

# **MAC5711 Análise de Algoritmos**

**Slides de Paulo Feofiloff**

**[com erros do coelho]**

# MAC5711 Análise de Algoritmos

Slides de **Paulo Feofiloff**

[com erros do coelho]

“A análise de algoritmos é uma disciplina de engenharia. Um engenheiro civil, por exemplo, tem métodos e tecnologia para **prever** o comportamento de uma estrutura antes de construí-la.

Da mesma forma, um projetista de algoritmos deve ser capaz de **prever** o comportamento de um algoritmo antes de implementá-lo.”

# Avisos

- **Página da disciplina:**

[www.ime.usp.br/~coelho/algoritmos/](http://www.ime.usp.br/~coelho/algoritmos/)

- **Panda:** Cadastro, fórum, entregas de trabalho

- **Monitor:** Paulo Silveira, 2a. das 12:30 às 14:00 (?)

- **Livros:**

- CLRS = Cormen, Leiserson, Rivest, Stein,  
*Introduction to Algorithms*

- AU = Aho, Ullman, *Foundations of Computer Science*

- TAOCP = Knuth, *The Art of Computer Programming*

- **Tarefas**

- **Alunos especiais:** formulário, comissão.

# Blá-Blá-Blá

Correção de algoritmos iterativos através de suas relações invariantes.

Tema recorrente e fundamental de MAC5711: a estrutura recursiva de muitos problemas.

Vamos estudar uma coleção de diferentes problemas. Eles servirão de **paradigma** para muitas técnicas e idéias importantes e muito gerais, aplicáveis a muitos outros problemas no futuro.

# Introdução à AA

CLRS 2.1–2.2

AU 3.3, 3.6

# Exercício 1.A

Quanto vale  $S$  no fim do algoritmo?

1  $S \leftarrow 0$

2 **para**  $i \leftarrow 2$  **até**  $n - 2$  **faça**

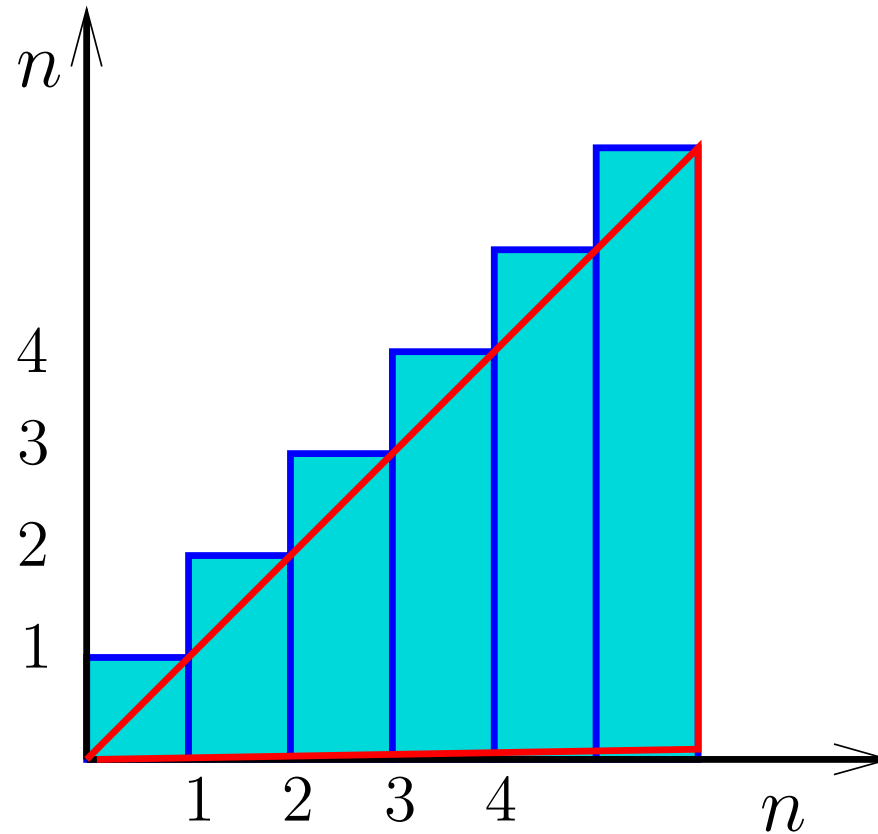
3     **para**  $j \leftarrow i$  **até**  $n$  **faça**

