Algoritmo e Estrutura de Dados II
COM-112

Ordenação por Seleção
Selection Sort

Vanessa Souza

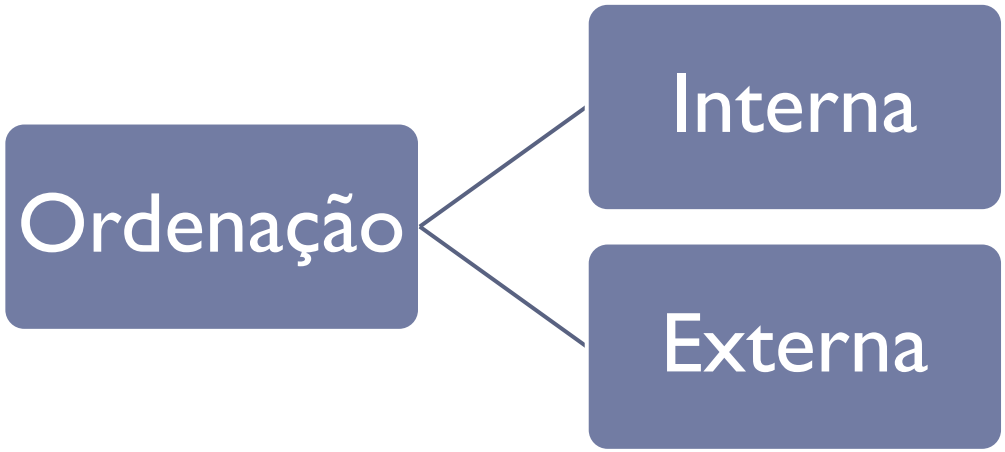
Universidade Federal de Itajubá – Instituto de Matemática e Computação

Ordenação

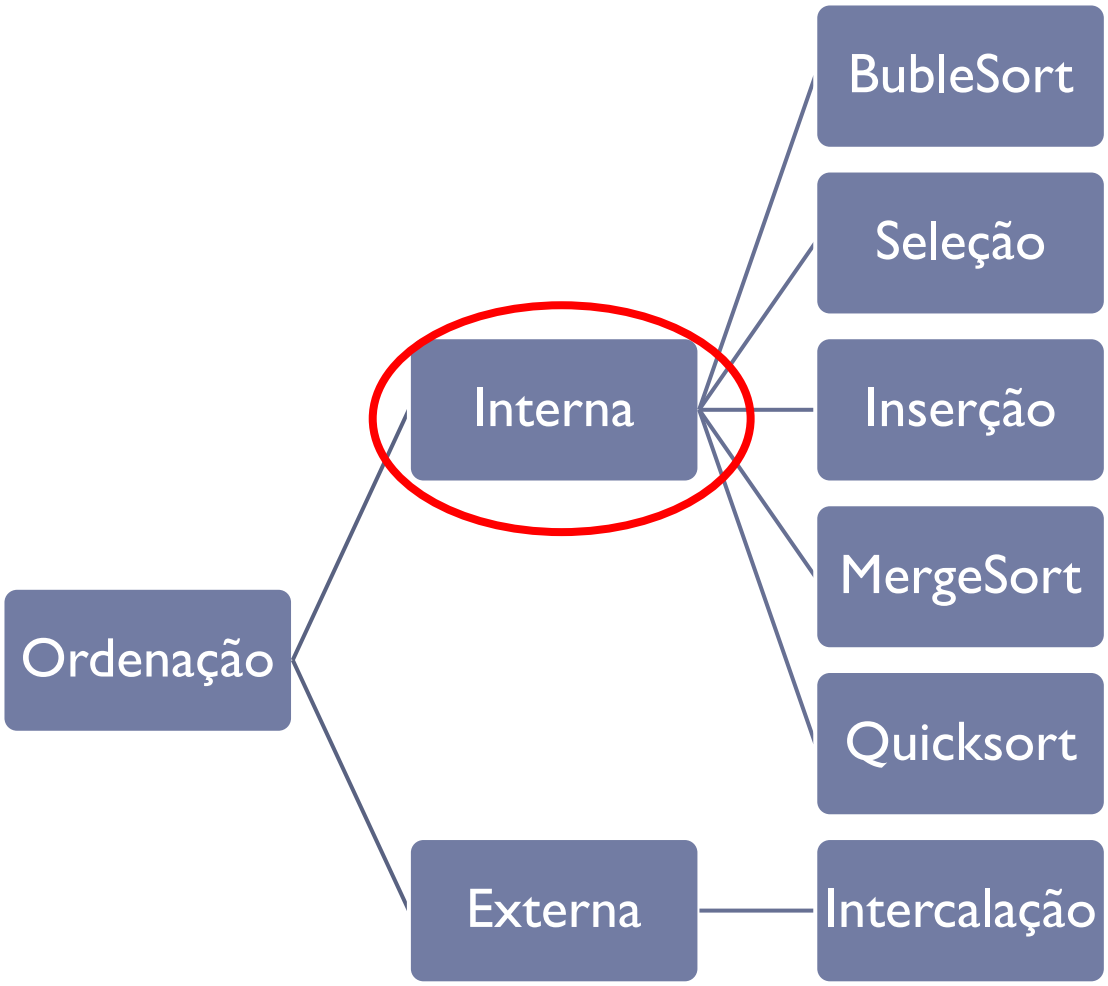
Ordenação

- Processo bastante utilizado na computação de uma estrutura de dados
- Ordenar significa colocar em ordem, segundo algum critério
- Alterar a ordem na qual os elementos de uma estrutura de dados aparece nessa estrutura
  - Rearranjar a estrutura

