

# ***Fundamentos de Análise de Algoritmos***

**(GABARITO)**

Unidade I: Análise de Algoritmos

# Exercício (1)

- Encontre o maior e menor valores em um *array* de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

# Exercício (1) - Versão 1

- Encontre o maior e menor valores em um *array* de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

```
void maxMin1 () {  
    max = min = array[0];  
    for (int i = 1; i < n; i++) {  
        if (array[i] > max) max = array[i];  
        if (array[i] < min) min = array[i];  
    }  
}
```

Todos os casos:

$$f(n) = 2(n - 1)$$

# Exercício (1) - Versão 2

- Encontre o maior e menor valores em um *array* de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

```
void maxMin2 () {  
    max = min = array[0];  
    for (int i = 1; i < n; i++) {  
        if (array[i] > max) max = array[i];  
        else if (array[i] < min) min = array[i];  
    }  
}
```

**Melhor caso:** elementos em ordem crescente  
 $f(n) = n - 1$

**Pior caso:** elementos em ordem decrescente  
 $f(n) = 2(n - 1)$

**Caso médio:**  $array[i] > max$  em metade das vezes  
 $f(n) = 3n/2 - 3/2$

# Exercício (1) - Versão 3

- Encontre o maior e menor valores em um *array* de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

```
void maxMin3 () {  
    int inicio = 1;  
    if ((n % 2) == 0) {  
        if (array[0] > array[1]) {  
            max = array[0]; min = array[1];  
        } else {  
            max = array[1]; min = array[0];  
        } inicio = 2;  
    } else {  
        max = min = array[0];  
    }  
}
```

```
...  
for (int i = inicio; i < n-1; i += 2) {  
    if (array[i] > array[i+1]) {  
        if (array[i] > max) max = array[i];  
        if (array[i+1] < min) min = array[i+1];  
    } else {  
        if (array[i] < min) min = array[i];  
        if (array[i+1] > max) max = array[i+1];  
    }  
}
```

Todos os casos: (n é ímpar)

$$f(n) = \frac{n-2}{2} + \frac{n-2}{2} + \frac{n-2}{2} = \frac{3(n-2)}{2}$$

# Exercício (2)

- Considerando o problema de encontrar o maior e menor valores em um *array*, veja os quatro códigos propostos e analisados no livro do Ziviani
- **Resposta:** Apresentamos as três primeiras versões no Exercício 1. A quarta versão utiliza o paradigma de projeto Dividir e Conquistar e será abordada mais na frente

## Exercício (3)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n \cdot \lg(n))$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(n^5)$	$\Theta(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \cdot \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

## Exercício (3)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n \cdot \lg(n))$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(n^5)$	$\Theta(n^{20})$
$f(n) = \lg(n)$		X						
$f(n) = n \cdot \lg(n)$				X				
$f(n) = 5n + 1$			X					
$f(n) = 7n^5 - 3n^2$							X	
$f(n) = 99n^3 - 1000n^2$						X		
$f(n) = n^5 - 99999n^4$							X	



## Exercício (4)

- Preencha verdadeiro ou falso na tabela abaixo:

	$O(1)$	$O(\lg n)$	$O(n)$	$O(n \cdot \lg(n))$	$O(n^2)$	$O(n^3)$	$O(n^5)$	$O(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \cdot \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

## Exercício (4)

- Preencha verdadeiro ou falso na tabela abaixo:

	$O(1)$	$O(\lg n)$	$O(n)$	$O(n \cdot \lg(n))$	$O(n^2)$	$O(n^3)$	$O(n^5)$	$O(n^{20})$
$f(n) = \lg(n)$		X	X	X	X	X	X	X
$f(n) = n \cdot \lg(n)$				X	X	X	X	X
$f(n) = 5n + 1$			X	X	X	X	X	X
$f(n) = 7n^5 - 3n^2$							X	X
$f(n) = 99n^3 - 1000n^2$						X	X	X
$f(n) = n^5 - 99999n^4$							X	X

## Exercício (5)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Omega(1)$	$\Omega(\lg n)$	$\Omega(n)$	$\Omega(n \cdot \lg(n))$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^5)$	$\Omega(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \cdot \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

