# Algoritmo QuickSort em C - Explicação e Código

## O que é o QuickSort?

QuickSort é um algoritmo de ordenação eficiente que usa a estratégia de dividir para conquistar. Ele organiza os elementos de um array particionando-o em dois subarrays, com base em um elemento pivô, e então aplica recursivamente a ordenação nesses subarrays.

## Funcionamento Passo a Passo

1. Escolhe-se um pivô (geralmente o último elemento do array).

2. Reorganizam-se os elementos para que os menores que o pivô fiquem à esquerda e os maiores à direita.

3. Aplica-se recursivamente o mesmo procedimento aos subarrays à esquerda e à direita do pivô.

4. Quando os subarrays têm tamanho 0 ou 1, a ordenação está completa.

## Complexidade do Algoritmo

- Melhor caso: O(n log n)

- Caso médio: O(n log n)

- Pior caso: O(n²), quando os elementos estão quase ordenados e o pivô sempre separa mal os dados.

## Implementação em C (com ponteiros)

#include <stdio.h>

void trocar(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int particionar(int \*arr, int inicio, int fim) {

int pivo = arr[fim];

int i = inicio - 1;

for (int j = inicio; j < fim; j++) {

if (arr[j] < pivo) {

i++;

trocar(&arr[i], &arr[j]);

}

}

trocar(&arr[i + 1], &arr[fim]);

return i + 1;

}

void quicksort(int \*arr, int inicio, int fim) {

if (inicio < fim) {

int indice\_pivo = particionar(arr, inicio, fim);

quicksort(arr, inicio, indice\_pivo - 1);

quicksort(arr, indice\_pivo + 1, fim);

}

}

void imprimir(int \*arr, int n) {

for (int i = 0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

}

int main() {

int vetor[10] = {8, 3, 1, 7, 0, 10, 2, 9, 4, 6};

int n = 10;

printf("Original: ");

imprimir(vetor, n);

quicksort(vetor, 0, n - 1);

printf("Ordenado: ");

imprimir(vetor, n);

return 0;

}

## Observações

- O uso de ponteiros permite troca eficiente de elementos e acesso direto à memória.

- Escolhas alternativas de pivô incluem: primeiro elemento, pivô aleatório, mediana de três.

- QuickSort é mais eficiente que Bubble Sort e Insertion Sort para grandes volumes de dados.

## Referências

- Cormen, T. H. et al. Introduction to Algorithms.

- Sedgewick, R. Algorithms in C.

- Visualização online: https://visualgo.net/en/sorting