

# Quicksort

Arthur Romani

Estevão Florencio

Guilherme Araujo

Isabel Valentim

Renan Manera

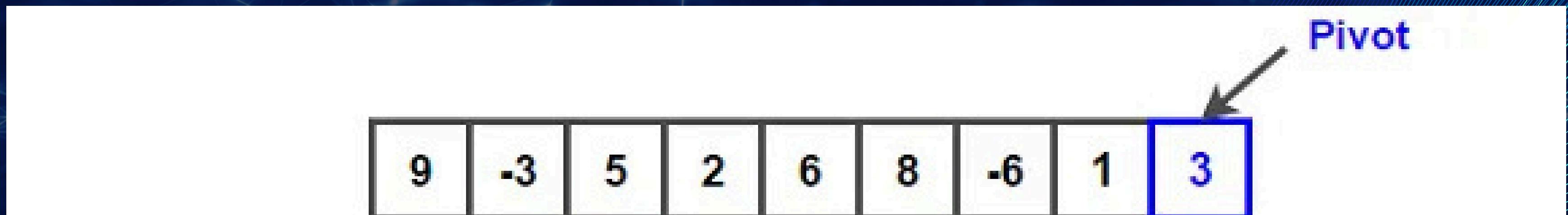
Marcos Vinicius

# DEFINIÇÃO

O Quicksort é um método que se utiliza a técnica dividir/divisão e conquista, selecionando um elemento e a lista em diferentes partes até que fiquem organizados.