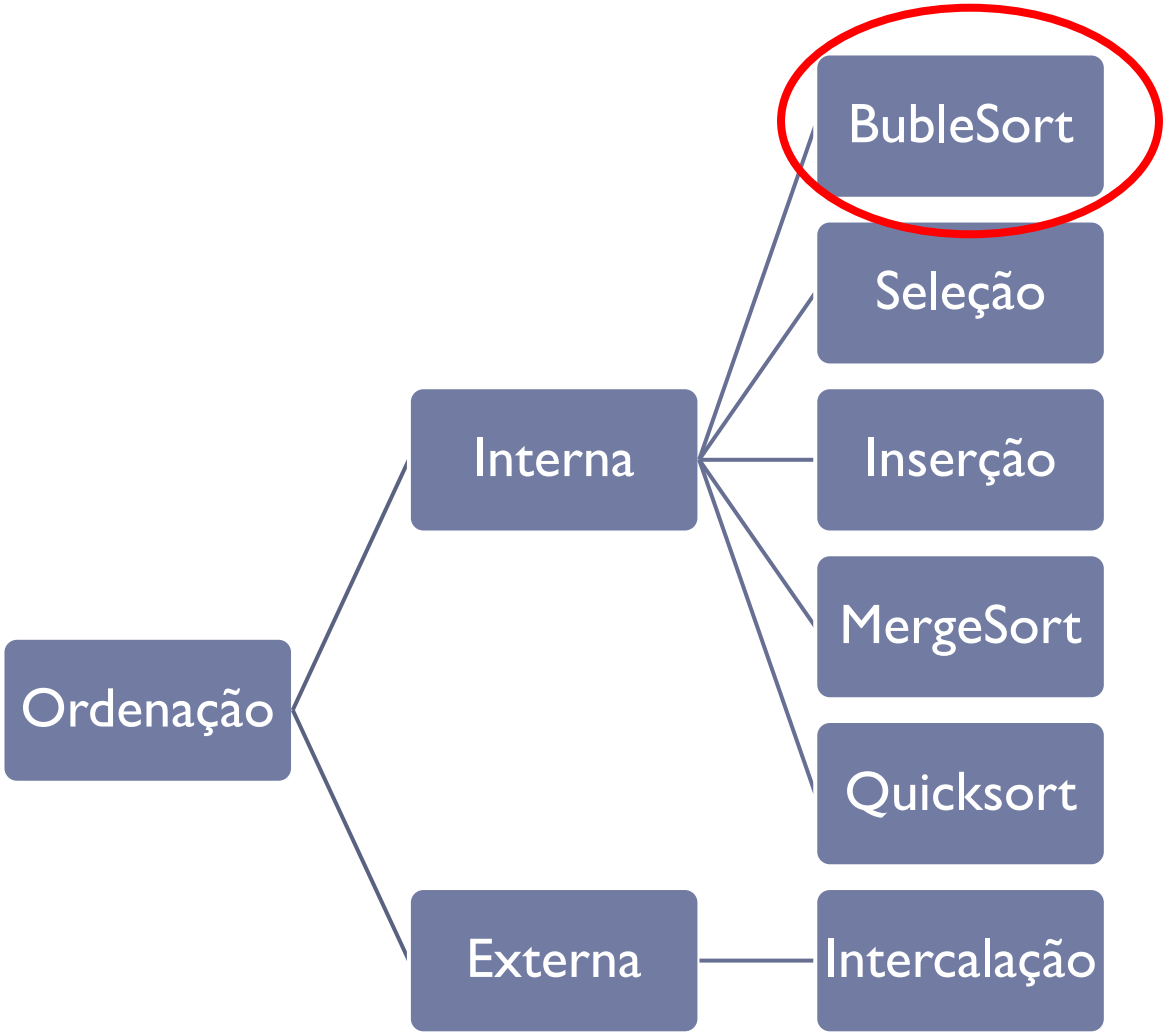
Classificação dos Métodos de Ordenação



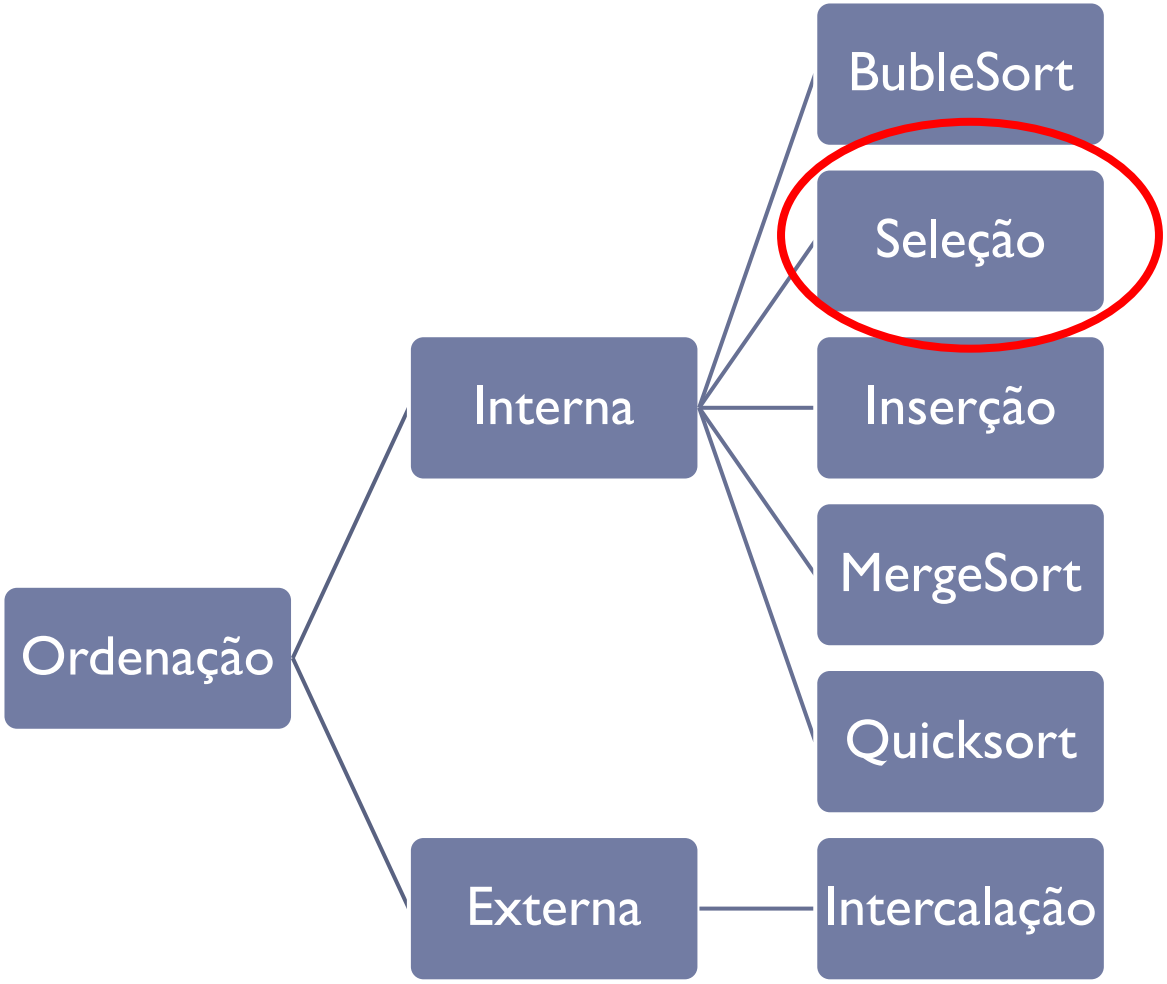
Classificação dos Métodos de Ordenação



Classificação dos Métodos de Ordenação



Classificação dos Métodos de Ordenação

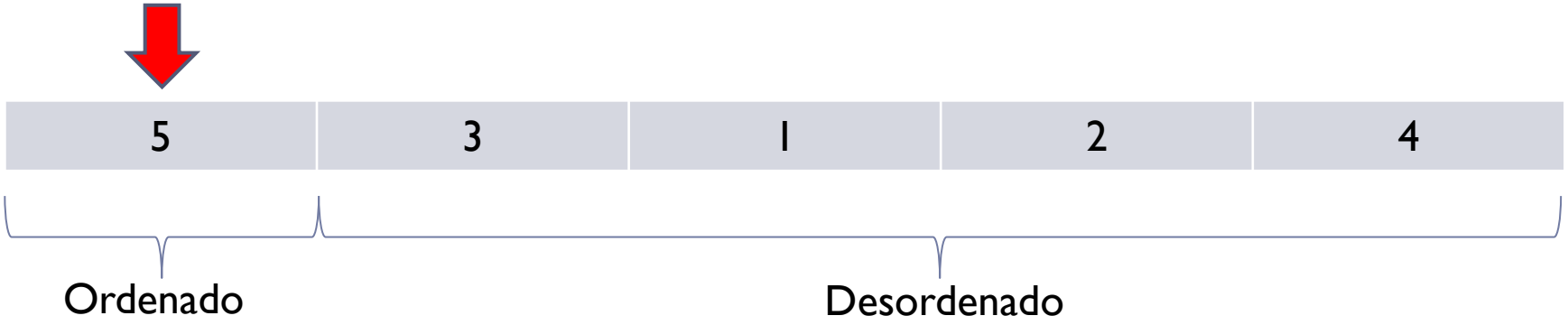


Seleção

Ordenação por Seleção

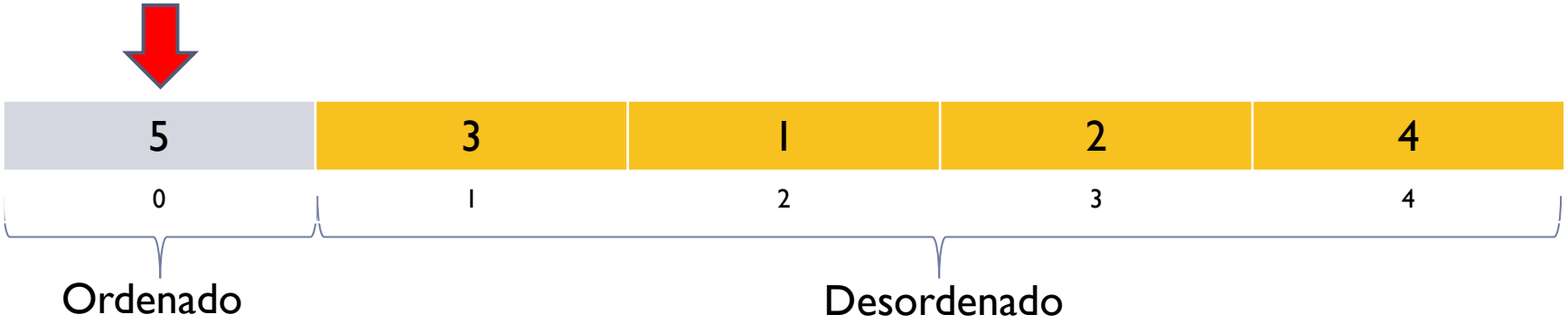
- Ideia
  1. **Divide o vetor em duas partes: ordenada e desordenada**
     - A divisão é feita usando um marcador (variável) que começa na posição 0 do vetor. Ou seja, a posição 0 está ordenada. O resto do vetor está desordenado.
  2. **Encontra o menor elemento na parte desordenada do vetor**
     - Uma busca é feita na parte desordenada do vetor para encontrar a posição do menor elemento
  3. **Verifica se o elemento do marcador é maior que o menor elemento encontrado**
     - Caso o elemento que está na posição do marcador seja maior que o menor elemento encontrado, eles trocam de posição.
  4. **Atualiza o marcador**
  5. **Repete o processo até a última posição do vetor**

Ordenação por Seleção



Divide o vetor na parte ordenada e desordenada utilizando uma variável chamada marcador.
O marcador é inicializado com 0.

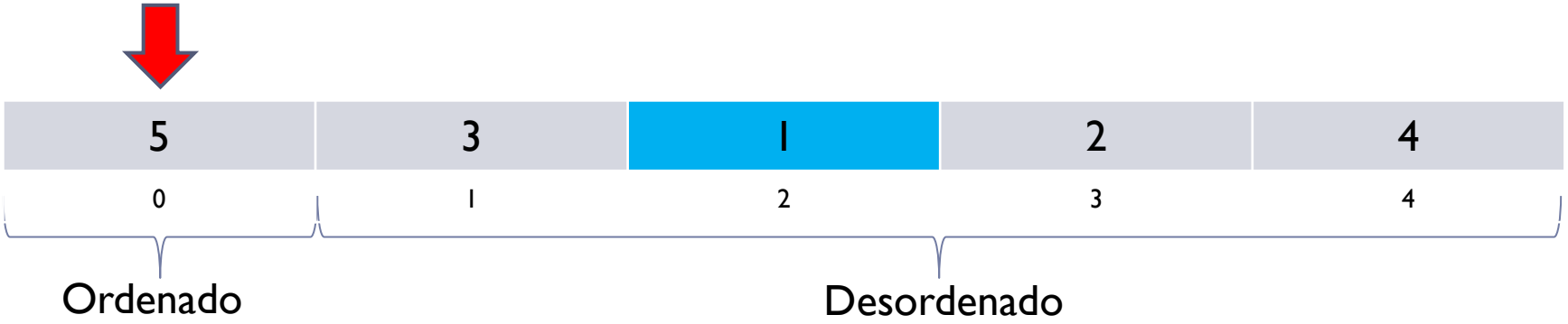
Ordenação por Seleção



Procura pelo menor elemento na parte desordenada do vetor.

Nesse caso, procura-se pela posição do menor elemento entre as posições 1 (marcador +1) e 4 (tam-1) do vetor.

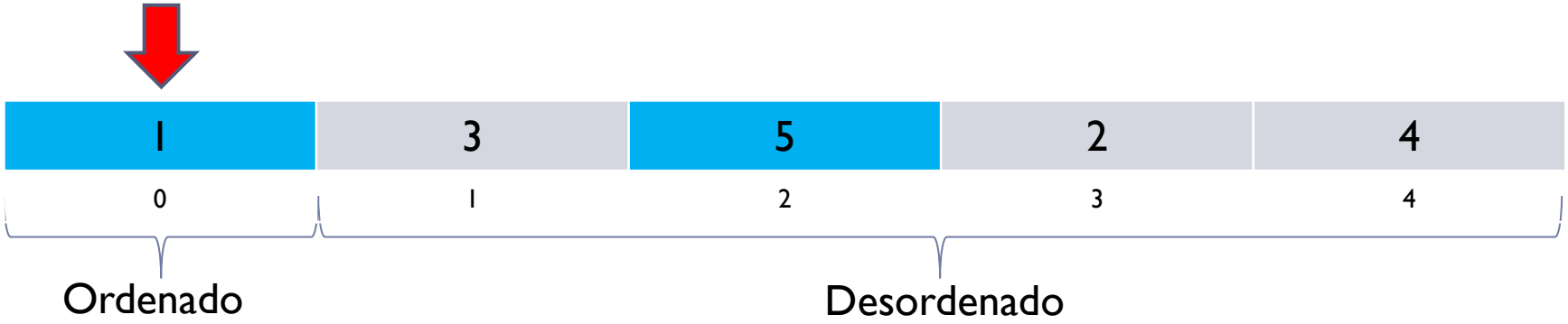
Ordenação por Seleção



Encontrado o menor elemento, verifica-se se ele é menor do que o elemento que está na posição do marcador.

