

# Algoritmos Rápidos de Ordenação

Técnicas de Programação - 2025/02

Última atualização: 03/09/2025 18:14

---

Neste atividade vamos estudar os passos 1 e 2 do algoritmo *Quick sort*, que é definido pela seguinte ideia em alto nível:

1. escolha um elemento do vetor (pode ser o primeiro). Vamos chamar ele de **pivô**
2. coloque o **pivô** no lugar certo em termos de ordenação.
  - índice  $i$  tal que todo  $A[j] \leq A[i], j < i$
  - índice  $i$  tal que todo  $A[j] > A[i], j > i$
3. ordene a parte à esquerda de  $i$
4. ordene a parte à direita de  $i$

Vamos **particionar** o array em torno de um elemento **pivô**. Supondo que **pivô** tenha sido movido para a posição  $i$ , teremos duas partes:

1. uma parte esquerda no intervalo início até  $i$
2. uma parte direita no intervalo  $i + 1$  até o fim

Note que, como o pivô já está na posição correta, ele não entra em nenhum das duas partes. Resta então pensar em como fazer essa partição de modo que, se o lugar certo do pivô é a posição  $i$ ,

- $A[i] > A[j]$  se  $i > j$
- $A[i] < A[j]$  se  $i < j$

Chamaremos os passos 1 e 2 de **PARTICIONA**( $A$ ,  $ini$ ,  $fim$ ). Vamos então tentar simular a seguinte ideia para estes passos:

1. vamos usar a variável  $lp = ini$  para marcar lugar atual do pivô no subarray  $[ini, fim)$
2. começamos então a percorrer o subarray, usando  $j \leftarrow ini + 1$  **to**  $fim$ , sendo que  $j$  marca o elemento atual.
  1. se o valor do elemento atual for **maior** que o valor do pivô, não faça nada
  2. se o valor do elemento atual for **menor** que o valor do pivô,
    1. aumente um em  $lp$
    2. troque  $A[j]$  com  $A[lp]$
3. troque  $A[ini]$  com  $A[lp]$

Como anteriormente, vamos simular esse algoritmo para entender melhor seu funcionamento. Desta vez ele não está em pseudo código completamente, mas já está bem mais estruturado e fácil de seguir do que se fosse descrito em texto corrido.

$j$	$lp$	$A$
1	1	$A = [6, 3, 1, 2, 7, 5, 4, 8]$

$j$	$lp$	$A$
-----	------	-----

**Exercício:** Mais uma vez, mas agora simule para o array  $A = [500, 1, 2, 3, 466, 2]$ .

---

$j$	$lp$	$A$
-----	------	-----

---

**Exercício:** e agora vamos formalizar esses passos em um algoritmo escrito em pseudo código.

---

**Algorithm 1:** Particiona

---

**Input** : array  $A$ , int  $ini$ , int  $fim$

**Output:** int

**Result:** Coloca o elemento  $A[ini]$  na posição correta dentro do subarray  $A[ini \dots fim]$ . Devolve o índice em que o valor  $A[ini]$  foi parar.

---