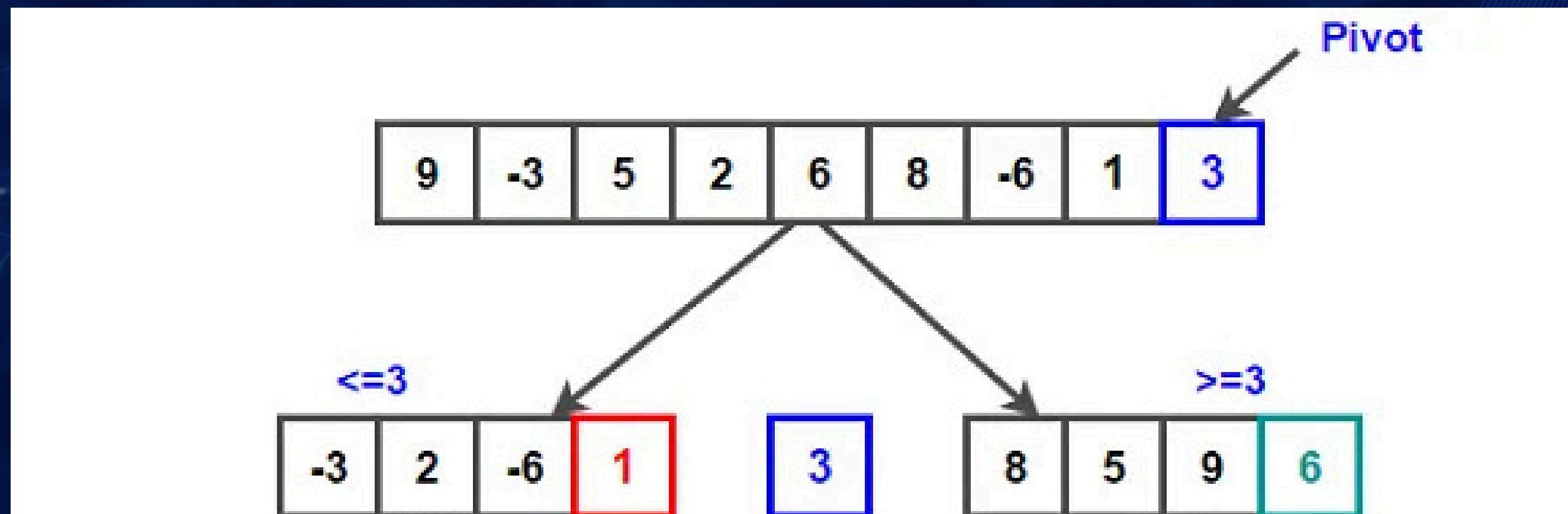
# 1 - ESCOLHER PIVÔ

Escolha um elemento da lista, denominado pivô



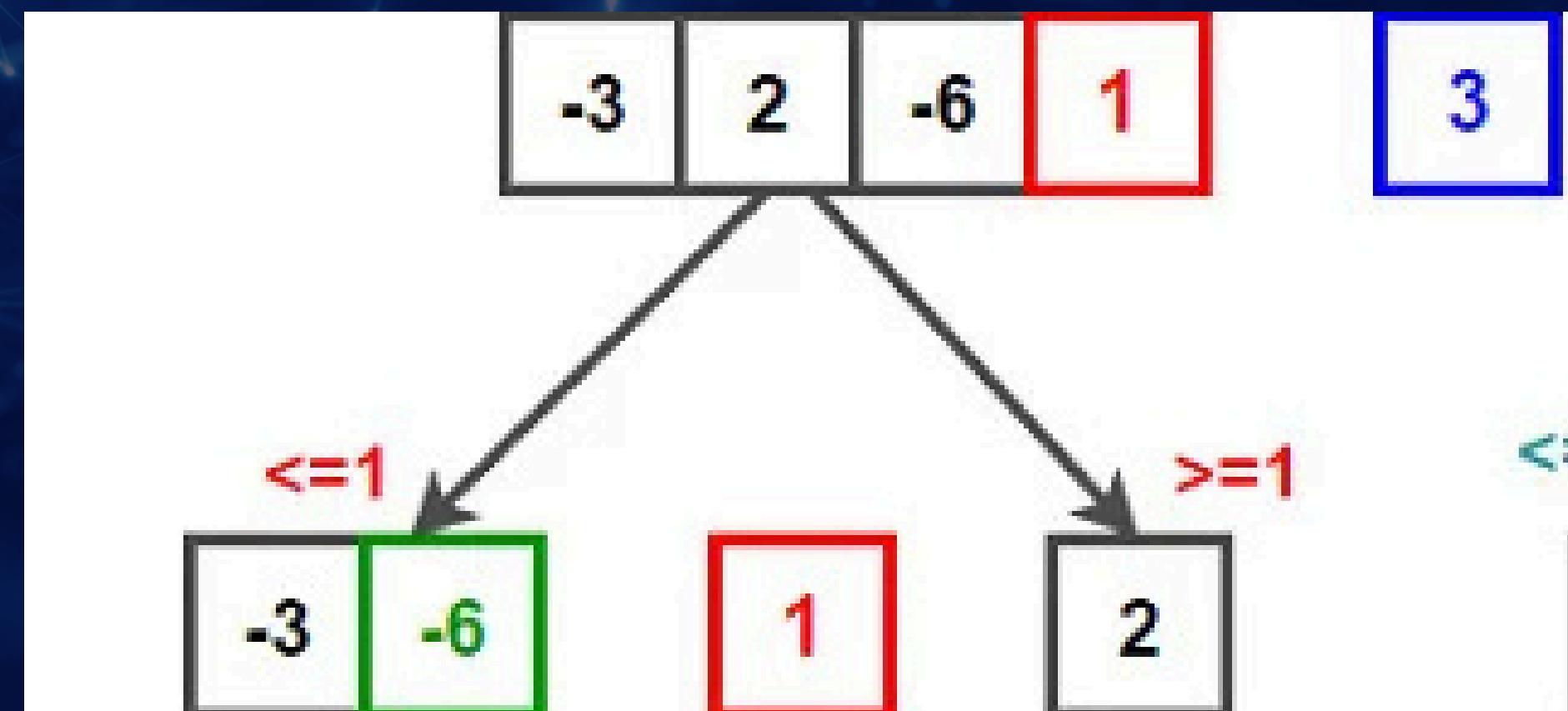
## 2 - PARTICIONAR

Rearranje a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele.