Se for menor, os dois elementos trocam de posição.

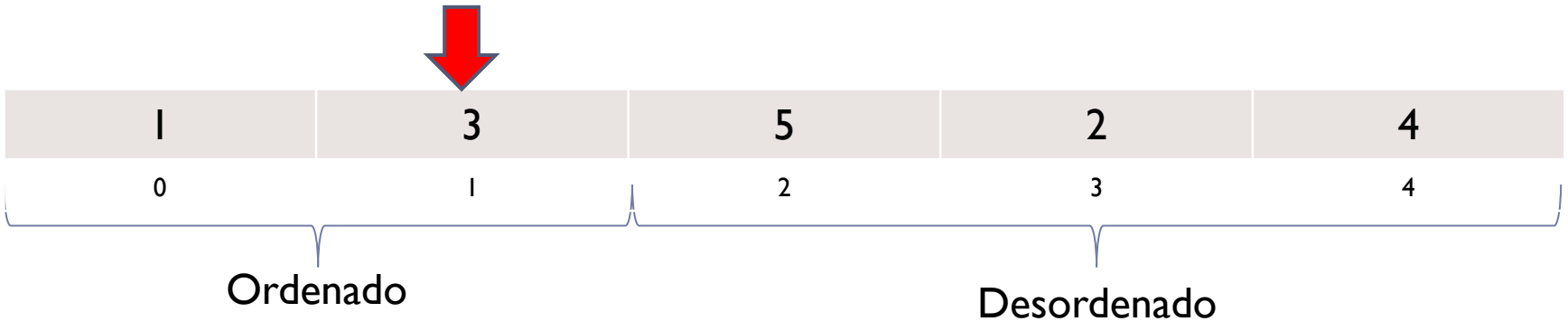
Ordenação por Seleção



Encontrado o menor elemento, verifica-se se ele é menor do que o elemento que está na posição do marcador.

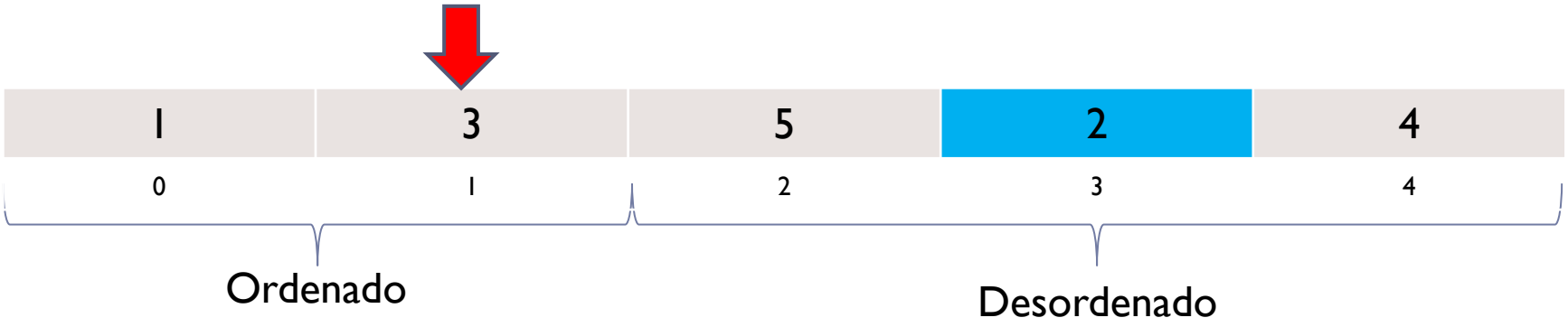
Se for menor, os dois elementos trocam de posição.

Ordenação por Seleção



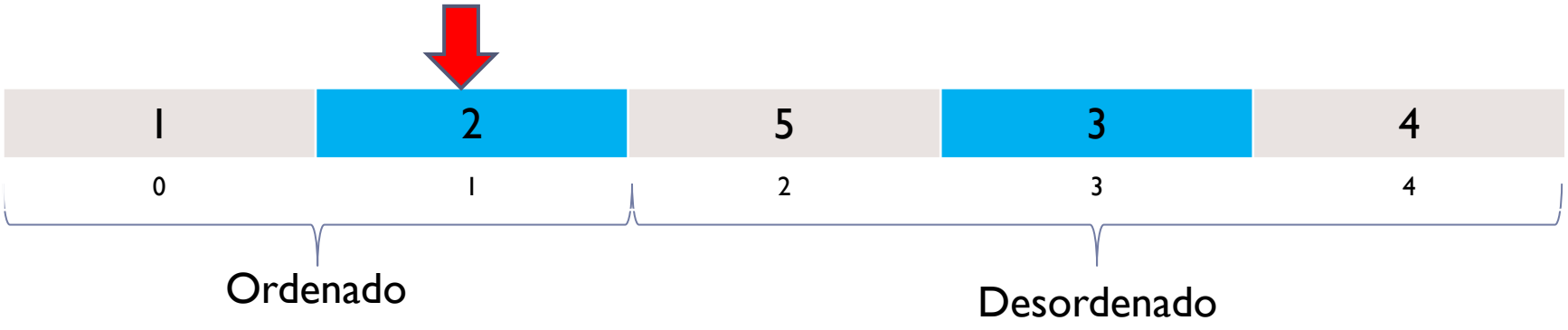
Atualiza o marcador

Ordenação por Seleção



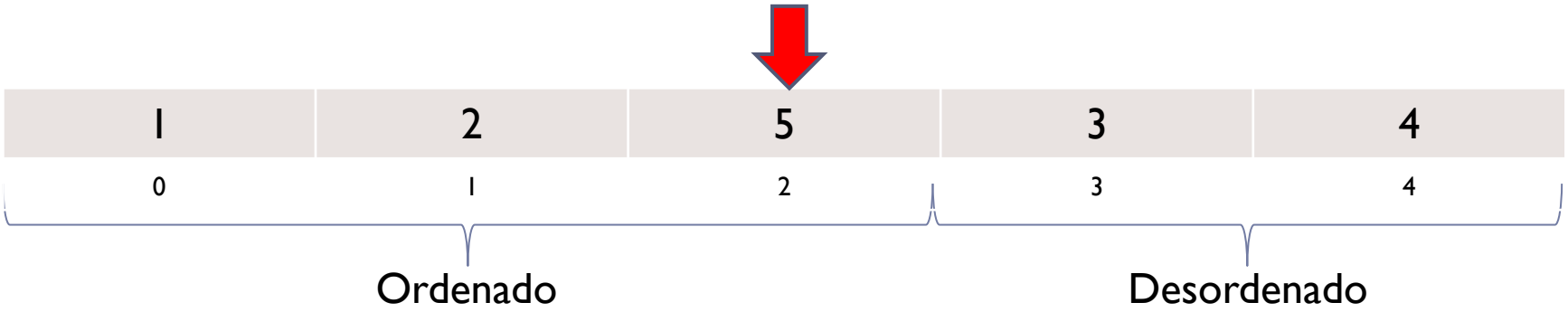
Procura pelo menor elemento na parte desordenada do vetor.

