lg n + 5$  é  $O(n^3)$ .

**Prova:** Para todo  $n \geq 2$ , tem-se que

$$20n^3 + 10n \lg n + 5 \leq 20n^3 + 10n^3 + 5n^3 = 35n^3.$$

**Outra prova:** Para todo  $n \geq 10$ , tem-se que

$$20n^3 + 10n \lg n + 5 \leq 20n^3 + n n \lg n + n \leq 20n^3 + n^3 + n^3 = 22n^3.$$

# Uso da notação $O$

$$O(f(n)) = \{T(n) : \text{existem } c \text{ e } n_0 \text{ tq } T(n) \leq cf(n), n \geq n_0\}$$

“ $T(n)$  é  $O(f(n))$ ” deve ser entendido como “ $T(n) \in O(f(n))$ ”.

“ $T(n) = O(f(n))$ ” deve ser entendido como “ $T(n) \in O(f(n))$ ”.

“ $T(n)$  é  $g(n) + O(f(n))$ ” significa que existe  $h(n) \in O(f(n))$  tal que

$$T(n) \leq g(n) + h(n)$$

para todo  $n$ .

“ $T(n) \leq O(f(n))$ ” é feio.

“ $T(n) \geq O(f(n))$ ” não faz sentido!

# Nomes de classes $O$

classe	nome
$O(1)$	constante
$O(\lg n)$	logarítmica
$O(n)$	linear
$O(n \lg n)$	$n \log n$
$O(n^2)$	quadrática
$O(n^3)$	cúbica
$O(n^k)$ com $k \geq 1$	polinomial
$O(2^n)$	exponencial
$O(a^n)$ com $a > 1$	exponencial

# De volta à ordenação por inserção

ORDENA-POR-INSERÇÃO ( $A, n$ )

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$        $i \leftarrow j - 1$ 
3      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
4           $A[i + 1] \leftarrow A[i]$        $\triangleright$  desloca
5           $i \leftarrow i - 1$ 
6       $A[i + 1] \leftarrow chave$        $\triangleright$  insere
```

linha	consumo de todas as execuções da linha
1	?
2	?
3	?
4	?
5	?
6	?
total	?

# De volta à ordenação por inserção

ORDENA-POR-INSERÇÃO ( $A, n$ )

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$        $i \leftarrow j - 1$ 
3      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
4           $A[i + 1] \leftarrow A[i]$        $\triangleright$  desloca
5           $i \leftarrow i - 1$ 
6       $A[i + 1] \leftarrow chave$        $\triangleright$  insere
```

linha	consumo de todas as execuções da linha
1	$O(n)$
2	$O(n)$
3	$O(n^2)$
4	$O(n^2)$
5	$O(n^2)$
6	$O(n)$
total	$3 \cdot O(n^2) + 3 \cdot O(n) = O(n^2)$

# Conclusão

O algoritmo ORDENA-POR-INSERÇÃO consome  
 $O(n^2)$  unidades de tempo.

Também escreve-se

O algoritmo ORDENA-POR-INSERÇÃO consome  
tempo  $O(n^2)$ .