

# EDCO4B

# ESTRUTURAS DE DADOS 2

Aula 02 - Bubble Sort

Prof. Rafael G. Mantovani  
Prof. Luiz Fernando Carvalho

# Roteiro



- 1** Introdução
- 2** Bubble Sort
- 3** Exemplo
- 4** Exercício
- 5** Referências

# Roteiro

- 1** Introdução
- 2** Bubble Sort
- 3** Exemplo
- 4** Exercício
- 5** Referências

# Introdução



**Algoritmos de  
Ordenação**

# Introdução

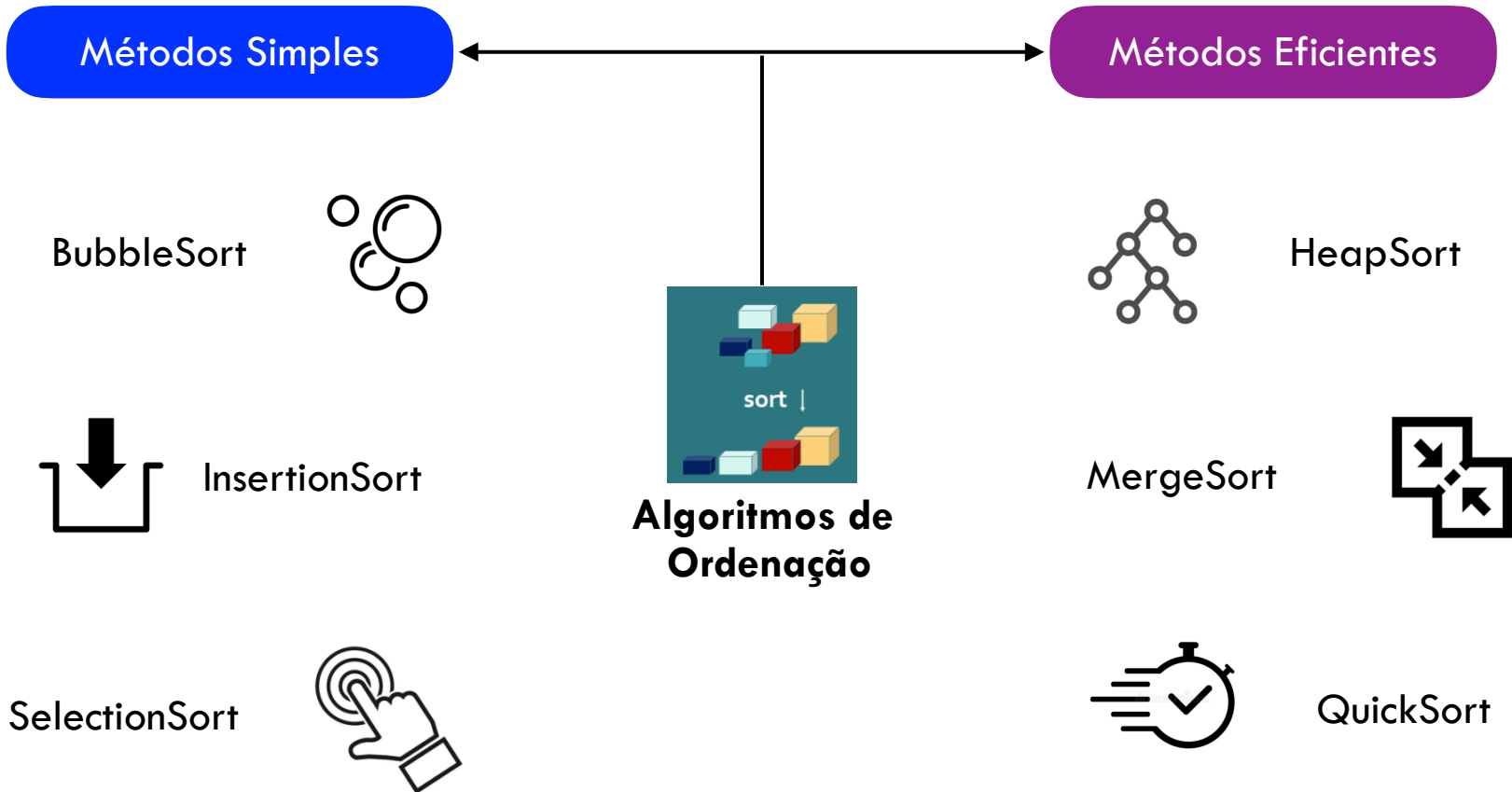
Métodos Simples

Métodos Eficientes

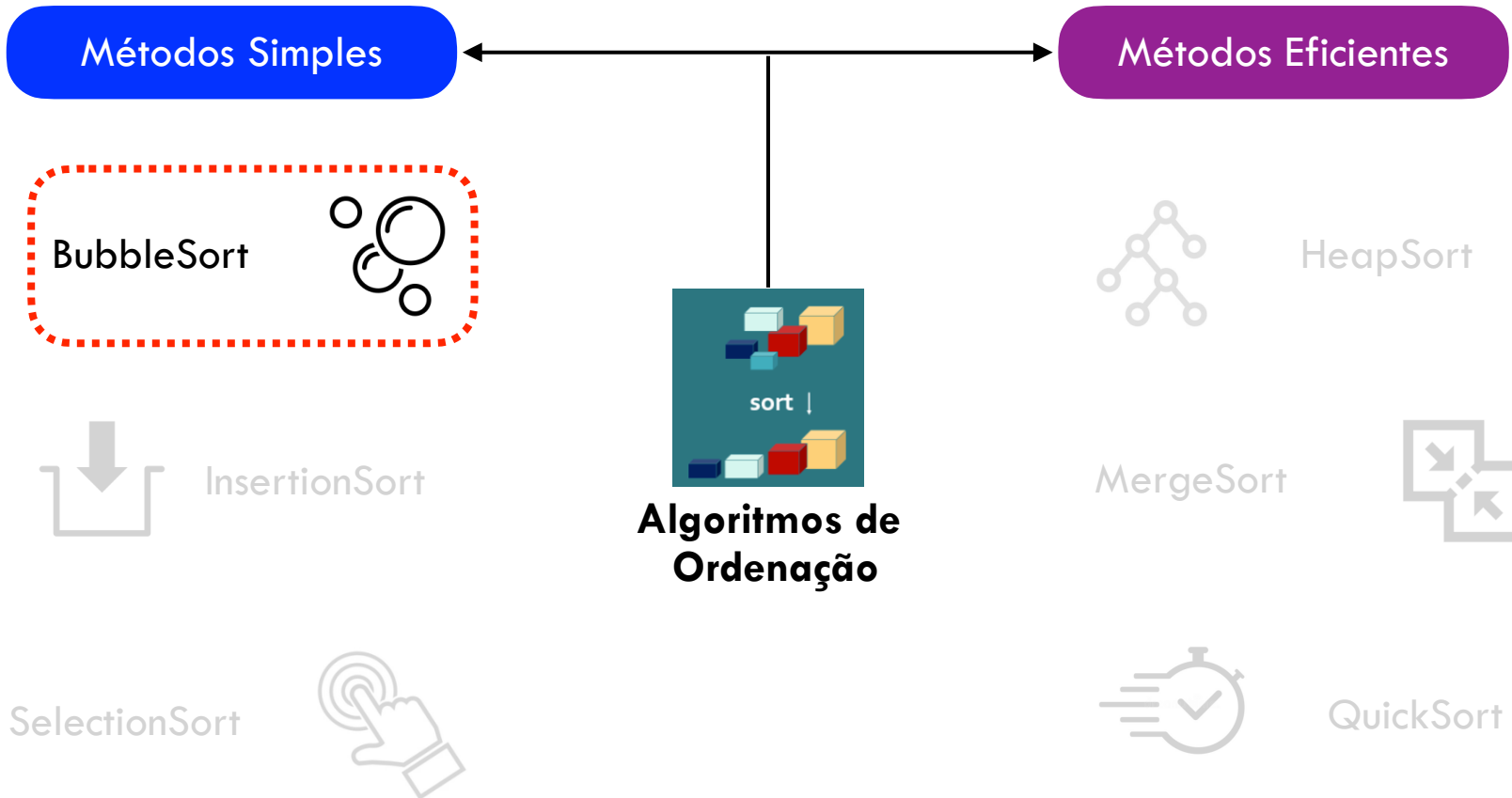


**Algoritmos de  
Ordenação**

# Introdução



# Introdução



# Roteiro



- 1 Introdução
- 2 Bubble Sort
- 3 Exemplo
- 4 Exercício
- 5 Referências



# Bubble Sort

- Ordenação por Bolha / Borbulhamento



# Bubble Sort

- Ordenação por Bolha / Borbulhamento



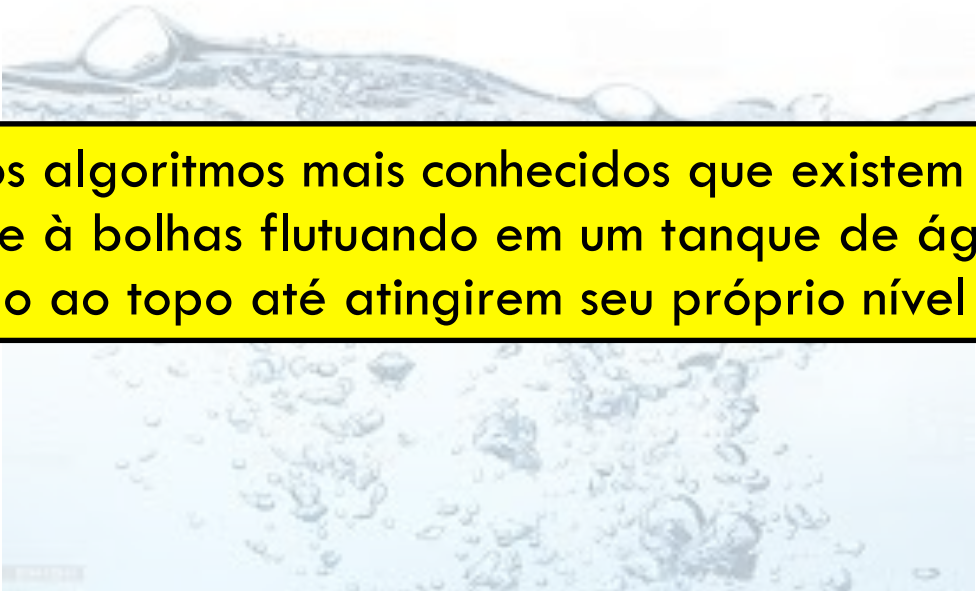
Topo



Fundo

# Bubble Sort

## □ Ordenação por Bolha / Borbulhamento

- 
- \* um dos algoritmos mais conhecidos que existem
  - \* remete à bolhas flutuando em um tanque de água: vão em direção ao topo até atingirem seu próprio nível