Ordenação por Seleção



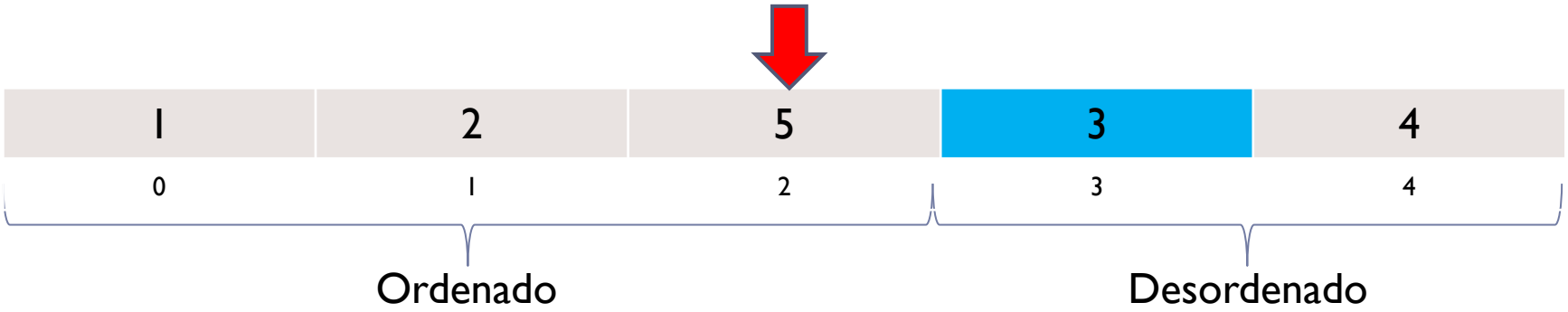
Troca de posição, se necessário

Ordenação por Seleção



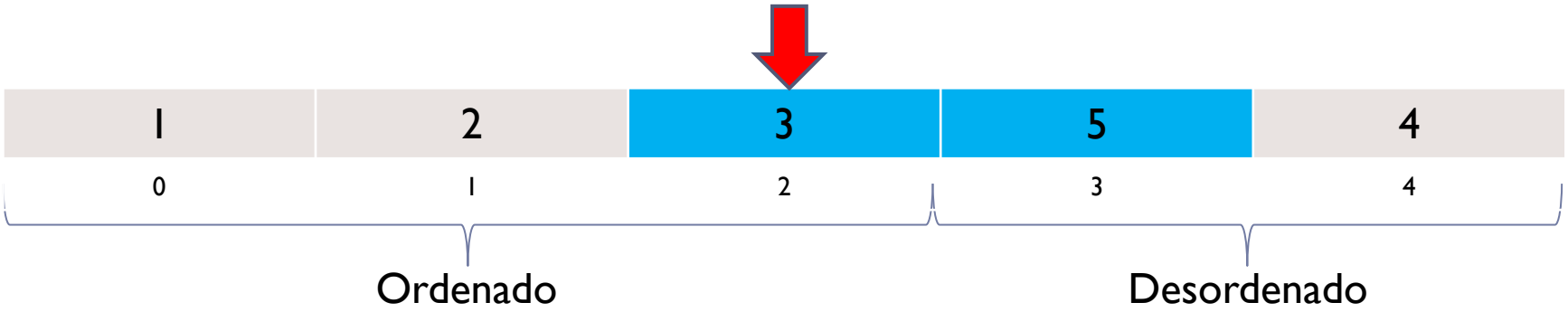
Atualiza o marcador

Ordenação por Seleção



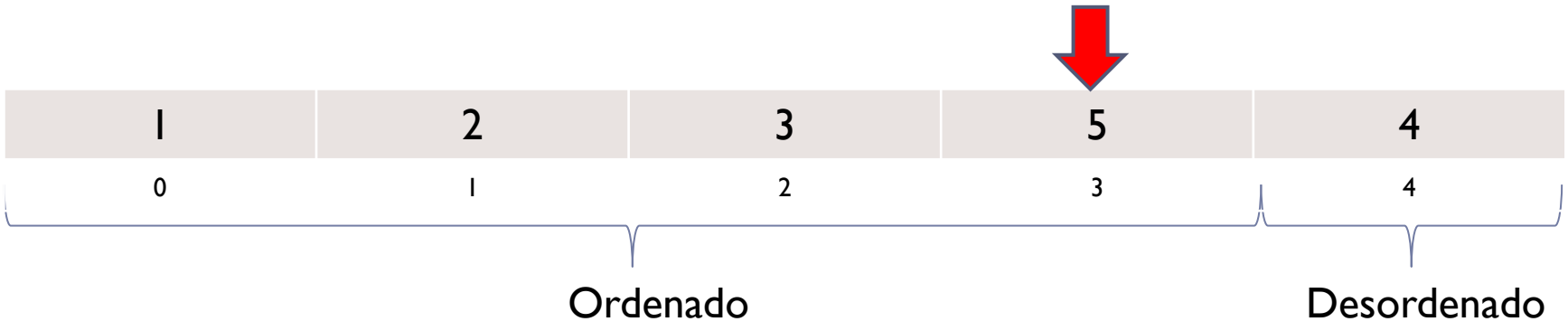
Procura pelo menor elemento na parte desordenada do vetor.

Ordenação por Seleção



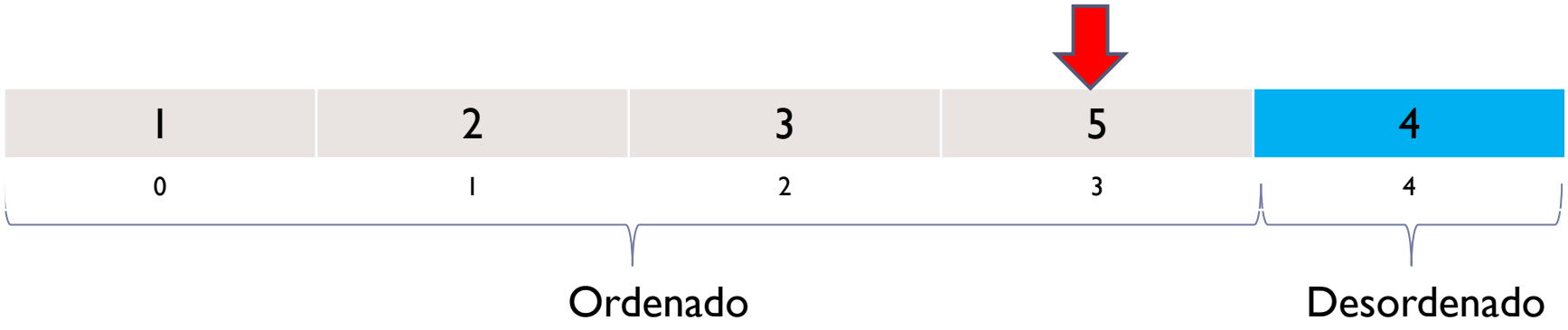
Troca de posição, se necessário

Ordenação por Seleção



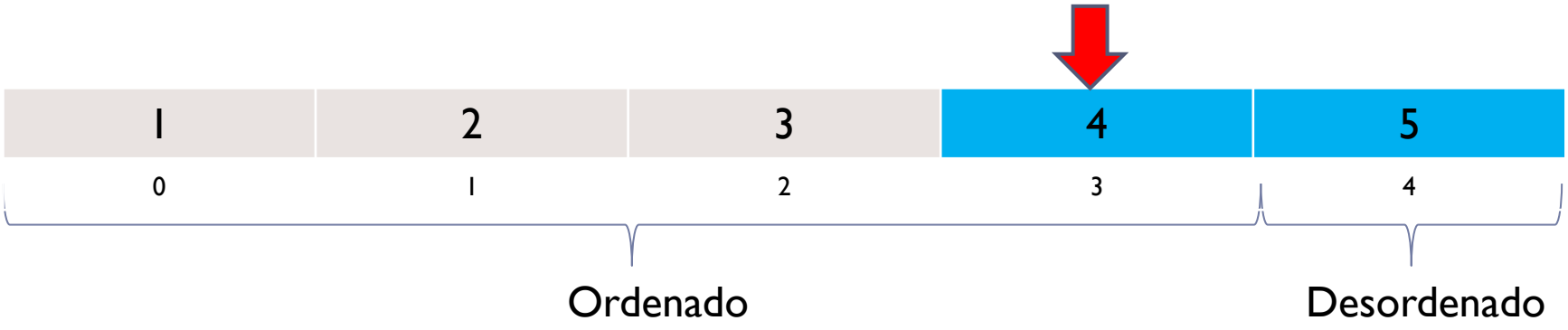
Atualiza o marcador

Ordenação por Seleção



Procura pelo menor elemento na parte desordenada do vetor.

Ordenação por Seleção



Troca de posição, se necessário

Ordenação por Seleção

Atualiza o marcador

Quando o marcador chegar na última posição do vetor, o mesmo estará ordenado.

Ordenação por Seleção

- Princípio da Ordenação
- Complexidade Assintótica

Ordenação por Seleção

- Algoritmo

Algoritmo: ORDENAÇÃO - SELEÇÃO

Entrada: Vet: Vetor de números naturais; **Tam:** Tamanho do vetor (inteiro)

Saída: Vet ordenado em ordem crescente

```
1  início
2      int marcador ← 0 ; // posição do marcador no vetor
3      int menor ; // posição do menor elemento no vetor desordenado
4      int aux ; // variável auxiliar para a troca
5      enquanto (marcador < Tam − 1) faça
6          menor ← encontraMenor(marcador + 1, tam, vet)
               \\menor recebe a posição do menor elemento na parte desordenada do vetor
           se Vet[menor] < Vet[marcador] então
7              aux ← Vet[menor]
8              Vet[menor] ← Vet[marcador]
9              Vet[marcador] ← aux
10         fim
11         marcador + +
12     fim
13 fim
```

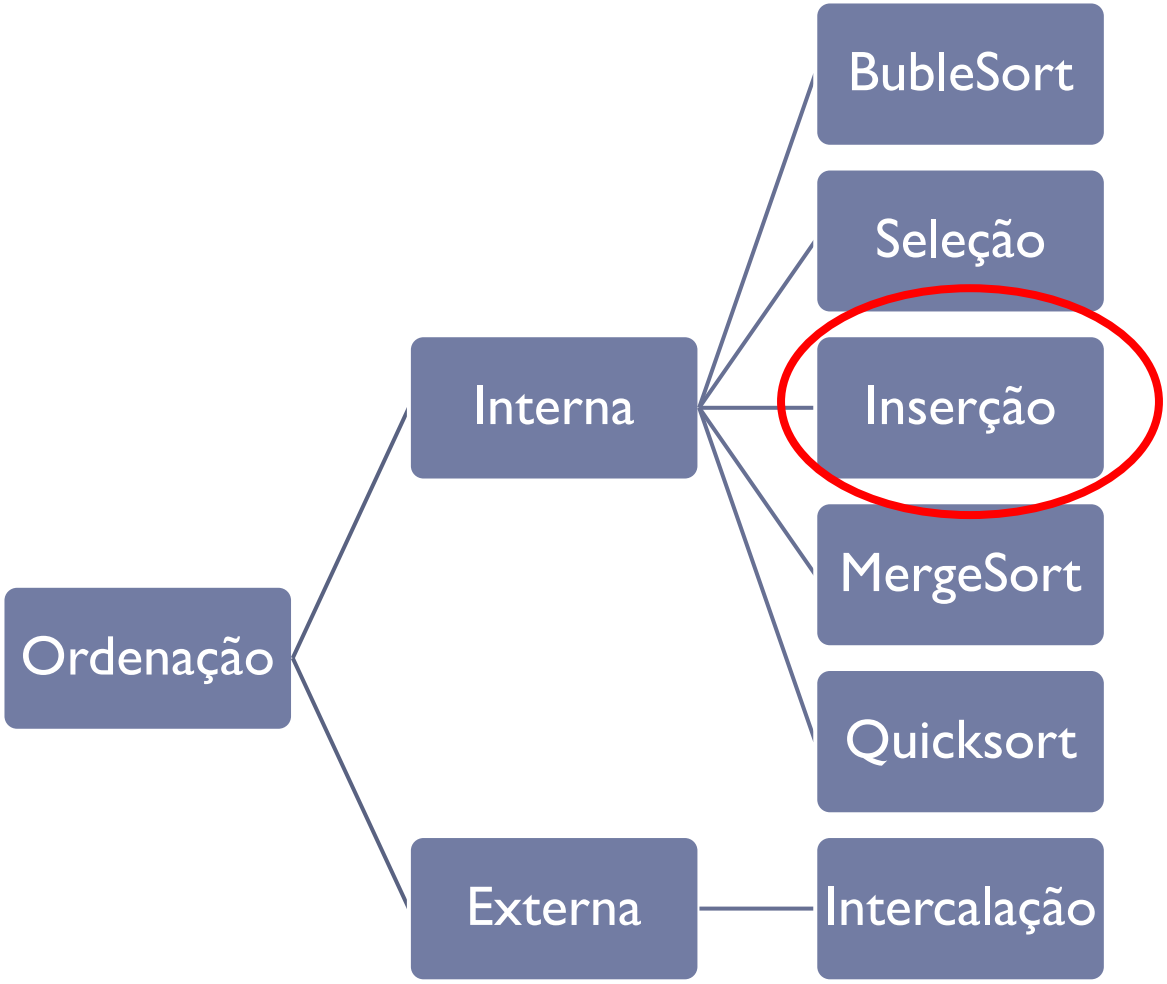
```
int encontraMenor(int inicio, int fim, int vet[]);
\\Função que procura o menor elemento na parte desordenada do vetor.
\\Retorna a posição do menor elemento.
```

Ordenação por Seleção

- https://visualgo.net/en/sorting
- https://www.hackerearth.com/practice/algorithms/sorting/selection-sort/visualize/
- Qual variação é possível de ser feita para melhorar o algoritmo?

