EDCO4B
ESTRUTURAS DE DADOS 2

Aula 04 - Insertion Sort

Prof. Rafael G. Mantovani
Prof. Luiz Fernando Carvalho

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

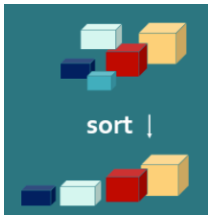
Roteiro

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

Roteiro

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

Introdução

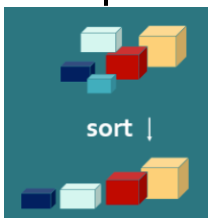


Algoritmos de Ordenação

Introdução

Métodos Simples

Métodos Eficientes



Algoritmos de Ordenação

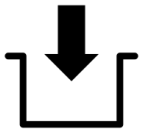
Introdução

Métodos Simples

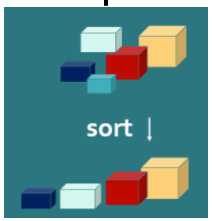
Métodos Eficientes

BubbleSort

InsertionSort



SelectionSort



Algoritmos de Ordenação

HeapSort

MergeSort

QuickSort

Introdução

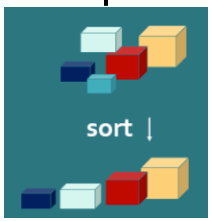
Métodos Simples

Métodos Eficientes

OK BubbleSort

HeapSort

InsertionSort



Algoritmos de Ordenação

MergeSort

OK SelectionSort

QuickSort

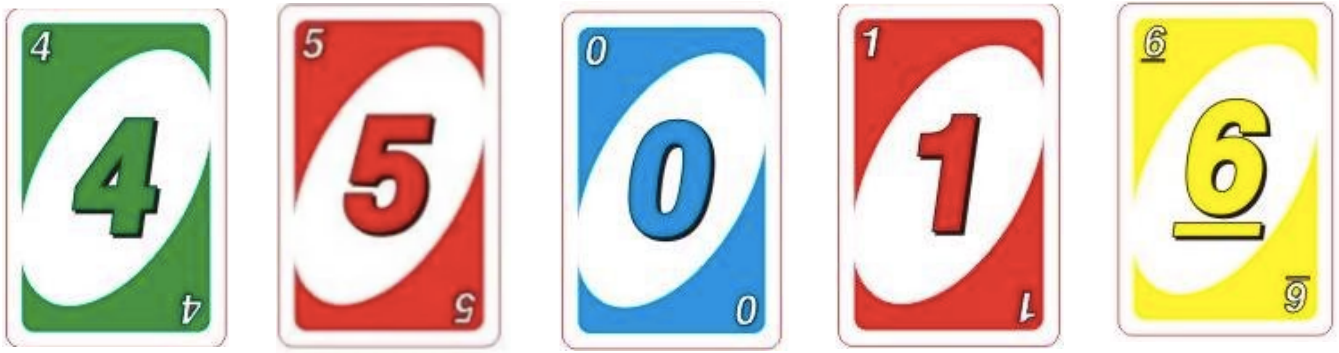
Roteiro

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

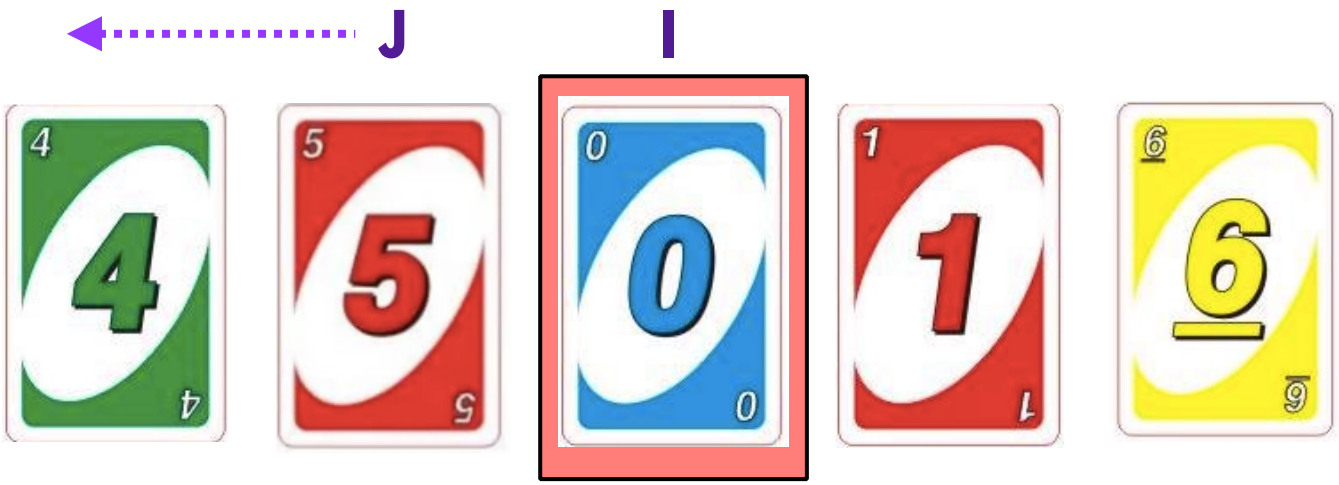
Insertion Sort

Insertion Sort

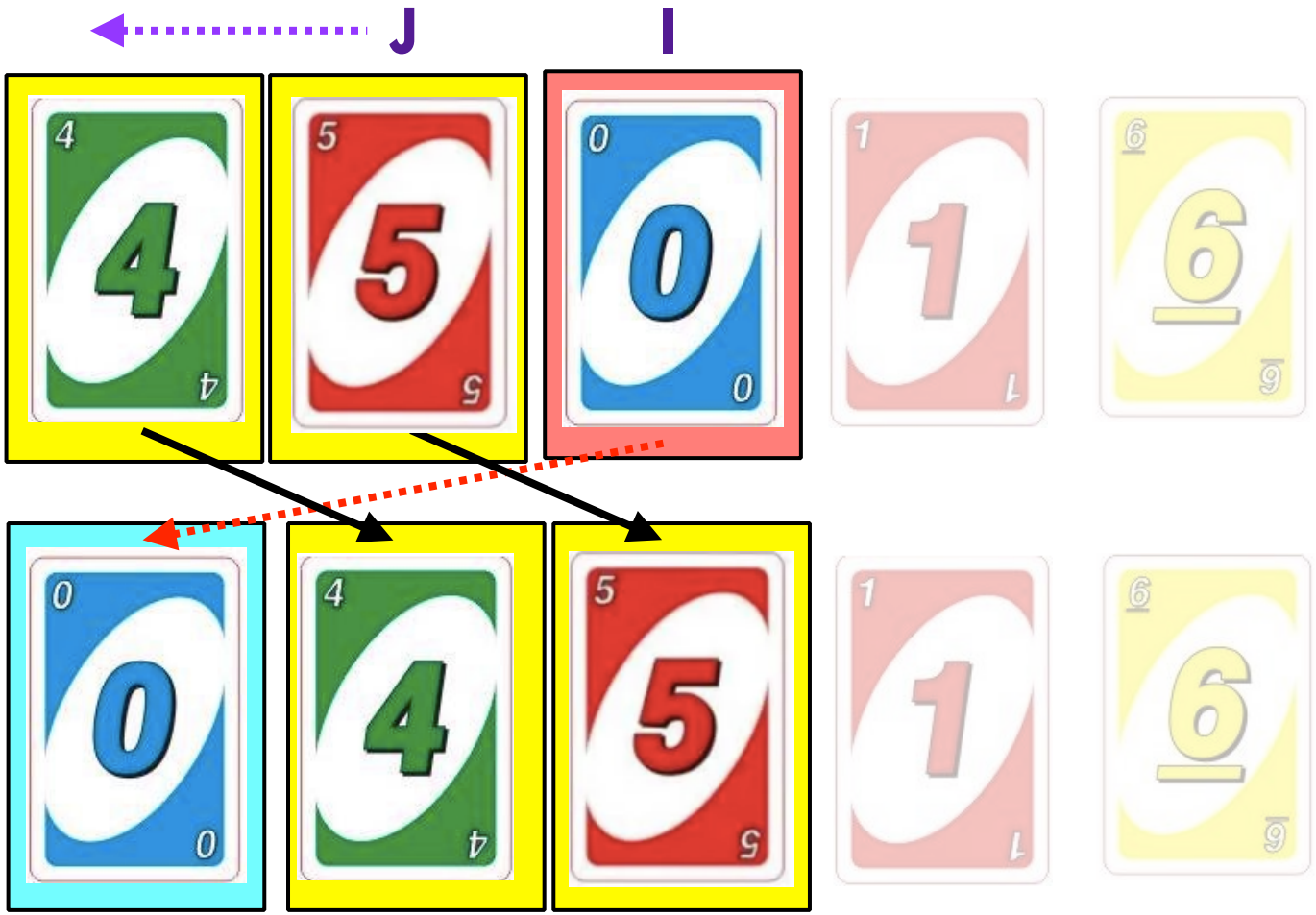
Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort

Insertion Sort

* 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

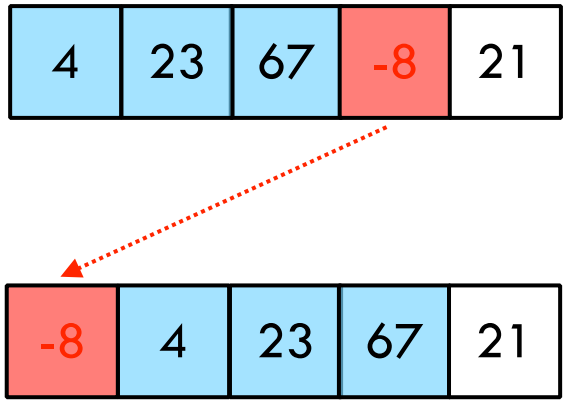
*o algoritmo percorre o array e **para cada posição X** verifica-se se o seu valor está na posição correta:

* 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

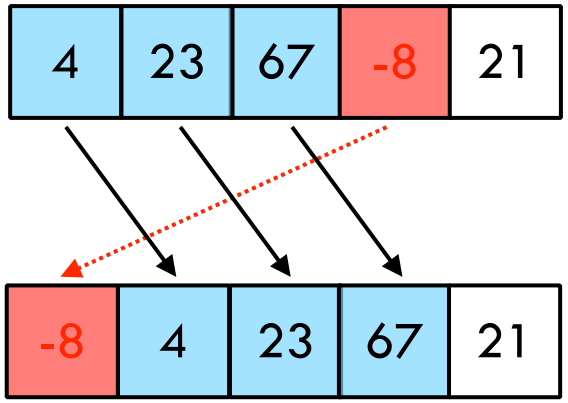
Insertion Sort

- Funcionamento



Insertion Sort

- Funcionamento



Insertion Sort

- **Desempenho**

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

Insertion Sort

- **Desempenho**

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

Insertion Sort

- **Desempenho**

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

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

Insertion Sort

- **Desempenho**

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

* **pior caso:** $O(N^2)$ //elementos estão em ordem decrescente

Insertion Sort

- **Desempenho**

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

* **pior caso:** $O(N^2)$ //elementos estão em ordem decrescente

* **caso médio:** $O(N^2)$

Pseudocódigo: Insertion Sort

InsertionSort (vetor, N):

Pseudocódigo: Insertion Sort

InsertionSort (vetor, N):

1. **Para** cada posição **i** entre 1 e N-1, faça: *// laço externo*

Pseudocódigo: Insertion Sort

InsertionSort (vetor, N):

1. **Para** cada posição **i** entre 1 e N-1, faça: *// laço externo*

   //guardar o valor da posição i na variável auxiliar

2. **auxiliar** = vetor[**i**]
3. **k** = **i** - 1

Pseudocódigo: Insertion Sort

InsertionSort (vetor, N):

```
1. Para cada posição i entre 1 e N-1, faça:                    // laço externo
       //guardar o valor da posição i na variável auxiliar
2.     auxiliar = vetor[ i ]
3.     k = i - 1
4.     Enquanto (k >=0 ) e (auxiliar < vetor[ k] ):            // laço interno
```

Pseudocódigo: Insertion Sort

InsertionSort (vetor, N):

```
1. Para cada posição i entre 1 e N-1, faça:                    // laço externo
     //guardar o valor da posição i na variável auxiliar
2.   auxiliar = vetor[ i ]
3.   k = i - 1
4.   Enquanto (k >=0 ) e (auxiliar < vetor[ k] ):              // laço interno
       // movimenta os elementos uma posição para frente
5.     vetor[ k + 1 ] = vetor[ k ]
6.     k = k - 1
```

