Introdução

CLRS 1.1, 1.2, 2.1 e 2.2
AU 3.3, 3.4 e 3.6

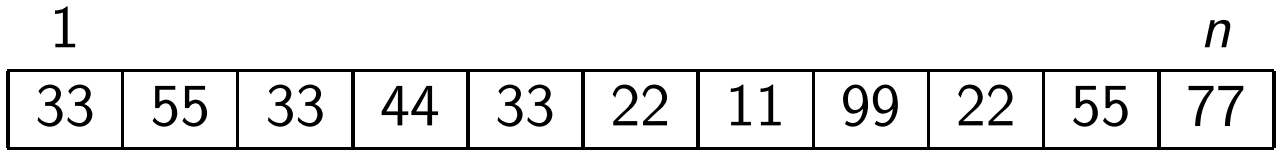
Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Ordenação

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

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

Entra:



Ordenação

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

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

Entra:

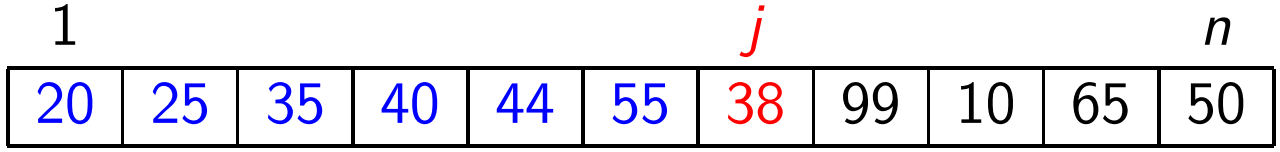
1										n
33	55	33	44	33	22	11	99	22	55	77

Sai:

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

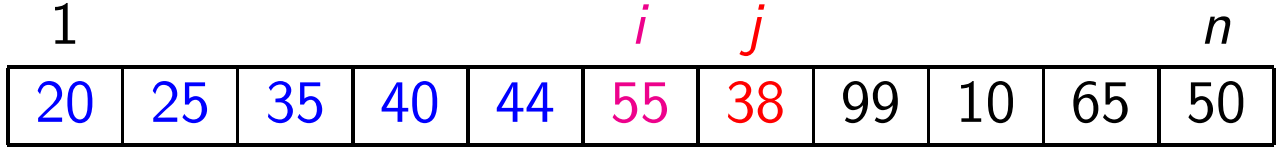
Ordenação por inserção

$chave = 38$



Ordenação por inserção

$chave = 38$



Ordenação por inserção

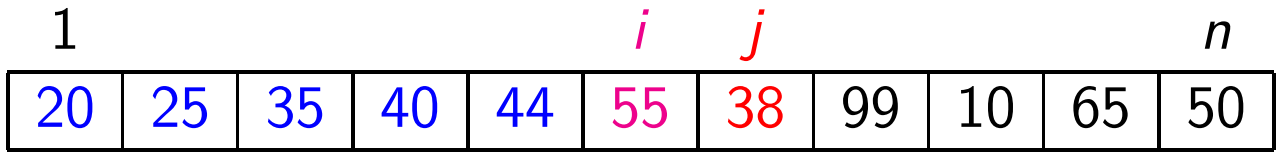
$chave = 38$

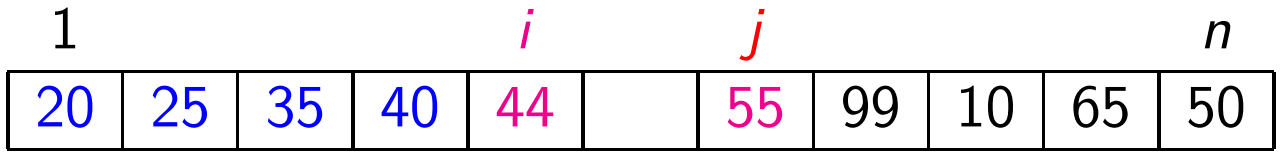
1					i	j				n
20	25	35	40	44	55	38	99	10	65	50

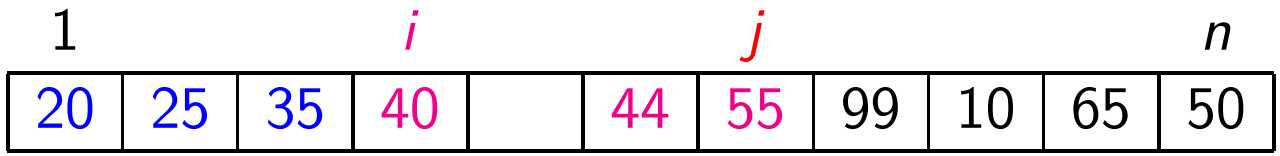
1				i		j				n
20	25	35	40	44		55	99	10	65	50

