

EDCO4B

ESTRUTURAS DE DADOS 2

Aula 03B - Insertion Sort

Prof. Rafael G. Mantovani

Licença

Este trabalho está licenciado com uma Licença CC BY-NC-ND 4.0:



maiores informações:

https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR

Roteiro



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

Roteiro

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

Introdução



**Algoritmos de
Ordenação**

Introdução

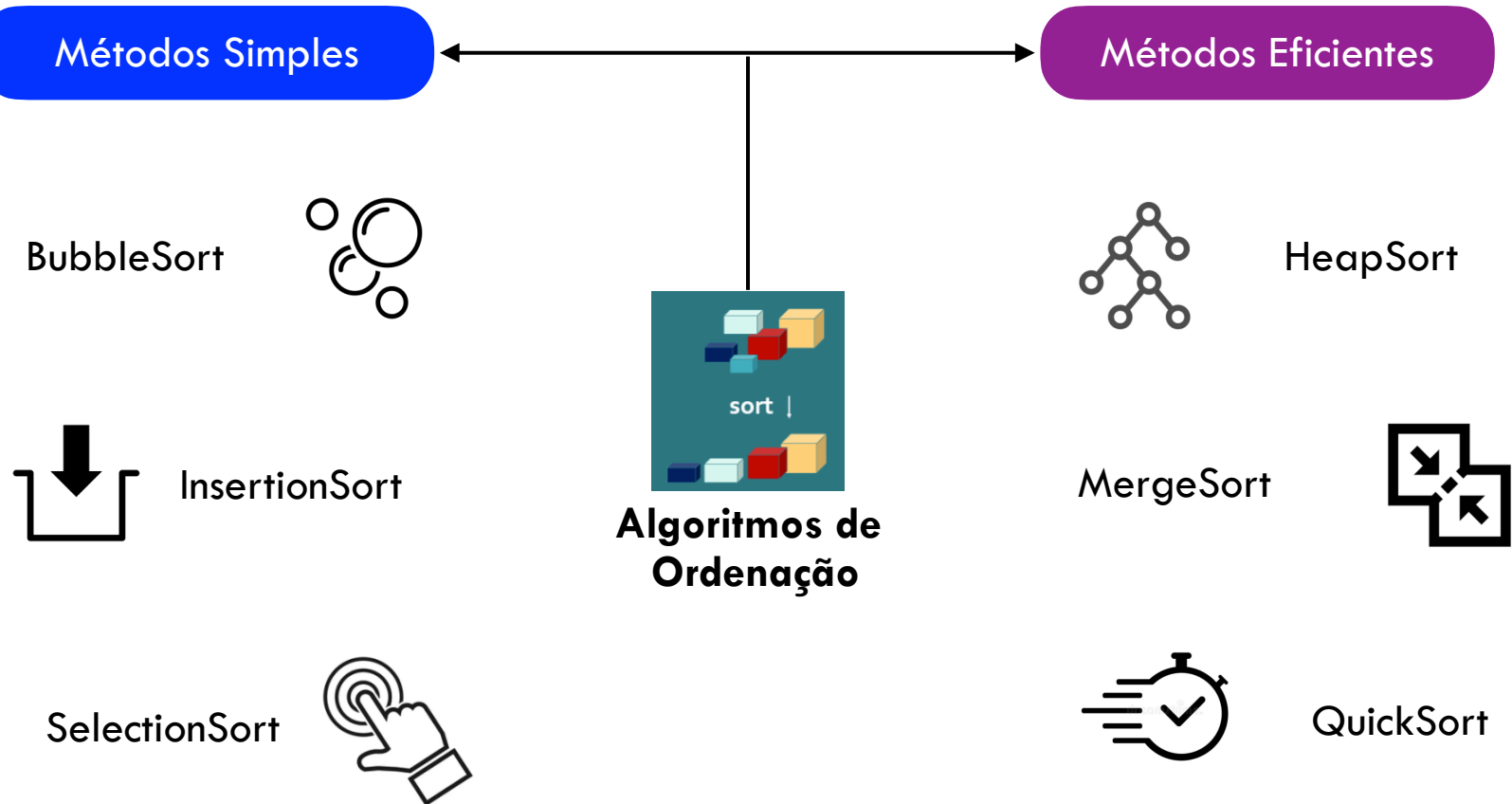
Métodos Simples

Métodos Eficientes

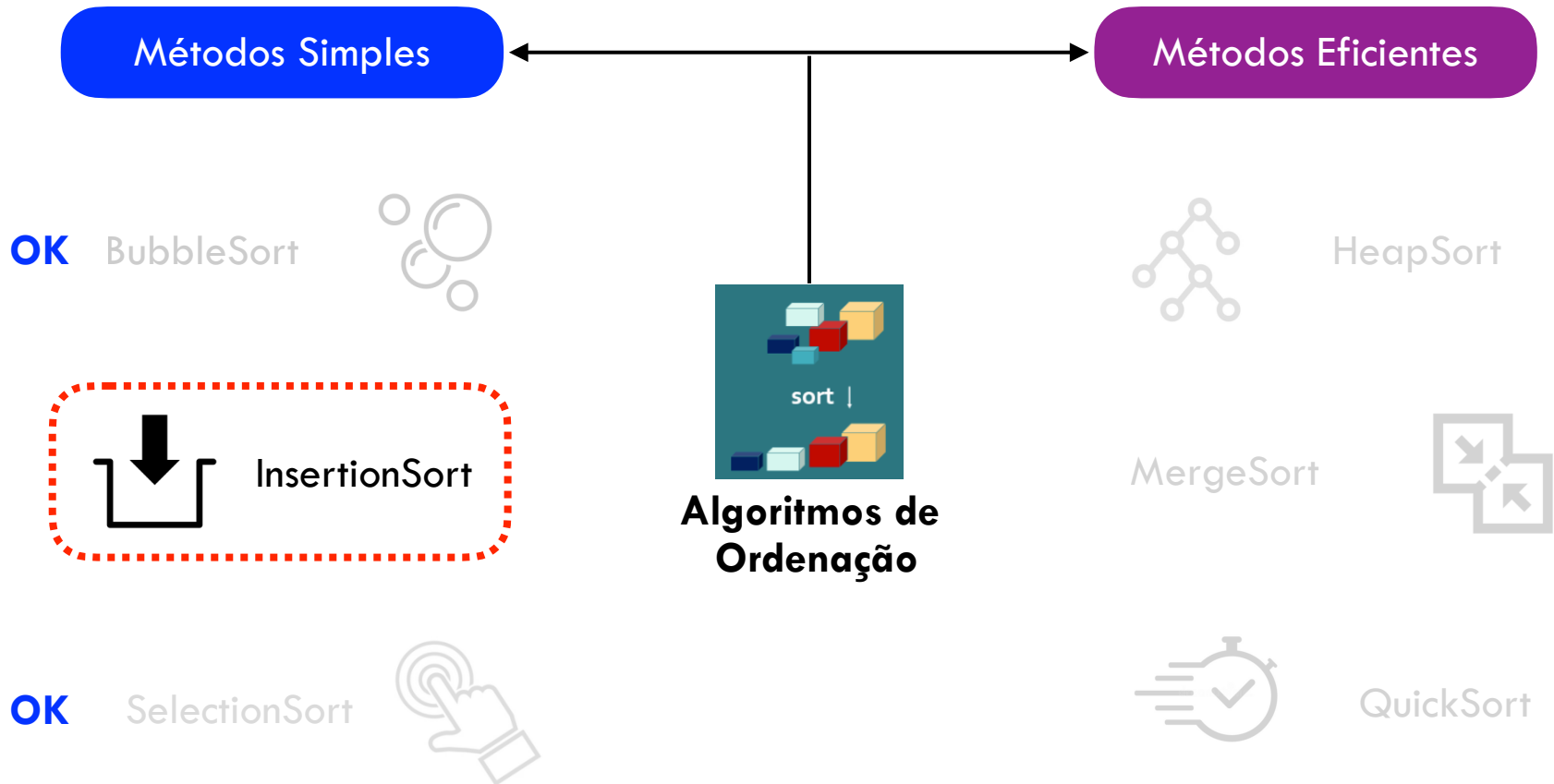


**Algoritmos de
Ordenação**

Introdução



Introdução



Roteiro



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

Insertion Sort

□ Ordenação por inserção

- * um dos algoritmos mais simples que existem
- * remete à ideia de ordenação de cartas quando jogamos baralho