# Bubble Sort

## □ Funcionamento

\* **compara pares de elementos adjacentes** em um array e os troca de lugar se estiverem na ordem errada

# Bubble Sort

## □ Funcionamento

- \* **compara pares de elementos adjacentes** em um array e os troca de lugar se estiverem na ordem errada
- \* trabalha de forma a movimentar, uma posição por vez, **o maior valor existente na posição não ordenada** de um vetor para sua respectiva posição no vetor ordenado

# Bubble Sort

## □ Funcionamento

- \* **compara pares de elementos adjacentes** em um array e os troca de lugar se estiverem na ordem errada
- \* trabalha de forma a movimentar, uma posição por vez, **o maior valor** existente **na posição não ordenada** de um vetor para sua respectiva posição no vetor ordenado
- \* processo se **repete** até que mais **nenhuma troca** seja necessária (elementos já estão ordenados)

# Bubble Sort

## □ Desempenho

\* **melhor caso:**  $O(N)$

# Bubble Sort

## □ Desempenho

\* **melhor caso:**  $O(N)$  // elementos já estão quase ordenados



# Bubble Sort

## □ Desempenho

\* **melhor caso:**  $O(N)$  // elementos já estão quase ordenados

\* **pior caso:**  $O(N^2)$

# Bubble Sort

## □ Desempenho

- \* **melhor caso:**  $O(N)$  // elementos já estão quase ordenados
- \* **pior caso:**  $O(N^2)$  // elementos estão em ordem decrescente

# Bubble Sort

## □ Desempenho

- \* **melhor caso:**  $O(N)$  // elementos já estão quase ordenados
- \* **pior caso:**  $O(N^2)$  // elementos estão em ordem decrescente
- \* **caso médio:**  $O(N^2)$

# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**



# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

1. **TROCOU = True** *// Variável de controle*

# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

1. **TROCOU** = **True** *// Variável de controle*
2. **Enquanto** **TROCOU** == **True** faça:

# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

1. **TROCOU** = **True** *// Variável de controle*
2. **Enquanto** **TROCOU** == **True** faça:
3.     **TROCOU** = **False**

# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

1. **TROCOU** = **True** *// Variável de controle*
2. **Enquanto** **TROCOU** == **True** faça:
3.     **TROCOU** = **False**
4.     **Para** todo índice **i** entre 0 e N-1



# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

1. **TROCOU** = **True** *// Variável de controle*
2. **Enquanto** **TROCOU** == **True** faça:
3.     **TROCOU** = **False**
4.     **Para** todo índice **i** entre 0 e N-1
5.         **Se** vetor[**i**] > vetor[**i** + 1], então:

# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

1. **TROCOU** = **True** *// Variável de controle*
2. **Enquanto** **TROCOU** == **True** faça:
3.     **TROCOU** = **False**
4.     **Para** todo índice **i** entre 0 e N-1
5.         **Se** vetor[**i**] > vetor[**i** + 1], então:
6.             trocar os conteúdos de vetor[**i**] e vetor[**i** + 1]
7.         **TROCOU** = **True**

# Pseudocódigo: Bubble Sort

**BubbleSort (vetor, N):**

```
1.  TROCOU = True                                // Variável de controle
2.  Enquanto TROCOU == True faça:
3.      TROCOU = False
4.      Para todo índice i entre 0 e N-1
5.          Se vetor[i] > vetor[i + 1], então:
6.              trocar os conteúdos de vetor[i] e vetor[i + 1]
7.              TROCOU = True

// Ao final do laço principal, o vetor "vetor" está ordenado
```

# Roteiro



- 1 Introdução
- 2 Bubble Sort
- 3 Exemplo
- 4 Exercícios
- 5 Referências

# Exemplo

23	4	67	-8	90	54	21
----	---	----	----	----	----	----

**vetor não ordenado**

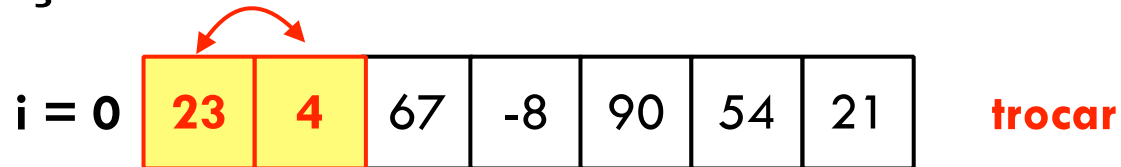
# Exemplo

**Iteração 01:**

$i = 0$	23	4	67	-8	90	54	21	<b>trocar</b>
---------	----	---	----	----	----	----	----	---------------

# Exemplo

**Iteração 01:**



# Exemplo

**Iteração 01:**

<b>i = 0</b>	<b>23</b>	<b>4</b>	67	-8	90	54	21	<b>trocar</b>
<b>i = 1</b>	<b>4</b>	<b>23</b>	67	-8	90	54	21	



# Exemplo

**Iteração 01:**

<b>i = 0</b>	23	4	67	-8	90	54	21	<b>trocar</b>  <b>OK</b>
<b>i = 1</b>	4	23	67	-8	90	54	21	

# Exemplo

## Iteração 01:

$i = 0$	23	4	67	-8	90	54	21	trocar
$i = 1$	4	23	67	-8	90	54	21	OK
$i = 2$	4	23	67	-8	90	54	21	trocar
$i = 3$	4	23	-8	67	90	54	21	OK
$i = 4$	4	23	-8	67	90	54	21	trocar
$i = 5$	4	23	-8	67	54	90	21	trocar

4	23	-8	67	54	21	90
---	----	----	----	----	----	----

# Exemplo

Iteração 01:

i = 0	23	4	67	-8	90	54	21	trocar
i = 1	4	23	67	-8	90	54	21	OK
i = 2	4	23	67	-8	90	54	21	trocar
i = 3	4	23	67	-8	90	54	21	OK
i = 4	4	23	-8	67	90	54	21	trocar
i = 5	4	23	-8	67	54	90	21	trocar

Tivemos trocas, então seguimos para uma próxima iteração

4	23	-8	67	54	21	90
---	----	----	----	----	----	----

Final da iteração 01

# Exemplo

**Iteração 02:**

$i = 0$	4	23	-8	67	54	21	90
---------	---	----	----	----	----	----	----

# Exemplo

## Iteração 02:

$i = 0$	4	23	-8	67	54	21	90	OK
$i = 1$	4	23	-8	67	54	21	90	trocar
$i = 2$	4	-8	23	67	54	21	90	OK
$i = 3$	4	-8	23	67	54	21	90	trocar
$i = 4$	4	-8	23	54	67	21	90	trocar