Ordenação por inserção

$chave = 38$

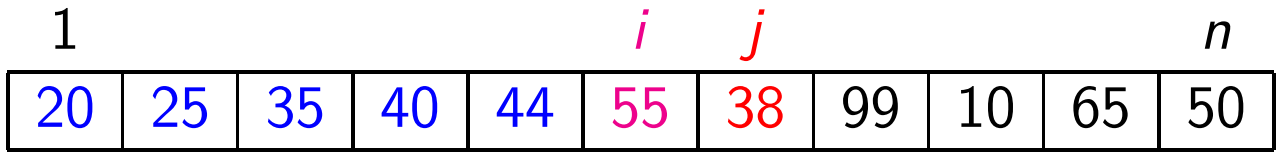


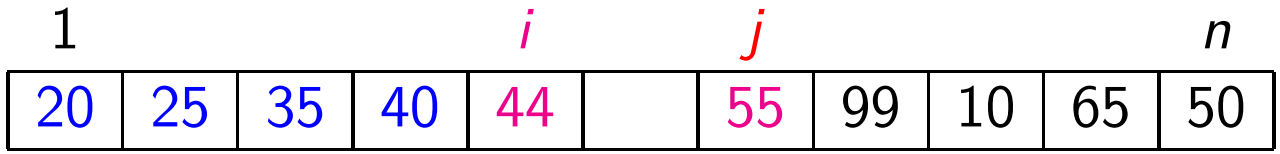


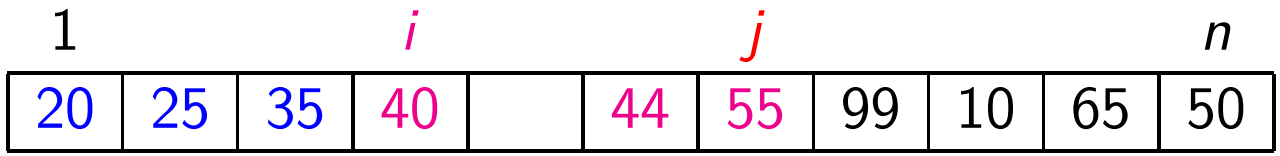


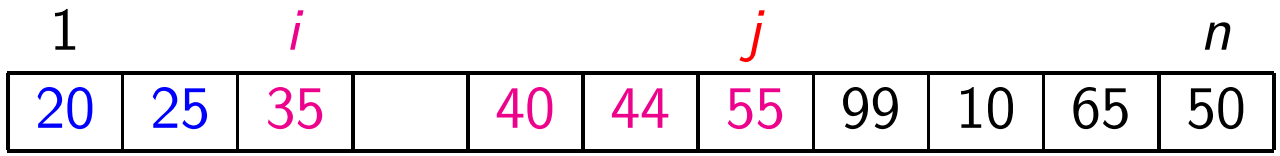
Ordenação por inserção

$chave = 38$



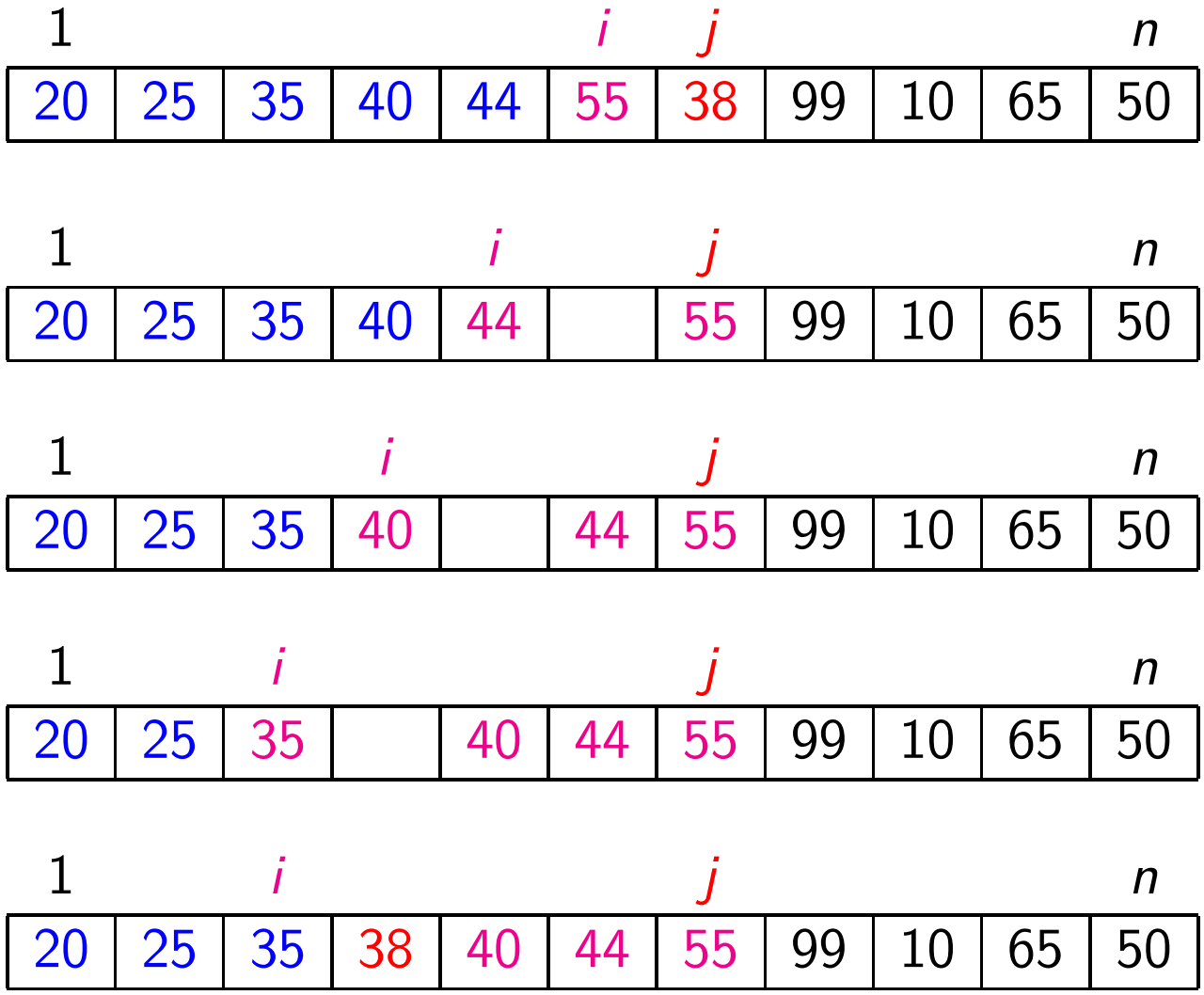




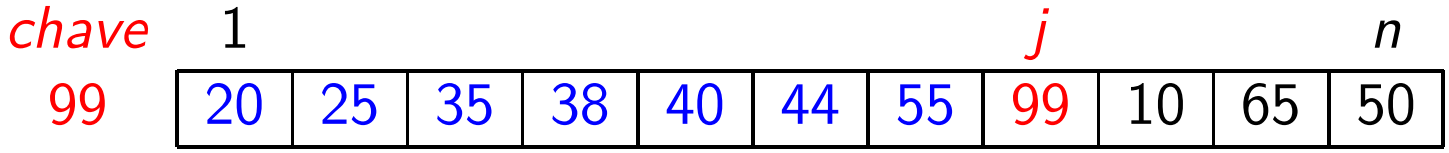


Ordenação por inserção

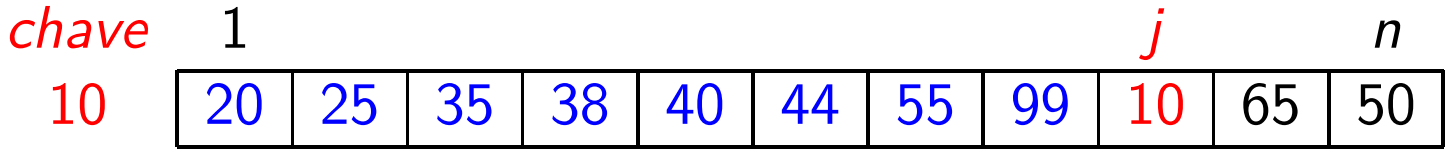
$chave = 38$



