

# ESTRUTURA DE DADOS

Prof.<sup>a</sup> Priscilla Abreu

[priscilla.braz@rj.senac.br](mailto:priscilla.braz@rj.senac.br)



# Estrutura de dados



## Roteiro de Aula

- Objetivo da aula
- Método de ordenação
  - Quicksort
- Revisão e dúvidas

# ORDENAÇÃO

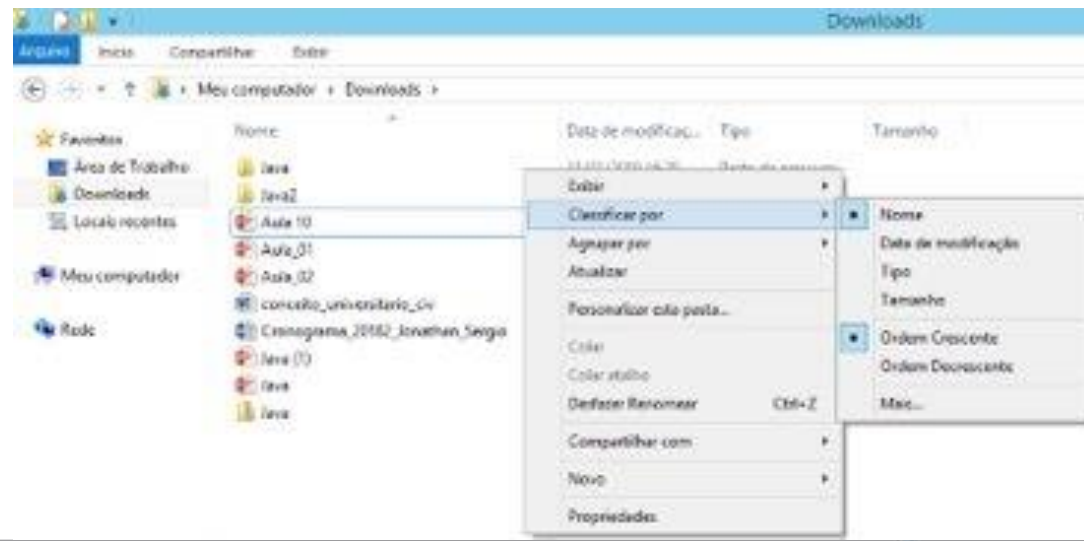
# Estrutura de dados



## ORDENAÇÃO

Em nosso cotidiano, nos deparamos com situações que precisam de alguma lógica para serem entendidas ou resolvidas.

Quando construímos um algoritmo, por exemplo, procuramos "ordenar" as instruções segundo alguma lógica.



# Estrutura de dados



## ORDENAR

É o processo de reorganizar objetos, dados, ..., de acordo com algum critério, para que posteriormente facilite recuperá-lo.

Para ordenar dados ou informações devemos determinar um ou mais campos de classificação e, em seguida, sua ordem de classificação: crescente ou decrescente.

## ORDENAR

A ordenação é feita segundo um dos campos do registro da lista. Este campo é denominado chave.

Ordenar uma lista  $L$  (em ordem não decrescente) e obter uma permutação dos seus índices  $p_1, p_2, \dots, p_m$  tal que:

$$L[p_1] \leq L[p_2] \leq \dots \leq L[p_m]$$

# Estrutura de dados



## ORDENAÇÃO

Considere o seguinte vetor

Como você ordenaria esses elementos de modo crescente?

48	37	15	25	29
----	----	----	----	----

# Estrutura de dados



## Métodos de ordenação

- Método da Bolha (Bubble sort)
- Seleção
- Inserção Direta (Insertion sort)
- Quicksort



# QUICKSORT

# Estrutura de dados



## Divisão e Conquista

- Envolve a resolução de um problema com uma entrada grande;
- Para facilitar a resolução do problema, a entrada é quebrada em pedaços menores (DIVISÃO);
- Cada pedaço da entrada é então tratado separadamente (CONQUISTA);
- Ao final, os resultados parciais são combinados para gerar o resultado final procurado.

## QUICKSORT

Tanto o **Quicksort** como o **Bubblesort** são métodos de ordenação por troca dos elementos que estão sendo ordenados.

Enquanto o **Bubblesort** troca pares de elementos consecutivos, o **Quicksort** compara e troca pares de elementos distantes, o que acelera o processo de ordenação.

# Estrutura de dados



## QUICKSORT

Vamos assistir uma simulação desse método para entendermos seu funcionamento!

