

# EDCO4B

# ESTRUTURAS DE DADOS 2

Aula 05 - Merge 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](https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR)

# Roteiro



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

# Roteiro

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

# Introdução



**Algoritmos de  
Ordenação**

# Introdução

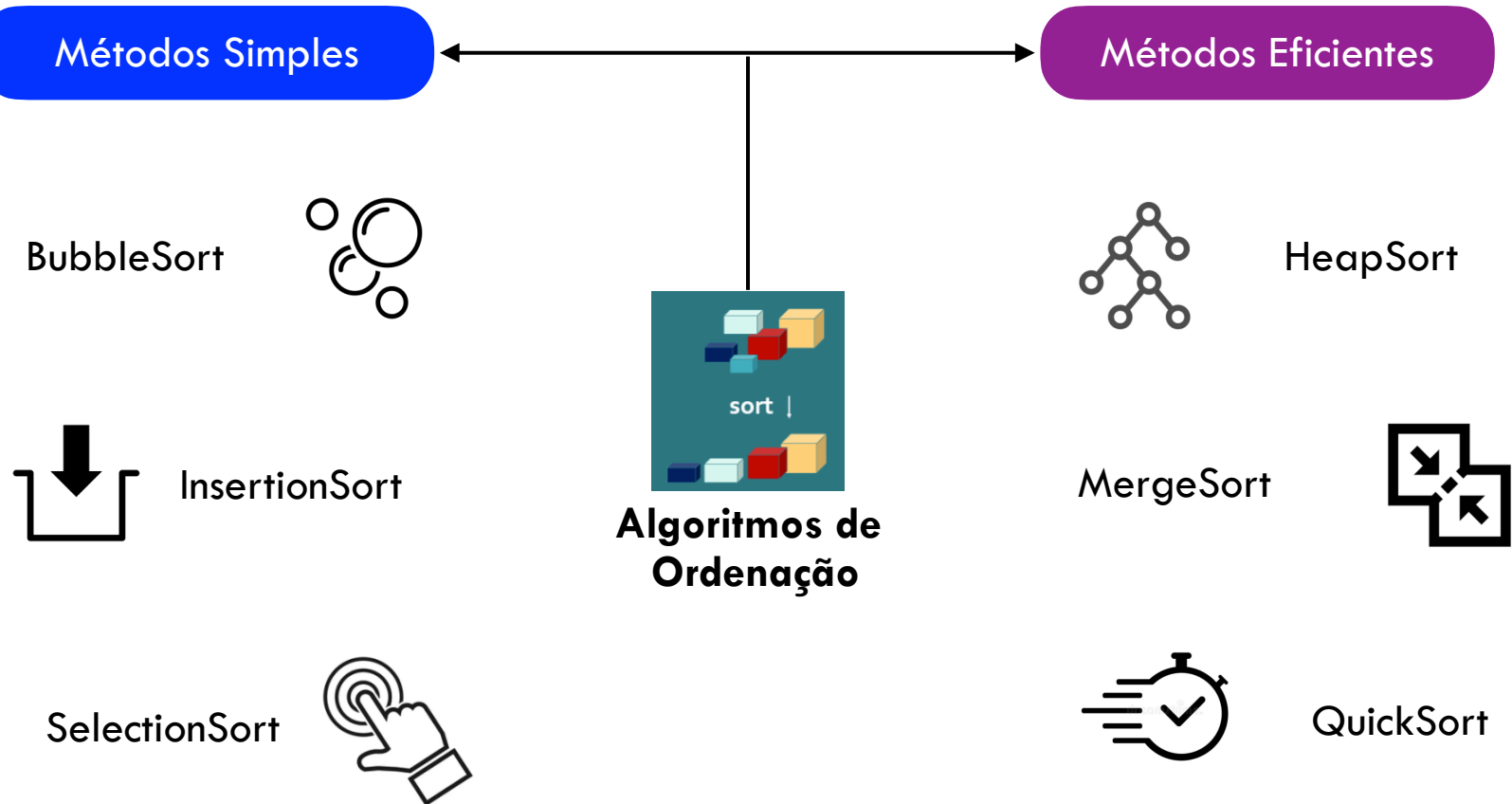
Métodos Simples

Métodos Eficientes

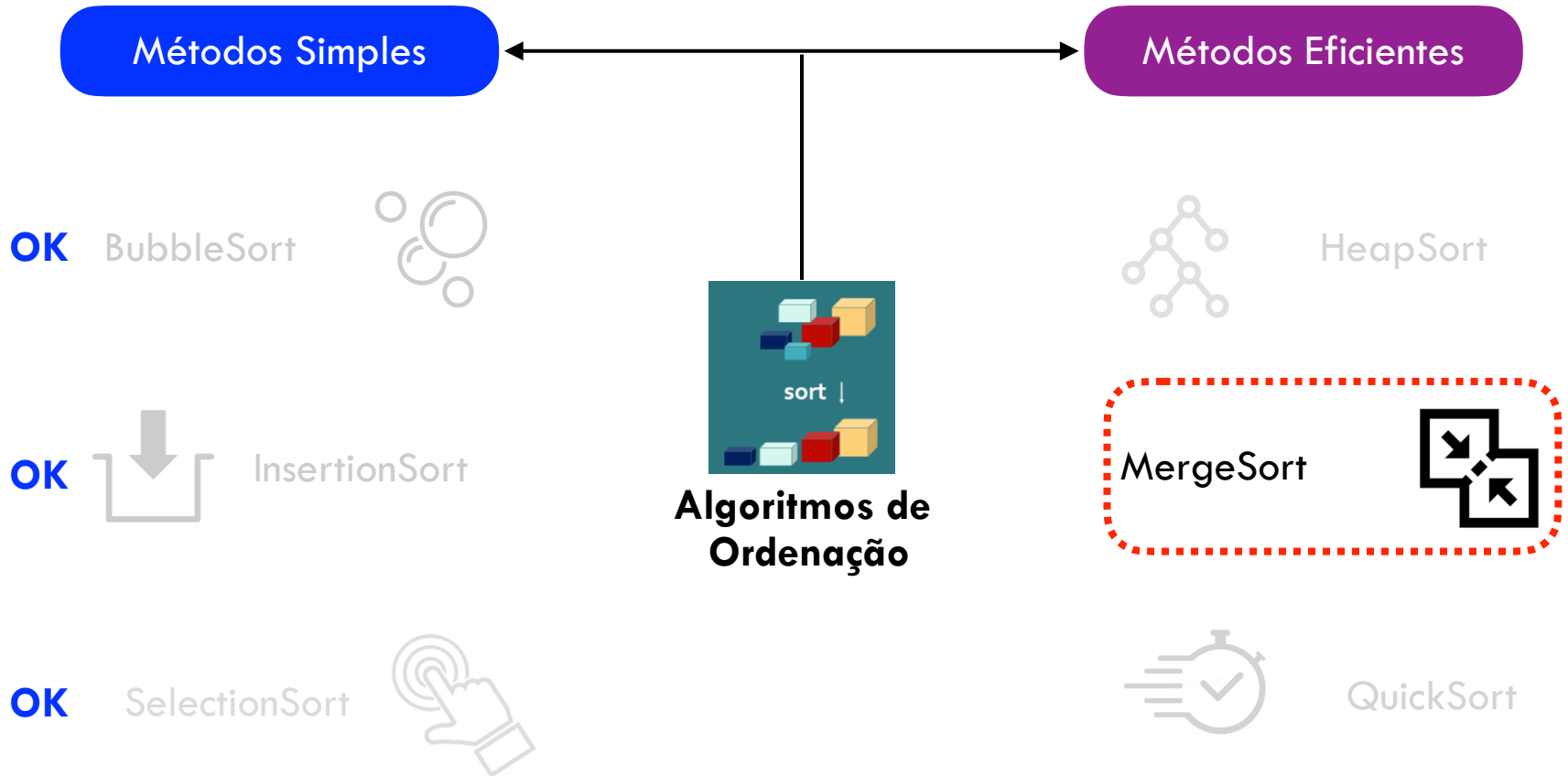


**Algoritmos de  
Ordenação**

# Introdução



# Introdução





# Roteiro



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

# Merge Sort



- Ordenação por Mistura