4              $S \leftarrow S + 1$

Escreva um algoritmo mais eficiente que tenha o mesmo efeito.

$$1 + 2 + \dots + (n - 1) + n = ?$$

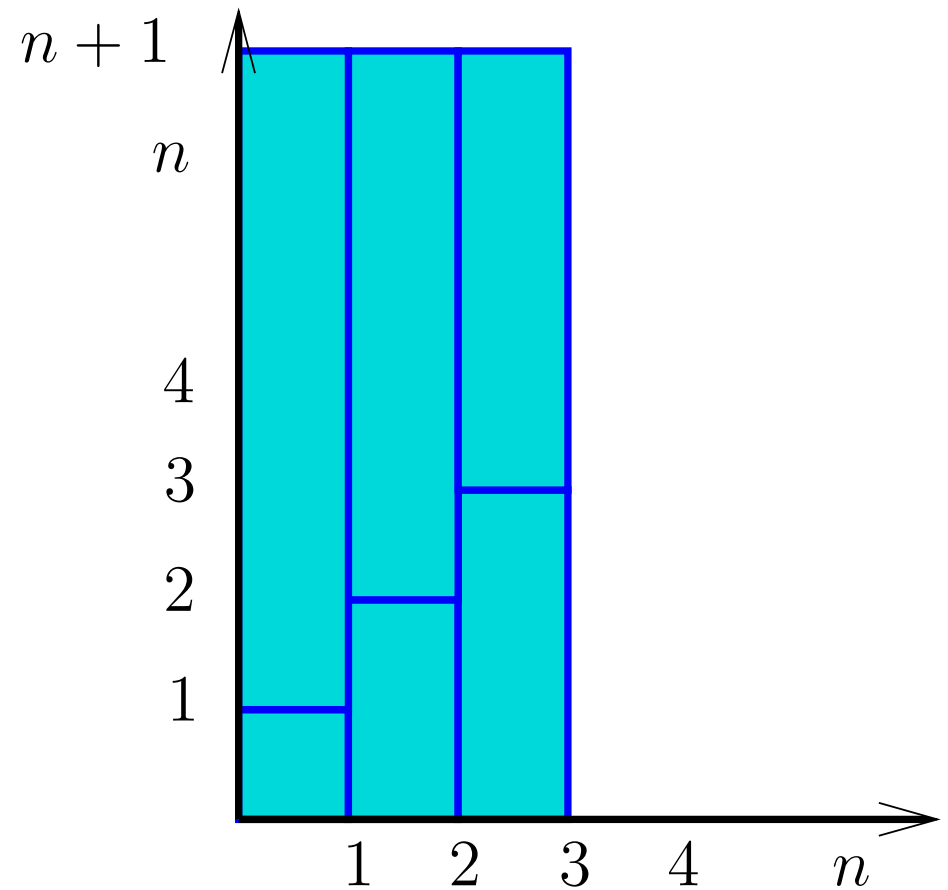
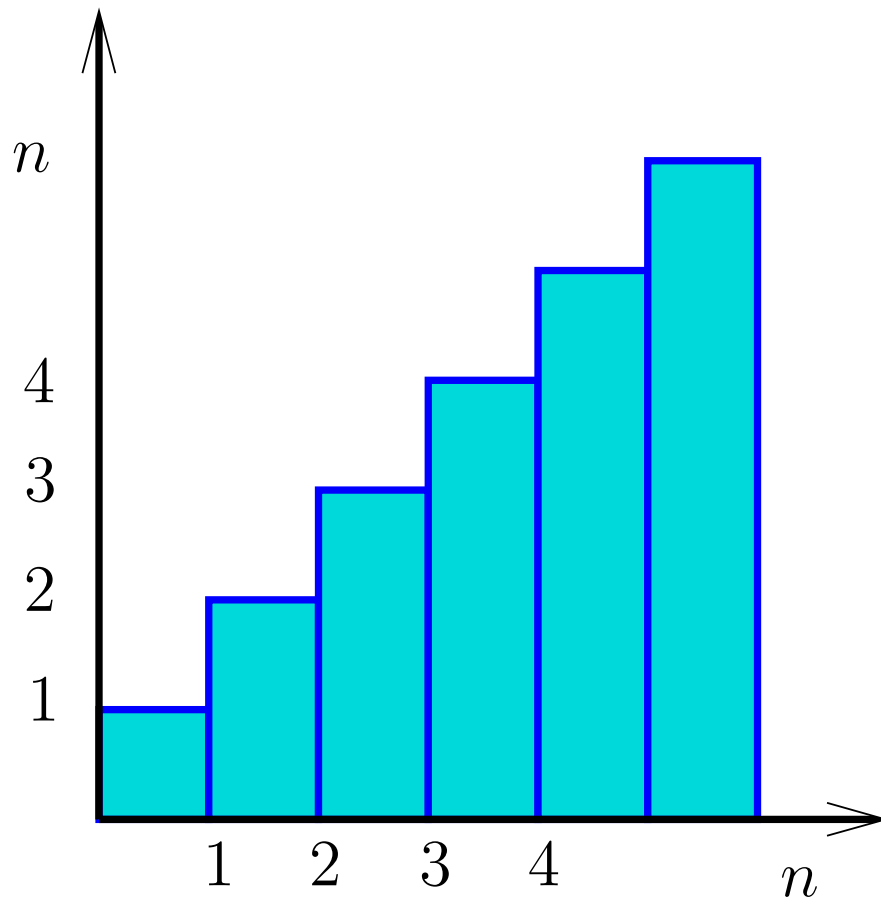
Carl Friedrich Gauss, 1787