4	-8	23	54	21	67	90
---	----	----	----	----	----	----

Final da iteração 02

# Exemplo

Iteração 02:

i = 0	4	23	-8	67	54	21	90	OK
i = 1	4	23	-8	67	54	21	90	trocar
i = 2	4	-8	23	67	54	21	90	OK
i = 3	4	-8	23	54	67	21	90	trocar
i = 4	4	-8	23	54	67	21	90	trocar

Tivemos trocas, então seguimos para uma próxima iteração

4	-8	23	54	21	67	90
---	----	----	----	----	----	----

Final da iteração 02

# Exemplo

**Iteração 03:**

$i = 0$	4	-8	23	54	21	67	90
---------	---	----	----	----	----	----	----

# Exemplo

## Iteração 03:

$i = 0$	4	-8	23	54	21	67	90	trocar
$i = 1$	-8	4	23	54	21	67	90	OK
$i = 2$	-8	4	23	54	21	67	90	OK
$i = 3$	-8	4	23	54	21	67	90	trocar

-8	4	23	21	54	67	90
----	---	----	----	----	----	----

Final da iteração 03



# Exemplo

## Iteração 03:

i = 0	4	-8	23	54	21	67	90	trocar
i = 1	-8	4	23	54	21	67	90	OK
i = 2	-8	4	23	54	21	67	90	OK

Tivemos trocas, então seguimos para uma próxima iteração

trocar

-8	4	23	21	54	67	90
----	---	----	----	----	----	----

Final da iteração 03

# Exemplo

**Iteração 04:**

$i = 0$	-8	4	23	21	54	67	90
---------	----	---	----	----	----	----	----

# Exemplo

## Iteração 04:

$i = 0$	-8	4	23	21	54	67	90	OK
$i = 1$	-8	4	23	21	54	67	90	OK
$i = 2$	-8	4	23	21	54	67	90	trocar

-8	4	21	23	54	67	90
----	---	----	----	----	----	----

Final da iteração 04

# Exemplo

## Iteração 04:

i = 0	-8	4	23	21	54	67	90	OK
i = 1	-8	4	23	21	54	67	90	OK
i = 2	-8	4	23	21	54	67	90	

**trocar**

Tivemos trocas, então seguimos para uma próxima iteração

-8	4	21	23	54	67	90
----	---	----	----	----	----	----

**Final da iteração 04**

# Exemplo

**Iteração 05:**

$i = 0$	-8	4	21	23	54	67	90
---------	----	---	----	----	----	----	----

# Exemplo

**Iteração 05:**

$i = 0$	-8	4	21	23	54	67	90	OK
$i = 1$	-8	4	21	23	54	67	90	OK

# Exemplo

## Iteração 05:

$i = 0$	-8	4	21	23	54	67	90	OK
$i = 1$	-8	4	21	23	54	67	90	OK

Não tivemos trocas, fim!

-8	4	21	23	54	67	90
----	---	----	----	----	----	----

Vetor Ordenado

# Bubble Sort

## Vantagens

- \* **simples** e de fácil entendimento e implementação
- \* está entre os métodos mais difundidos



# Bubble Sort

## Vantagens

- \* **simples** e de fácil entendimento e implementação
- \* está entre os métodos mais difundidos

## Desvantagens

- \* **não** é um algoritmo **eficiente**
- \* Sua eficiência diminui de acordo com o número de elementos
- \* é estudado apenas para fins didáticos

# Roteiro



- 1 Introdução
- 2 Bubble Sort
- 3 Exemplo
- 4 Exercícios
- 5 Referências

# Exercícios



**HANDS ON :)))**

# Exercícios



1) Execute o teste de mesa (simulação) do algoritmo **Bubble Sort** para a sua sequência de números aleatórios, definida na planilha da disciplina.

# Exercícios

2) Implemente o **bubbleSort** em Python considerando a seguinte assinatura de função:

```
/* Ordena o vetor usando BubbleSort
```

```
Parâmetros:
```

```
    array: vetor a ser ordenado
```

```
    option: opção que define se a ordenação é crescente ou  
            decrescente
```

```
Esse algoritmo tem um comportamento assintótico  $O(N^2)$  */
```