# Merge Sort

- Ordenação por Mistura



# Merge Sort

## □ Ordenação por Mistura



# Merge Sort

- Ordenação por Mistura



League: levar os 6 mais fortes



# Merge Sort



# Merge Sort

Lv 65   Lv 16   Lv 54   Lv 15   Lv 60   Lv 5   Lv 40   Lv 39   Lv 50



# Merge Sort

Lv 65



Lv 16



Lv 54



Lv 15



Lv 60



Meio

Lv 5



Lv 40



Lv 39

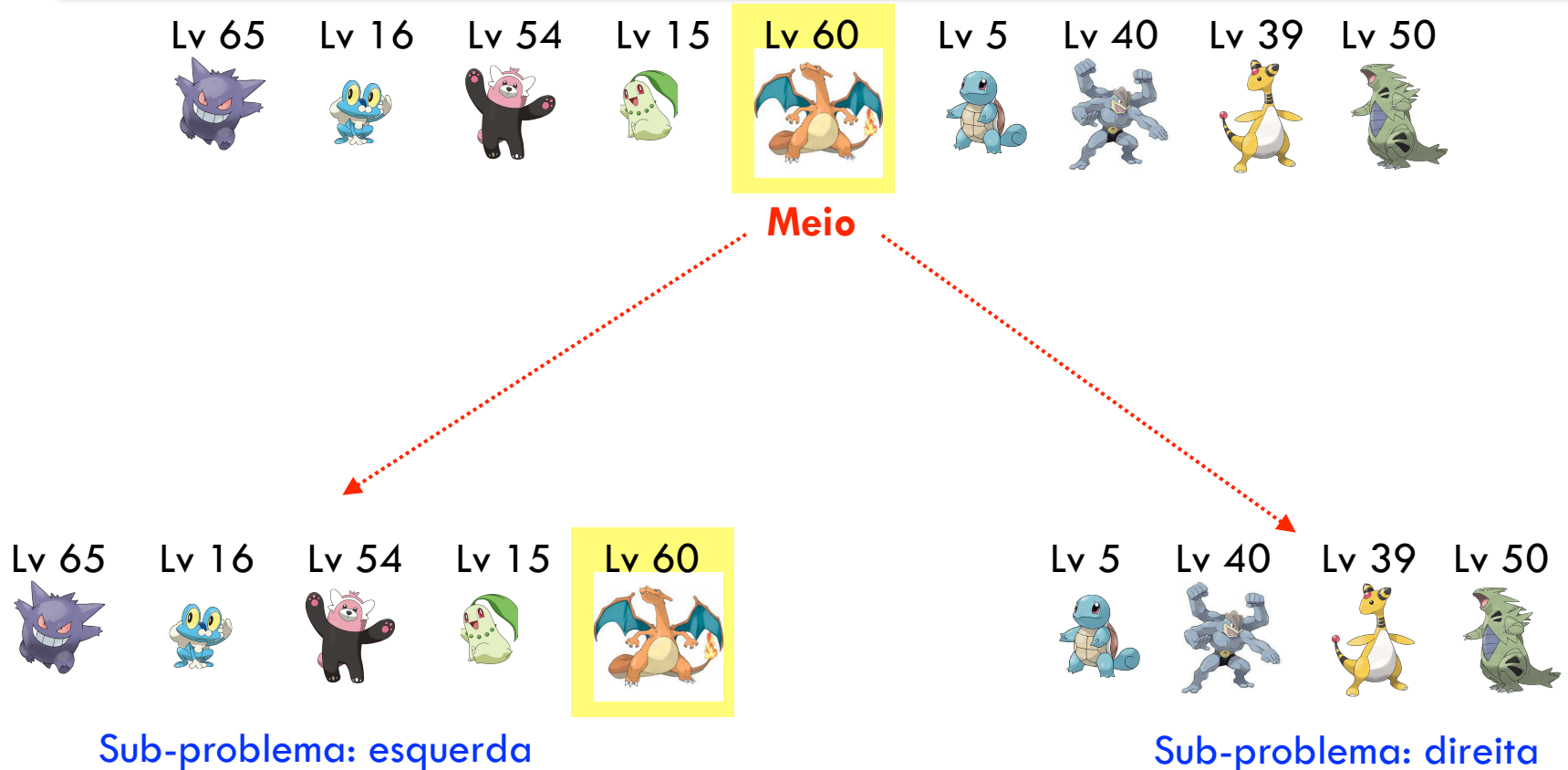


Lv 50





# Merge Sort



# Merge Sort

Lv 65



Lv 16



Lv 54



Meio

Lv 15



Lv 60



Lv 5



Lv 40



Lv 39

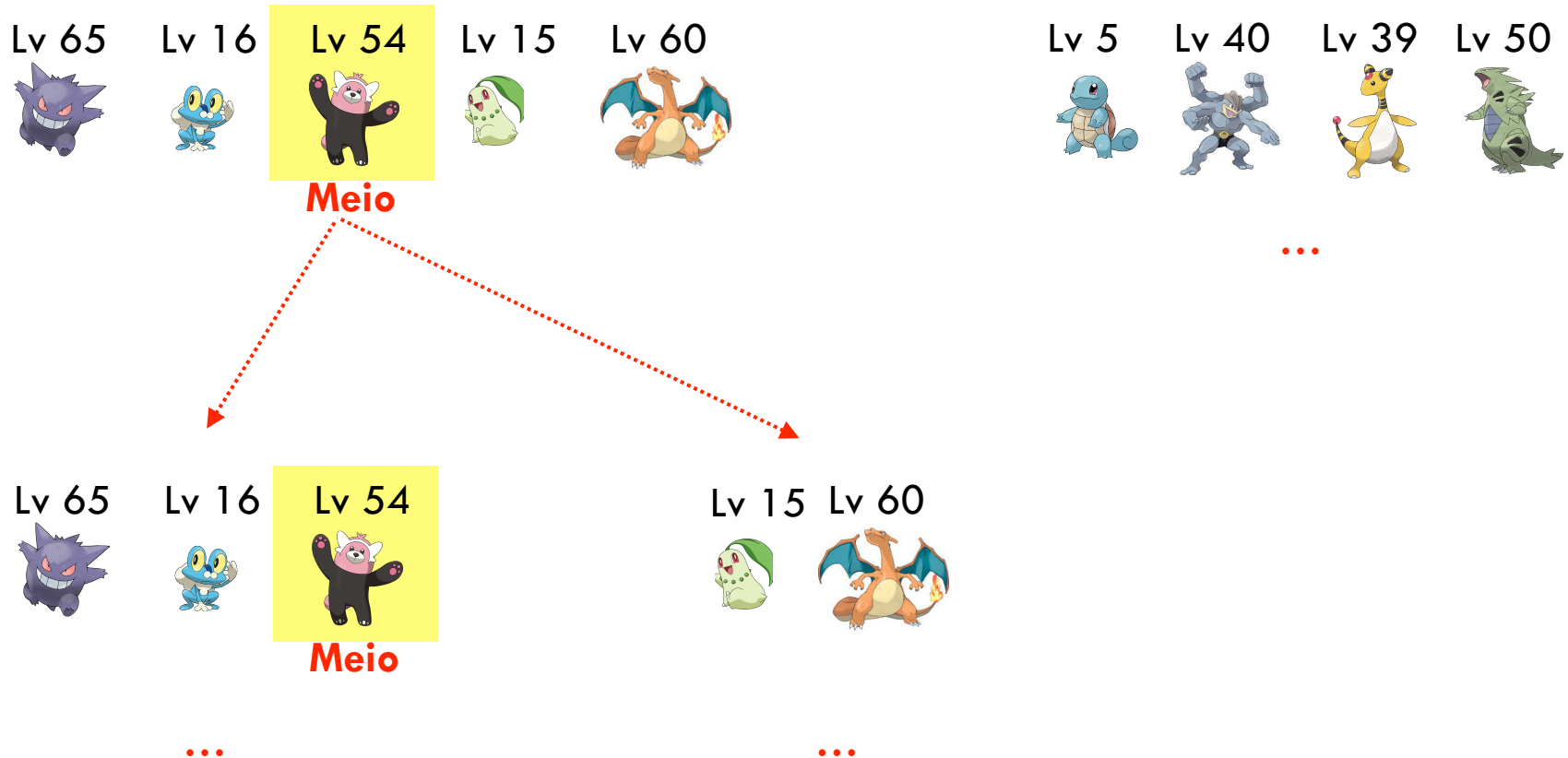


Lv 50

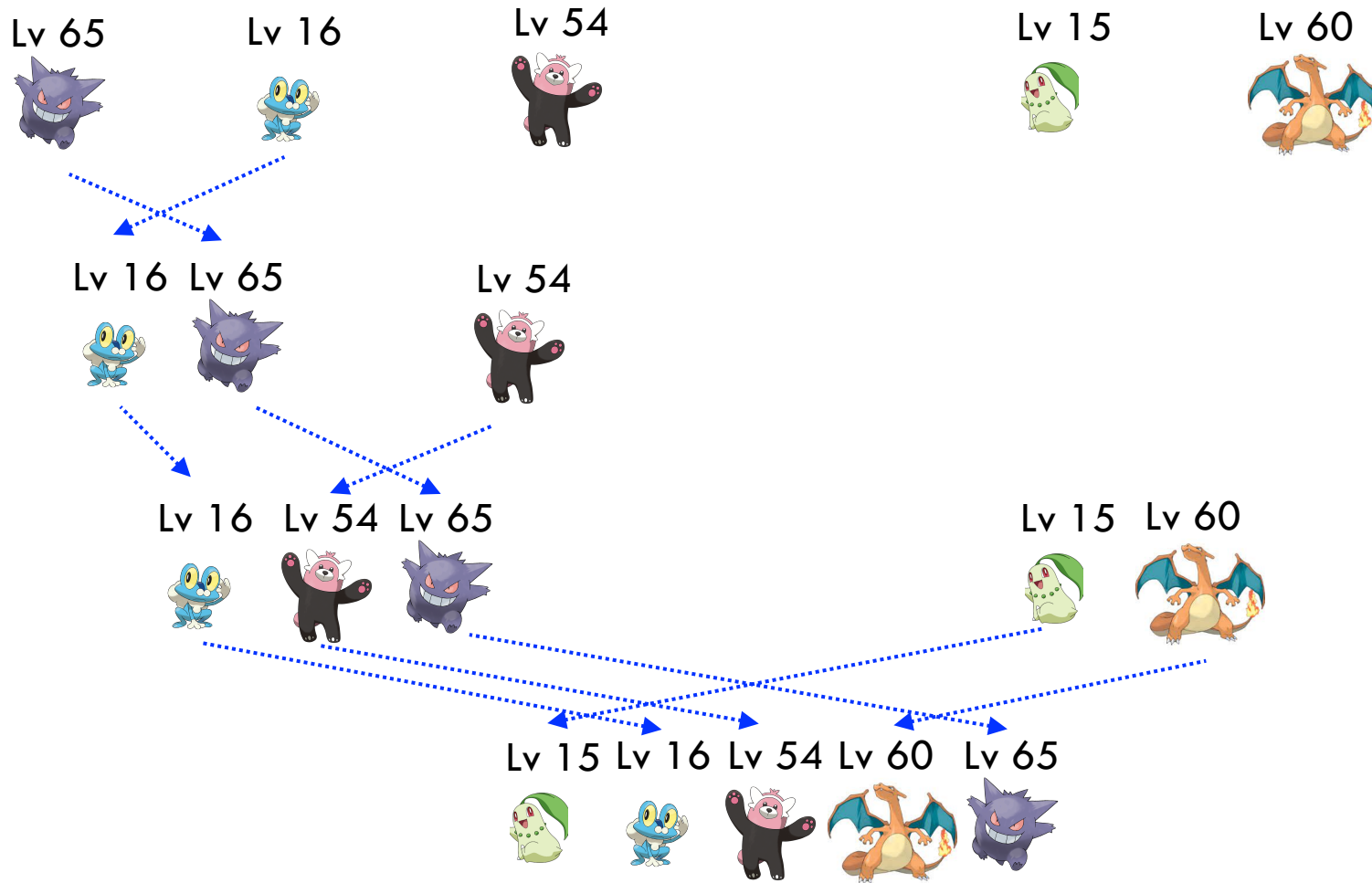


...

# Merge Sort



# Merge Sort



# Merge Sort



# Merge Sort

Lv 5   Lv 15   Lv 16



**Lv 39**



**Lv 40**



**Lv 50**



**Lv 54**



**Lv 60**



**Lv 65**



**Good Team :)**

# Merge Sort

## □ Ordenação por Mistura

- \* ideia básica: dividir e conquistar
- \* divide recursivamente o conjunto de dados até que cada subconjunto possua um elemento

# Merge Sort

## □ Funcionamento

\* Dividir e conquistar:

1. Divide recursivamente o array até obter subconjuntos com elementos únicos
2. Volta da recursão combinando 2 conjuntos de forma a obter um conjunto maior e mais ordenado
3. Processo se repete até que existe apenas um conjunto único e ordenado