Ordenação por Inserção

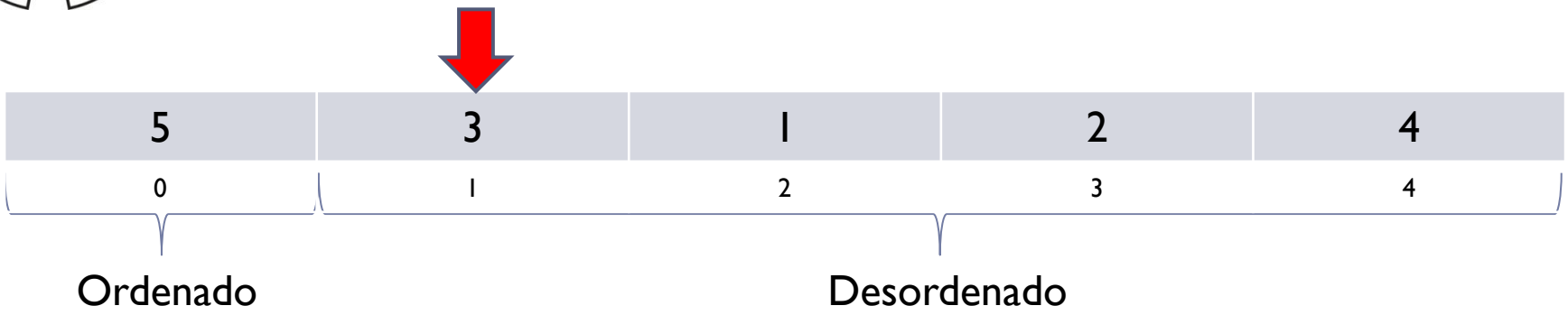
Classificação dos Métodos de Ordenação



Ordenação por Inserção

- Ideia
  1. **Divide o vetor em duas partes: ordenada e desordenada**
     - A divisão é feita usando um marcador (variável) que começa na posição 1 do vetor.
  2. **Insere o elemento que está na posição do marcador em seu local correto na parte ordenada do vetor**
     - O elemento do marcador é comparado com os elementos anteriores
     - A comparação é feita enquanto o elemento do marcador for menor que os anteriores OU até que se chegue ao início do vetor.
  3. **Atualiza o marcador**
  4. **Repete o processo até a última posição do vetor**

InsertionSort



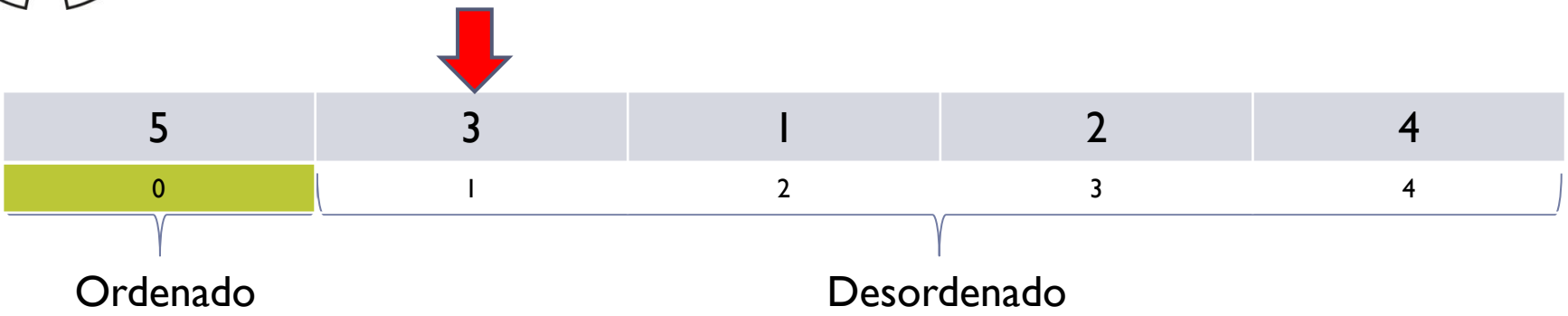
Procura na parte ordenada do vetor, o local correto de 3

Isso se faz comparando o 3 com os elementos anteriores a ele no vetor

A comparação é feita enquanto 3 for menor que os elementos anteriores OU chegar no início do vetor

Uma variável auxiliar (pos) controla essas comparações.

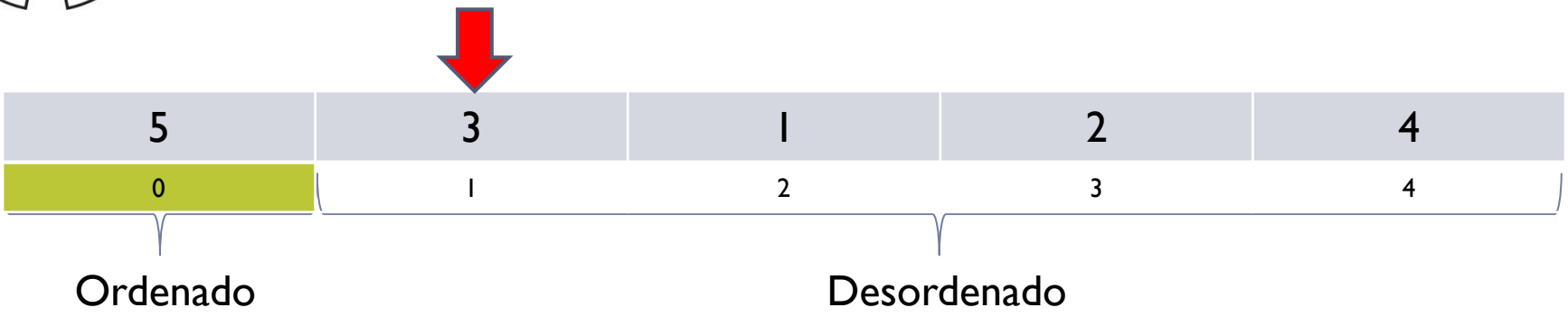
InsertionSort



Uma variável auxiliar (aux) é utilizada para armazenar o valor do elemento do marcador.

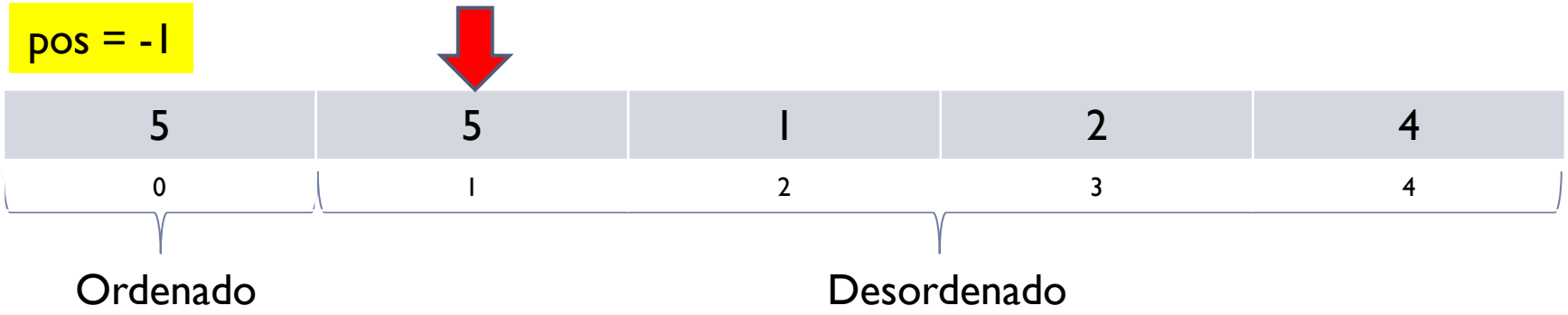
marcador = 1
aux = 3
pos = 0

InsertionSort

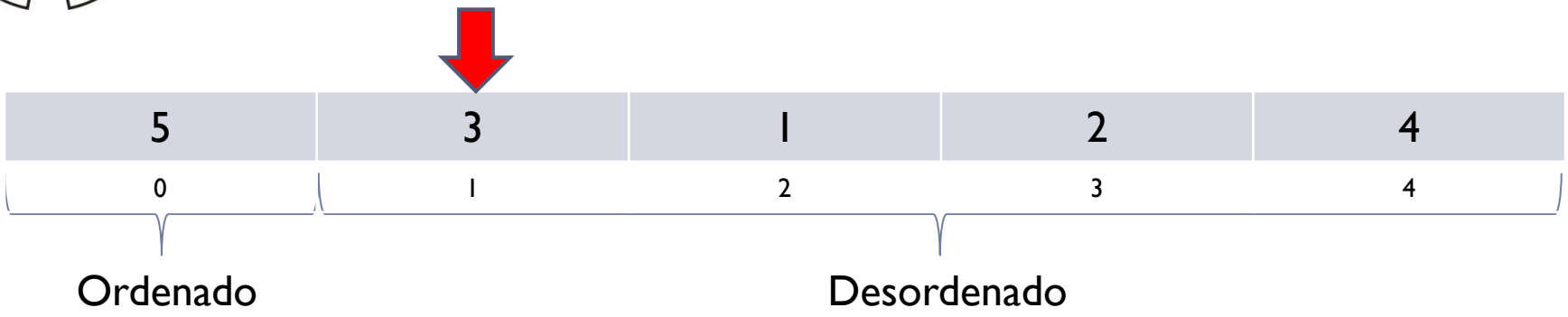


Compara o 3 com 5.
Se 3 é menor que 5, o 5 é copiado para a posição do 3.
A variável pos é atualizada (pos--).