$$\frac{n^2}{2} + \frac{n}{2} = \frac{n(n+1)}{2}$$

$$1 + 2 + \dots + (n - 1) + n = ?$$

Carl Friedrich Gauss, 1787



$$(n + 1) \times \frac{n}{2} = \frac{n(n + 1)}{2}$$



# Solução

Se  $n \geq 4$  então no fim da execução das linhas 1–4,

$$\begin{aligned} S &= (n - 1) + (n - 2) + \cdots + 4 + 3 \\ &= (n + 2)(n - 3)/2 \\ &= \frac{1}{2}n^2 - \frac{1}{2}n - 3. \end{aligned}$$

# Ordenação

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

**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

# Ordenação

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

**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*

	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

# Ordenação por inserção

*chave* 1  $j$   $n$

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

*chave* 1  $j$   $n$

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

# Ordenação por inserção

*chave* 1  $j$   $n$

<i>99</i>	20	25	35	38	40	44	55	<i>99</i>	10	65	50
-----------	----	----	----	----	----	----	----	-----------	----	----	----

*chave* 1  $j$   $n$

<i>10</i>	<i>10</i>	20	25	35	38	40	44	55	<i>99</i>	65	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	99	65	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
----	----	----	----	----	----	----	----	----	----	----	----

# 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	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      $j \leftarrow j + 1$

# O algoritmo faz o que promete?

Correção do algoritmo!

# O algoritmo faz o que promete?

Correção do algoritmo!

Relação **invariante** chave:

(i0) na linha 1 vale que:  $A[1..j-1]$  é crescente.

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

# O algoritmo faz o que promete?

Correção do algoritmo!