# Merge Sort

- Dividir

4	23	67	-8	21
---	----	----	----	----

# Merge Sort

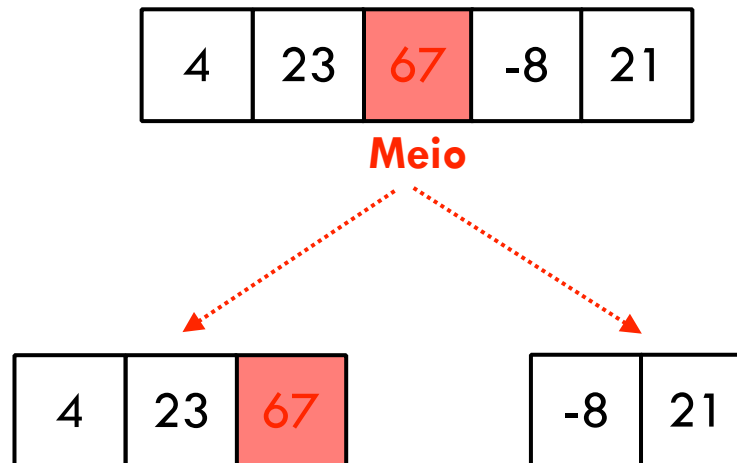
- Dividir

4	23	67	-8	21
---	----	----	----	----

Meio

# Merge Sort

## □ Dividir



# Merge Sort

## □ Recombinar

4	23	67
---	----	----

-8	21
----	----

# Merge Sort

## □ Recombinar

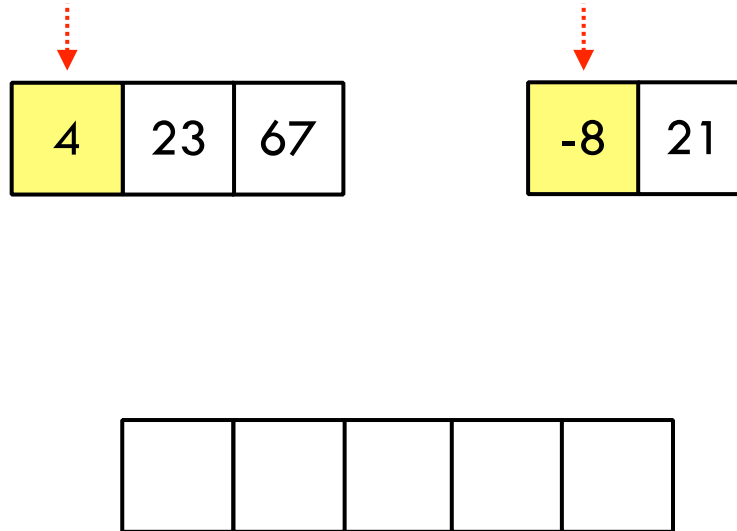
4	23	67
---	----	----

-8	21
----	----

--	--	--	--	--

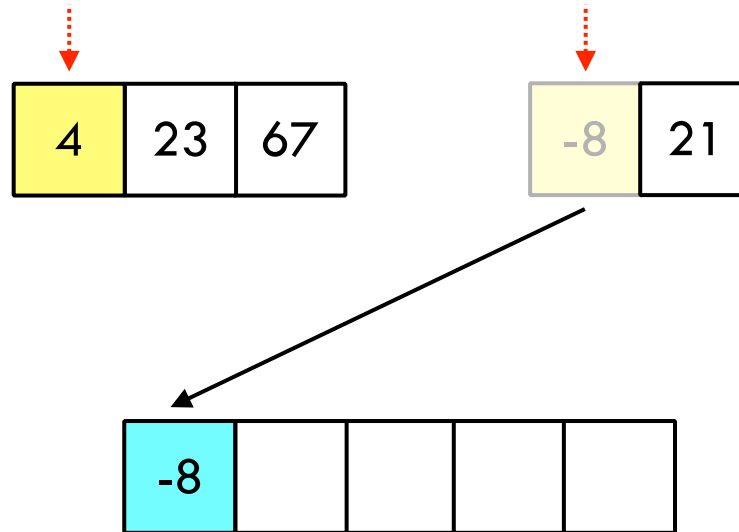
# Merge Sort

## □ Recombinar



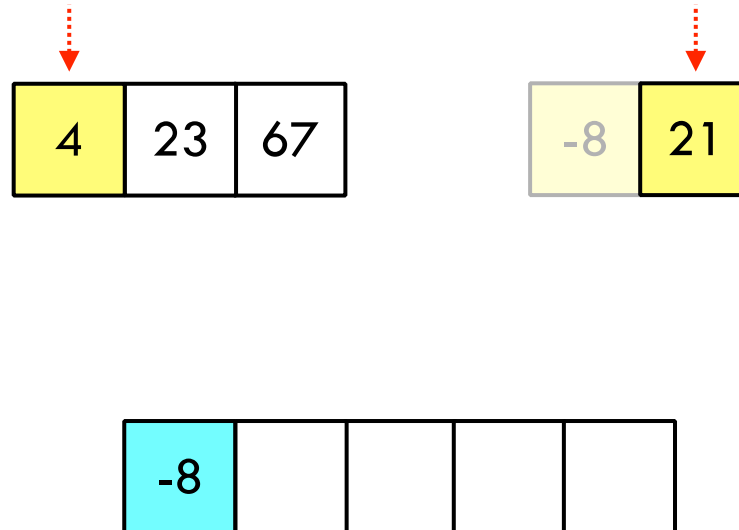
# Merge Sort

## □ Recombinar



# Merge Sort

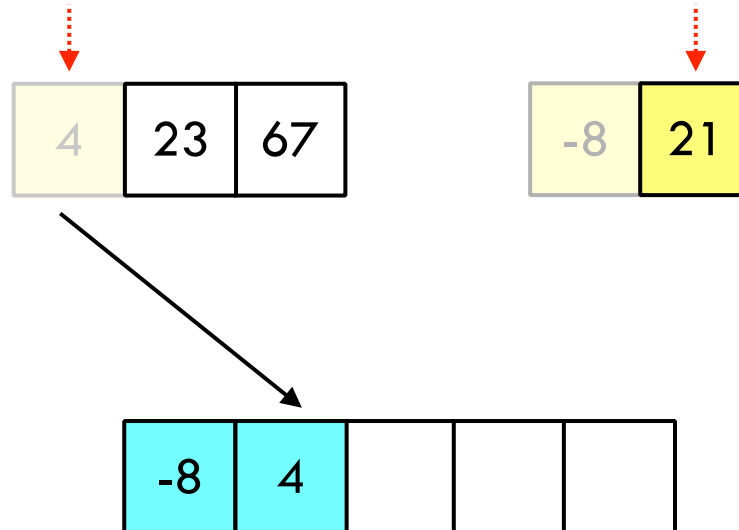
## □ Recombinar





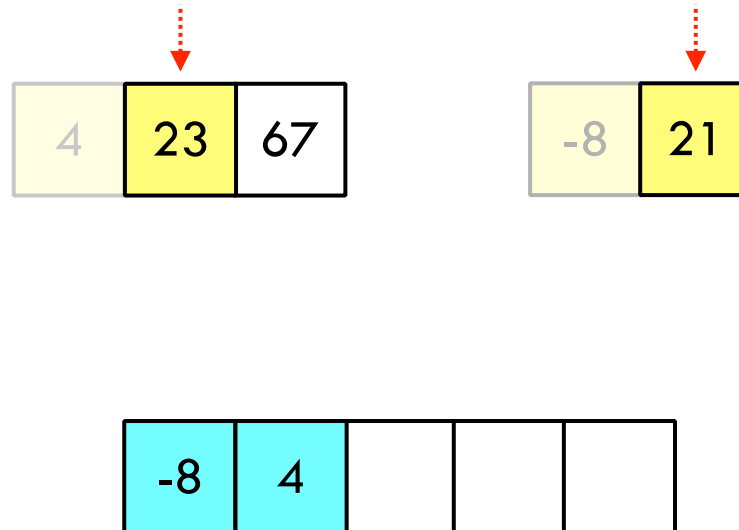
# Merge Sort

## □ Recombinar



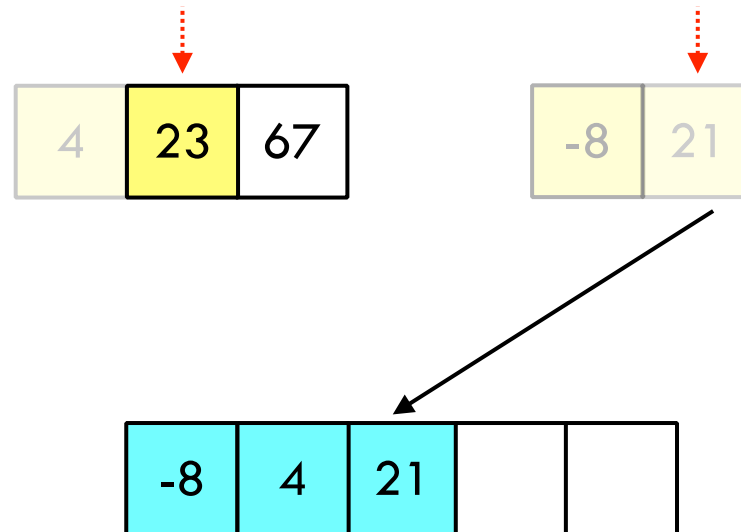
# Merge Sort

## □ Recombinar



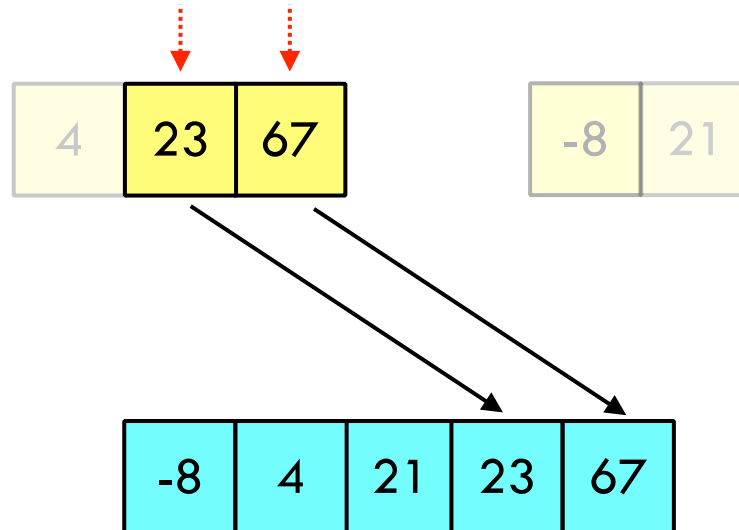
# Merge Sort

## □ Recombinar



# Merge Sort

## □ Recombinar



# Merge Sort

## □ Desempenho

- \* **melhor caso:**  $O(N \log N)$ , elementos já ordenados
- \* **pior caso:**  $O(N \log N)$ , elementos na ordem decrescente
- \* **caso médio:**  $O(N \log N)$

# Merge Sort

## □ Desempenho

- \* **melhor caso:**  $O(N \log N)$ , elementos já ordenados
- \* **pior caso:**  $O(N \log N)$ , elementos na ordem decrescente
- \* **caso médio:**  $O(N \log N)$

**Obs:** Eficiente para grandes conjuntos de dados. Estável.

# Merge Sort

## □ Pseudocódigo (função principal)

```
1. MergeSort (V, Inicio, Fim)
2.   Se (Inicio < Fim), então:
3.     Meio = ((Inicio + Fim)/2)
4.     MergeSort(V, Inicio, Meio)
5.     MergeSort(V, Meio+1, Fim)
6.     Merge(V, Inicio, Meio, Fim)
```

# Merge Sort

## □ Pseudocódigo (função auxiliar)

1. **Merge** (V, Inicio, Meio, Fim)
2.     Alocar dinamicamente um vetor auxiliar
3.      $P1 = \text{Inicio}$
4.      $P2 = \text{Meio} + 1$
5.     **Enquanto** ( $P1 < \text{Meio}$  E  $P2 < \text{Fim}$ ) **faça**:
6.         Copia para o vetor auxiliar o menor valor entre  $V[P1]$  e  $V[P2]$
7.         Incrementa o contador correspondente
8.     **Se**  $P1 == \text{Meio}$  **então**:
9.         Copia o que sobrou a partir de  $P2$
10.    **Senão**:
11.         Copia o que sobrou a partir de  $P1$
12.    Copia o vetor auxiliar no original