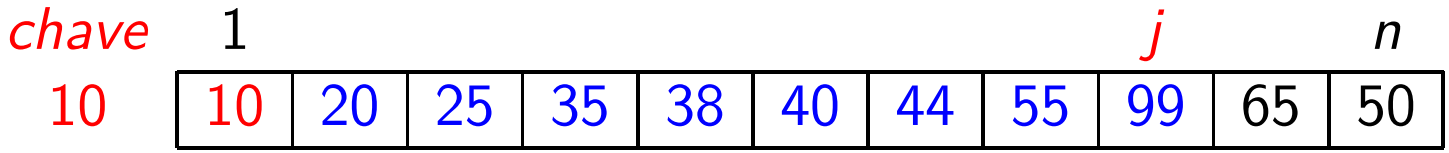
Ordenação por inserção



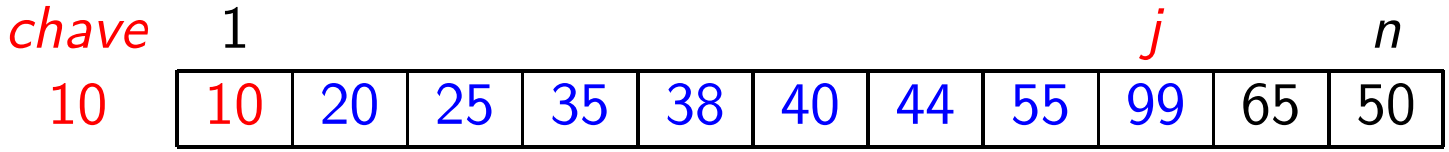
Ordenação por inserção

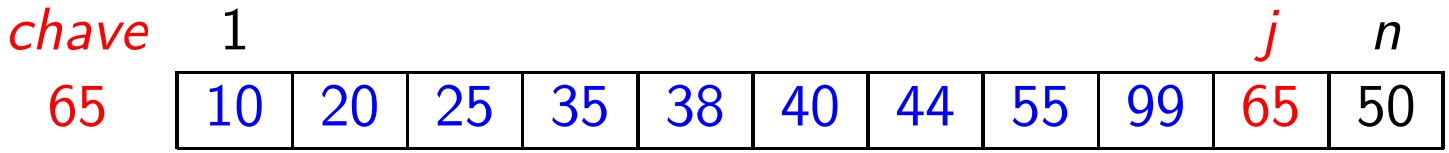


Ordenação por inserção

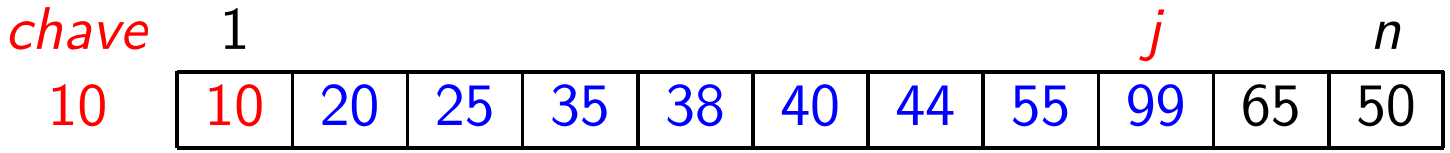


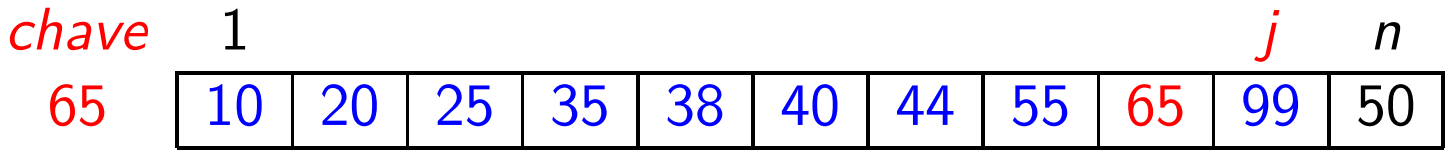
Ordenação por inserção



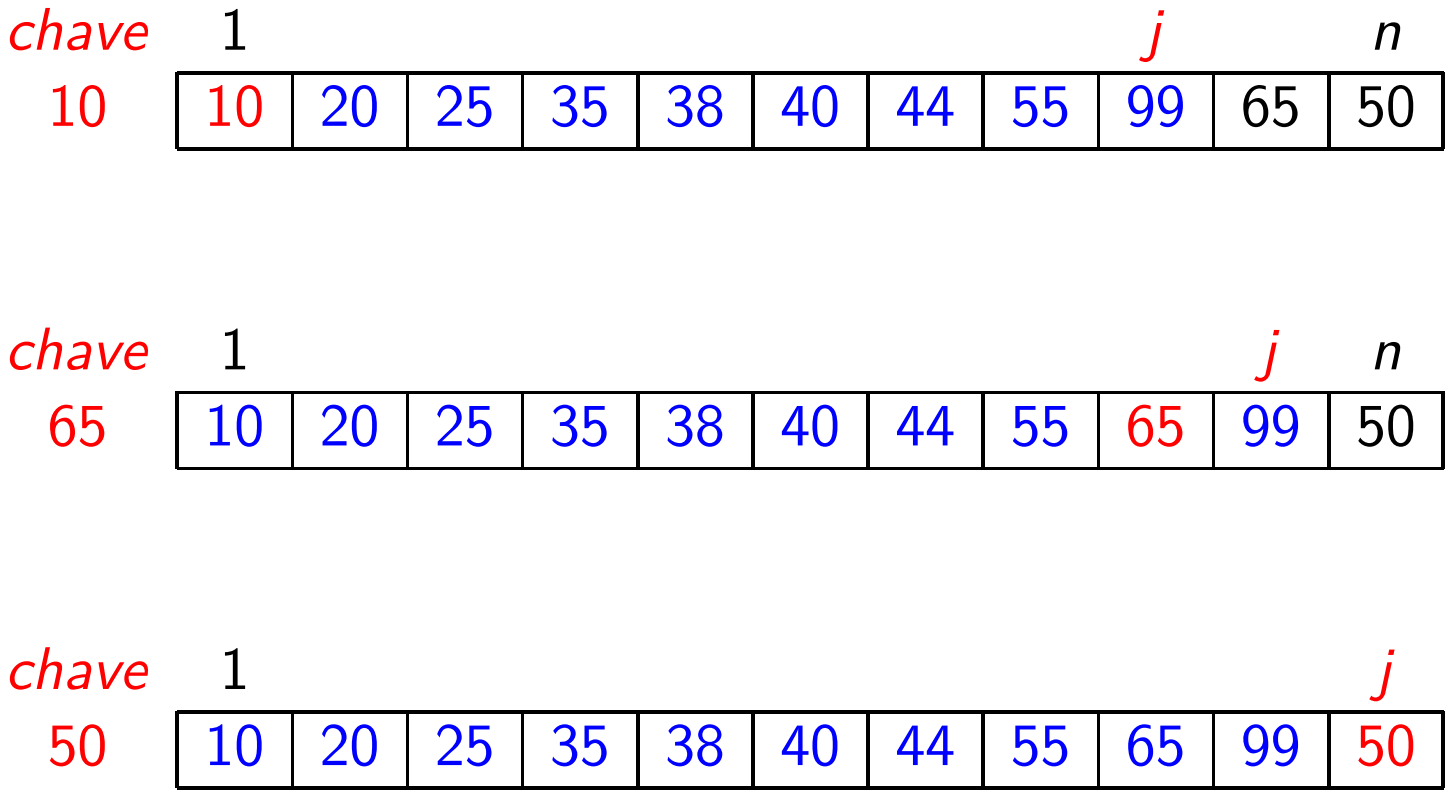


Ordenação por inserção

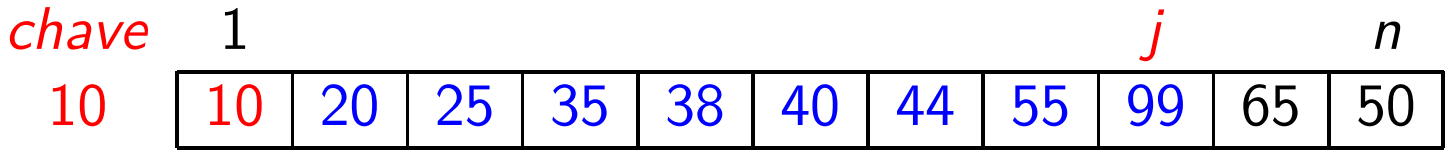