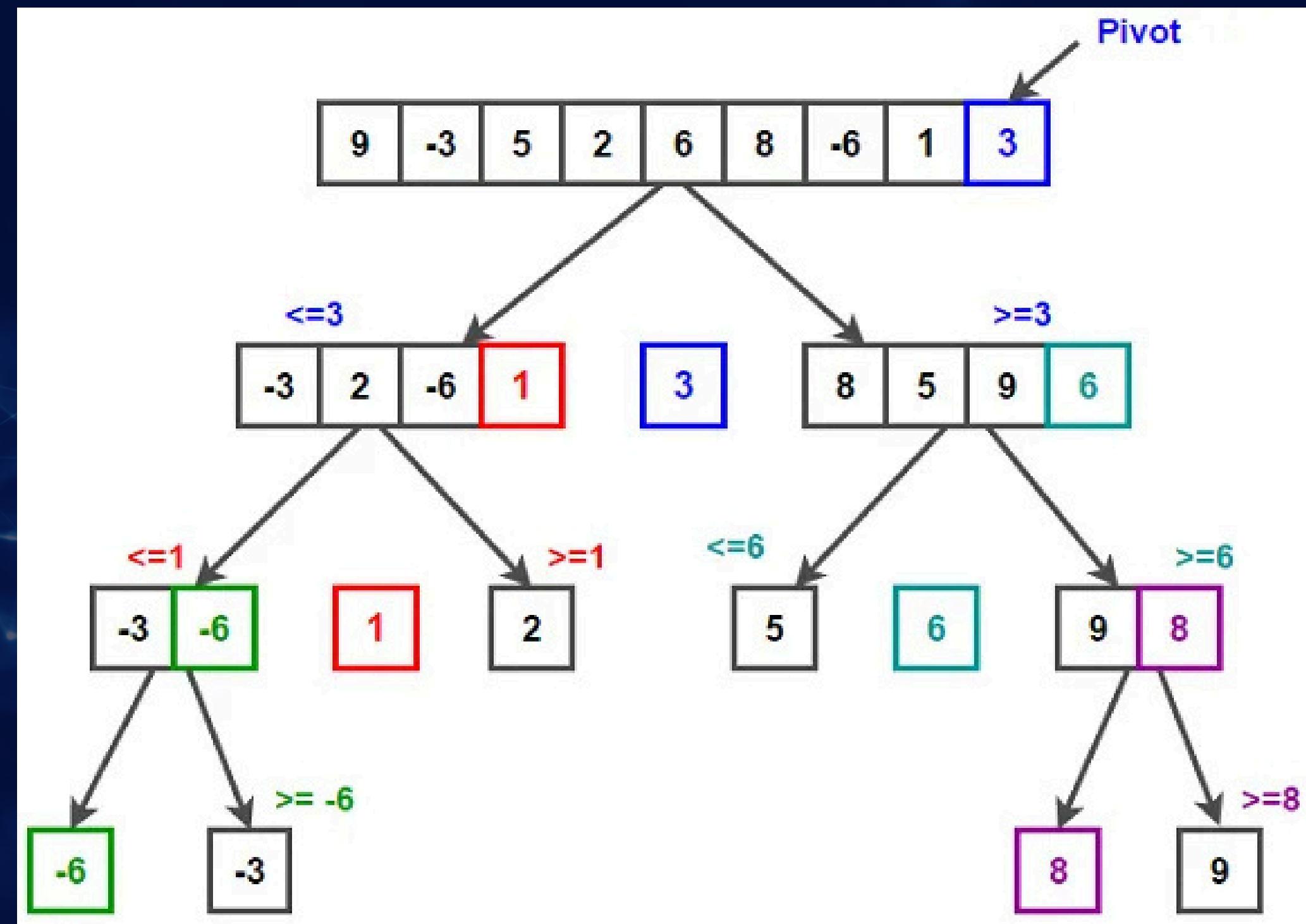


# 3 - FUNÇÃO RECURSIVA

O Quicksort é então aplicado recursivamente, escolhendo outro pivô para as novas sub listas e reordenando até que contenha apenas um elemento.