pos = -1

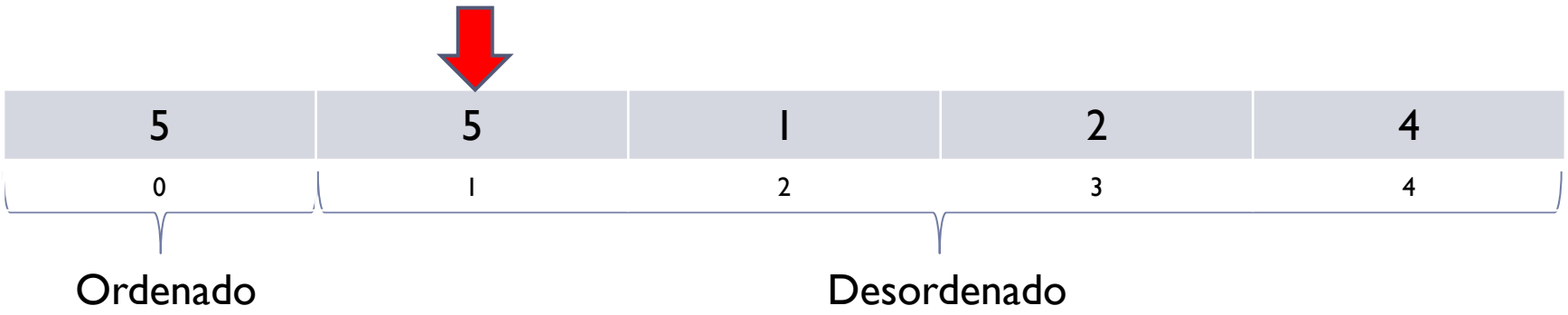


InsertionSort

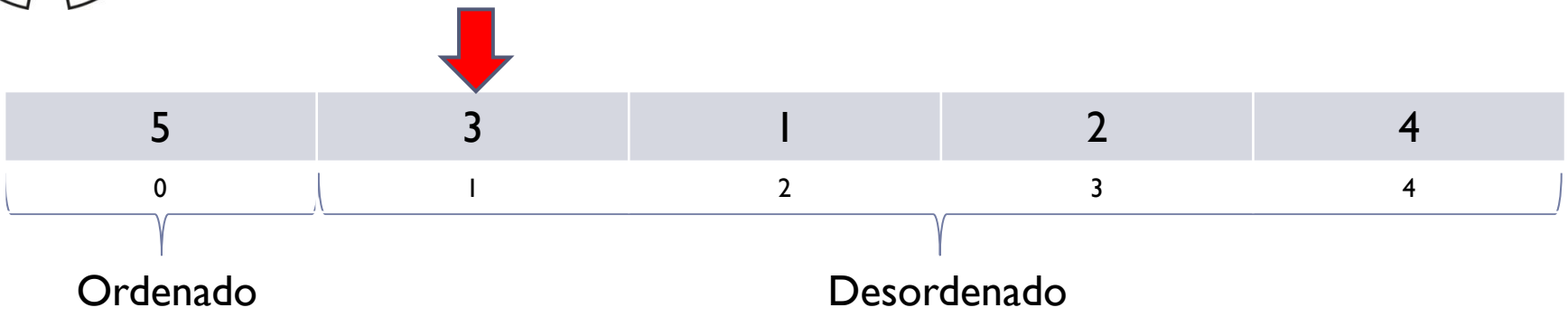


marcador = 1
aux = 3
pos = -1

Como pos é menor que 0, as comparações são encerradas.

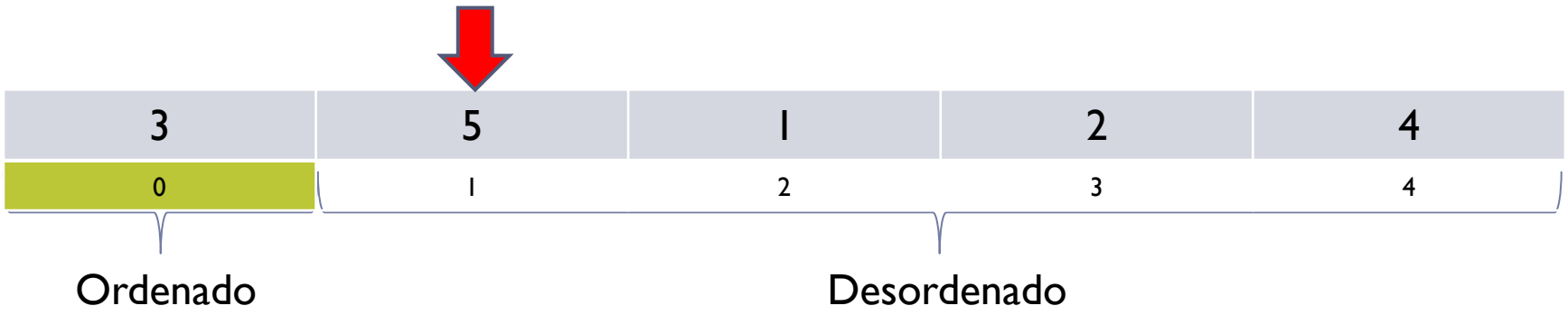


InsertionSort

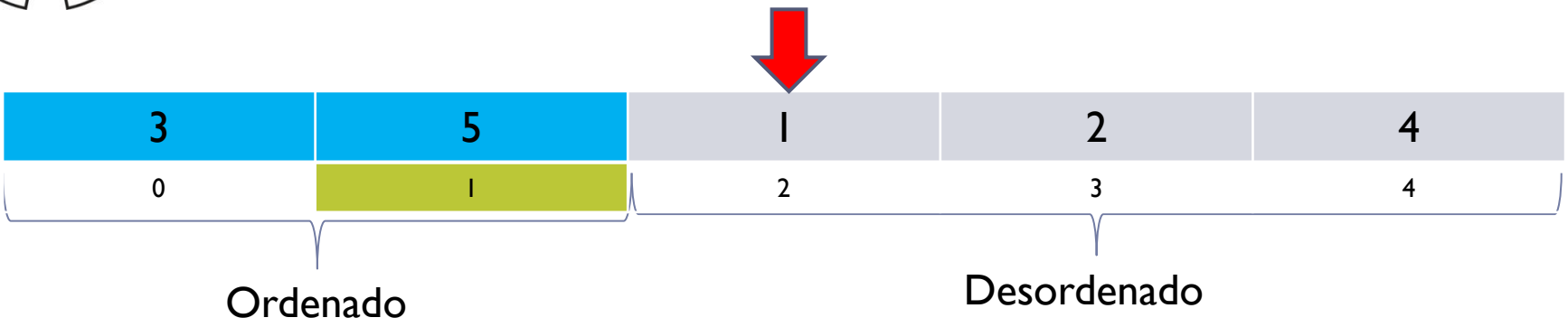


marcador = 1
aux = 3
pos = -1

O valor que está em aux é copiada para vet[pos+1].



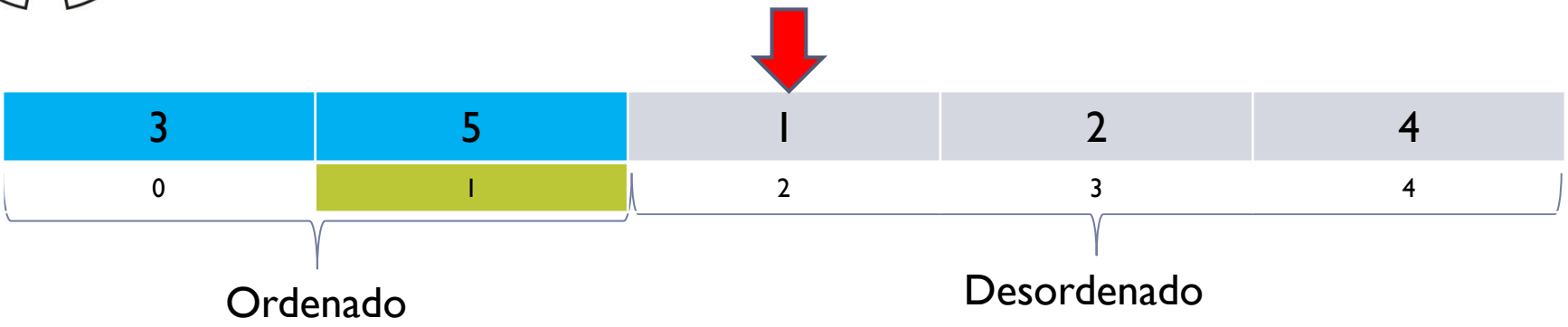
InsertionSort



Atualiza as variáveis

marcador = 2
aux = 1
pos = 1

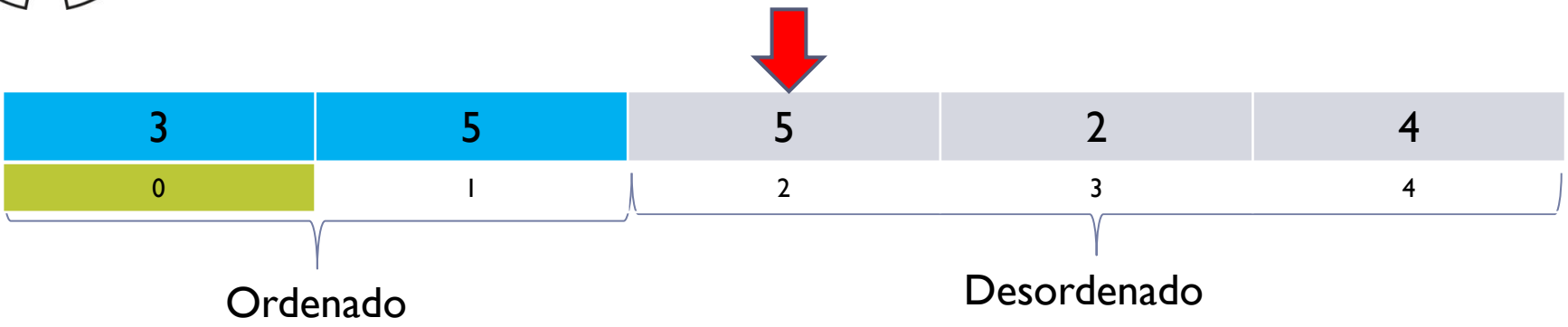
InsertionSort



Compara o 1 com os elementos anteriores.