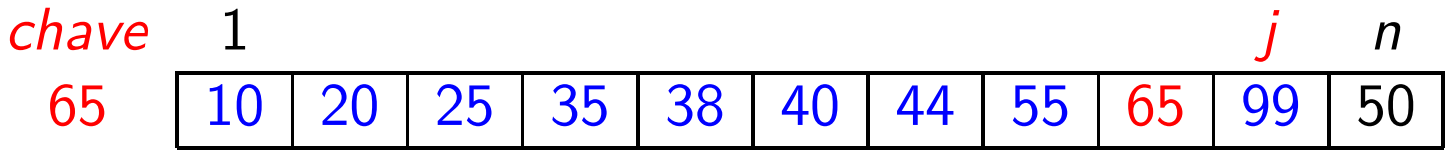


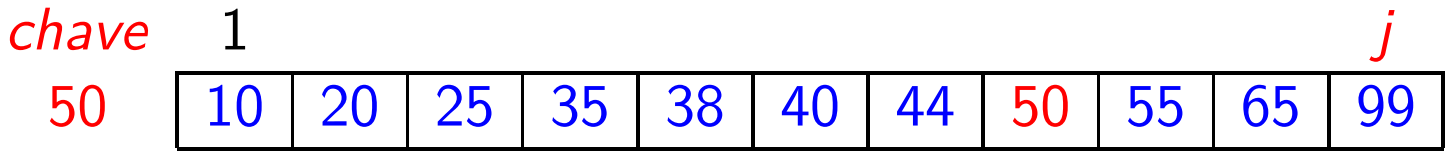
Ordenação por inserção



Ordenação por inserção







Ordenação por inserção

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

```
ORDENA-POR-INSERÇÃO (A, n)
1  para j ← 2 até n faça
2      chave ← A[j]
3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]        ▷ desloca
6          i ← i − 1
7      A[i + 1] ← chave           ▷ insere