Relação **invariante** chave:

(i0) na linha 1 vale que:  $A[1 \dots j-1]$  é crescente.

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

Supondo que a invariante vale. Correção do algoritmo é evidente.

No início da última iteração das linhas 1–7 tem-se que  $j = n + 1$ . Da invariante concluí-se que  $A[1 \dots n]$  é crescente.

# Mais invariantes

Na linha 4 vale que:

(i1)  $A[1 \dots i]$  e  $A[i + 2 \dots j]$  são crescentes

(i2)  $A[1 \dots i] \leq A[i + 2 \dots j]$

(i3)  $A[i + 2 \dots j] > chave$

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

# Mais invariantes

Na linha 4 vale que:

(i1)  $A[1 \dots i]$  e  $A[i + 2 \dots j]$  são crescentes

(i2)  $A[1 \dots i] \leq A[i + 2 \dots j]$

(i3)  $A[i + 2 \dots j] > chave$

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

invariantes (i1), (i2) e (i3)

+ condição de parada do **enquanto** da linha 4

+ atribuição da linha 7  $\Rightarrow$  validade (i0)

Demonstre!



# Correção de algoritmos iterativos

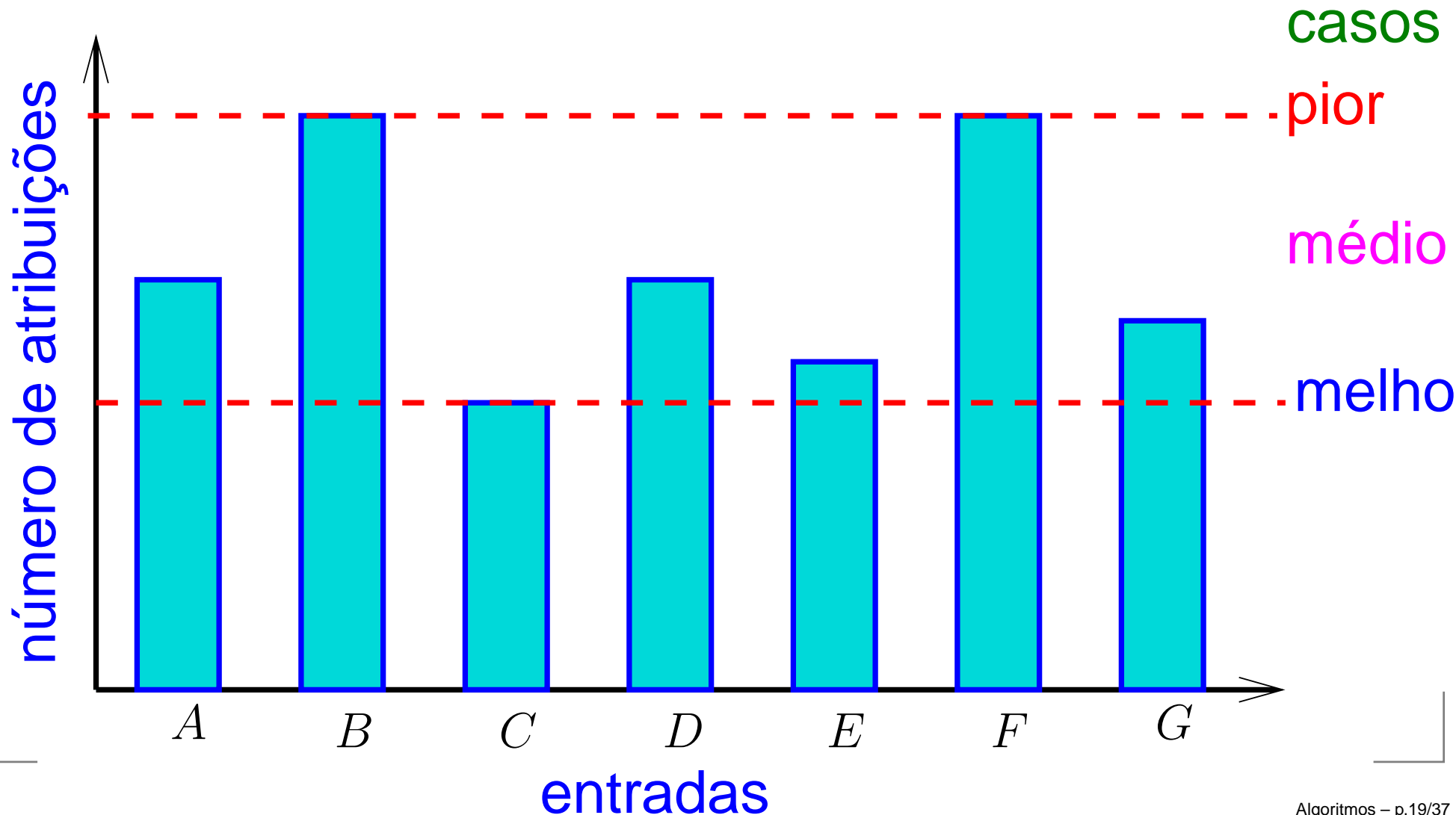
Estrutura “típica” de demonstrações da correção de algoritmos iterativos através de suas relações invariantes consiste em:

1. verificar que a relação **vale no início** da primeira iteração;
2. demonstrar que  
se a relação **vale no início** da iteração, **então** ela **vale no final** da iteração (com os papéis de alguns atores possivelmente trocados);
3. concluir que, se **relação vale** no início da **última iteração**, **então** a **relação junto com a condição** de parada **implicam na correção** do algoritmo.

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

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	?
<b>total</b>	?

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

LINHAS 3–6 ( $A, j, chave$ )

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

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	?
<b>total</b>	?

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

LINHAS 3–6 ( $A, j, chave$ )

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

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	?
<b>total</b>	?

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

LINHAS 3–6 ( $A, j, chave$ )

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

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	$\leq j - 1$

**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  $chave \leftarrow A[j]$
- 3 LINHAS 3–6 ( $A, j, chave$ )
- 7  $A[i + 1] \leftarrow chave$

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



# 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         $chave \leftarrow A[j]$

3        LINHAS 3–6 ( $A, j, chave$ )

7         $A[i + 1] \leftarrow chave$

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	$= n - 1$
4	$= 0$
5	$\leq 1 + 2 + \dots + (n-1) = n(n-1)/2$
6	$\leq 1 + 2 + \dots + (n-1) = n(n-1)/2$
7	$= n - 1$

---

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

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

$n$	$n^2 + 3n - 3$	$n^2$
1	1	1
2	7	4

# $n^2 + 3n - 3$ 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  $j \leftarrow 2$  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	$= n - 1$
<b>total</b>	$\leq (3/2)n^2 + (7/2)n - 4$



# 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           $A[i + 1] \leftarrow A[i]$     ▷ desloca
6           $i \leftarrow i - 1$ 

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

# Solução para $t_i = 1$

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

# 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	$= n - 1$	$\times t_7$
<b>total</b>	$\leq ?$	

# 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	$= n - 1$	$\times t_7$

---

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

# 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	$\leq$	$2 + 3 + \dots + n = (n - 1)(n + 2)/2$	$\times t_4$
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	=	$n - 1$	$\times t_7$

$$\text{total} \leq c_2 \times n^2 + c_1 \times n + c_0$$

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

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

# Exercícios

## Exercício 1.D (bom!)

Quanto vale  $S$  no fim do seguinte algoritmo?

```
1    $S \leftarrow 0$ 
2    $i \leftarrow n$ 
3   enquanto  $i > 0$  faça
4       para  $j \leftarrow 1$  até  $i$  faça
5            $S \leftarrow S + 1$ 
6        $i \leftarrow \lfloor i/2 \rfloor$ 
```

## Exercício 1.E

Prove a invariante do algoritmo **ORDENA-POR-INSERÇÃO**.