# 3 - FUNÇÃO RECURSIVA



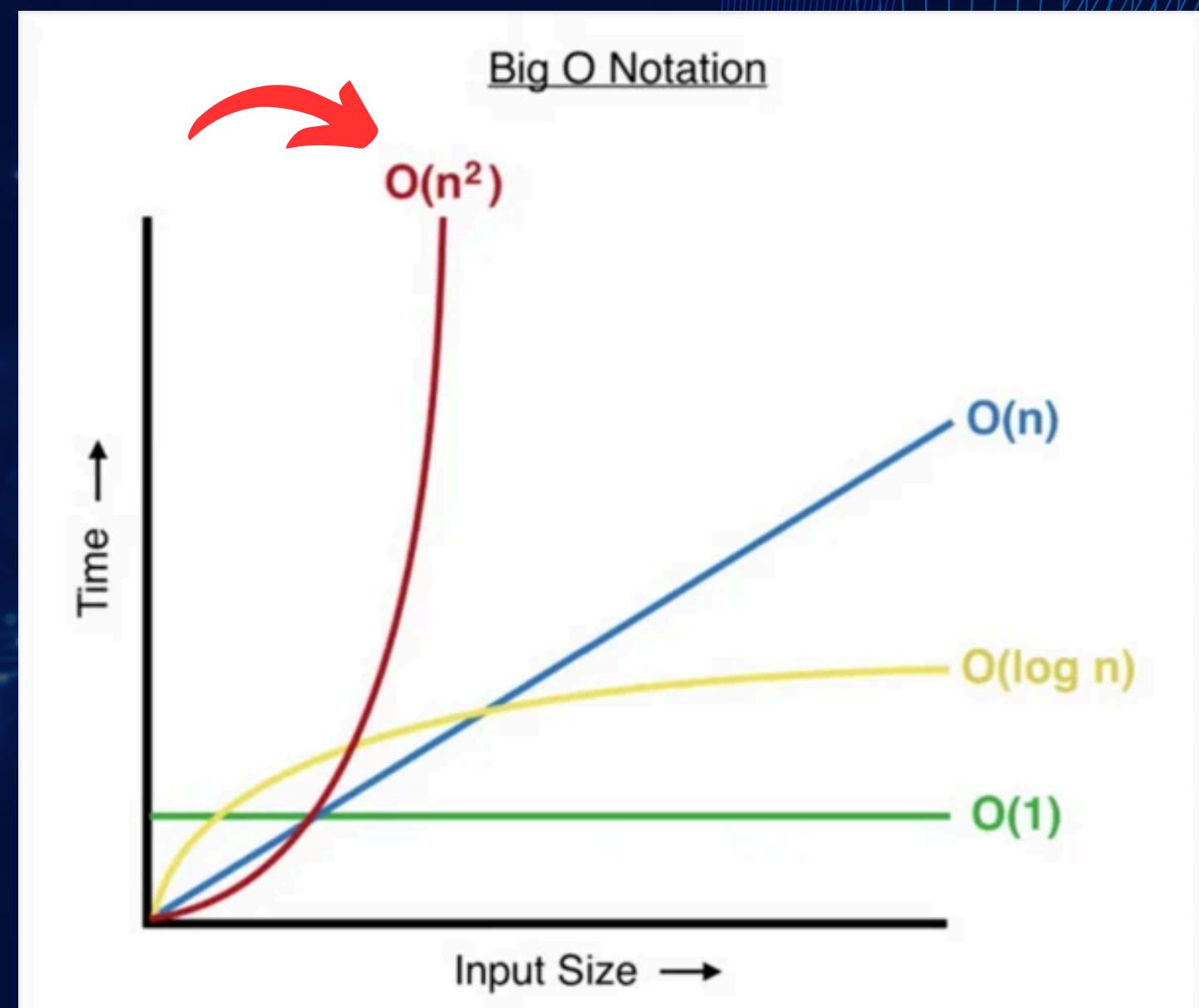
# PIOR CENÁRIO

Quando o pivô escolhido é sempre o **maior ou o menor** elemento da lista.

Array = [5, 8, 2, 9, 3, 6, 1, 4, 7]