```

Ordenação por inserção

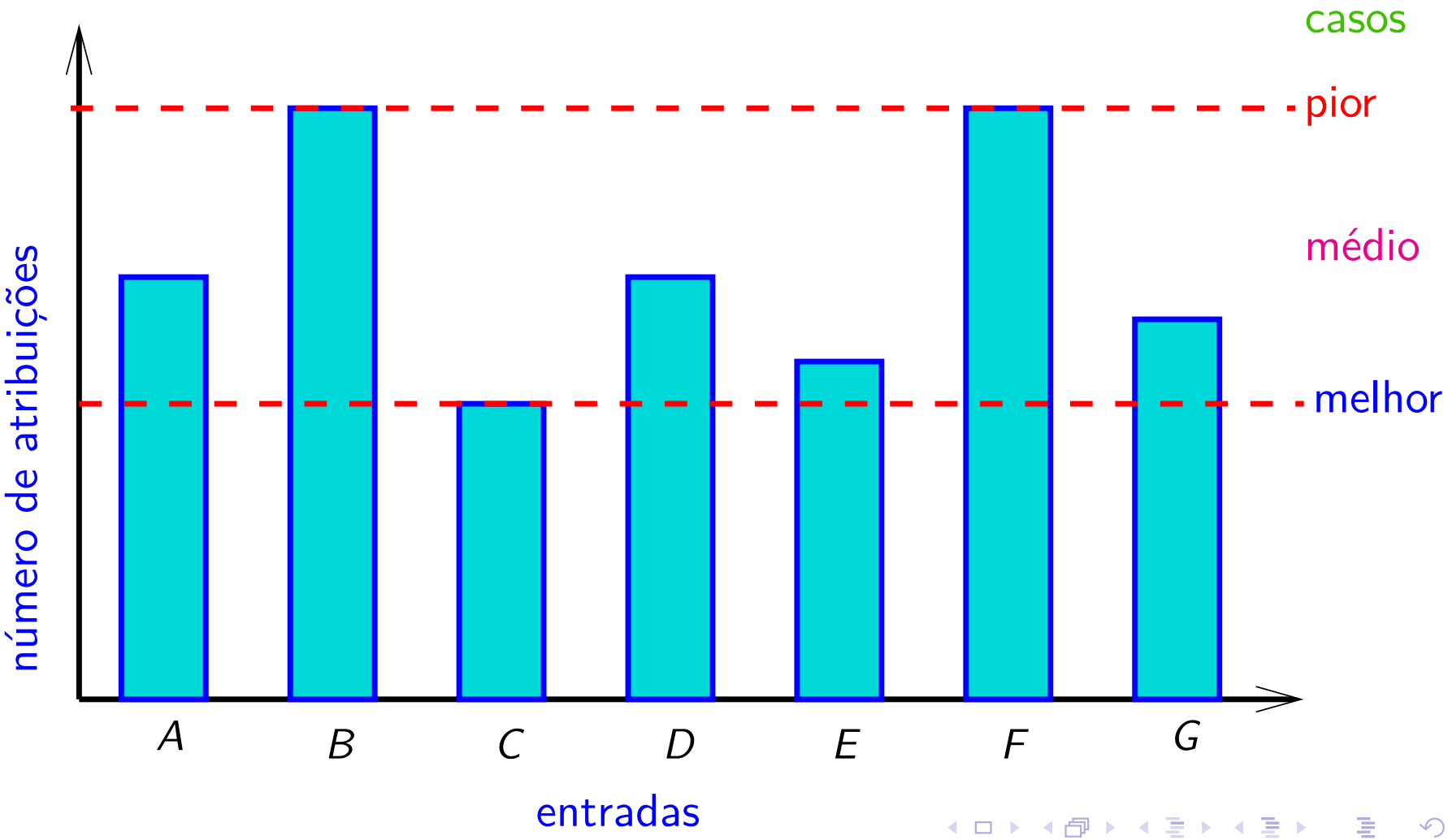
Algoritmo rearranja $A[1..n]$ em ordem crescente

```
ORDENA-POR-INSERÇÃO (A, n)
0  j ← 2
1  enquanto j ≤ n faça
2      chave ← A[j]

3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]      ▷ desloca
6          i ← i − 1

7      A[i + 1] ← chave         ▷ insere
8      j ← j + 1
```

Quantas atribuições ($\leftarrow$) algoritmo faz?