## QUICKSORT

<https://www.youtube.com/watch?v=xmhQMbEvVOQ>

# Estrutura de dados



## Método Quicksort

{5, 9, 6, 4, 3, 8, 7, 1}

{4, 1, 3} {5} {6, 8, 7, 9}

{3, 1} {4} {5} {6} {8, 7, 9}

{1} {3} {4} {5} {6} {7} {8} {9}

{1, 3, 4, 5, 6, 7, 8, 9}

## Método Quicksort – Partição

- Adota a estratégia de divisão e conquista;
- Um elemento  $x$  (pivô) será alocado na sua posição correta na lista ordenada produzindo 3 sublistas:
  1. A sublista que contém os elementos com chave menor ou igual à do elemento  $x$ ;
  2. A sublista formada unicamente pelo elemento  $x$ ;
  3. A sublista que contém os elementos com chave maior ou igual à do elemento  $x$ ;

# Estrutura de dados



## Método Quicksort – Partição

O elemento  $x$  (pivô) a ser alocado na posição correta poderá ser escolhido. Adotaremos como pivô o primeiro elemento da lista.

A mesma estratégia de partição será repetida nas sublistas obtidas em 1. e 3.

# Estrutura de dados



## Método Quicksort – Código

Vetor: `v` - declaração `int v[n];`

```
void quicksort (int ini, int fim){  
    int p;  
    if (ini < fim){  
        p= particao(ini, fim);  
        quicksort(ini, p-1);  
        quicksort(p + 1, fim);  
    }  
}
```

Divide a lista limitada por `ini` e `fim` em outras duas listas:

1- sublista com os elementos menores do que `v[p]`;

2- sublista com os elementos maiores do que `v[p]`.