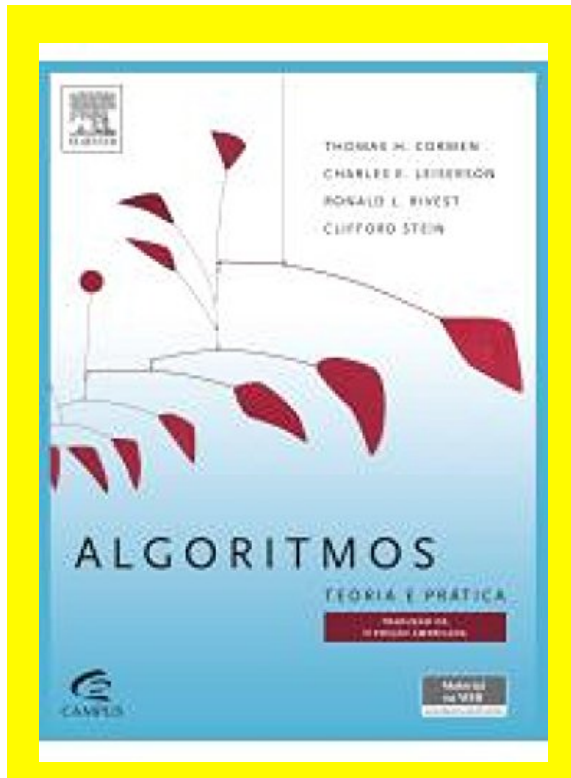
```
bubbleSort(array, option)
```

# Roteiro



- 1** Introdução
- 2** Bubble Sort
- 3** Exemplo
- 4** Exercícios
- 5** Referências

# Referências sugeridas

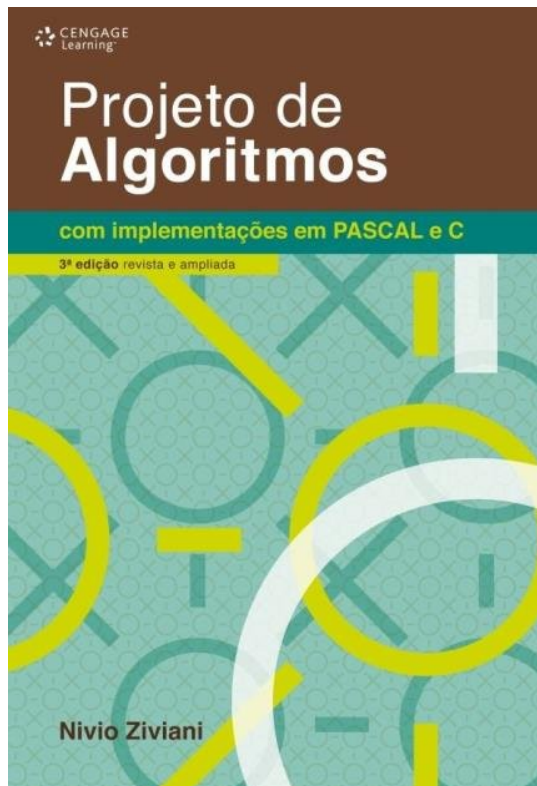


[Cormen et al, 2018]

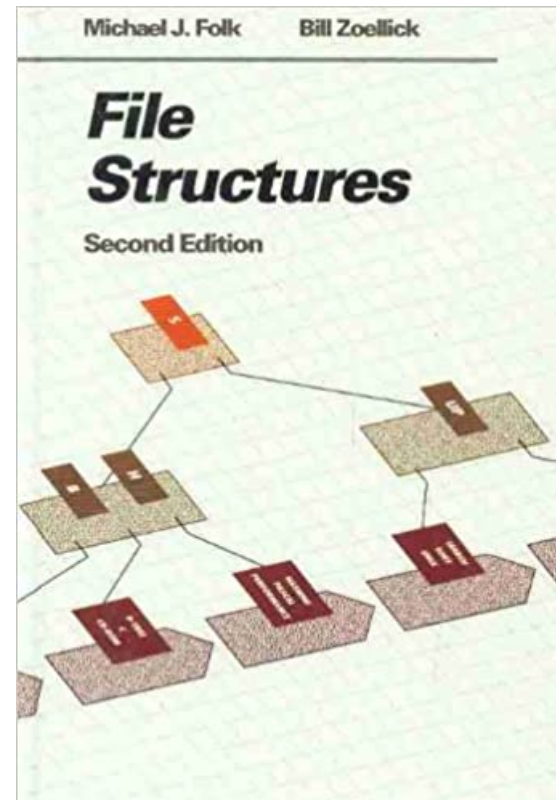


[Drozdek, 2017]

# Referências sugeridas



[Ziviani, 2010]



[Folk & Zoellick, 1992]



# Perguntas?

Prof. Rafael G. **Mantovani**

[rafaelmantovani@utfpr.edu.br](mailto:rafaelmantovani@utfpr.edu.br)