Quantas atribuições (←) 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 ← j − 1          ▷  2 ≤ j ≤ n
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]
6          i ← i − 1
```

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

Quantas atribuições ($\leftarrow$) algoritmo faz?

LINHAS 3–6 $(A, j, chave)$

```
3      i ← j − 1          ▷  2 ≤ j ≤ n
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]
6          i ← 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 ← j − 1          ▷  2 ≤ j ≤ n
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]
6          i ← i − 1
```

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

Quantas atribuições ($\leftarrow$) algoritmo faz?

LINHAS 3–6 $(A, j, chave)$

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

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

5 $\quad\quad A[i+1] \leftarrow A[i]$

6 $\quad\quad i \leftarrow i-1$

linha	atribuições (número máximo)
3	$= 1$
4	$= 0$
5	$\leq j-1$
6	$\leq j-1$
total	$\leq 2j-1 \leq 2n-1$

Quantas atribuições ($\leftarrow$) algoritmo faz?

ORDENA-POR-INSERÇÃO (A, n)

1 **para** $j \leftarrow 2$ **até** n **faça** $\triangleright$ $j \leftarrow j+1$ escondido

2 $\quad chave \leftarrow A[j]$

3 $\quad$ LINHAS 3–6 $(A, j, chave)$

7 $\quad A[i+1] \leftarrow chave$

linha	atribuições (número máximo)
1	?
2	?
3–6	?
7	?
total	?

Quantas atribuições ($\leftarrow$) algoritmo faz?

ORDENA-POR-INSERÇÃO (A, n)

1 **para** $j \leftarrow 2$ **até** n **faça** $\quad \triangleright$ $j \leftarrow j+1$ escondido

2 $\quad$ $chave \leftarrow A[j]$

3 $\quad$ LINHAS 3–6 $(A, j, chave)$

7 $\quad$ $A[i+1] \leftarrow chave$

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

Análise mais fina

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

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

n	$n^2 + 3n - 3$	n^2
1	1	1
2	7	4
3	15	9
10	127	100

$n^2 + 3n - 3$ **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

$n^2 + 3n - 3$ **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 ← 2 até n faça
2       chave ← A[j]

3       i ← j − 1
4       enquanto i ≥ 1 e A[i] > chave faça
5           A[i + 1] ← A[i]      ▷ desloca
6           i ← i − 1

7       A[i + 1] ← chave      ▷ insere
```

Solução

linha		todas as execuções da linha
1	$=$	n
2	$=$	$n-1$
3	$=$	$n-1$
4	$\leq$	$2+3+\cdots+n \;=\; (n-1)(n+2)/2$
5	$\leq$	$1+2+\cdots+(n-1) \;=\; n(n-1)/2$
6	$\leq$	$1+2+\cdots+(n-1) \;=\; n(n-1)/2$
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 $i = 1, \ldots, 7$, qual o consumo total?

```
ORDENA-POR-INSERÇÃO (A, n)
1   para j ← 2 até n faça
2       chave ← A[j]
3       i ← j − 1
4       enquanto i ≥ 1 e A[i] > chave faça
5           A[i + 1] ← A[i]      ▷ desloca
6           i ← i − 1
7       A[i + 1] ← chave         ▷ insere
```

Solução para $t_i = 1$

linha		todas as execuções da linha
1	$=$	n
2	$=$	$n-1$
3	$=$	$n-1$
4	$\leq$	$2+3+\cdots+n \;=\; (n-1)(n+2)/2$
5	$\leq$	$1+2+\cdots+(n-1) \;=\; n(n-1)/2$
6	$\leq$	$1+2+\cdots+(n-1) \;=\; n(n-1)/2$
7	$=$	$n-1$
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	$\leq$	$2+3+\cdots+n \;=\; (n-1)(n+2)/2$	$\times t_4$
5	$\leq$	$1+2+\cdots+(n-1) \;=\; n(n-1)/2$	$\times t_5$
6	$\leq$	$1+2+\cdots+(n-1) \;=\; n(n-1)/2$	$\times t_6$
7	$=$	$n-1$	$\times t_7$
total	$\leq$	?	

Solução

linha	todas as execuções da linha	
1	$= \quad n$	$\times t_1$
2	$= \quad n-1$	$\times t_2$
3	$= \quad n-1$	$\times t_3$
4	$\leq \quad 2+3+\cdots+n \;=\; (n-1)(n+2)/2$	$\times t_4$
5	$\leq \quad 1+2+\cdots+(n-1) \;=\; n(n-1)/2$	$\times t_5$
6	$\leq \quad 1+2+\cdots+(n-1) \;=\; n(n-1)/2$	$\times t_6$
7	$= \quad n-1$	$\times t_7$

$$
\begin{aligned}
\text{total} \quad &\leq \quad ((t_4+t_5+t_6)/2)\times n^2 \\
&+ \quad (t_1+t_2+t_3+t_4/2-t_5/2-t_6/2+t_7)\times n \\
&- \quad (t_2+t_3+t_4+t_7)
\end{aligned}
$$