# Roteiro



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

# Exemplo

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

**vetor não ordenado**

# Exemplo

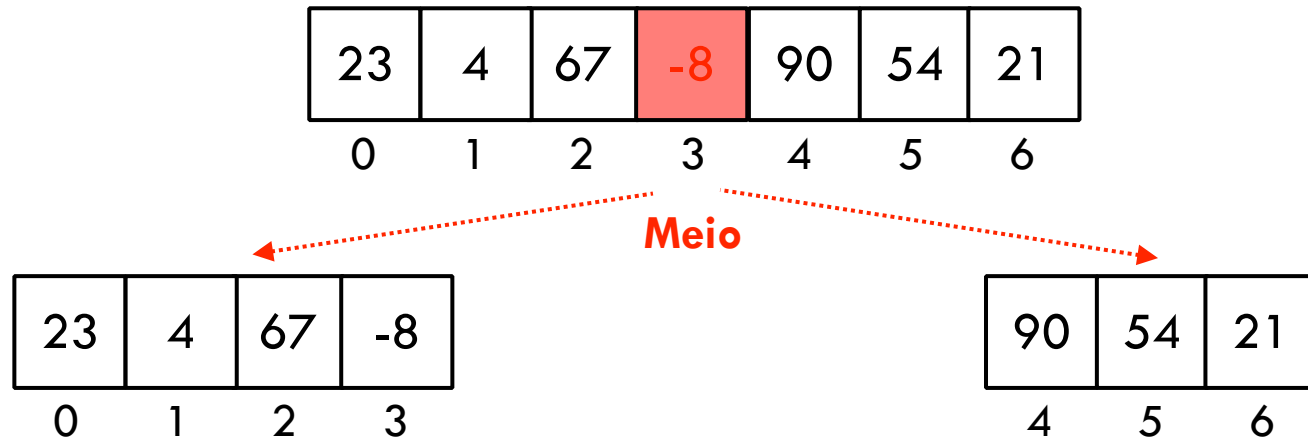
23	4	67	-8	90	54	21
0	1	2	3	4	5	6

# Exemplo

23	4	67	-8	90	54	21
0	1	2	3	4	5	6

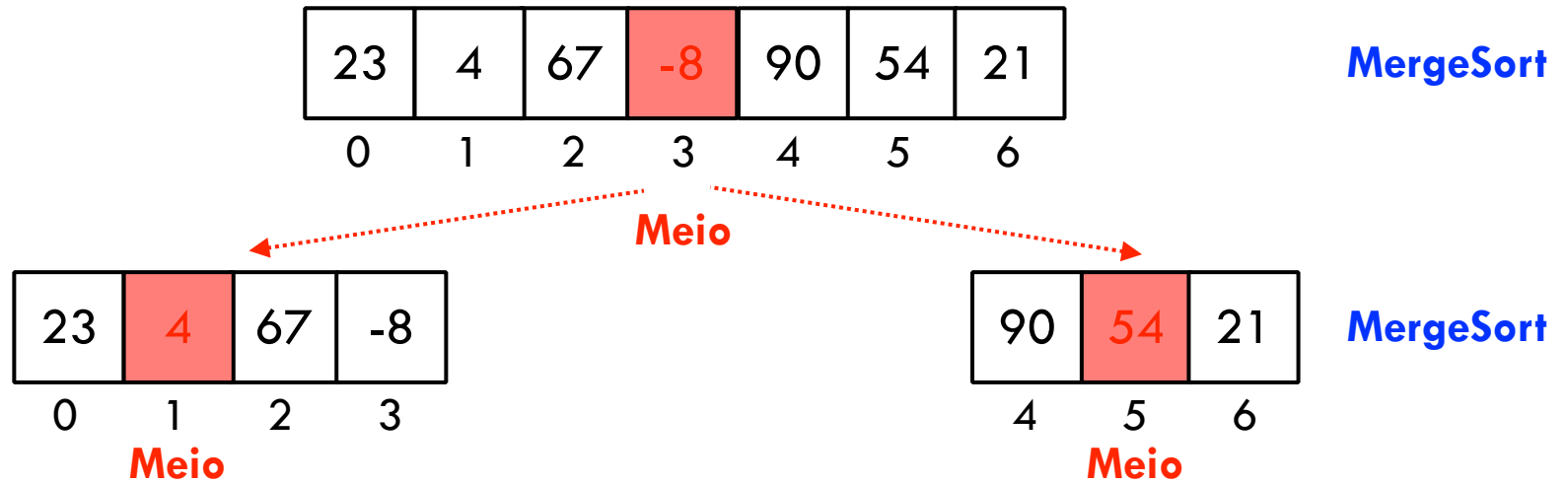
Meio