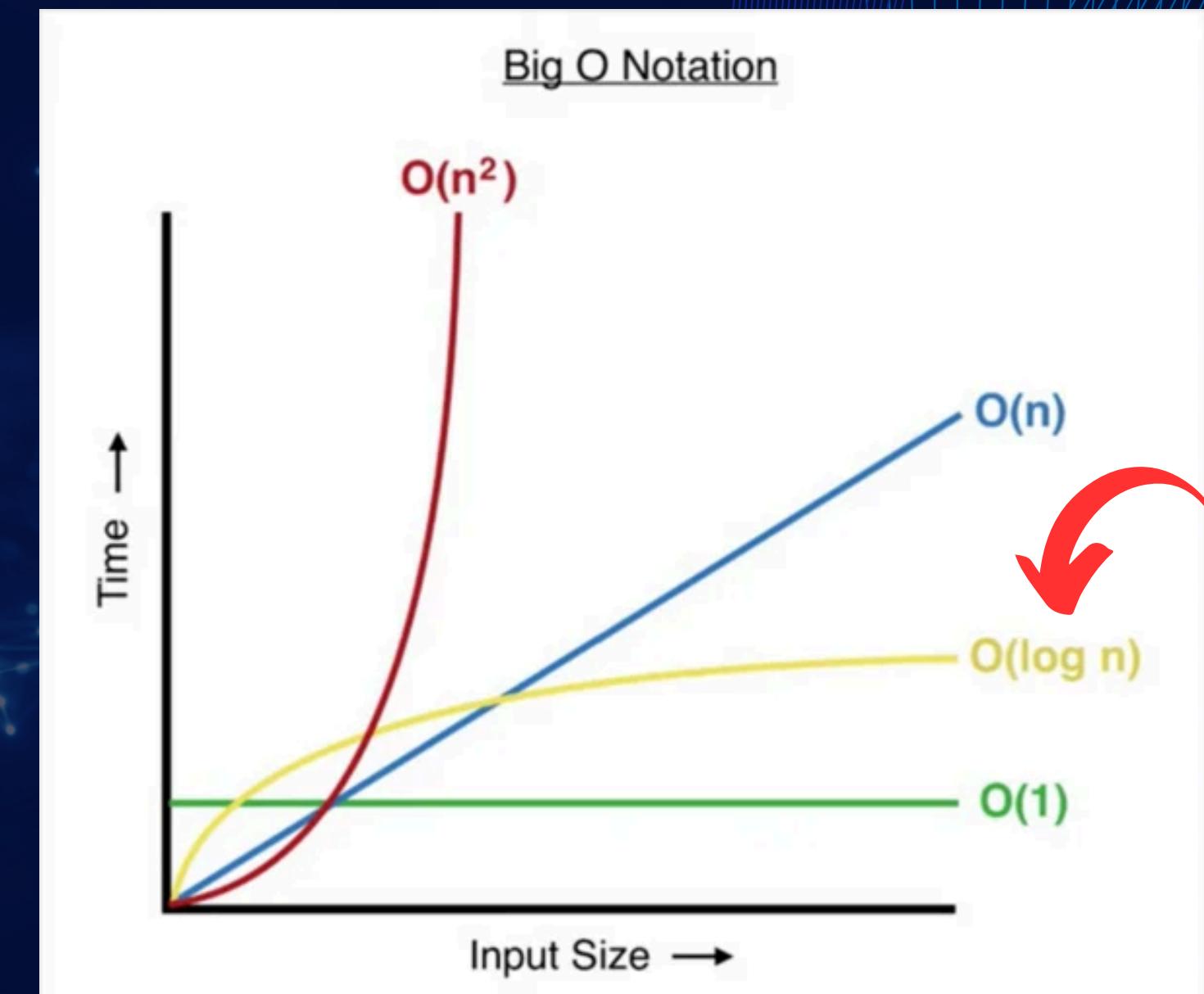
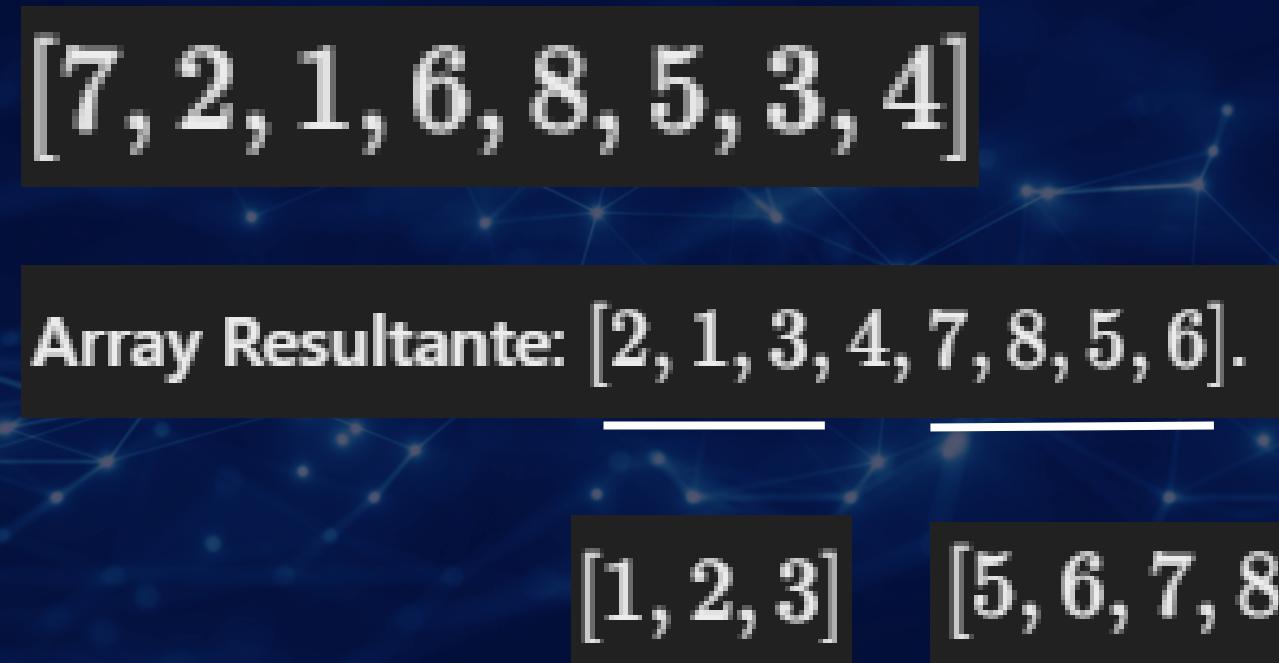
Depois do Particionamento = [1|5, 8, 2, 9, 3, 6, 4, 7]