## Exercício 1.F

Quantas comparações faz o algoritmo **ORDENA-POR-INSERÇÃO**, no pior caso, ao receber um vetor  $A[1..n]$ ?

## Exercício 1.G

Quantas atribuições faz o algoritmo abaixo?

```
 $s \leftarrow 0$ 
para  $i \leftarrow 1$  até  $n$  faça
     $s \leftarrow s + i$ 
devolva  $s$ 
```

Escreva um algoritmo melhorado que produza o mesmo efeito com menos atribuições.

# Exercícios

## Exercício 1.H (AU 3.7.1)

O algoritmo abaixo opera sobre um vetor  $A[1..n]$ . Quantas atribuições ele faz no pior caso? Quantas comparações?

```
1    $s \leftarrow 0$ 
2   para  $i \leftarrow 1$  até  $n$  faça
3        $s \leftarrow s + A[i]$ 
4    $m \leftarrow s/n$ 
5    $k \leftarrow 1$ 
6   para  $i \leftarrow 2$  até  $n$  faça
7       se  $(A[i] - m)^2 < (A[k] - m)^2$ 
8           então  $k \leftarrow i$ 
9   devolva  $k$ 
```

## Exercício 1.I (AU 3.7.2)

O fragmento abaixo opera sobre uma matriz  $A[1..n, 1..n]$ . Quantas atribuições faz?

```
1   para  $i \leftarrow 1$  até  $n - 1$  faça
2       para  $j \leftarrow i + 1$  até  $n$  faça
3           para  $k \leftarrow i$  até  $n$  faça
4                $A[j, k] \leftarrow A[j, k] - A[i, k] \cdot A[j, i] / A[i, i]$ 
```

# Exercícios

## Exercício 1.J (AU 3.7.3\*)

Quantas atribuições faz o algoritmo?

SUMPOWERSOFTWO ( $n$ )