marcador = 2
aux = 1
pos = 1

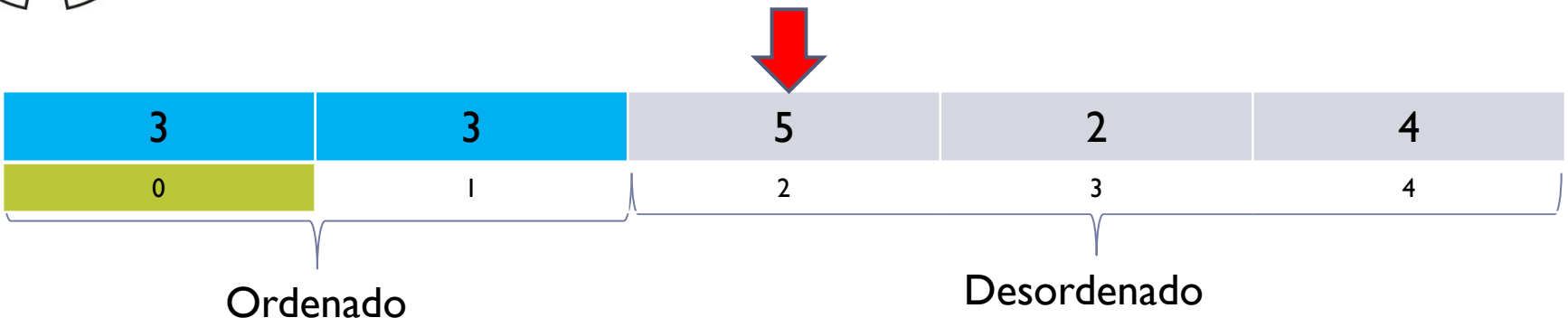
InsertionSort



Compara o 1 com todos os elementos anteriores.

marcador = 2
aux = 1
pos = 0

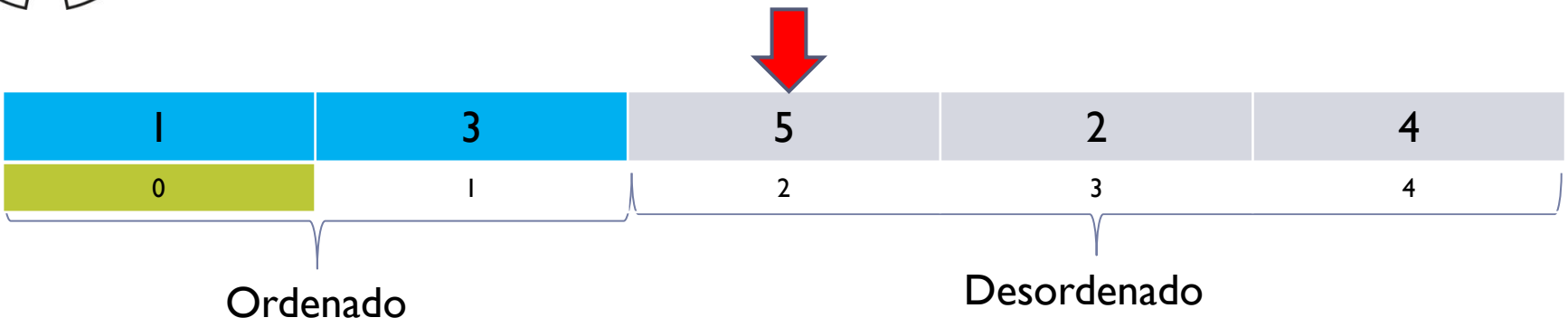
InsertionSort



Compara o 1 com todos os elementos anteriores.

marcador = 2
aux = 1
pos = 0

InsertionSort



Copia aux para vet[pos + 1].

marcador = 2
aux = 1
pos = -1

InsertionSort

Atualiza as variáveis

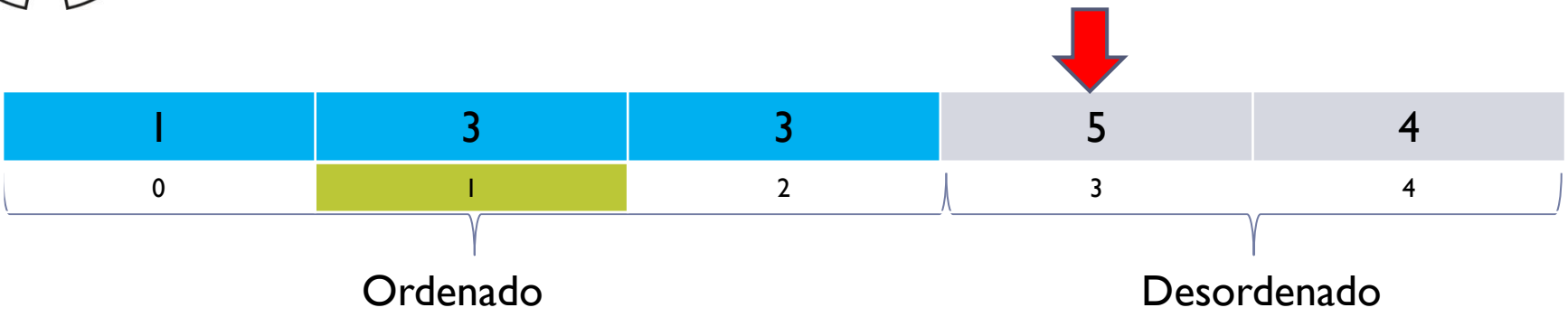
marcador = 3
aux = 2
pos = 2

InsertionSort

Compara o 2 com os elementos anteriores.

marcador = 3
aux = 2
pos = 2

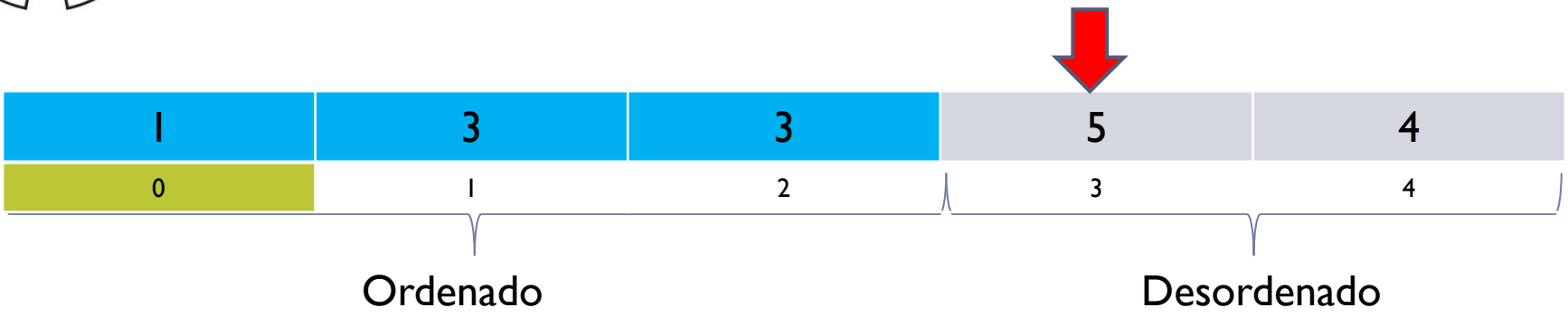
InsertionSort



Compara o 2 com todos os elementos anteriores.

marcador = 3
aux = 2
pos = 1

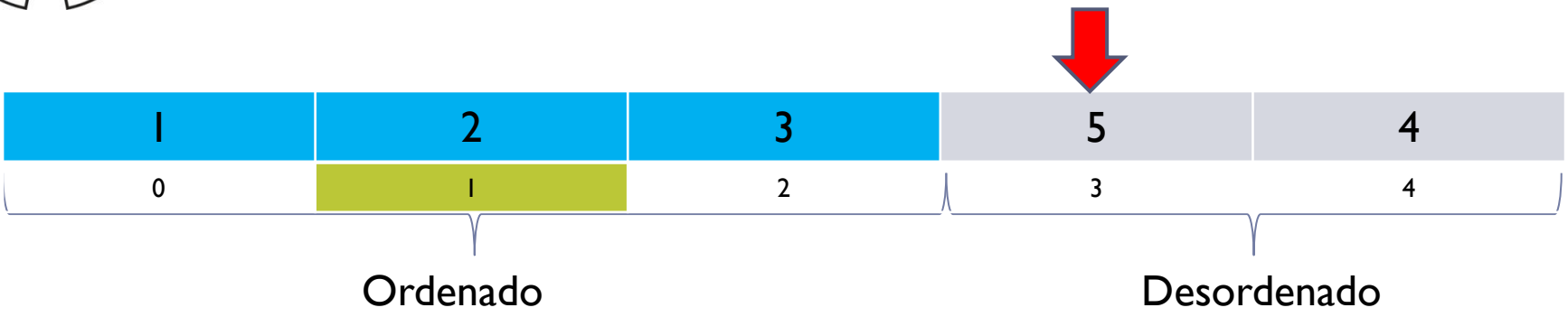
InsertionSort



Compara o 2 com todos os elementos anteriores.

marcador = 3
aux = 2
pos = 0

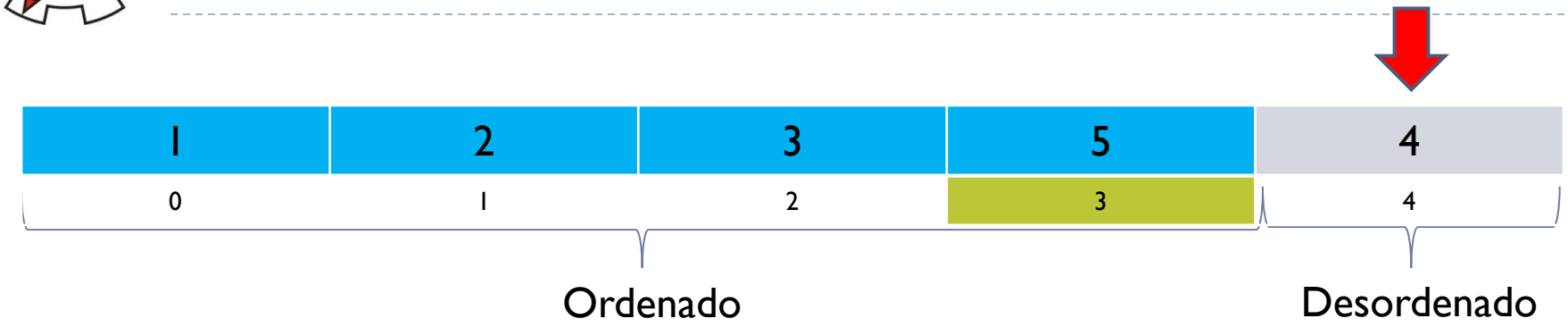
InsertionSort



Copia aux para vet[pos + 1].