5, 9, 6, 4, 3, 8, 7, 1

4, 1, 3

5

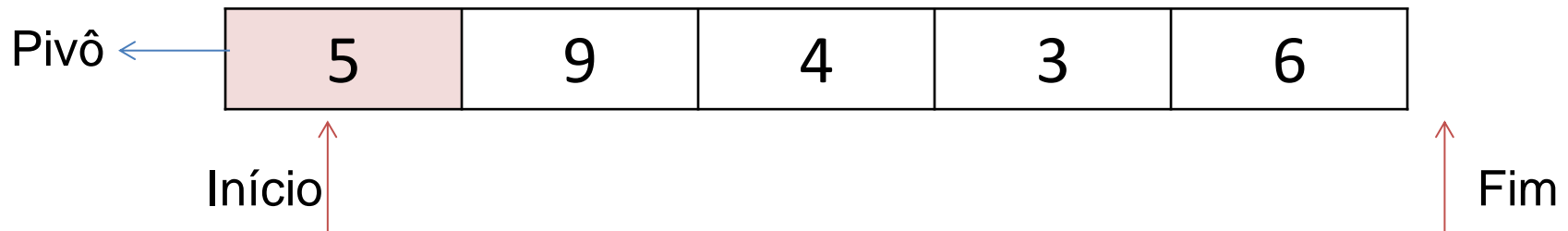
6, 8, 7, 9



# Estrutura de dados



## Método Quicksort – Partição



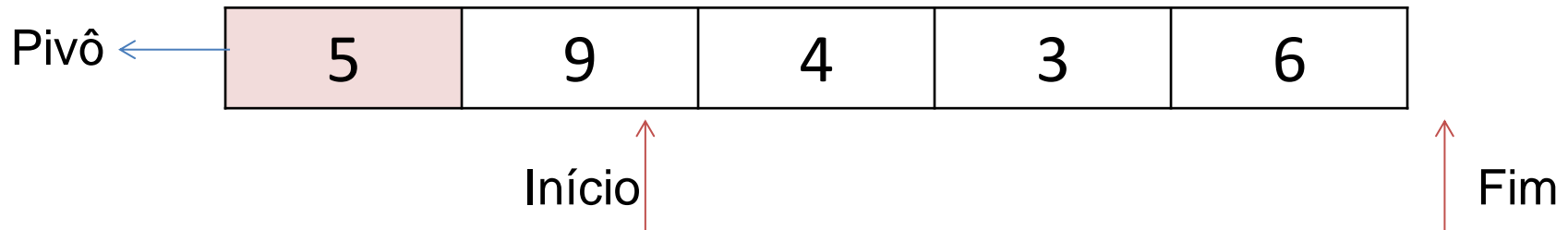
**enquanto (início < fim)**

**Movimento a variável início até  
encontrar um elemento maior que o pivô**

# Estrutura de dados



## Método Quicksort – Partição



**ENCONTREI!!!**

**Agora, movimento a variável Fim até encontrar um elemento menor do que o pivô.**

# Estrutura de dados



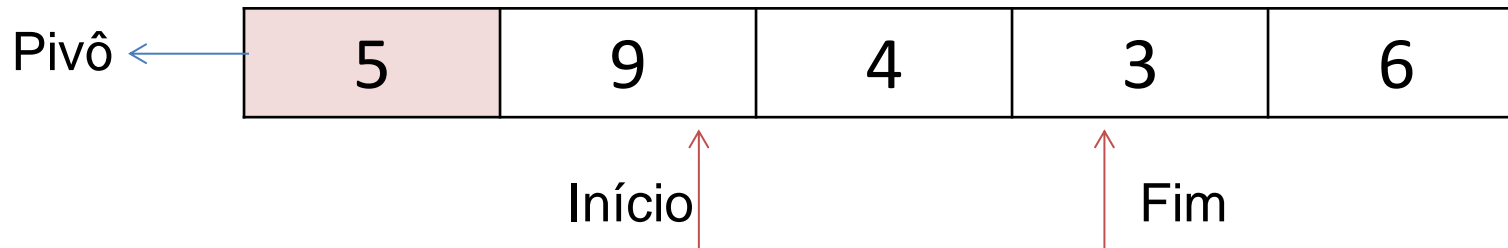
## Método Quicksort – Partição



# Estrutura de dados



## Método Quicksort – Partição

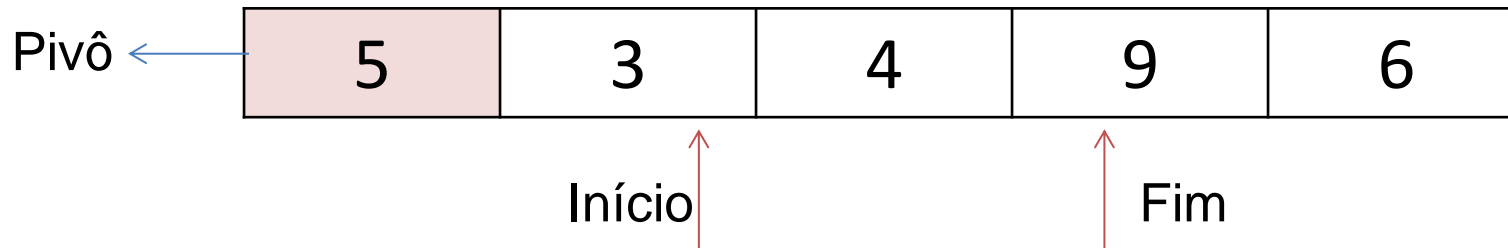


**Encontrei!!!**  
**Início < Fim => Troco os elementos**

# Estrutura de dados



## Método Quicksort – Partição

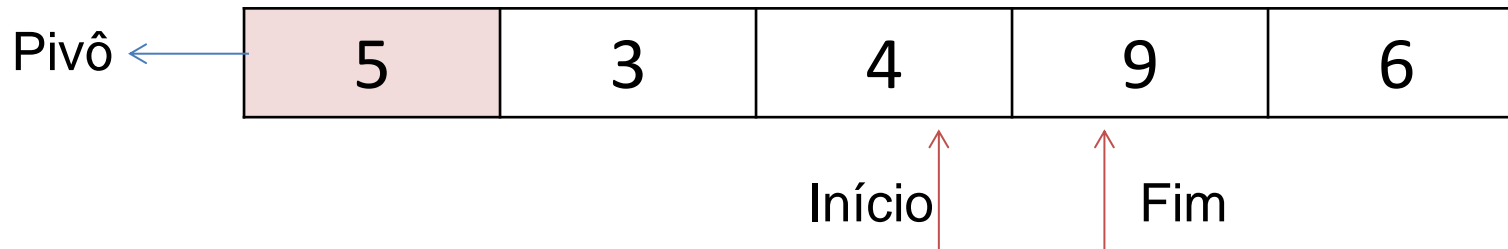


**Início < Fim???**  
**Repito o processo!**

# Estrutura de dados



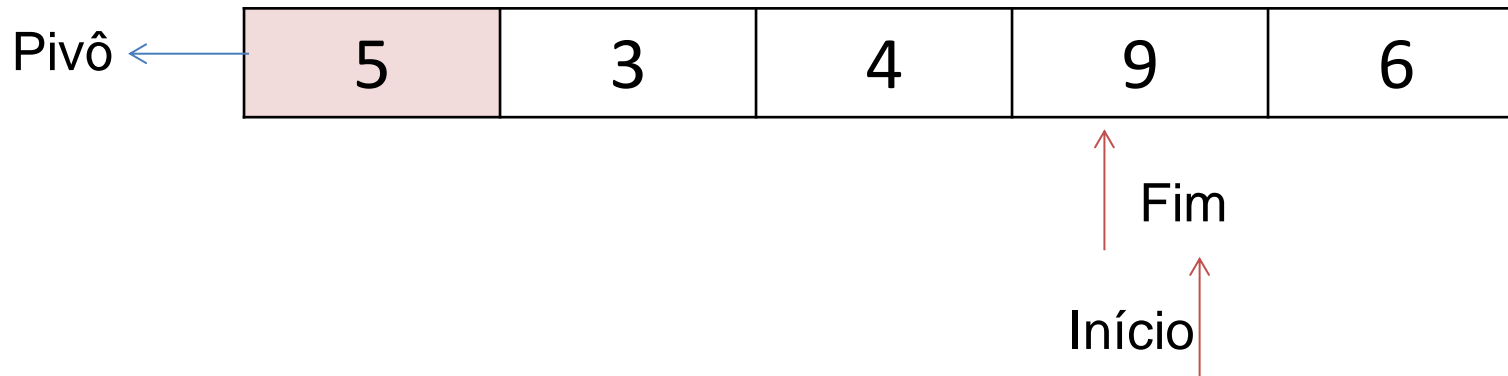
## Método Quicksort – Partição



# Estrutura de dados



## Método Quicksort – Partição

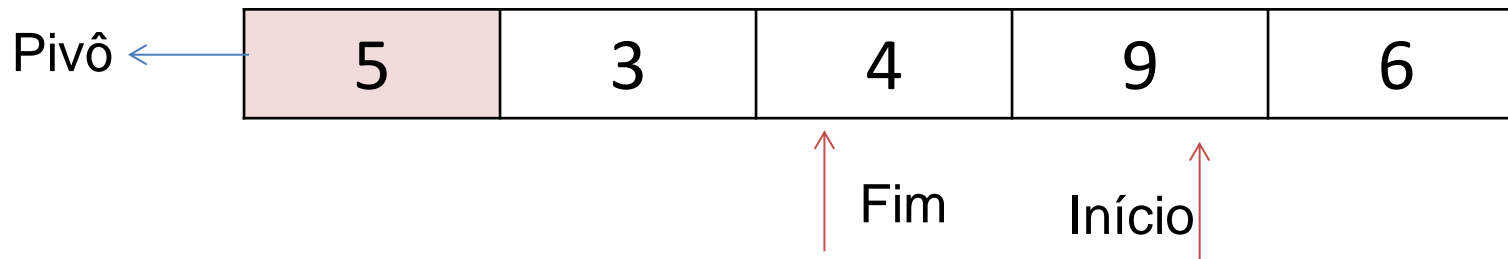


**Encontrei!!!**  
**Repetir o processo para a variável Fim.**

# Estrutura de dados



## Método Quicksort – Partição



**Encontrei!!!**

Início < Fim ???

NÃO!!! => Então não tem troca entre V[início] e V[fim]