1      $s \leftarrow 0$

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

3          $j \leftarrow i$

4         **enquanto**  $2 \lfloor j/2 \rfloor = j$  **faça**

5              $j \leftarrow \lfloor j/2 \rfloor$

6              $s \leftarrow s + 1$

7     **devolva**  $s$



# Chão, teto, log etc

CLRS 3.2, A.1

AU 2.9

# Definições

$\lfloor x \rfloor :=$  inteiro  $i$  tal que  $i \leq x < i + 1$

$\lceil x \rceil :=$  inteiro  $j$  tal que  $j - 1 < x \leq j$

Diz-se que

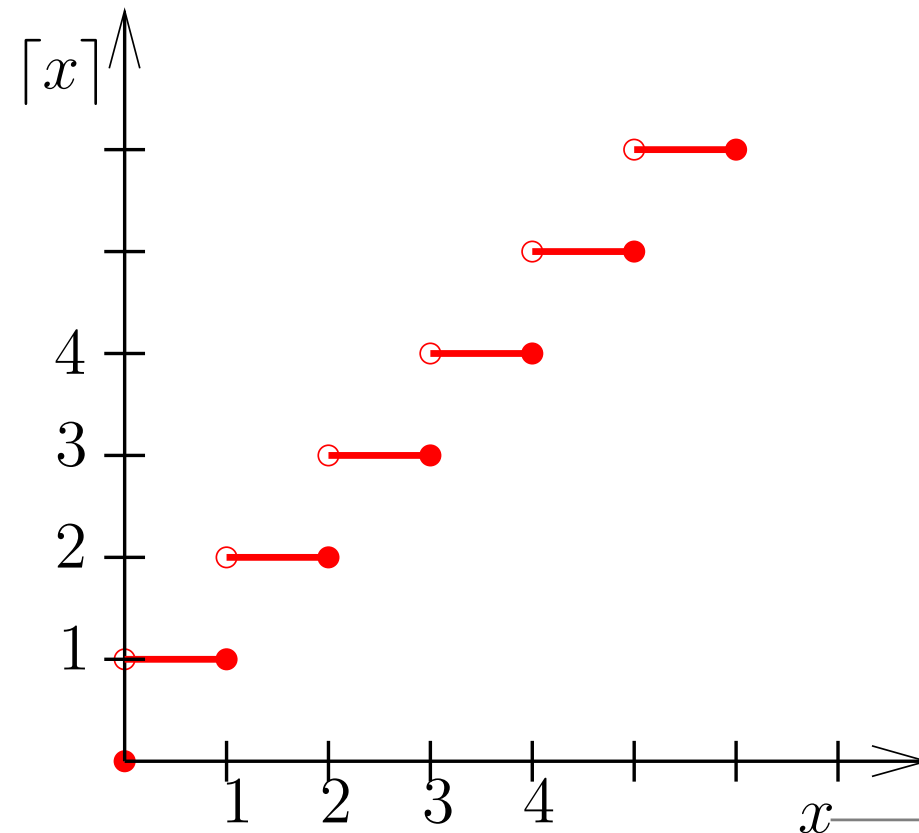
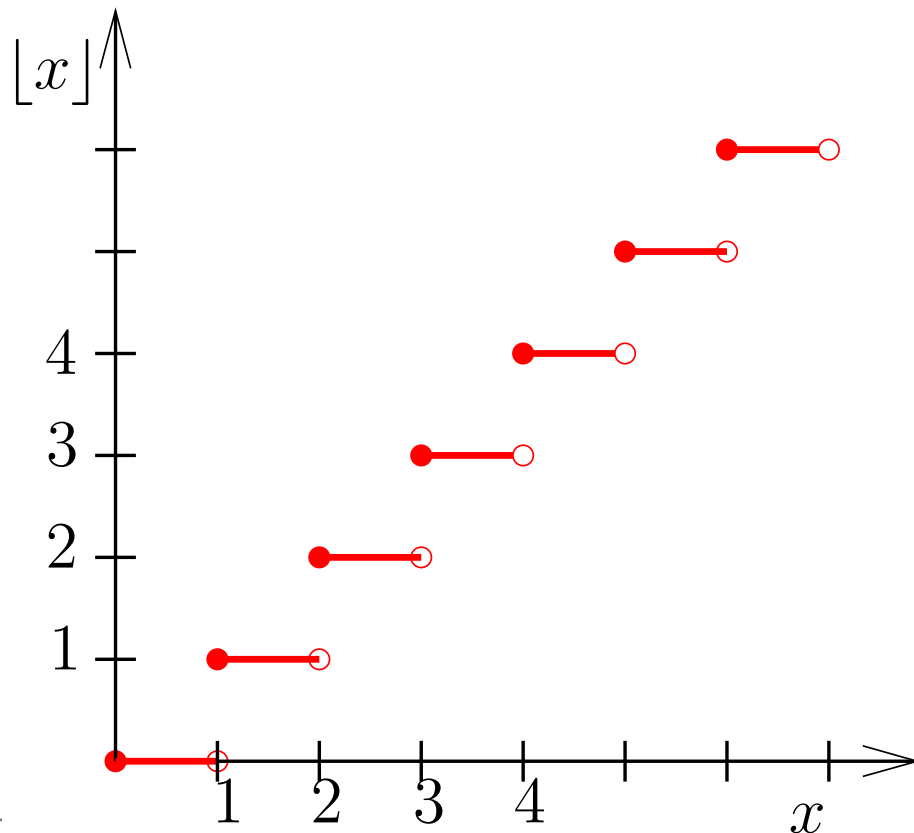
$\lfloor x \rfloor$  é o **chão** de  $x$

$\lceil x \rceil$  é o **teto** de  $x$

# Exercício A1.A

Desenhe os gráficos das funções  $\lfloor x \rfloor$  e  $\lceil x \rceil$  para  $x$  não-negativo.