- **Pivô Escolhido:** O menor elemento, 2.
- **Particionamento:** 2|5, 8, 9, 3, 6, 4, 7



# MELHOR CENÁRIO

Dividir o array em **partes iguais** pois a profundidade da árvore de recursão será mínima.



# VANTAGENS

- **Rápido:** Muito eficiente em grandes conjuntos de dados.
- **Complexidade Média  $O(n \log n)$ :** Melhor que muitos algoritmos com  $O(n^2)$ .
- **In-place:** Usa pouca memória adicional.
- **Divisão Recursiva:** Quebra em subproblemas menores.

# DESVANTAGENS

- **Pior Caso  $O(n^2)$ :** Se o pivô for mal escolhido.
- **Instável:** Não preserva a ordem de elementos iguais.
- **Escolha do Pivô:** Impacta o desempenho.
- **Recursivo:** Pode consumir muita memória.

# COMPARAÇÃO

Algoritmo	Vantagens	Desvantagens	Cenário Ideal
Heapsort	Complexidade $O(n \log n)$ garantida, in-place	Não é estável, mais lento que Quicksort	Listas grandes com restrição de memória
Mergesort	Estável, complexidade $O(n \log n)$ garantida	Requer memória extra	Grandes volumes de dados, listas encadeadas
Quicksort	Rápido em média, in-place, bom com pivô certo	Pior caso $O(n^2)$ , não é estável	Listas grandes, sistemas de alta performance
Radix Sort	Complexidade quase linear, estável	Restrito a inteiros/strings, consome memória	Números inteiros ou strings fixas
Shellsort	Bom para listas médias, baixo consumo de memória	Complexidade depende da sequência	Listas médias e parcialmente ordenadas

# EXEMPLO EM CÓDIGO

[HTTPS://ONECOMPILER.COM/C/42WKAB2VN](https://onecompiler.com/c/42WKAB2VN)

Aqui em nosso exemplo é usada a função particionar que escolhe o último elemento como pivô e organiza o array ao redor dele: elementos menores vão à esquerda e maiores à direita. A função quicksort aplica o Quicksort recursivamente às partes esquerda e direita do pivô. A função trocar realiza trocas entre dois elementos, e imprimirArray exibe o array. Ao final, o array está ordenado.

**Output:**

Fila original: 10 7 8 9 1 5

Fila ordenada: 1 5 7 8 9 10

**OBRIGADO!**