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	$\leq 2+3+\cdots+n = (n-1)(n+2)/2$	$\times t_4$
5	$\leq 1+2+\cdots+(n-1) = n(n-1)/2$	$\times t_5$
6	$\leq 1+2+\cdots+(n-1) = n(n-1)/2$	$\times t_6$
7	$= n-1$	$\times t_7$
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 menores ou iguais a um múltiplo de $f(n)$

n^2 $\quad (3/2)n^2$ $\quad 9999n^2$ $\quad n^2/1000$ $\quad$ 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 menores ou iguais a um múltiplo de $f(n)$

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

crescem todas com a mesma velocidade

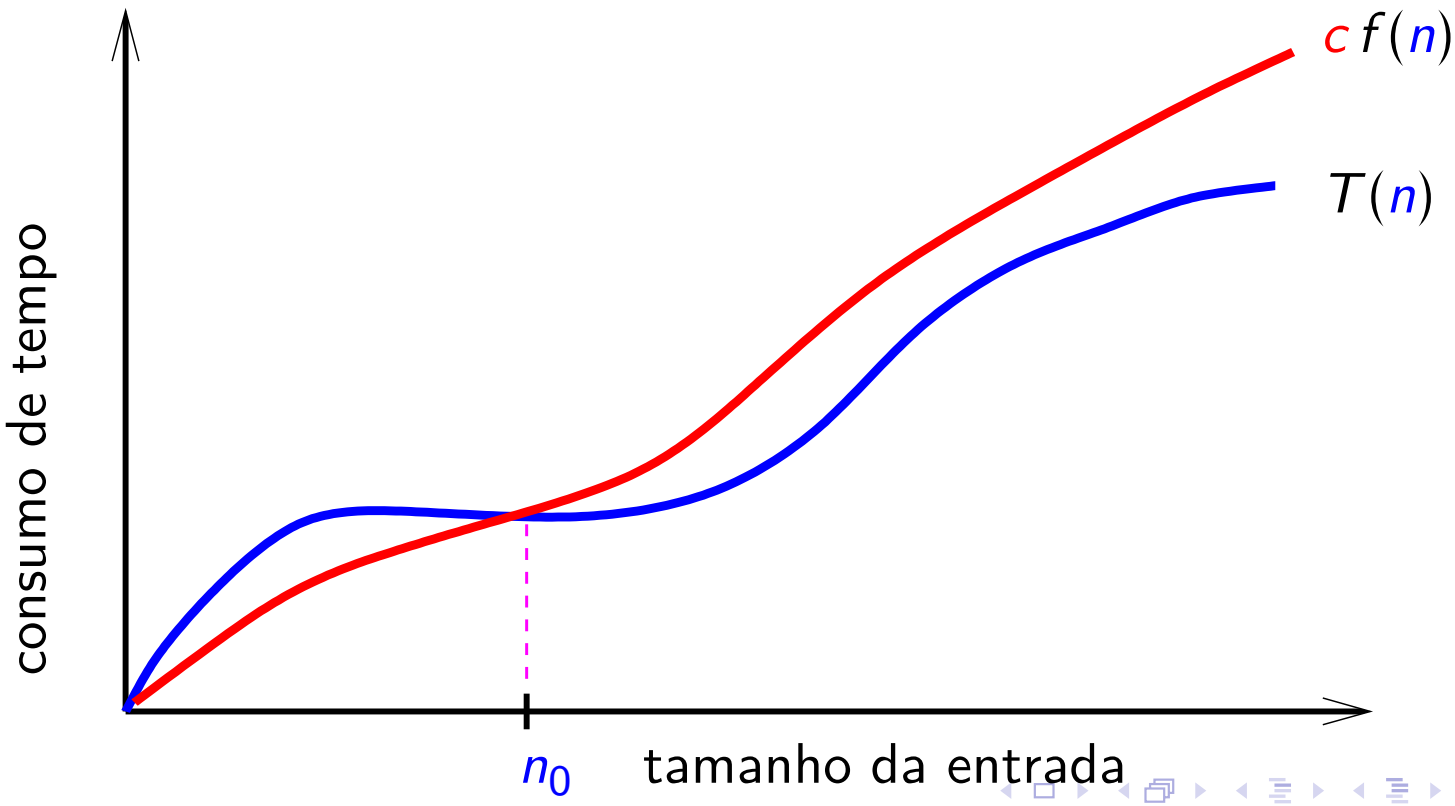
- $n^2 + 99n$ é $O(n^2)$
- $33n^2$ é $O(n^2)$
- $9n + 2$ é $O(n^2)$
- $0{,}00001n^3 - 200n^2$ não é $O(n^2)$

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros nos reais.
Dizemos que $T(n)$ é $\mathrm{O}(f(n))$ se existem constantes positivas c e n_0 tais que

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

para todo $n \geq n_0$.