Solução



# Exercícios

## Exercício A1.B

Mostre que

$$\frac{n-1}{2} \leq \left\lfloor \frac{n}{2} \right\rfloor \leq \frac{n}{2} \quad \text{e} \quad \frac{n}{2} \leq \left\lceil \frac{n}{2} \right\rceil \leq \frac{n+1}{2}$$

para qualquer inteiro  $n \geq 1$ .

## Exercício A1.C

É verdade que  $\lfloor x \rfloor + \lfloor y \rfloor = \lfloor x + y \rfloor$  para quaisquer  $x$  e  $y$ ?

## Exercício A1.D

Desenhe os gráficos das funções  $\lg x$  e  $2^x$  para  $x$  inteiro não-negativo.

# Exercícios

## Exercício A1.E

Explique o significado da expressão  $\log_{3/2} n$ .

## Exercício A1.F

Calcule  $5^{\log_5 n}$ ,  $\log_3 3^n$  e  $\lg 2^n$ .

## Exercício A1.G

Qual a relação entre  $\log_8 n$  e  $\log_2 n$ ?

## Exercício A1.H

Se  $i := \lfloor \lg n \rfloor$ , qual a relação entre  $n$  e  $2^i$ ?

Se  $j := \lceil \lg n \rceil$ , qual a relação entre  $n$  e  $2^j$ ?

## Exercício A1.I

É verdade que  $\lfloor \lg n \rfloor + 1 = \lceil \lg(n+1) \rceil$  para todo inteiro  $n \geq 1$ ?

## Exercício A1.J

Escreva um algoritmo que calcule  $\lfloor \lg n \rfloor$ .

## Exercício A1.L

Mostre que para qualquer número real  $x$  tem-se  $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$ .

## Exercício A1.M

Mostre que  $\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$  para todo inteiro positivo  $n$ .

# Mais exercícios

## Exercício A1.N

Mostre que  $\lfloor x \rfloor + \lfloor y \rfloor \leq \lfloor x + y \rfloor$ , com igualdade se e somente se  $x + y - 1 < \lfloor x \rfloor + \lfloor y \rfloor$ .  
Encontre uma fórmula análoga para  $\lceil \cdot \rceil$ .

## Exercício A1.O

Se  $c$  é inteiro e  $x$  é racional, é verdade que  $\lceil cx \rceil = c \lceil x \rceil$ ?

## Exercício A1.P

Use a notação  $\lfloor \cdot \rfloor$  para representar o resto da divisão de  $n$  por 7.

## Exercício A1.Q

É verdade que  $\lceil 2 \lceil 2n/3 \rceil / 3 \rceil = \lceil 4n/9 \rceil$ ? É verdade que  $\lfloor 2 \lfloor 2n/3 \rfloor / 3 \rfloor = \lfloor 4n/9 \rfloor$ ?

## Exercício A1.R

É verdade que  $\lfloor \lfloor n/2 \rfloor / 2 \rfloor = \lfloor n/4 \rfloor$ ?

## Exercício A1.S

Se  $n, a, b$  são inteiros positivos, é verdade que  $\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor$ ?

## Exercício A1.T

É verdade que  $\lfloor \lg n \rfloor \geq \lg(n-1)$  para todo inteiro  $n \geq 2$ ? É verdade que  $\lceil \lg n \rceil \leq \lg(n+1)$  para todo inteiro  $n \geq 1$ ?

# Mais exercícios ainda

## Exercício A1.U

Prove que para qualquer número racional  $x > 1$  tem-se

$$\lfloor \lg x \rfloor \leq \lg \lfloor x \rfloor \leq \lg x \leq \lg \lceil x \rceil \leq \lceil \lg x \rceil$$

## Exercício A1.V

Quanto vale  $1/2 + 1/4 + 1/8 + \dots + 1/2^n + \dots$ ?