# Exemplo

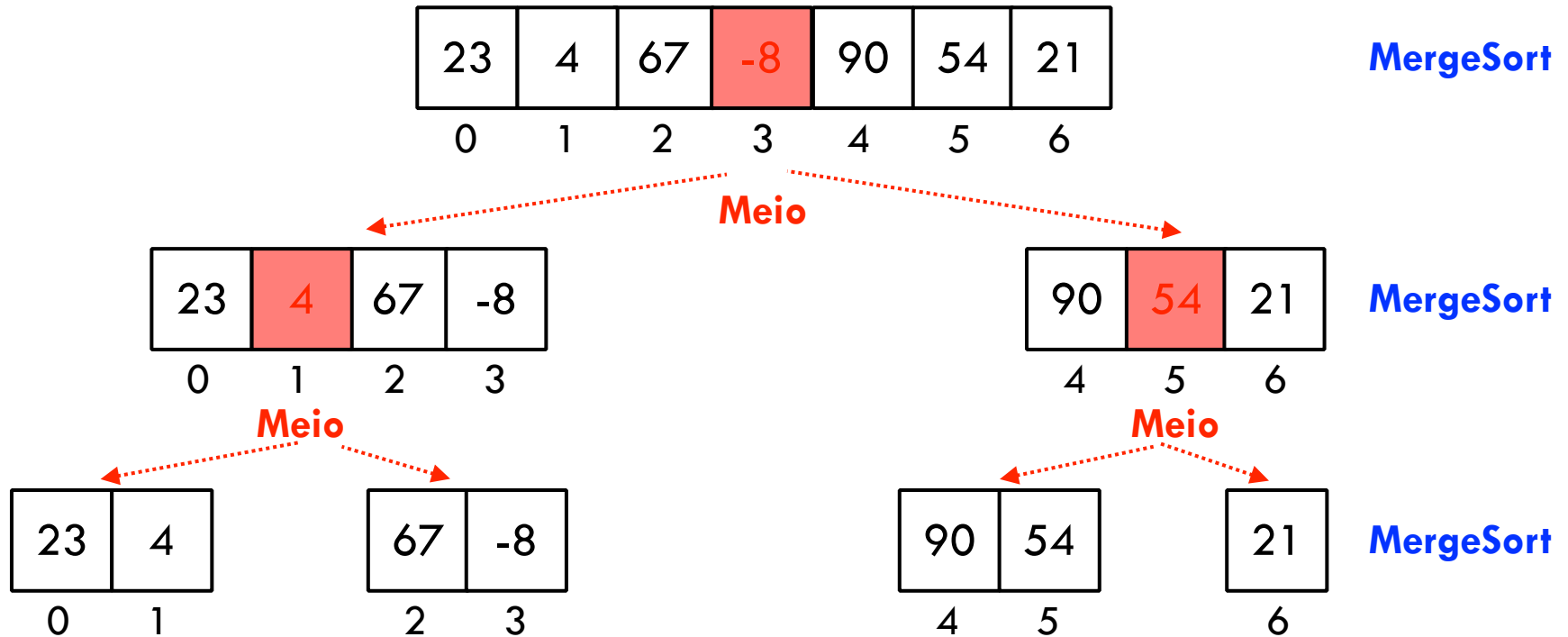


MergeSort

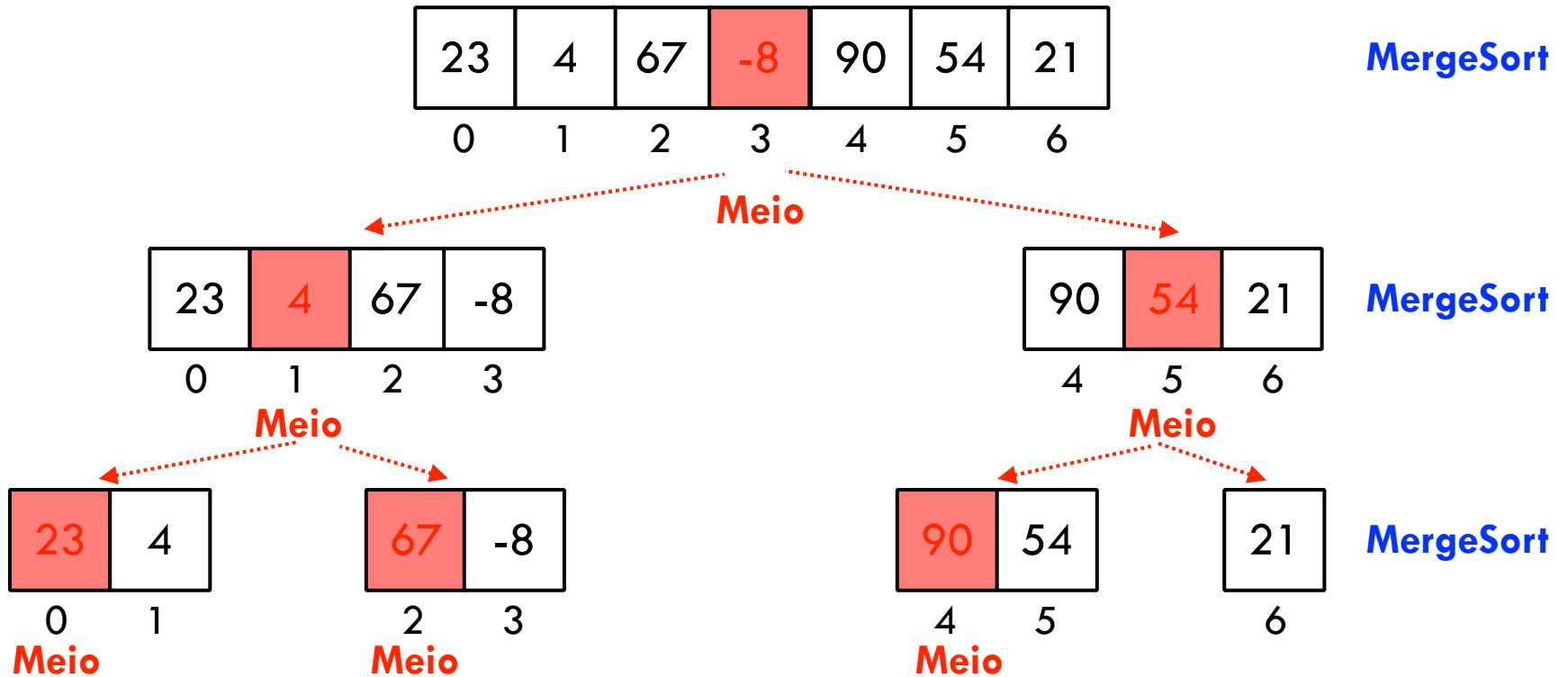
# Exemplo



# Exemplo

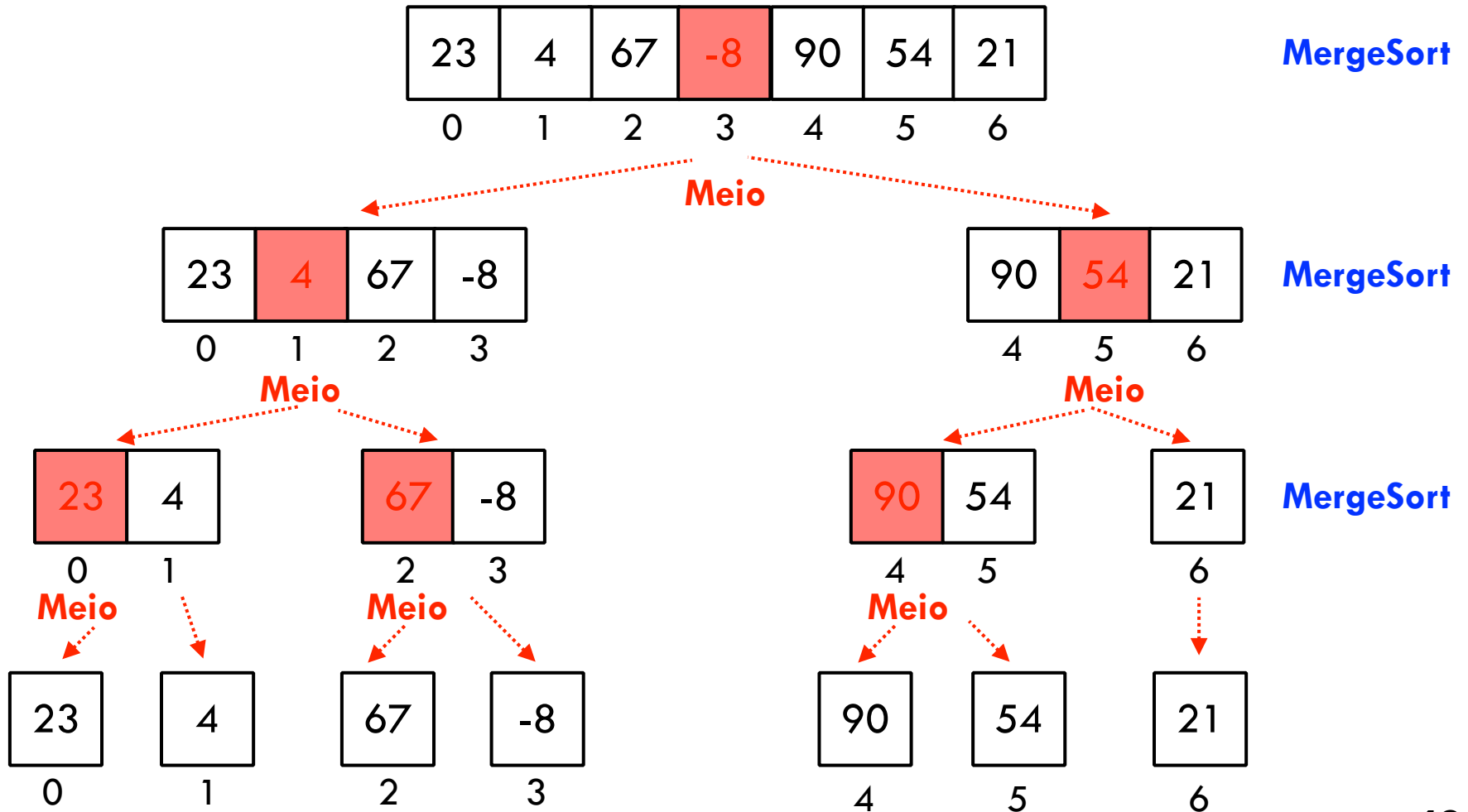


# Exemplo





# Exemplo



# Exemplo

23

0

4

1

67

2

-8

3

90

4

54

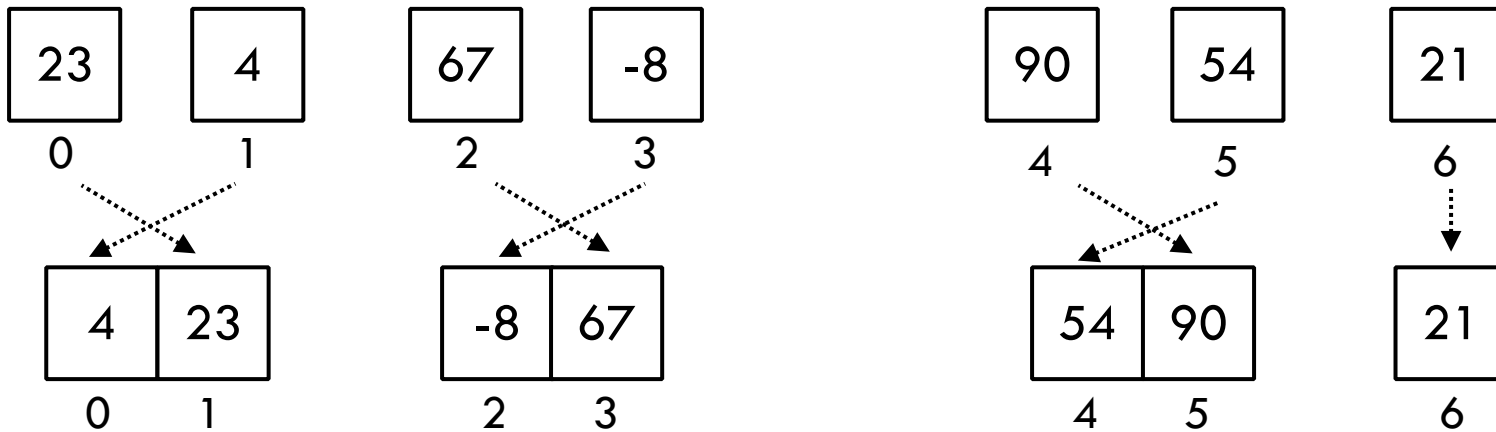
5

21

6

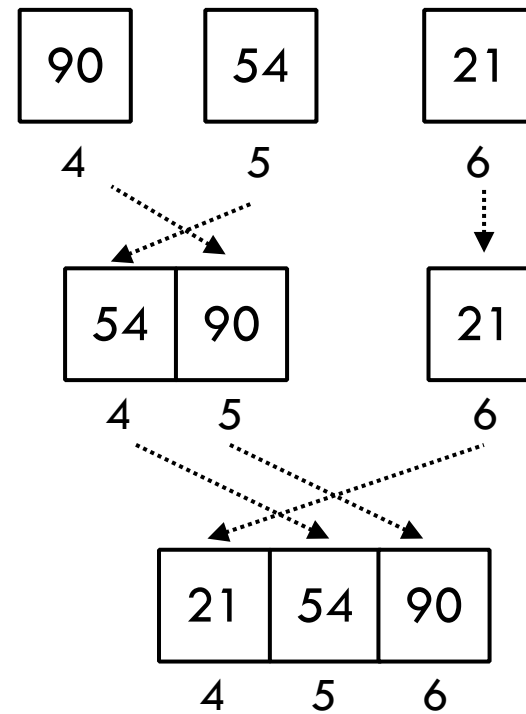
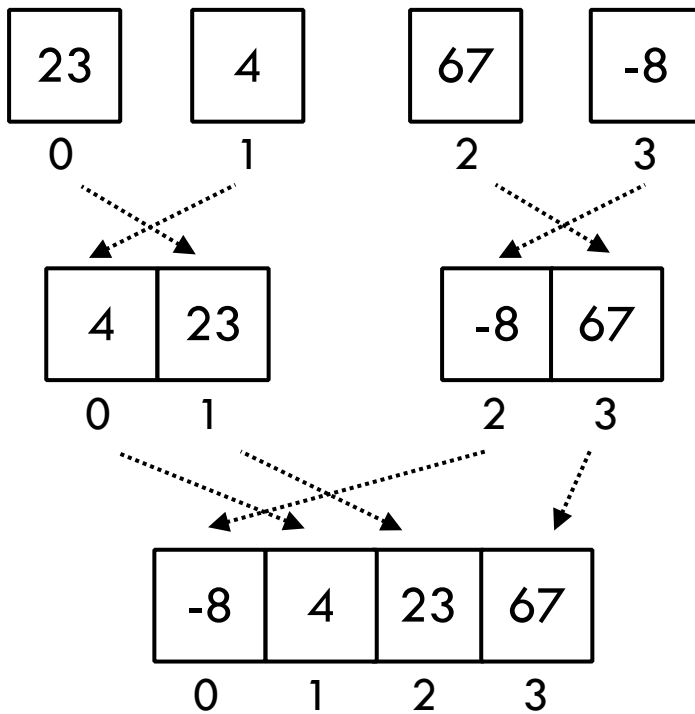
Merge

