

## Aula 03

### *Engenharia da Computação – 3ª série*

## *Função de Complexidade de Algoritmos* *(E1, E2)*

**2025**

## Função de Complexidade

### Pergunta

- O que é Função de Complexidade de Algoritmos?



## Função de Complexidade

### Resposta



- Função de Complexidade:
  - ✓ Para medir o custo de execução de um algoritmo é comum definir uma função de custo ou função de complexidade  $f$ ;
  - ✓ Função de complexidade de tempo:  $f(n)$  mede o tempo necessário para executar um algoritmo em um problema de tamanho  $n$ ;
  - ✓ Função de complexidade de espaço:  $f(n)$  mede a memória necessária para executar um algoritmo em um problema de tamanho  $n$ ;

## Função de Complexidade

### Resposta



- Função de Complexidade:
  - ✓ A função de complexidade na realidade não representa tempo diretamente, mas o número de vezes que determinada operação, considerada relevante, é executada.

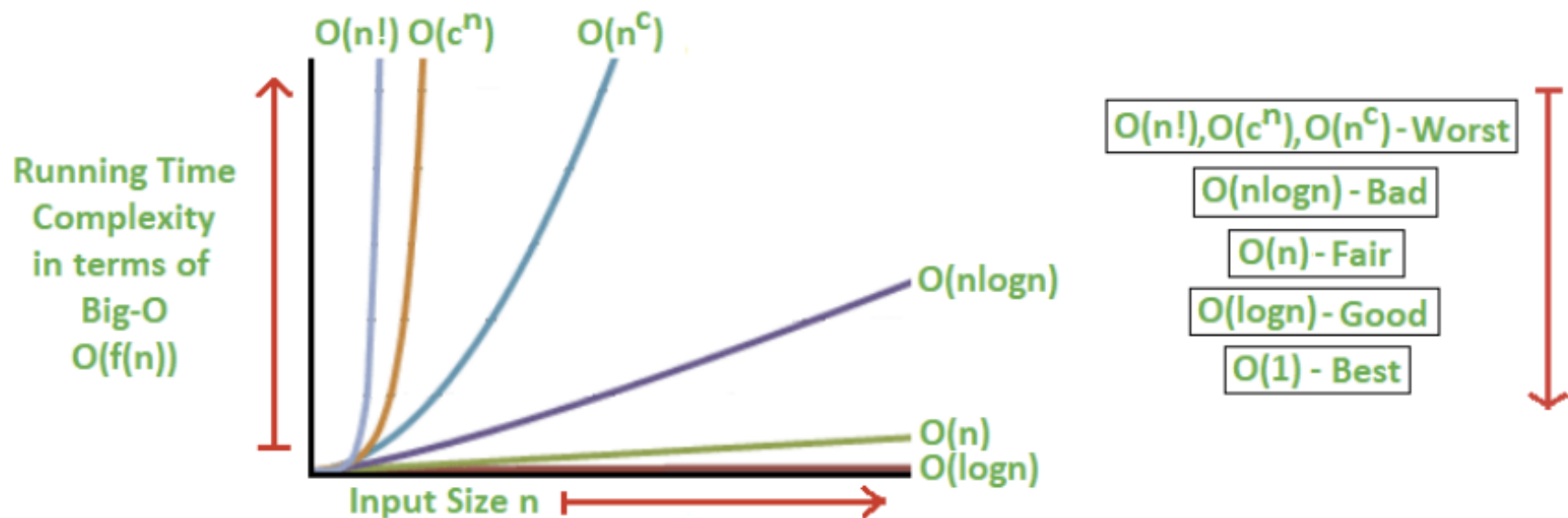
# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Resposta



- Função de Complexidade:



## Função de Complexidade

### Exemplo 1



- Usando o modelo simplificado de *Knuth*, apresente a equação que define o tempo de processamento do algoritmo que calcula a somatória de uma série aritmética simples, ou seja, a Função de Complexidade que relaciona a entrada de dados ( $n$ ) com a quantidade de operações executadas pelo algoritmo para tratamento desta entrada  $n$ :

$$\sum_{i=1}^n i$$

### Exemplo 1



- O algoritmo pode ser implementado pelo código abaixo:

```
1 import java.util.Scanner;
2
3 public class Somatoria {
4     public static int soma(int n) {
5         int resultado = 0;
6         for(int i = 1; i <= n; i++) {
7             resultado = resultado + i;
8         }
9         return resultado;
10    }
11
12    public static void main(String args[]) {
13        Scanner input = new Scanner(System.in);
14        int n = input.nextInt();
15        int resposta = soma(n);
16        System.out.println(resposta);
17    }
18 }
```




# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



```
1 import java.util.Scanner;
2
3 public class Somatoria {
4     public static int soma(int n) {
5         int resultado = 0;
6         for(int i = 1; i <= n; i++) {
7             resultado = resultado + i;
8         }
9         return resultado;
10    }
11
12    public static void main(String args[]) {
13        Scanner input = new Scanner(System.in);
14        int n = input.nextInt();
15        int resposta = soma(n);
16        System.out.println(resposta);
17    }
18 }
```

L1    

L2    5        int resultado = 0;

L3    6        for(int i = 1; i <= n; i++) {

L4    7           resultado = resultado + i;

      8        }

L5    9        return resultado;

     10     }

     11

     12     public static void main(String args[]) {

     13         Scanner input = new Scanner(System.in);

     14         int n = input.nextInt();

     15         int resposta = soma(n);

     16         System.out.println(resposta);

     17     }

     18 }



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1

L1 → 15

```
int resposta = soma(n);
```



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1 → 15

```
int resposta = soma(n);
```

- 1 operação: operação para se recuperar a variável *n*;

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1    15

```
int resposta = soma(n);
```

- 1 operação: operação para se recuperar a variável ***n***;
- 1 operação: operação para passagem do parâmetro ***n*** ao método ***soma( )***;

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1    15

```
int resposta = soma(n);
```

- 1 operação: operação para se recuperar a variável ***n***;
- 1 operação: operação para passagem do parâmetro ***n*** ao método ***soma( )***;
- 1 operação: operação para invocar o método ***soma( )***;

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1    15   `int resposta = soma(n);`

- 1 operação: operação para se recuperar a variável ***n***;
- 1 operação: operação para passagem do parâmetro ***n*** ao método ***soma( )***;
- 1 operação: operação para invocar o método ***soma( )***;
- $\Sigma \text{ op}_{\text{soma}(n)}$ : total de operações processadas pelo método ***soma( )***;

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1    15   `int resposta = soma(n);`

- 1 operação: operação para se recuperar a variável ***n***;
- 1 operação: operação para passagem do parâmetro ***n*** ao método ***soma( )***;
- 1 operação: operação para invocar o método ***soma( )***;
- $\Sigma \text{ op}_{\text{soma}(n)}$ : total de operações processadas pelo método ***soma( )***;
- 1 operação: operação de retorno do método ***soma( )***;

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1    15   `int resposta = soma(n);`

- 1 operação: operação para se recuperar a variável ***n***;
- 1 operação: operação para passagem do parâmetro ***n*** ao método ***soma( )***;
- 1 operação: operação para invocar o método ***soma( )***;
- $\Sigma$  op ***soma(n)***: total de operações processadas pelo método ***soma( )***;
- 1 operação: operação de retorno do método ***soma( )***;
- 1 operação: operação de atribuição na variável ***resposta***.



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1    15   `int resposta = soma(n);`

- 1        operação para se recuperar a variável *n*;
- 1        operação para passagem do parâmetro *n* ao método *soma( )*;
- 1        operação para invocar o método *soma( )*;
- $\sum \text{op}_{soma(n)}$     total de operações processadas pelo método *soma( )*;
- 1        operação de retorno do método *soma( )*;
- 1        operação de atribuição na variável *resposta*.