Pseudocódigo: Insertion Sort

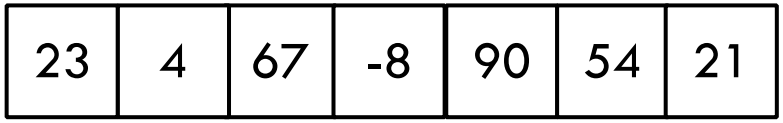
InsertionSort (vetor, N):

```
1.  Para cada posição i entre 1 e N-1, faça:                    // laço externo
        //guardar o valor da posição i na variável auxiliar
2.      auxiliar = vetor[ i ]
3.      k = i - 1
4.      Enquanto (k >=0 ) e (auxiliar < vetor[ k] ):            // laço interno
            // movimenta os elementos uma posição para frente
5.          vetor[ k + 1 ] = vetor[ k ]
6.          k = k - 1
7.      vetor[ k + 1] = auxiliar

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

Roteiro

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

Exemplo



vetor não ordenado

Exemplo

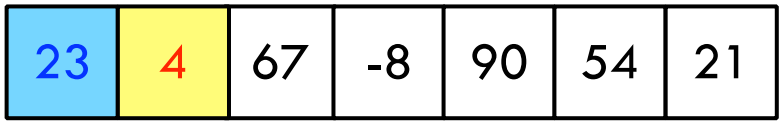
Iteração 1:

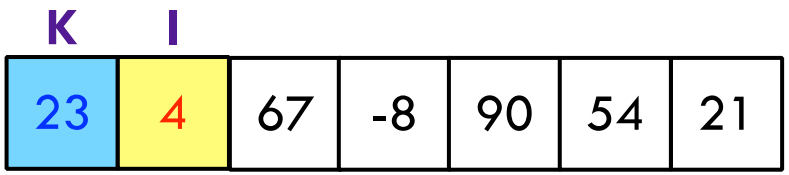
i = 1	23	4	67	-8	90	54	21

Exemplo

Iteração 1:

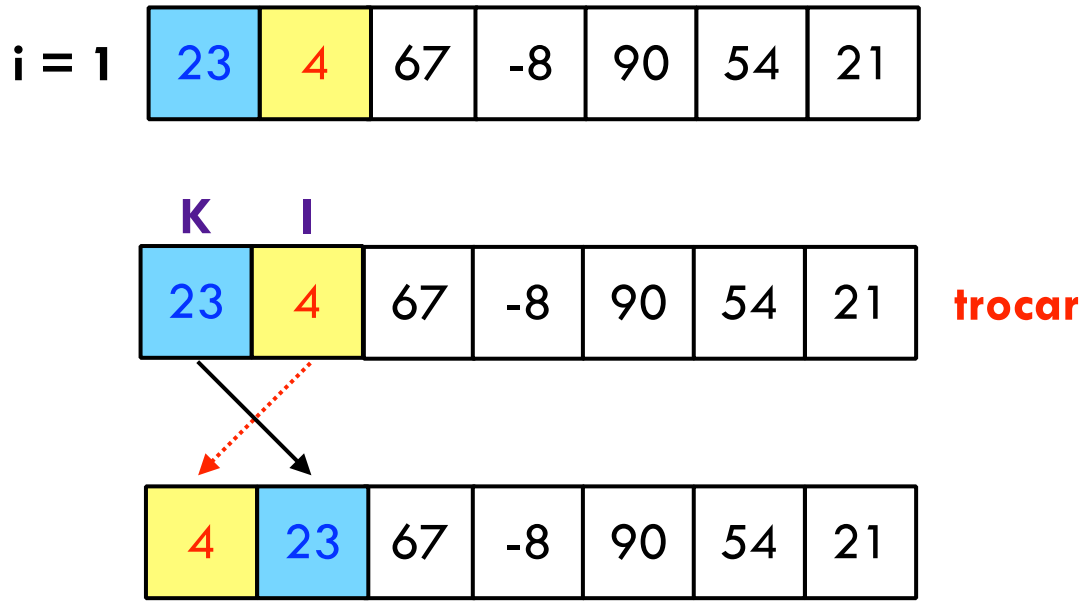
i = 1





Exemplo

Iteração 1:



Exemplo

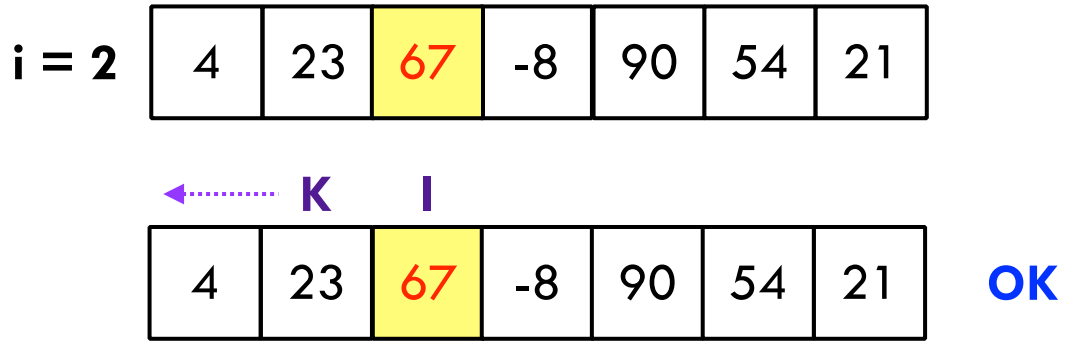
Iteração 2:

i = 2

4	23	67	-8	90	54	21

Exemplo

Iteração 2:



Exemplo

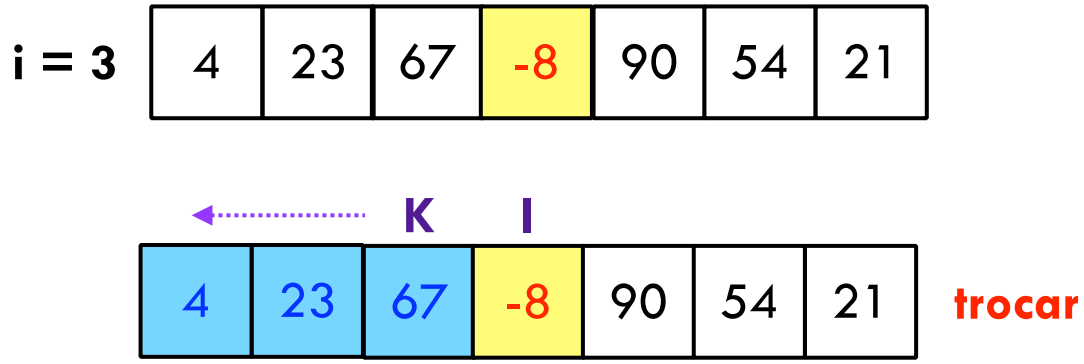
Iteração 3:

i = 3

4	23	67	-8	90	54	21

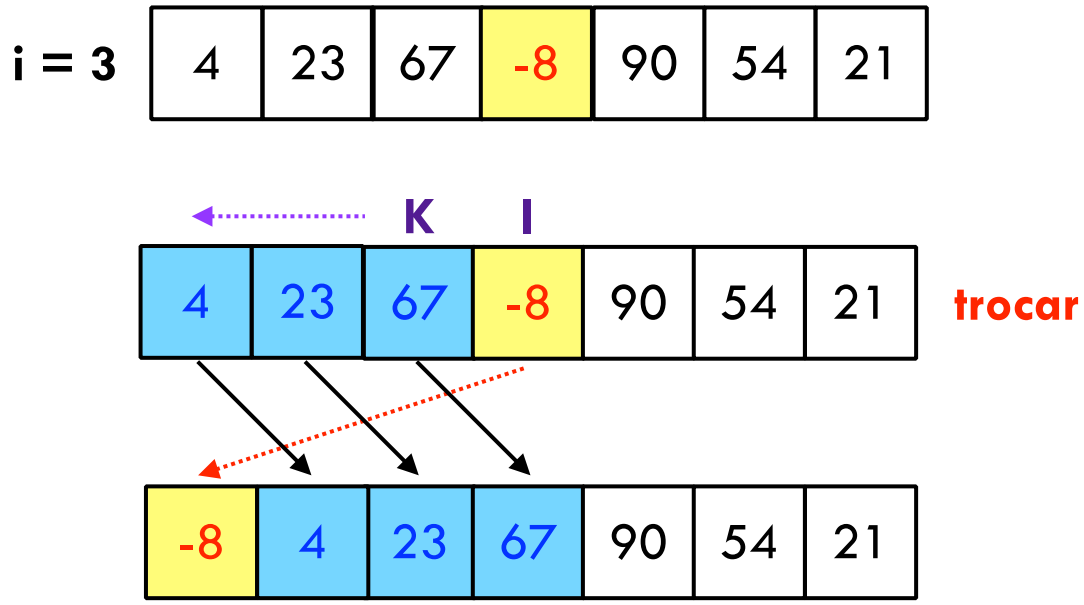
Exemplo

Iteração 3:



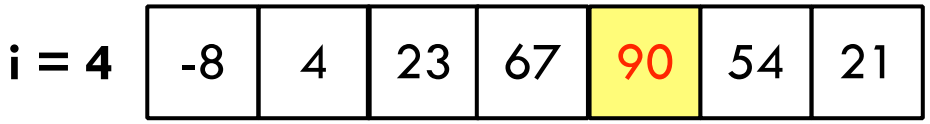
Exemplo

Iteração 3:



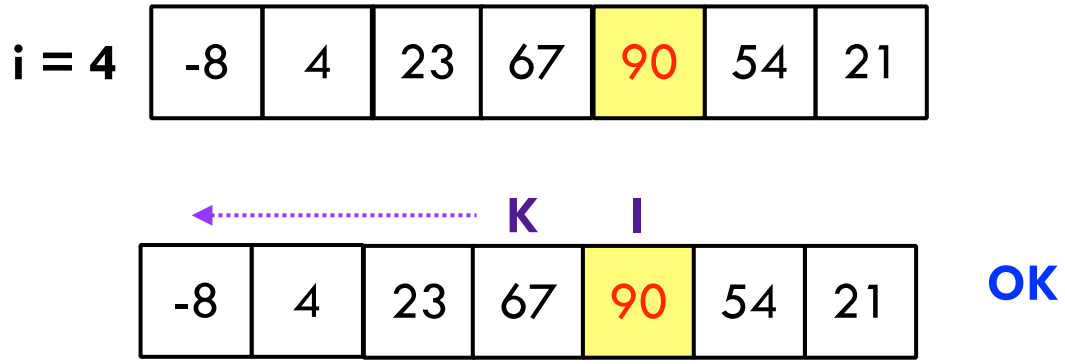
Exemplo

Iteração 4:



Exemplo

Iteração 4:



Exemplo

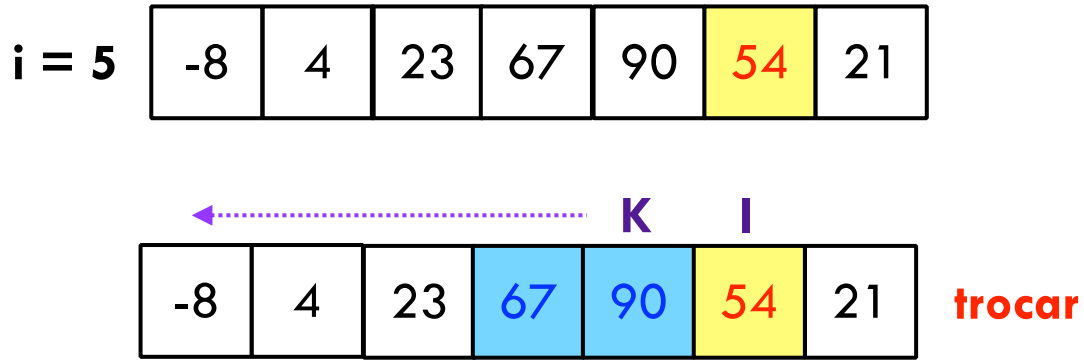
Iteração 5:

i = 5

-8	4	23	67	90	54	21

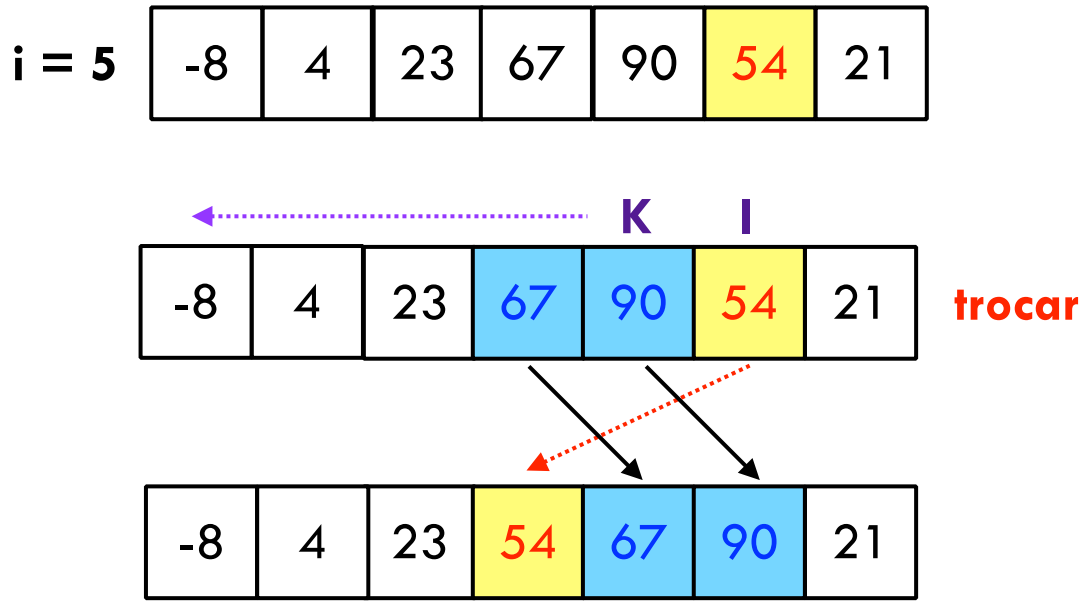
Exemplo

Iteração 5:



Exemplo

Iteração 5:



Exemplo