marcador = 3
aux = 2
pos = 0

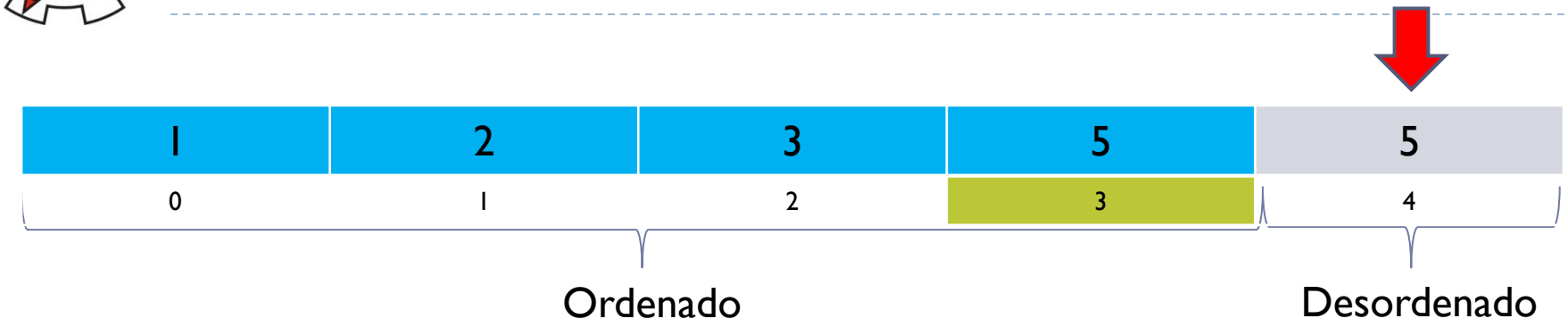
InsertionSort



Atualiza as variáveis

marcador = 4
aux = 4
pos = 3

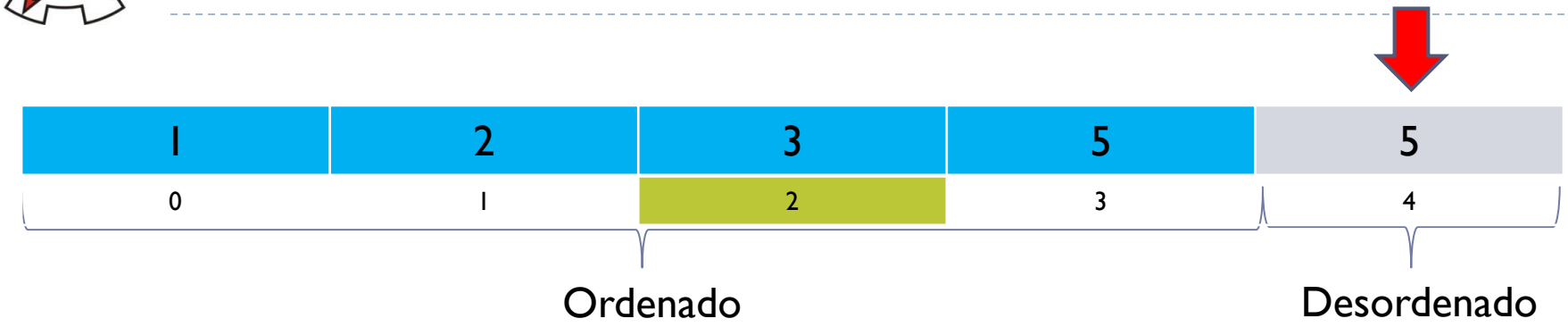
InsertionSort



Compara o 4 com os elementos anteriores.

marcador = 4
aux = 4
pos = 3

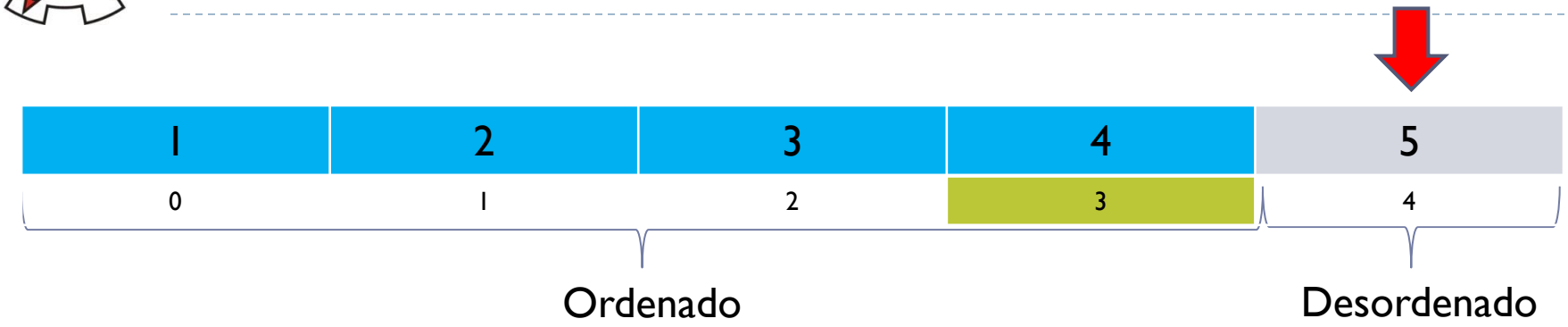
InsertionSort



Compara o 4 com os elementos anteriores.

marcador = 4
aux = 4
pos = 2

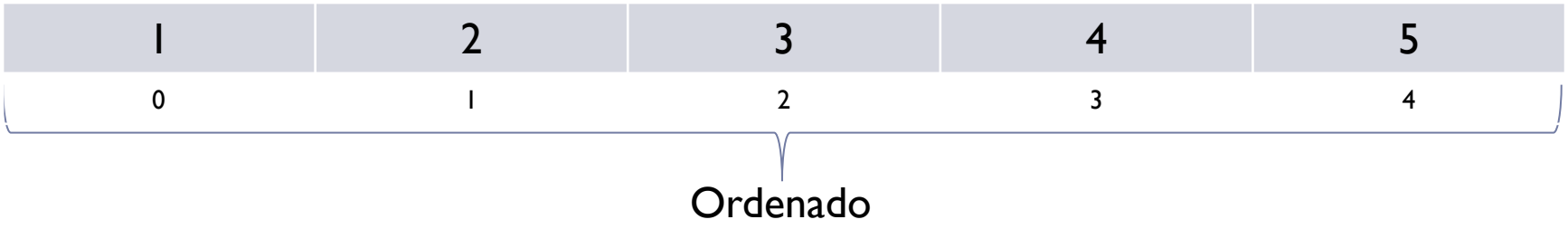
InsertionSort



Copia aux para vet[pos + 1].

marcador = 4
aux = 4
pos = 2

InsertionSort



Atualiza as variáveis

Como marcador é maior que 4, o procedimento para.
Garantia de vetor ordenado.

Ordenação por Inserção

- Princípio da Ordenação
- Complexidade Assintótica

InsertionSort

- Algoritmo

Algoritmo: ORDENAÇÃO - INSERÇÃO

Entrada: **Vet:** Vetor de números naturais; **Tam:** Tamanho do vetor (inteiro)

Saída: **Vet** ordenado em ordem crescente

```
1  início
2      int marcador ; // posição do marcador no vetor
3      int aux ; // valor armazenado na posição do marcador
4      int pos ; // percorre a parte ordenada do vetor
5      para (marcador ← 1 até Tam) faça
6          pos ← marcador − 1
7          aux ← Vet[marcador]
8          enquanto (aux < vet[pos]) E (pos ≥ 0) faça
9              Vet[pos + 1] ← Vet[pos]
10             pos ← pos − 1
11         fim
12         Vet[pos + 1] ← aux
13     fim
14 fim
```

Ordenação por Inserção

- https://visualgo.net/en/sorting
- https://www.hackerearth.com/practice/algorithms/sorting/selection-sort/visualize/

Comparação entre os métodos

Comparação entre os métodos

Algoritmo	Melhor Caso*	Caso Médio	Pior Caso	Estratégia
Bolha	$O(n^2)$	$O(n^2)$	$O(n^2)$	Maior elemento na última posição
Bolha Inteligente	$O(n)$	$O(n^2)$	$O(n^2)$	
Seleção	$O(n^2)$	$O(n^2)$	$O(n^2)$	Menor elemento na primeira posição
Inserção	$O(n)$	$O(n^2)$	$O(n^2)$	Cada elemento em sua posição correta na parte ordenada do vetor

* vetor ordenado

Comparação entre os métodos

- Os algoritmos vistos até agora são muito citados por sua simplicidade.
- Todos possuem complexidade assintótica do pior caso de $O(n^2)$.
- Costumam ser bons para arquivos pequenos e já quase ordenados.

Exercício

- Usando os algoritmos de ordenação seleção e inserção, ordene os vetores abaixo avaliando o número de trocas e comparações. Qual foi o melhor e o pior caso de cada algoritmo?

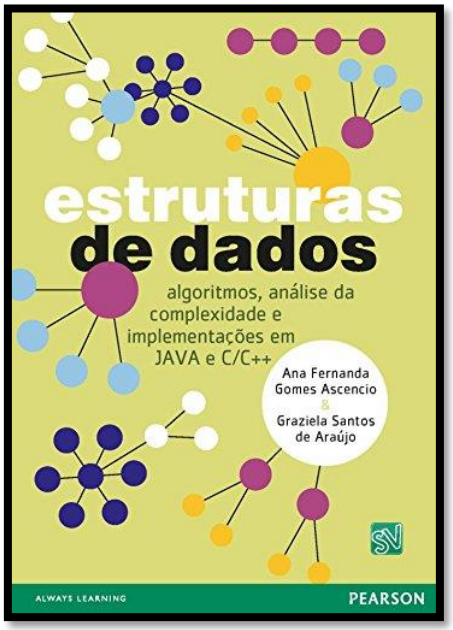
a) 12, 43, 1, 6, 56, 23

b) 10, 25, 36, 47, 88, 92

a) 92, 88, 47, 36, 25, 10

Leitura Recomendada

- Capítulos 1 e 2 do livro **Estruturas de Dados** – Ascencio & Araújo



Leitura Recomendada

- Livro **Projeto de Algoritmos** – Nivio Ziviani

- Capítulo 1.3
- Capítulo 4
  - 4.1.1 e 4.1.2