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L1 → 15

```
int resposta = soma(n);
```


$$5 + \sum \text{op}_{\text{soma}(n)}$$

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L2 


```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L2 

```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```

**2 operações**

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L3a



```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L3a



```
4      public static int soma(int n) {  
5          int resultado = 0;  
6          for(int i = 1; i <= n; i++) {  
7              resultado = resultado + i;  
8          }  
9          return resultado;  
10     }
```

Código	Tempo
<b>int i = 1</b>	<b>2 operações</b>

- Este tempo corresponde à primeira parte do código for que representa a etapa de inicialização;
- É executado uma única vez antes da primeira iteração do loop.



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L3b



```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L3b



```
4      public static int soma(int n) {  
5          int resultado = 0;  
6          for(int i = 1; i <= n; i++) {  
7              resultado = resultado + i;  
8          }  
9          return resultado;  
10     }
```

Código	Tempo
$i \leq n$	$3 \times (n+1)$ operações

- Este tempo corresponde ao teste de término do loop;
- É executado antes do início de cada iteração do loop;
- O número de vezes em que o teste de término do loop é feito é um a mais que o número de vezes em que o corpo do loop é executado.

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L3c



```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```

### Exemplo 1



L3c



```
4      public static int soma(int n) {  
5          int resultado = 0;  
6          for(int i = 1; i <= n; i++) {  
7              resultado = resultado + i;  
8          }  
9          return resultado;  
10     }
```

Código	Tempo
<b>++i</b>	<b>4 x (n) operações</b>

- Este tempo corresponde ao terceiro elemento do for, o passo de incremento do contador do loop. Equivale a **i = i + 1;**
- É executado uma vez a cada iteração do loop. Portanto, n vezes.

### Exemplo 1



L4



```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L4



```
4      public static int soma(int n) {  
5          int resultado = 0;  
6          for(int i = 1; i <= n; i++) {  
7              resultado = resultado + i;  
8          }  
9          return resultado;  
10     }
```

Código	Tempo
<b>resultado += i</b>	<b>4 x (n) operações</b>

- **resultado = resultado + i;**
- Este tempo corresponde ao corpo do loop;
- É executado n vezes.



## Função de Complexidade

### Exemplo 1



L5


```
4 public static int soma(int n) {  
5     int resultado = 0;  
6     for(int i = 1; i <= n; i++) {  
7         resultado = resultado + i;  
8     }  
9     return resultado;  
10 }
```

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Função de Complexidade

### Exemplo 1



L5 

```
4      public static int soma(int n) {  
5          int resultado = 0;  
6          for(int i = 1; i <= n; i++) {  
7              resultado = resultado + i;  
8          }  
9          return resultado;  
10     }
```

Código	Tempo
<b>return resultado;</b>	<b>2 operações</b>

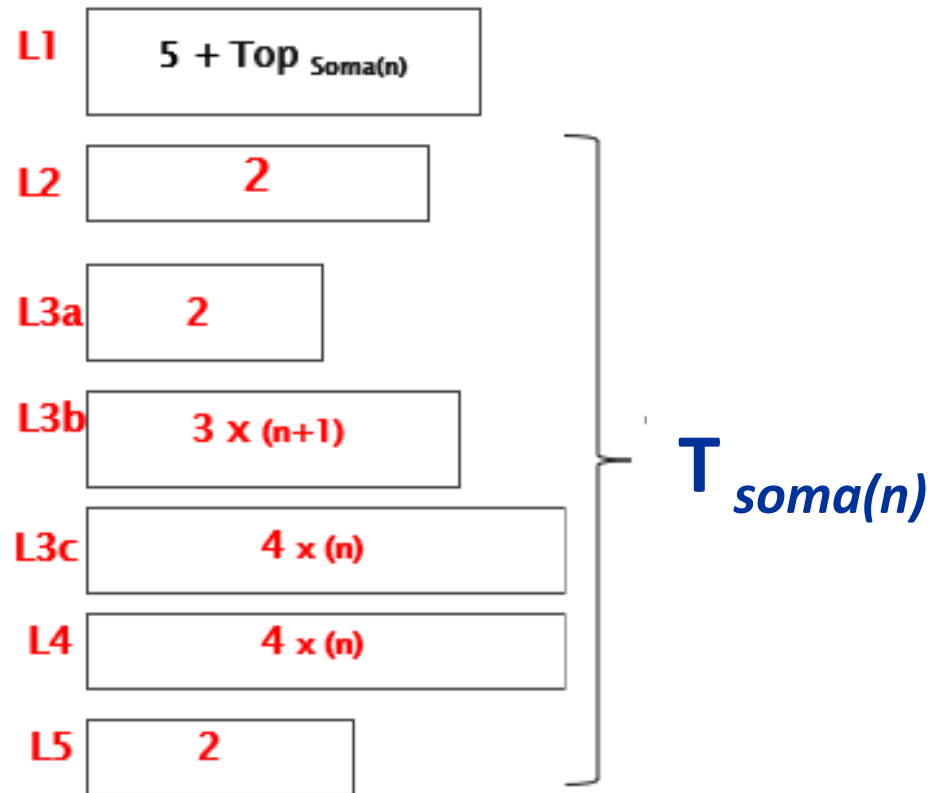
- Este tempo corresponde ao retorno da variável resultado;
- A variável é lida na memória e armazenada na pilha (registro de ativação).