## Exercício (5)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Omega(1)$	$\Omega(\lg n)$	$\Omega(n)$	$\Omega(n \cdot \lg(n))$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^5)$	$\Omega(n^{20})$
$f(n) = \lg(n)$	X	X						
$f(n) = n \cdot \lg(n)$	X	X	X	X				
$f(n) = 5n + 1$	X	X	X					
$f(n) = 7n^5 - 3n^2$	X	X	X	X	X	X	X	
$f(n) = 99n^3 - 1000n^2$	X	X	X	X	X	X		
$f(n) = n^5 - 99999n^4$	X	X	X	X	X	X	X	

## Exercício (6)

- Dada a definição da notação  $\Omega$ :
  - a) Mostre os valores de  $c$  e  $m$  tal que, para  $n \geq m$ ,  $|g(n)| \geq c \times |f(n)|$ , provando que  $3n^2 + 5n + 1$  é  $\Omega(n^2)$
  - b) Mostre os valores de  $c$  e  $m$  tal que, para  $n \geq m$ ,  $|g(n)| \geq c \times |f(n)|$ , provando que  $3n^2 + 5n + 1$  é  $\Omega(n)$
  - c) Prove que  $3n^2 + 5n + 1$  **não é**  $\Omega(n^3)$

## Exercício (6)

- Dada a definição da notação  $\Omega$ :
  - a) Mostre os valores de  $c$  e  $m$  tal que, para  $n \geq m$ ,  $|g(n)| \geq c \times |f(n)|$ , provando que  $3n^2 + 5n + 1$  é  $\Omega(n^2) \Rightarrow c = 3$  e  $m = 1$
  - b) Mostre os valores de  $c$  e  $m$  tal que, para  $n \geq m$ ,  $|g(n)| \geq c \times |f(n)|$ , provando que  $3n^2 + 5n + 1$  é  $\Omega(n) \Rightarrow c = 3$  e  $m = 1$
  - c) Prove que  $3n^2 + 5n + 1$  **não é**  $\Omega(n^3) \Rightarrow$  Não existe par  $(c, m)$  tal que para  $n \geq m$ ,  $|3n^2 + 5n + 1| \geq c \times |n^3|$  seja verdadeira. Aumentando o valor de  $c$ , apenas retardamos o momento em que a curva cúbica supera a quadrática

## Exercício (7)

- Dada a definição da notação  $\Theta$ :
  - a) Mostre um valor para  $c_1$ ,  $c_2$  e  $m$  tal que, para  $n \geq m$ ,  $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$ , provando que  $3n^2 + 5n + 1$  é  $\Theta(n^2)$
  - b) Prove que  $3n^2 + 5n + 1$  **não** é  $\Theta(n)$
  - c) Prove que  $3n^2 + 5n + 1$  **não** é  $\Theta(n^3)$

## Exercício (7)

- Dada a definição da notação  $\Theta$ :
  - a) Mostre um valor para  $c_1$ ,  $c_2$  e  $m$  tal que, para  $n \geq m$ ,  $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$ , provando que  $3n^2 + 5n + 1$  é  $\Theta(n^2) \Rightarrow c_1 = 3, c_2 = 4$  e  $m = 5.2$
  - b) Prove que  $3n^2 + 5n + 1$  **não** é  $\Theta(n)$   $\Rightarrow$  Não existe par  $(c_2, m)$  tal que para  $n \geq m$ ,  $|3n^2 + 5n + 1| \leq c_2 \times |n|$  seja verdadeira. Aumentando o valor de  $c_2$ , apenas retardamos o momento em que a curva quadrática supera a linear
  - c) Prove que  $3n^2 + 5n + 1$  **não** é  $\Theta(n^3)$   $\Rightarrow$  Não existe par  $(c_1, m)$  tal que para  $n \geq m$ ,  $|3n^2 + 5n + 1| \geq c_1 \times |n^3|$  seja verdadeira. Aumentando o valor de  $c_1$ , apenas retardamos o momento em que a curva cúbica supera a quadrática