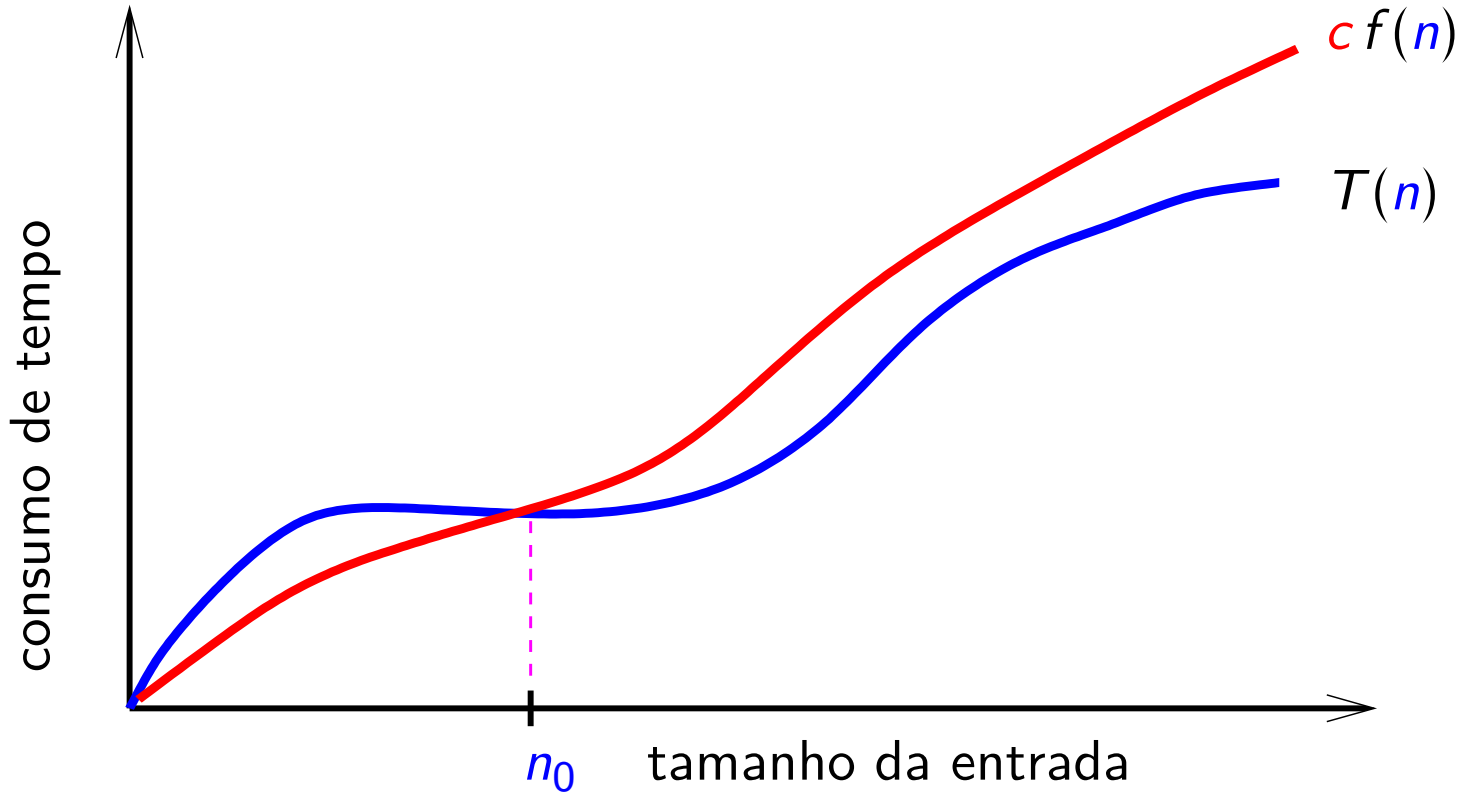


Mais informal

$T(n)$ 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 $\mathrm{O}(f(n))$ lê-se "$T(n)$ é O de $f(n)$" ou
"$T(n)$ é da ordem de $f(n)$"

Exemplos

$T(n)$ é $\mathrm{O}(f(n))$ lê-se "$T(n)$ é O de $f(n)$" ou
"$T(n)$ é da ordem de $f(n)$"

Exemplo 1

$10n^2$ é $\mathrm{O}(n^3)$.

Exemplos

$T(n)$ é $\mathrm{O}(f(n))$ lê-se "$T(n)$ é O de $f(n)$" ou
"$T(n)$ é da ordem de $f(n)$"

Exemplo 1

$10n^2$ é $\mathrm{O}(n^3)$.

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

Exemplos

$T(n)$ é $\mathrm{O}(f(n))$ lê-se "$T(n)$ é O de $f(n)$" ou
"$T(n)$ é da ordem de $f(n)$"

Exemplo 1

$10n^2$ é $\mathrm{O}(n^3)$.

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

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

Exemplos

$T(n)$ é $\mathrm{O}(f(n))$ lê-se "$T(n)$ é O de $f(n)$" ou
"$T(n)$ é da ordem de $f(n)$"

Exemplo 1

$10n^2$ é $\mathrm{O}(n^3)$.

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

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

Exemplo 2

$\lg n$ é $\mathrm{O}(n)$.

Exemplos

$T(n)$ é $\mathrm{O}(f(n))$ lê-se "$T(n)$ é O de $f(n)$" ou
"$T(n)$ é da ordem de $f(n)$"

Exemplo 1

$10n^2$ é $\mathrm{O}(n^3)$.

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

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

Exemplo 2

$\lg n$ é $\mathrm{O}(n)$.

Prova: Para $n \geq 1$, tem-se que $\lg n \leq 1\, n$.

Mais exemplos

Exemplo 3

$20n^3 + 10n \log n + 5$ é $\mathrm{O}(n^3)$.

Mais exemplos

Exemplo 3

$20n^3 + 10n \log n + 5$ é $\mathrm{O}(n^3)$.

Prova: Para $n \geq 1$, tem-se que

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

Mais exemplos

Exemplo 3

$20n^3 + 10n \log n + 5$ é $\mathrm{O}(n^3)$.

Prova: Para $n \geq 1$, tem-se que

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

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

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

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

"$T(n) \leq O(f(n))$" é feio.

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

"$T(n)$ é $g(n) + O(f(n))$" significa que existe constantes positivas c e n_0 tais que

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

para todo $n \geq n_0$.

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