### Exemplo 1



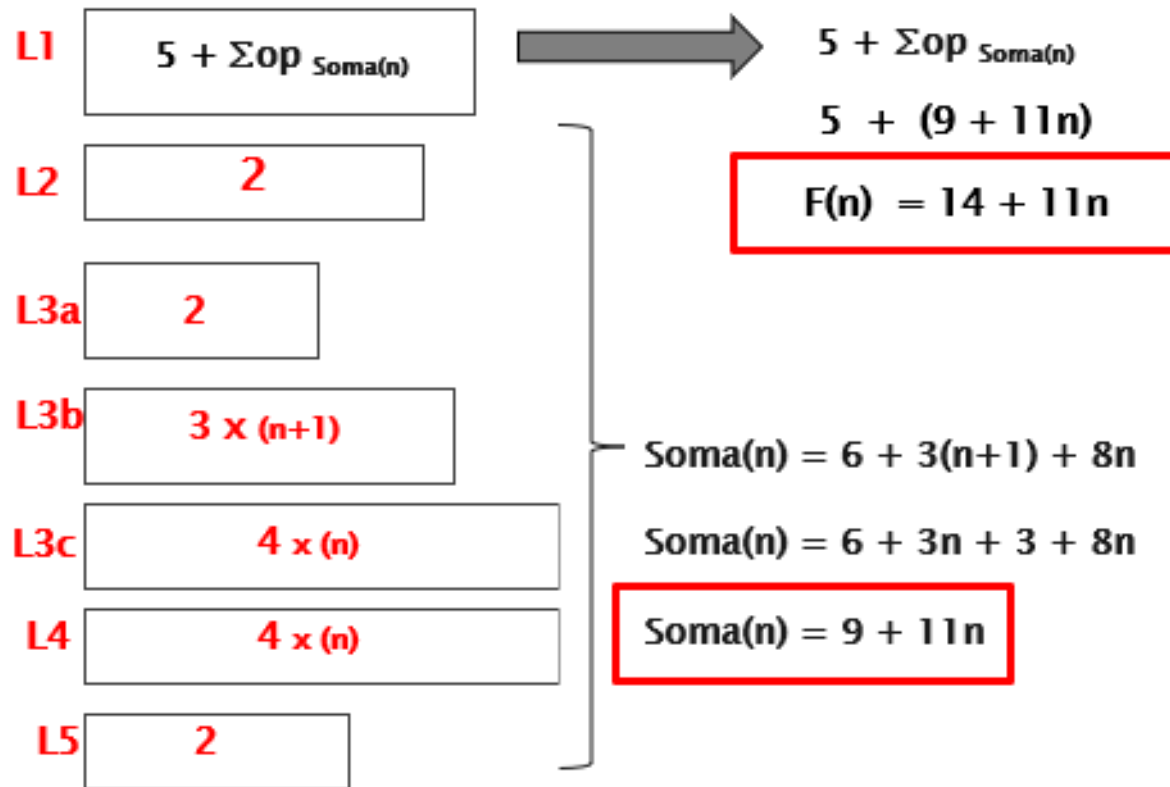
- Contagem Total de Operações:



### Exemplo 1



- Contagem Total de Operações:

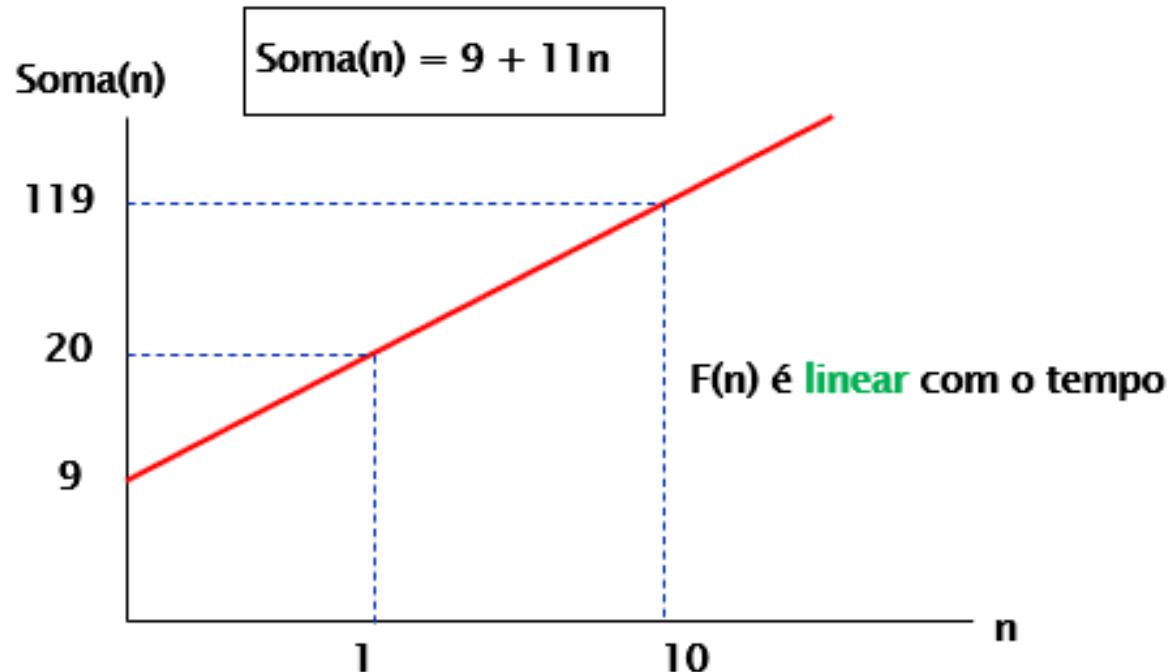


## Função de Complexidade

### Exemplo 1



- A ordem de complexidade do algoritmo é **LINEAR**.
- Ordem de complexidade é  **$O(n)$** .



## Função de Complexidade

### Exemplo 1



- Função de Complexidade:
  - ✓ Para medir o custo de execução de um algoritmo é comum definir uma função de custo ou função de complexidade  $f$ ;
  - ✓ Função de complexidade de **tempo**:  $f(n)$  mede o **tempo** necessário para executar um algoritmo em um problema de tamanho  $n$ ;
  - ✓ Função de complexidade de **espaço**:  $f(n)$  mede a memória necessária para executar um algoritmo em um problema de tamanho  $n$ ;

## Função de Complexidade

### Exemplo 1



- Função de Complexidade:
  - ✓ A função de **tempo** na realidade **não** representa tempo diretamente, mas o **número de vezes que determinada operação**, considerada relevante, é executada.

### Exemplo 2



- Maior elemento de um **array** de inteiros:
  - ✓ Seja V um array de n elementos inteiros,  $n \geq 1$ :

```
public class Max {  
  
    public static void main(String[] args) {  
  
        int[] V = { 2,5,6,4,2,10,2,3,5,8,1,2 };  
        // vetor tem 12 elementos  
  
        int Max=V[0],contador = 0;  
  
        for (int i=1; i< V.length; i++) {  
            if(V[i] > Max) Max = V[i]; contador++;  
        }  
  
        System.out.println("Maior Valor do Vetor V = " + Max);  
        System.out.println("Contador = " + contador );  
    }  
}
```

### Exemplo 2



- Maior elemento de um **array** de inteiros:
  - ✓ Trace de Execução:

Maior Valor do Array  $V = 10$   
Contador = 11

- ✓ O vetor tem 12 elementos e foram executadas **11** comparações
- ✓ Se o array tivesse 1000 elementos seriam necessárias 999 comparações
- ✓ Assim, se o array tiver **n** elementos, serão executadas **n-1** comparações . . .