- * pega-se uma carta de cada vez e a coloca em seu devido lugar, sempre deixando as cartas da mão em ordem

Insertion Sort

□ Funcionamento

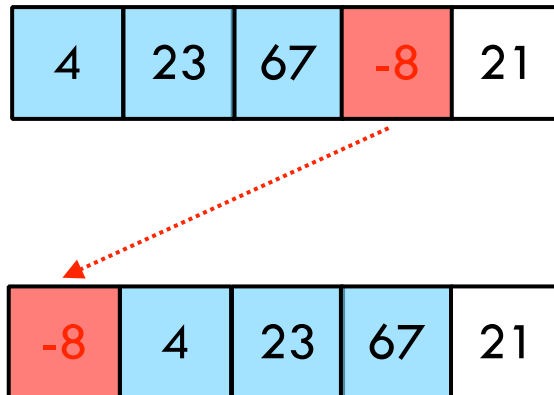
*o algoritmo percorre o array e **para cada posição X** verifica-se se o seu valor está na posição correta:

* isso é feito **andando para o começo do array a partir da posição X**, e **movimentando uma posição para frente os valores que são maiores que** o valor da posição X

* desse modo, teremos uma posição livre para inserir o valor da posição X em seu devido lugar

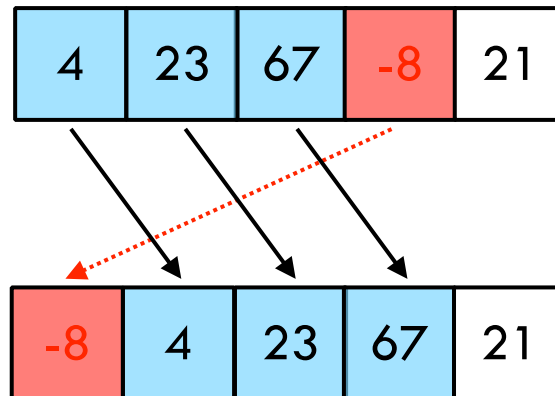
Insertion Sort

□ Funcionamento



Insertion Sort

□ Funcionamento



Insertion Sort

□ Desempenho

- * **melhor caso:** $O(N)$, os elementos já estão ordenados
- * **pior caso:** $O(N^2)$, os elementos estão na ordem decrescente
- * **caso médio:** $O(N^2)$

Insertion Sort

□ Desempenho

- * **melhor caso:** $O(N)$, os elementos já estão ordenados
- * **pior caso:** $O(N^2)$, os elementos estão na ordem decrescente
- * **caso médio:** $O(N^2)$

Obs: Eficiente para grandes conjuntos pequenos. Estável.
Capaz de ordenar dados em tempo real.

Insertion Sort

□ Pseudocódigo

1. InsertionSort (V, TAM)
2. **Para** cada posição i entre 1 e $N-1$, ($i = i+1$) faça:
3. auxiliar = valor na posição i
4. $j = i - 1$
5. **Enquanto** ($j > 0$) && ($\text{auxiliar} < V[j]$):
6. $V[j+1] = V[j]$
7. $j = j - 1$
8. $V[j+1] = \text{auxiliar}$

Roteiro



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

Exemplo

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

vetor não ordenado

Exemplo

Iteração 1:

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

Exemplo

Iteração 1:

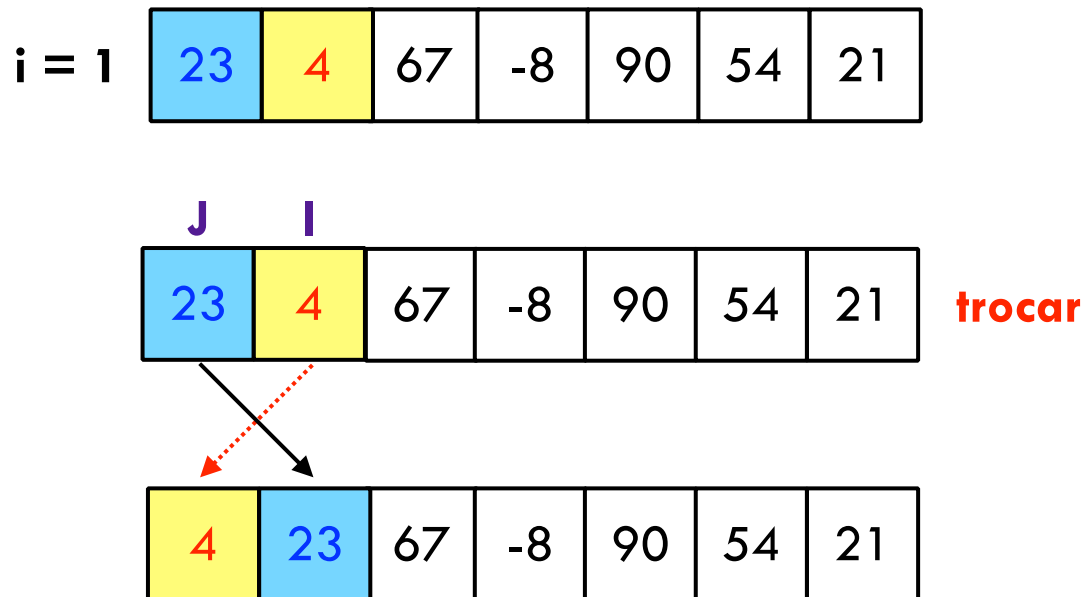
$i = 1$

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

J	I					
23	4	67	-8	90	54	21

Exemplo

Iteração 1:



Exemplo

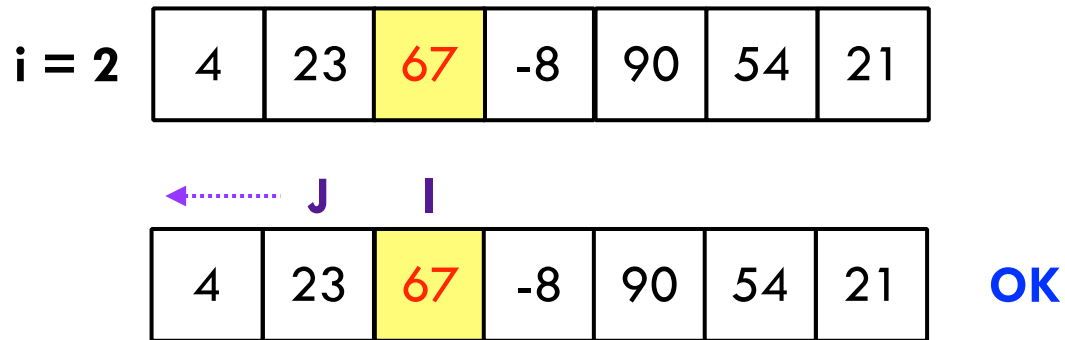
Iteração 2:

i = 2

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

Exemplo

Iteração 2:



Exemplo

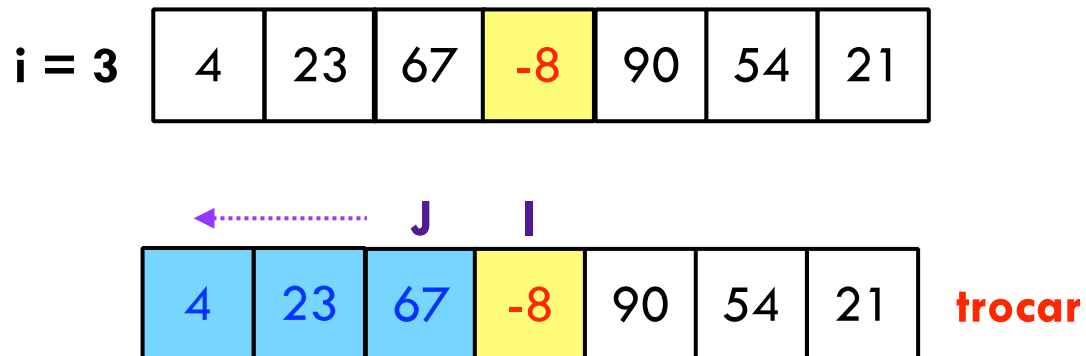
Iteração 3:

i = 3

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

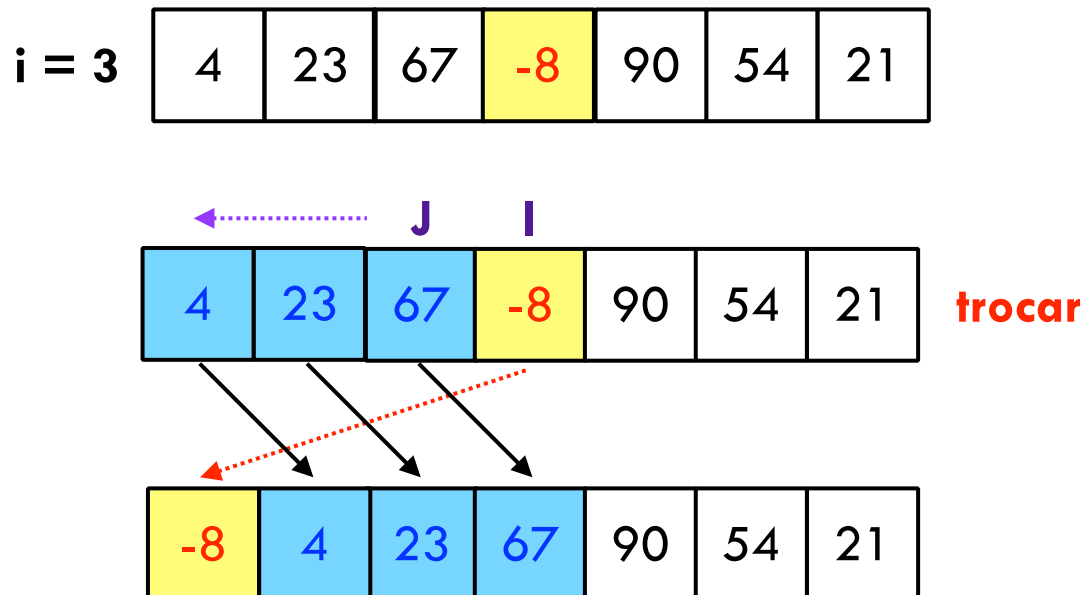
Exemplo

Iteração 3:



Exemplo

Iteração 3:



Exemplo

Iteração 4:

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

Exemplo

Iteração 4:

$i = 4$

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

				J	I		
←							
-8	4	23	67	90	54	21	

OK

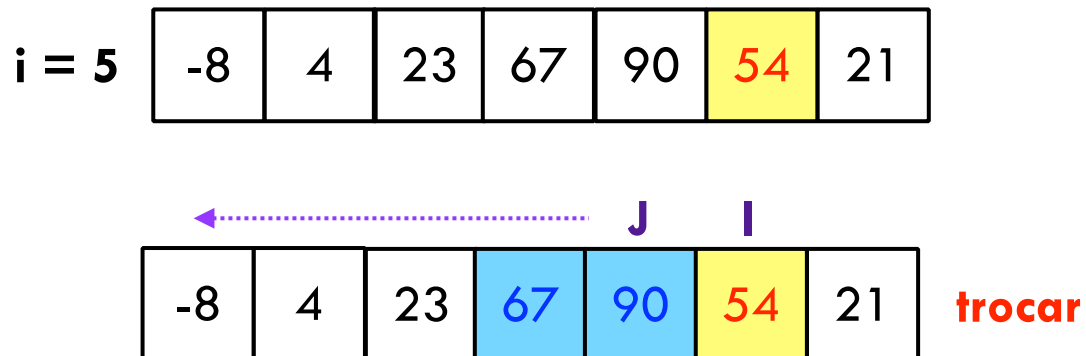
Exemplo

Iteração 5:

$i = 5$	-8	4	23	67	90	54	21
---------	----	---	----	----	----	----	----

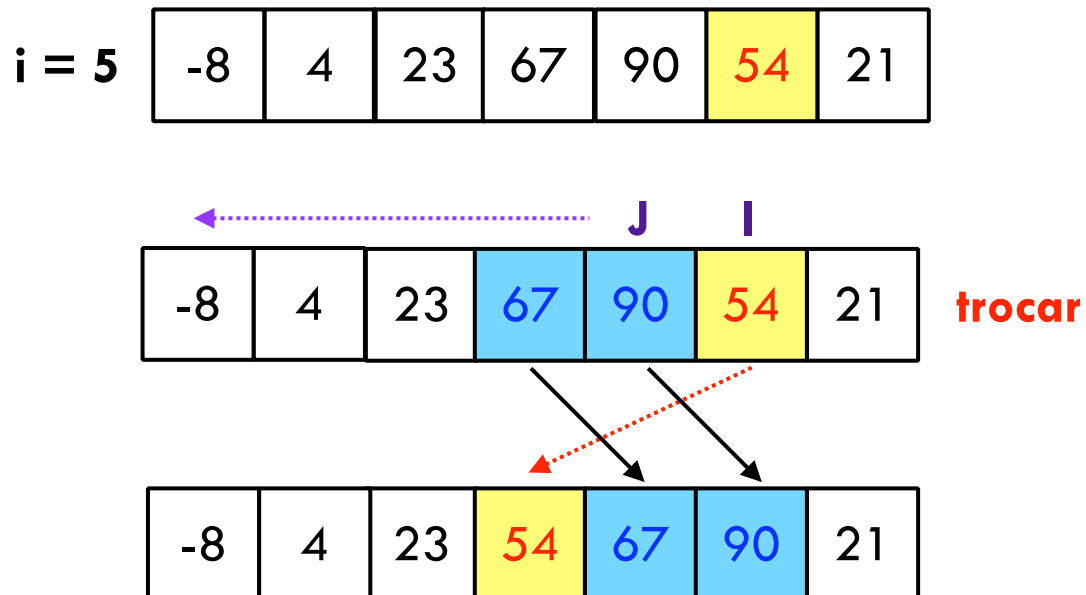
Exemplo

Iteração 5:



Exemplo

Iteração 5:



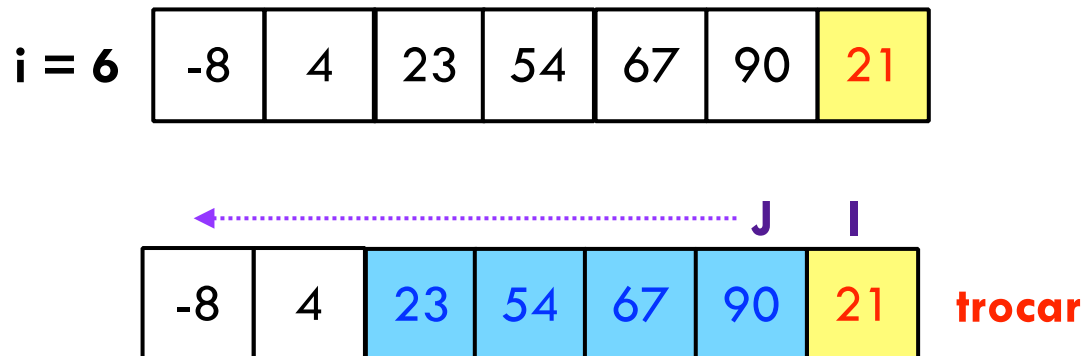
Exemplo

Iteração 6:

i = 6	-8	4	23	54	67	90	21
--------------	----	---	----	----	----	----	----

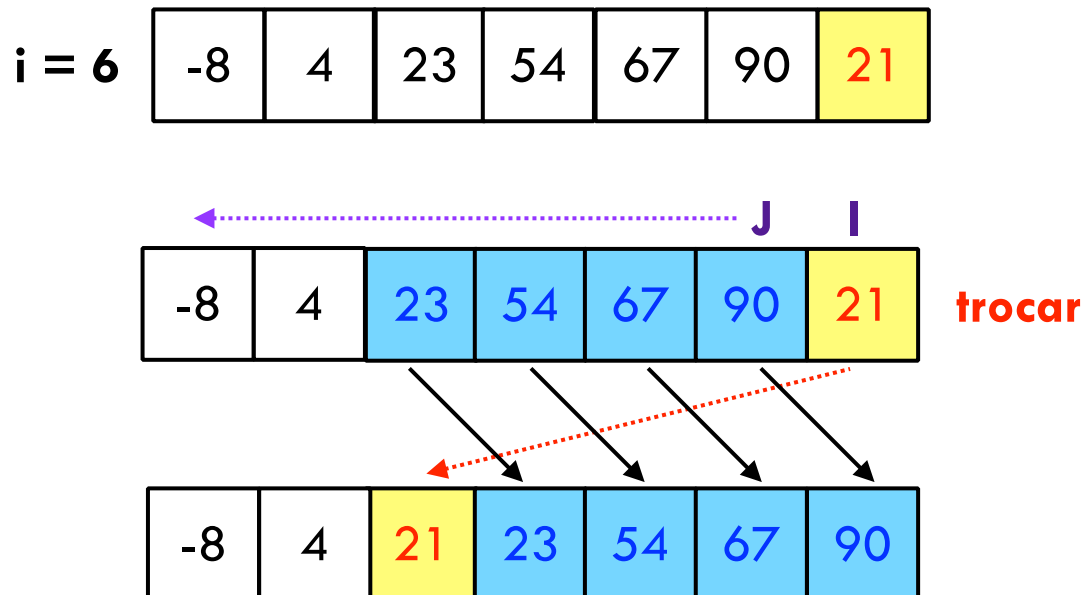
Exemplo

Iteração 6:



Exemplo

Iteração 6:



Exemplo

Final:

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

Vetor Ordenado

Bubble Sort

Vantagens

- * simples e de fácil entendimento e implementação
- * não altera a ordem dos dados (estável)
- * na prática, mais eficiente que os outros algoritmos simples (Bubble e Selection sort)
- * um dos mais rápidos para conjuntos pequenos (Superando quick sort)

Selection Sort

Vantagens


- * simples e de fácil entendimento e implementação
- * não altera a ordem dos dados (estável)
- * na prática, mais eficiente que os outros algoritmos simples (Bubble e Selection sort)
- * um dos mais rápidos para conjuntos pequenos (Superando quick sort)

Desvantagens

- * sua eficiência diminui de acordo com o número de elementos
- * não é recomendado para aplicações com grandes quantidades de dados

Roteiro



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

Exercícios



HANDS ON :)))

Exercícios



1) No Discord, reúna-se com seu grupo e execute o teste de mesa (simulação) do algoritmo para as sequências de números apresentadas

Exercícios

2) Implemente o **insertionSort** em C considerando a seguinte assinatura de função:

/ Ordena o vetor usando Insertion Sort*

Parâmetros:

v: vetor a ser ordenado

n: número de elemento do vetor, tamanho do vetor

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

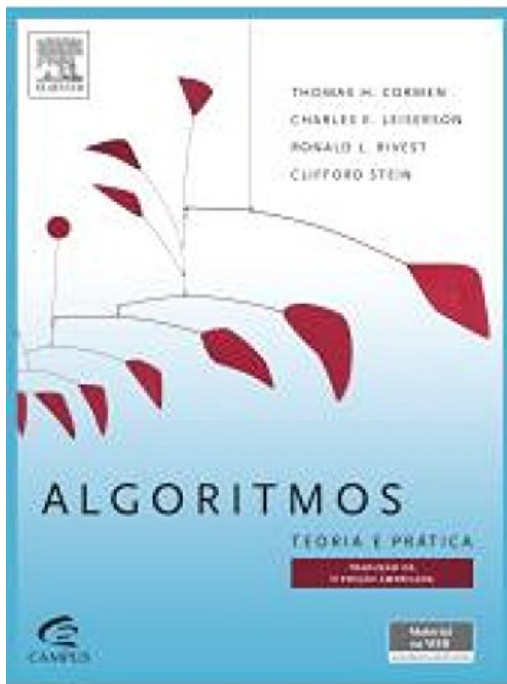
```
void insertionSort(int *v, int n);
```

Roteiro



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

Referências sugeridas

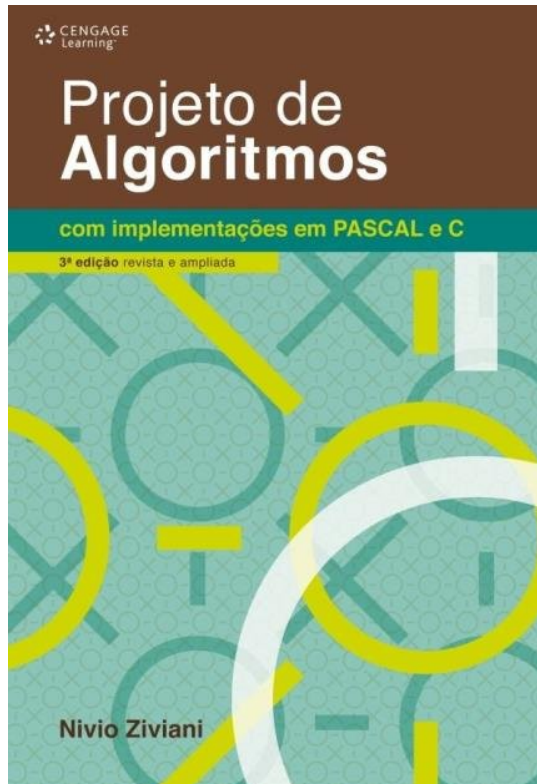


[Cormen et al, 2018]

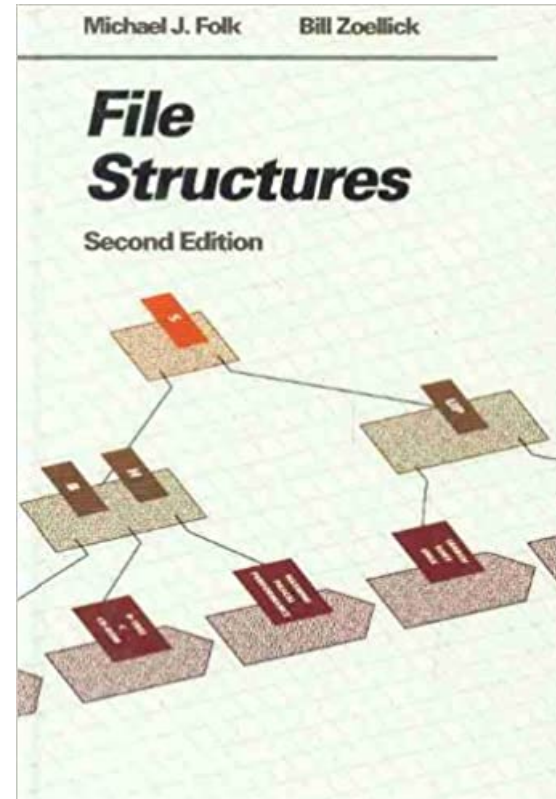


[Drozdek, 2017]

Referências sugeridas



[Ziviani, 2010]



[Folk & Zoellick, 1992]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br