# Exemplo



Merge

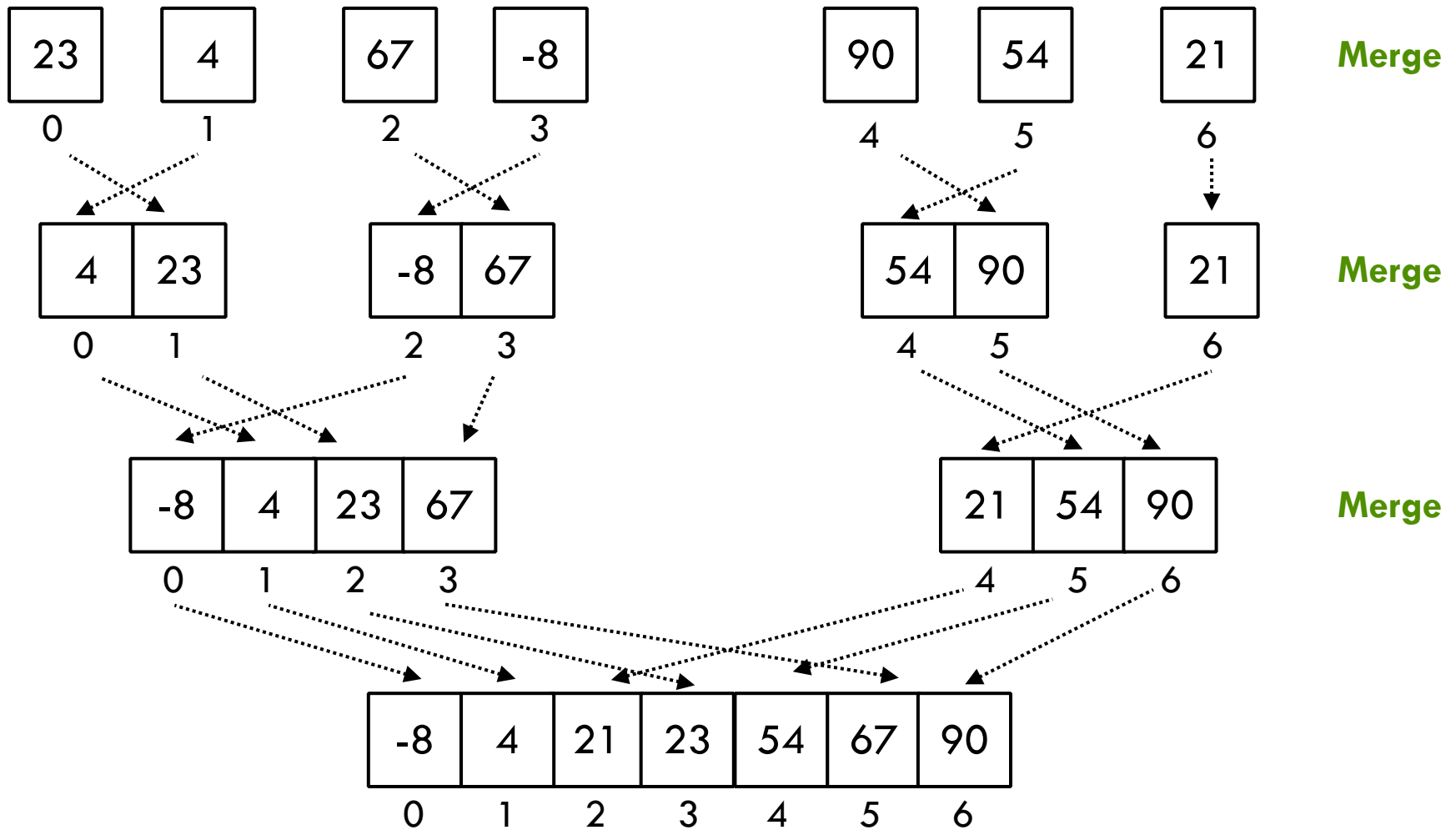
# Exemplo



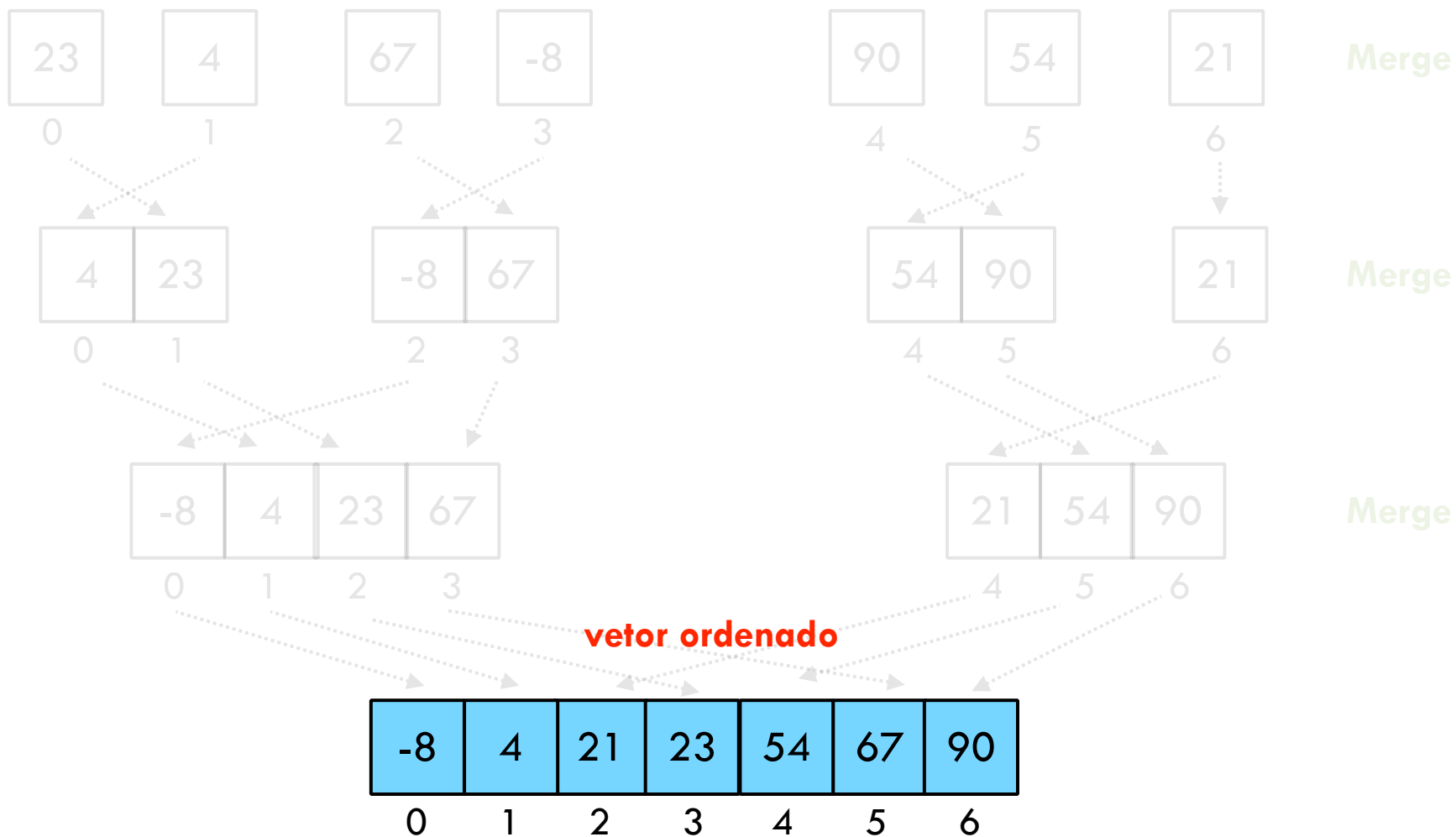
Merge

Merge

# Exemplo



# Exemplo



# Merge Sort

## Vantagens

- \* elegante e eficiente
- \* não altera a ordem dos dados (estável)
- ...

## Desvantagens

- \* Recursivo
- \* Uso de memória - usa um vetor auxiliar durante a ordenação
- \* Pode ser custoso dependendo do tamanho do array

# Roteiro



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



# Exercícios



**HANDS ON :)))**

# Exercícios

1) Reuna-se com seu grupo e execute o teste de mesa (simulação) do algoritmo para as sequências de números apresentadas

Link planilha grupos/sequências de teste:

<https://docs.google.com/spreadsheets/d/1X9IGtcZeAt7j0lIR1W3JPupl2wCtPQgybH0KZm8j-iE/edit?usp=sharing>

# Exercícios

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

*/\* Ordena o vetor usando Merge Sort*

*Parâmetros:*

*v: vetor a ser ordenado*

*n: número de elemento do vetor, tamanho do vetor*