### Exemplo 2



- Maior elemento de um **array** de inteiros:
  - Seja  $f$  um função de complexidade tal que  $f(n)$  corresponda ao número de comparações entre os elementos de  $V$ , considerando  $V$  com  $n$  elementos;
  - $f(n) = n - 1$ , para  $n > 0$ ;
  - Logo,  $n - 1$  comparações são necessárias.

## Função de Complexidade

### Exemplo 2



- Maior elemento de um **array** de inteiros:
  - ✓ Função Complexidade:

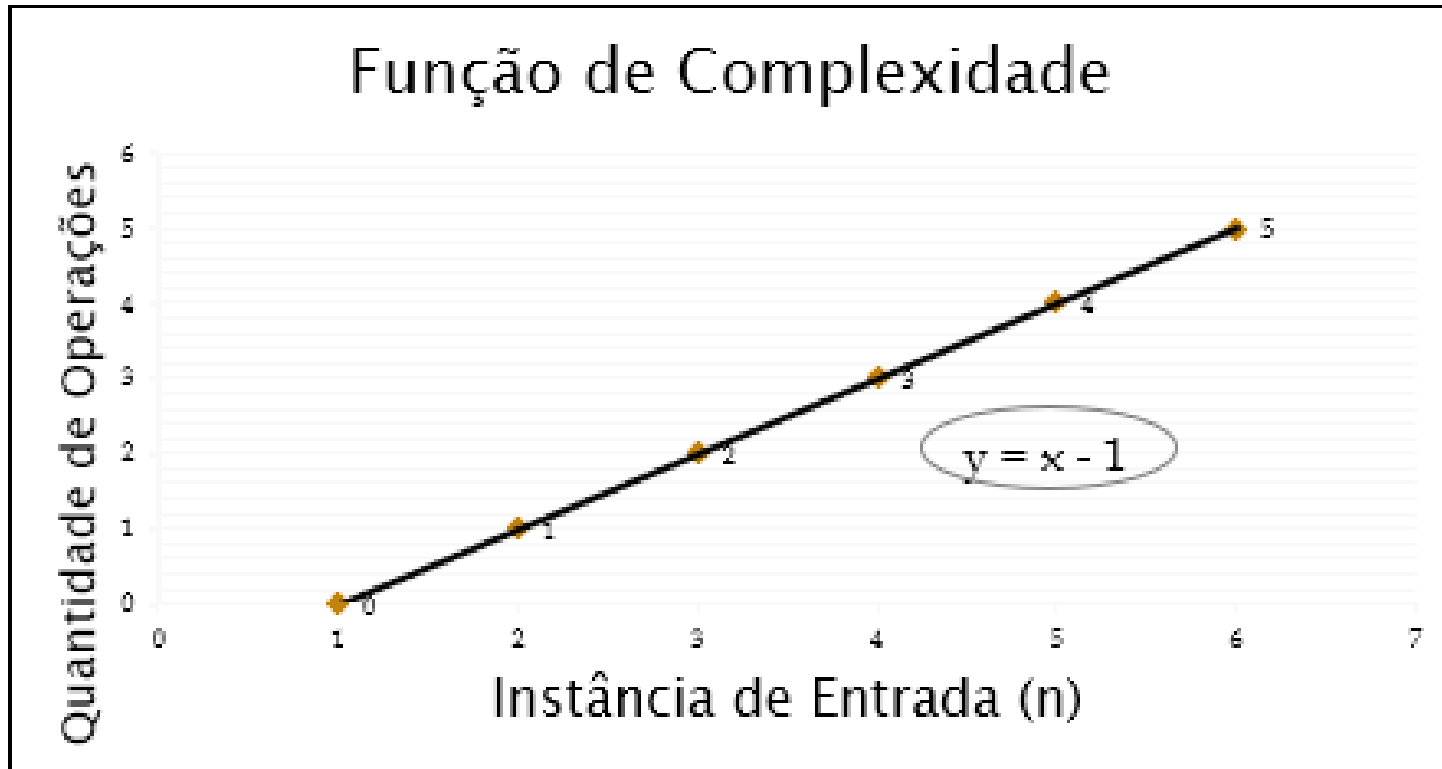
n	$f(n) = n - 1$
1	0
2	1
3	2
4	3
10	9
20	19

## Função de Complexidade

### Exemplo 2



- Maior elemento de um **array** de inteiros:



## Função de Complexidade

### Tamanho da entrada de dados

- No caso do método **max** do programa do exemplo 2, o custo é proporcional à entrada de dados submetida ao algoritmo;
- Já para um algoritmo de ordenação, isso não ocorre: se os dados de entrada já estiverem quase ordenados, então o algoritmo irá trabalhar menos.