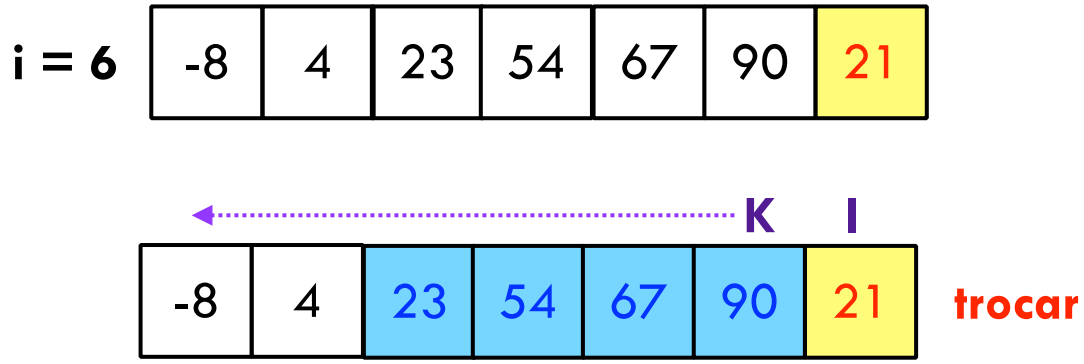
Iteração 6:

i = 6

-8	4	23	54	67	90	21

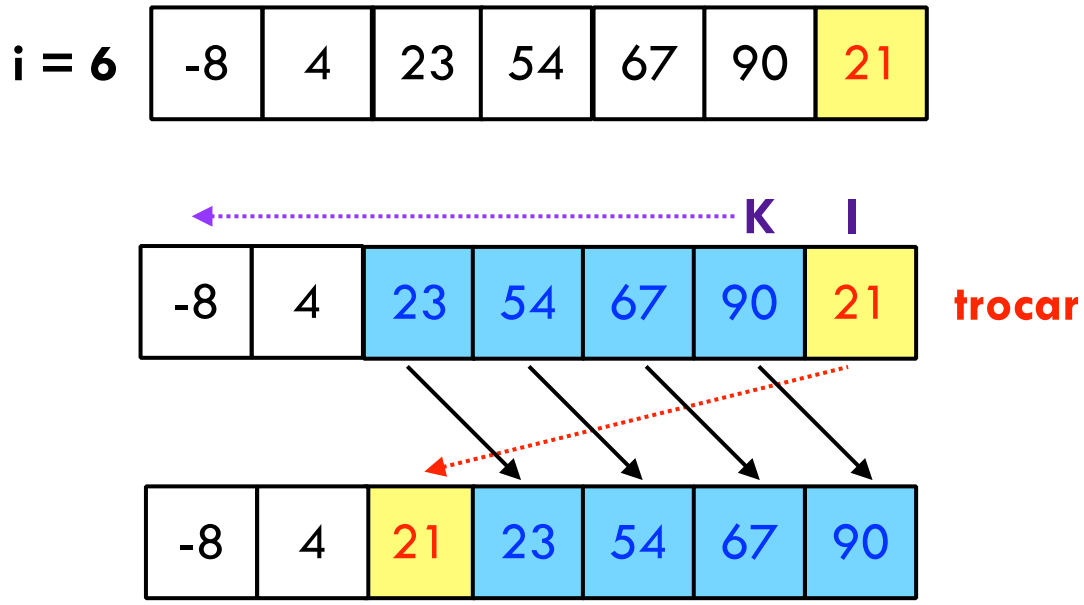
Exemplo

Iteração 6:



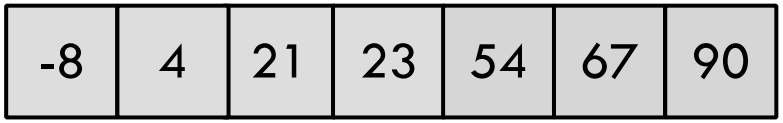
Exemplo

Iteração 6:



Exemplo

Final:



Vetor Ordenado

Insertion 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)

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

Exercícios

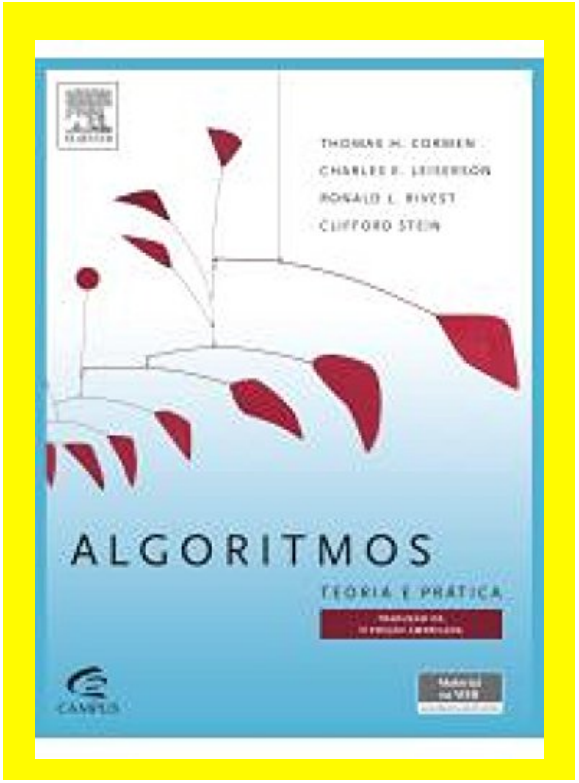
2) Implemente o **insertionSort** em **Python** considerando a seguinte assinatura de função:

```
/* Ordena o vetor usando Insertion Sort
Parâmetros:
   array: vetor a ser ordenado
   option: 1 - ordenação crescente, 2 - ordenação decrescente

Esse algoritmo tem um comportamento assintótico O(N²) */

def insertionSort(array, option):
```

Roteiro

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

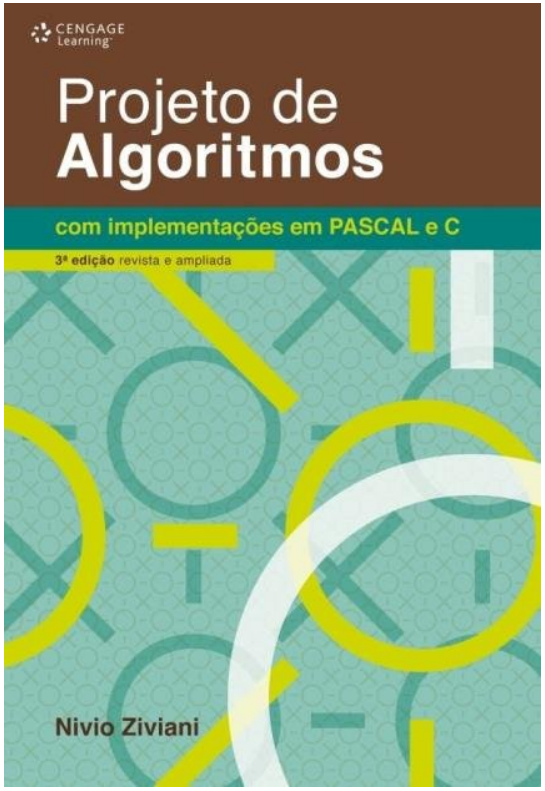
Referências sugeridas



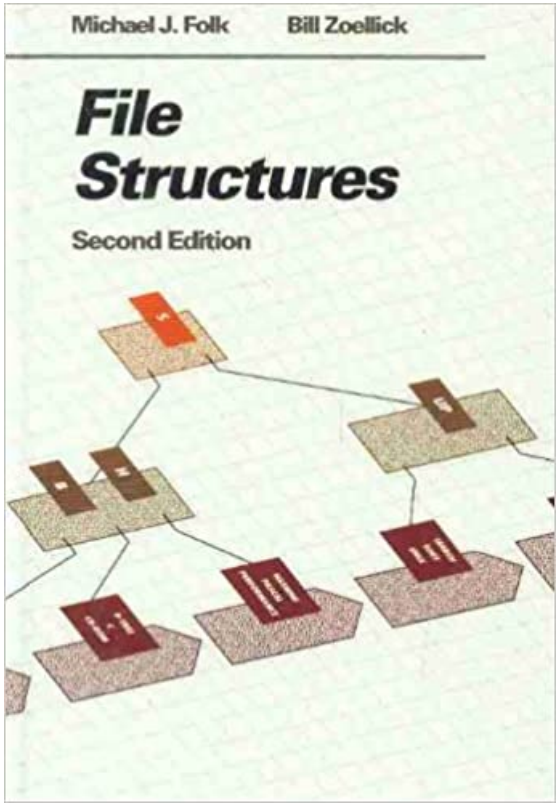
[Cormen et al, 2018]

[Drozdek, 2017]

Referências sugeridas



[Ziviani, 2010]



[Folk & Zoellick, 1992]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br