# Exercício (8)

- Suponha um sistema de monitoramento contendo os métodos telefone, luz, alarme, sensor e câmera, apresente a função e ordem de complexidade para o **pior** e **melhor** caso: (a) **método alarme**; (b) **outros métodos**.

```
void sistemaMonitoramento() {
    alarme(((telefone() == true && luz() == true)) ? 0 : 1);
    for (int i = 2; i < n; i++){
        if (sensor(i- 2) == true){
            alarme (i - 2);
        } else if (camera(i- 2) == true){
            alarme (i - 2 + n);
        }
    }
}
```

# Exercício (8)

- Suponha um sistema de monitoramento contendo os métodos telefone, luz, alarme, sensor e câmera, apresente a função e ordem de complexidade para o **pior** e **melhor** caso: (a) **método alarme**; (b) **outros métodos**.

```
void sistemaMonitoramento() {  
    alarme(((telefone() == true && luz() == true)) ? 0 : 1);  
    for (int i = 2; i < n; i++){  
        if (sensor(i- 2) == true){  
            alarme (i - 2);  
        } else if (camera(i- 2) == true){  
            alarme (i - 2 + n);  
        }  
    }  
}
```

Ordem de Complexidade  
alarme()

Pior  $O(n)$ ,  $\Omega(n)$  e  $\Theta(n)$

Melhor  $O(1)$ ,  $\Omega(1)$  e  $\Theta(1)$

Função de Complexidade  
alarme()

Pior  $f(n) = 1 + (n-2)$

Melhor  $f(n) = 1 + (n-2) \times 0$

# Exercício (8)

- Suponha um sistema de monitoramento contendo os métodos telefone, luz, alarme, sensor e câmera, apresente a função e ordem de complexidade para o **pior** e **melhor** caso: (a) **método alarme**; (b) **outros métodos**.

```
void sistemaMonitoramento() {  
    alarme(((telefone() == true && luz() == true)) ? 0 : 1);  
    for (int i = 2; i < n; i++){  
        if (sensor(i- 2) == true){  
            alarme (i - 2);  
        } else if (camera(i- 2) == true){  
            alarme (i - 2 + n);  
        }  
    }  
}
```

Ordem de Complexidade  
outros métodos

Pior

$O(n)$ ,  $\Omega(n)$  e  $\Theta(n)$

Melhor

Função de Complexidade  
outros métodos

Pior

$f(n) = 2 + 2x(n-2)$

Melhor

$f(n) = 2 + (n-2)$

# Exercício (9)

- Apresente um código, defina duas operações relevantes e apresente a função e a complexidade para as operações escolhidas no pior e melhor caso

# Exercício (9)

- Apresente um código, defina duas operações relevantes e apresente a função e a complexidade para as operações escolhidas no pior e melhor caso

**RESPOSTA:** Cada aluno terá uma resposta distinta

# Exercício (10)

- Anteriormente, verificamos que quando desejamos pesquisar a existência de **um** elemento em um *array* de números reais é adequado executar uma pesquisa sequencial cujo custo é  $\Theta(n)$ . Nesse caso, o custo de ordenar o *array* e, em seguida, aplicar uma pesquisa binária é mais elevado,  $\Theta(n \times \lg(n)) + \Theta(\lg(n)) = \Theta(n \times \lg(n))$ . Agora, supondo que desejamos efetuar  $n$  pesquisas, responda qual das duas soluções é mais eficiente

# Exercício (10)

- Anteriormente, verificamos que quando desejamos pesquisar a existência de **um** elemento em um *array* de números reais é adequado executar uma pesquisa sequencial cujo custo é  $\Theta(n)$ . Nesse caso, o custo de ordenar o *array* e, em seguida, aplicar uma pesquisa binária é mais elevado,  $\Theta(n \times \lg(n)) + \Theta(\lg(n)) = \Theta(n \times \lg(n))$ . Agora, supondo que desejamos efetuar  $n$  pesquisas, responda qual das duas soluções é mais eficiente

**RESPOSTA:** As  $n$  pesquisas sequenciais terão o custo de  $n \times \Theta(n) = \Theta(n^2)$  e a ordenação mais as  $n$  pesquisas binárias,  $\Theta(n \times \lg(n)) + n \times \Theta(\lg(n)) = \Theta(n \times \lg(n))$ . Logo, a segunda solução é mais indicada