### Exemplo 3



- Registros de um arquivo:
  - Cada registro contém uma **chave única**, que é utilizada para recuperar registros do arquivo, sem ordenação;
  - Dada uma chave qualquer, o problema consiste em localizar o registro que contenha esta chave;
  - O algoritmo de pesquisa mais simples que existe é o que faz uma **pesquisa sequencial**;
  - Este algoritmo examina os registros na ordem em que eles aparecem no arquivo, até que o registro procurado seja encontrado ou pode ser que o registro não exista.

### Exemplo 3



- Registros de um arquivo:
  - Seja  $f$  uma função de complexidade tal que  $f(n)$  é o número de registros consultados no arquivo, isto é, o número de vezes que a chave de consulta é comparada com a chave de cada registro;
  - Melhor caso:  $f(n) = 1$ , onde o registro procurado é o primeiro;
  - Pior caso:  $f(n) = n$ , onde o registro procurado é o último ou o registro não existe.
  - Qual seria a função de complexidade para o caso médio?

### Exemplo 3



- Registros de um arquivo – Análise do Caso Médio:
  - No estudo do caso médio, considera-se que toda pesquisa recupera um registro, não existindo a pesquisa sem sucesso;
  - Se  $p_i$  for a probabilidade de que o  $i$ -ésimo registro seja procurado e considerando que para recuperar o  $i$ -ésimo registro são necessárias  $i$  comparações, então:

$$f(n) = 1 \times p_1 + 2 \times p_2 + 3 \times p_3 + \dots + n \times p_n$$

- Para calcular  $f(n)$  basta conhecer a distribuição de probabilidades  $p_i$ .



### Exemplo 3



- Registros de um arquivo – Análise do Caso Médio:

$$f(n) = 1 \times p_1 + 2 \times p_2 + 3 \times p_3 + \dots + n \times p_n$$

- Se cada registro tiver a mesma probabilidade de ser acessado que todos os outros, então:

$$p_i = 1/n, \quad 0 \leq i < n$$

- Neste caso:  $f(n) = 1/n (1+2+3+\dots+n) = 1/n \cdot n(n+1)/2$

- Assim,  $f(n) = (n + 1) / 2$

- A análise do caso esperado revela que uma pesquisa com sucesso examina aproximadamente **metade dos registros**.

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Referências bibliográficas

- CORMEN, T.H. et al. Algoritmos: Teoria e Prática (Caps. 13). Campus. 2002.
- ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C (Cap. 1). 2.ed. Thomson, 2004.
- FEOFILOFF, P. Minicurso de Análise de Algoritmos, 2010. Disponível em:  
<http://www.ime.usp.br/~pf/livrinho-AA/>
- DOWNEY, A.B. *Analysis of algorithms* (Cap. 2), Em: *Computational Modeling and Complexity Science*. Disponível em:  
<http://www.greenteapress.com/compmo/html/book003.html>
- ROSA, J.L. Notas de Aula de Introdução a Ciência de Computação II. Universidade de São Paulo. Disponível em:  
<http://coteia.icmc.usp.br/mostra.php?ident=639>

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Referências bibliográficas

- GOODRICH, Michael T. et al: *Algorithm Design and Applications*. Wiley, 2015.
- LEVITIN, Anany. *Introduction to the Design and Analysis of Algorithms*. Pearson, 2012.
- SKIENA, Steven S. *The Algorithm Design Manual*. Springer, 2008.
- Série de Livros Didáticos. *Complexidade de Algoritmos*. UFRGS.
- BHASIN, Harsh. *Algorithms – Design and Analysis*. Oxford University Press, 2015.
- FREITAS, Aparecido V. de – 2022 – Estruturas de Dados: Notas de Aula.
- CALVETTI, Robson - 2015 – Estruturas de Dados: Notas de Aula.

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Aula 03

FIM