*Esse algoritmo tem um comportamento assintótico  $O(N \log N)$  \*/*

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

# Exercícios

3) Adapte sua implementação do **mergeSort** para realizar tanto a ordenação crescente, como decrescente.

*/\* Ordena o vetor usando Merge Sort*

*Parâmetros:*

*v: vetor a ser ordenado*

*n: número de elemento do vetor, tamanho do vetor*

*op: 1 para realizar ordenação crescente, 2 para ordenação decrescente*

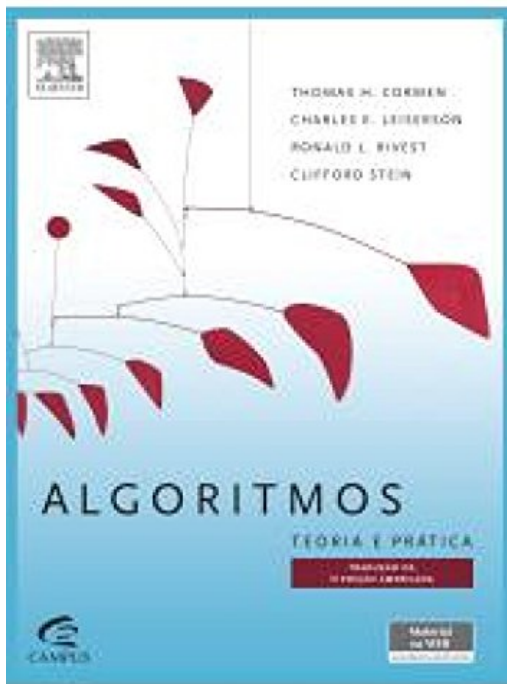
*Esse algoritmo tem um comportamento assintótico  $O(N \log N)$  \*/*

**void mergeSort(int \*v, int n, int op);**

# Roteiro

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

# Referências sugeridas

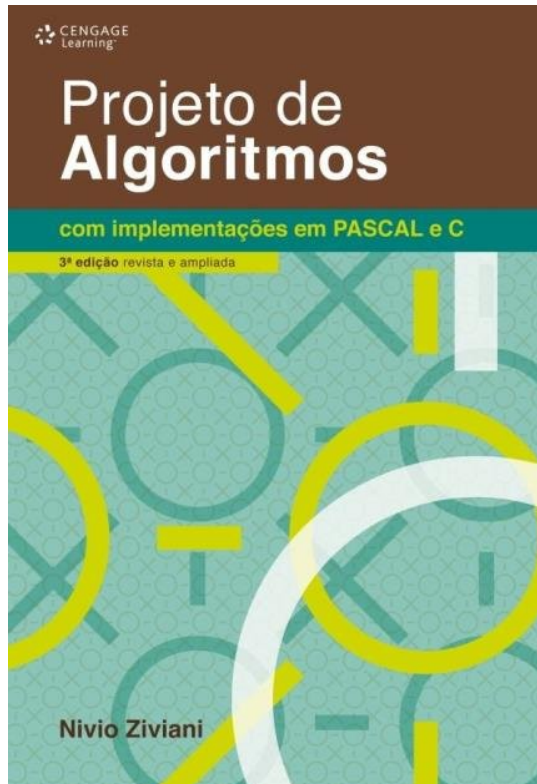


[Cormen et al, 2018]

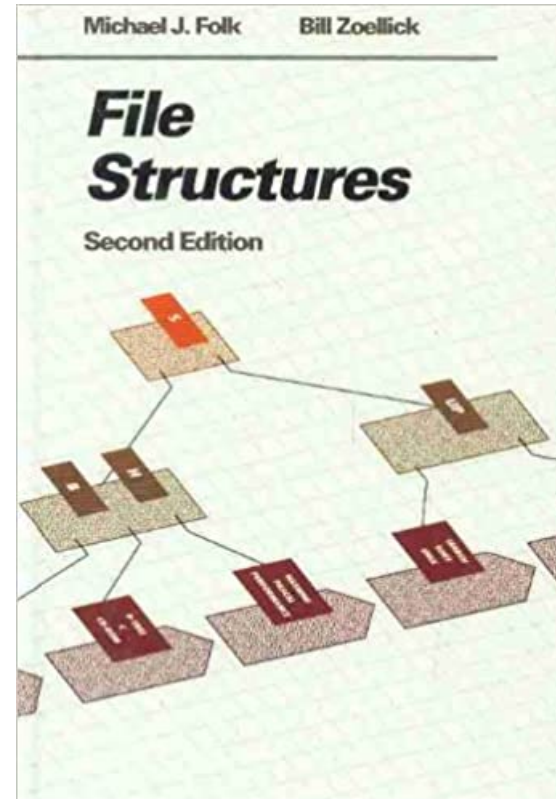


[Drozdek, 2017]

# Referências sugeridas



[Ziviani, 2010]



[Folk & Zoellick, 1992]

# Perguntas?

Prof. Rafael G. **Mantovani**

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