Troco o pivô com o elemento da posição Fim.



# Estrutura de dados



## Método Quicksort – Partição

4	3	5	9	6
---	---	---	---	---

Agora, retornamos para o procedimento quicksort para fazer o mesmo processo para as sublistas à direita e à esquerda do pivô.

## Método Quicksort – Partição

```
int particao(int i, int f){  
    int ini, fim, pivo;  
    ini = i;    fim = f;    pivo= v[i];  
    while (ini < fim){  
        while (pivo >= v[ini])  
            ini++;  
        while (pivo < v[fim])  
            fim--;  
        if (ini < fim)  
            troca(ini, fim);  
    }  
    troca(i, fim);  
    return fim ;  
}
```

### Observação:

Ao final do procedimento, **dir** guardará a posição onde o elemento **v[i]** deve ser inserido. Isto é, todos os elementos após **dir** possuem chave maior ou igual a chave de **v[i]**.

## EXERCÍCIO

Para o método de ordenação Quicksort, apresente a ordenação do seguinte vetor.

Mostre a sequência de todas as etapas executadas (comparações e trocas).

$$V = \{5; 7; 2; 8; 1; 6\}$$

# Estrutura de dados



## Considerações

Busca e ordenação são operações frequentemente utilizadas em sistemas computacionais.

Desse modo, torna-se essencial conhecer os diferentes métodos e analisar o contexto de aplicação para decidir por qual deles utilizar.