QUICK SORT

Entendendo um dos algoritmos de ordenação mais eficientes da computação.

O QUE É O QUICK SORT?

- Quick Sort é um algoritmo de ordenação eficiente.
- Baseado no princípio de Dividir e Conquistar.
- Escolhe um pivô e divide os elementos em dois grupos:
  - Menores ou iguais ao pivô (esquerda).
  - Maiores que o pivô (direita).
- Repete esse processo até o vetor estar ordenado.

COMPLEXIDADE:

- Melhor Caso: O(n log n) – ocorre quando o pivô é sempre o elemento central da lista.

- Pior Caso: $O(n^2)$ – ocorre quando o pivô é sempre o menor ou o maior elemento da lista, resultando em sublistas com tamanhos muito desequilibrados.

- Caso Médio: O(n log n) – é o desempenho típico do Quicksort.

🚀 COMO O QUICK SORT FUNCIONA?

Escolhe-se um elemento como pivô (neste código, sempre o último elemento da lista analisada).

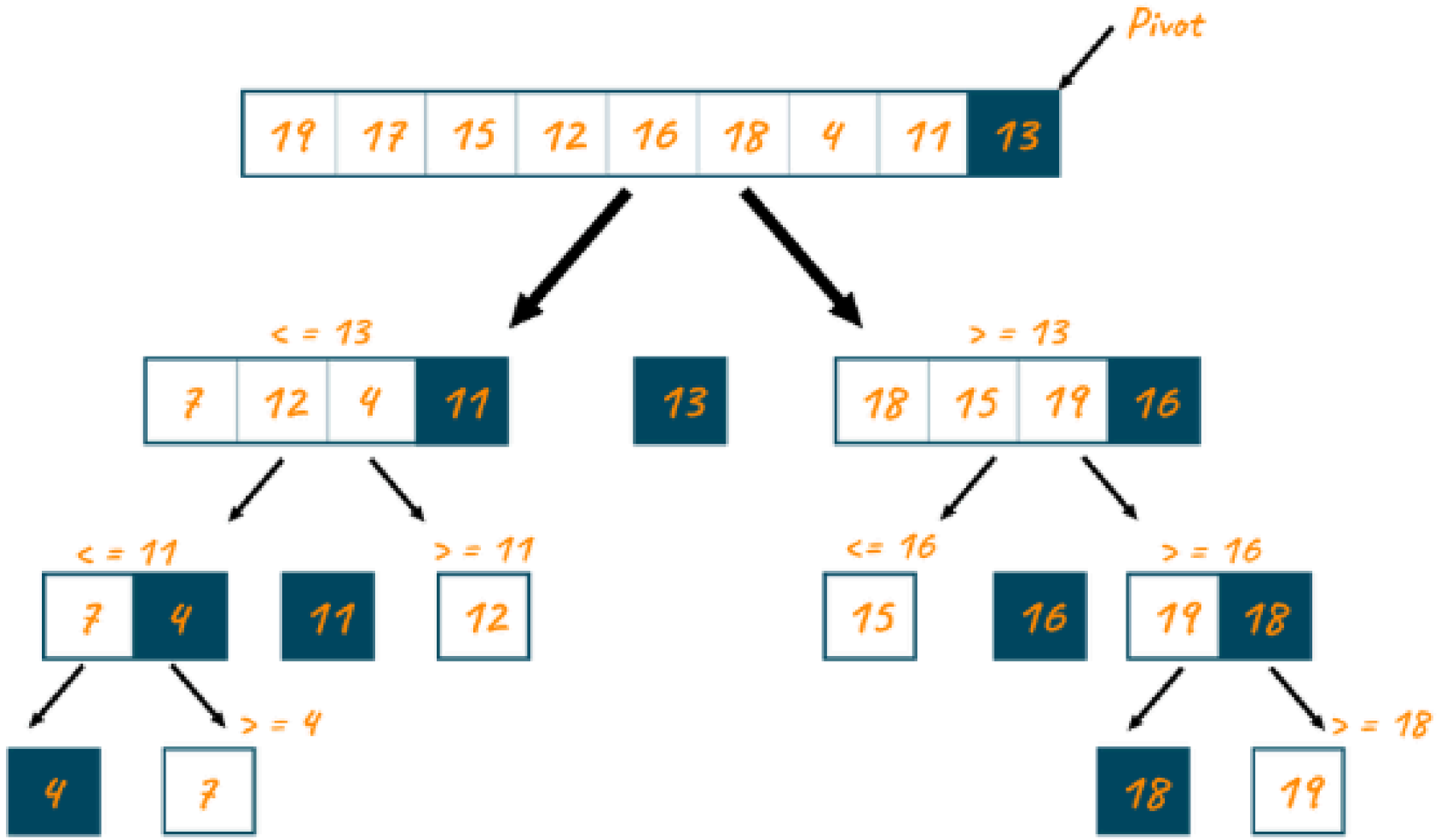
Reorganiza-se o vetor em duas partes:

- 🔸 Menores ou iguais ao pivô.
- 🔸 Maiores que o pivô.

Aplica-se o Quick Sort de forma recursiva nessas duas partes.

Isso se repete até as sublistas terem tamanho 1.

Quick Sort Algorithm



💻 ESTRUTURA DO CÓDIGO EM C

```c
//troca de valores de duas variáveis
void trocar(int *x, int *y){
      int temp = *x;
      *x = *y;
      *y = temp;
}
```

💻 ESTRUTURA DO CÓDIGO EM C

```c
//divide o vetor em duas partes
int particionar(int vetor[], int inicio, int fim){
    int pivo = vetor[fim];
    int i = inicio - 1;

    for(int j=inicio; j<fim; j++){
        if(vetor[j] <= pivo){
            i++;
            trocar(&vetor[i], &vetor[j]);
        }
    }
    trocar(&vetor[i+1], &vetor[fim]);
    return i + 1;
}
```

💻 ESTRUTURA DO CÓDIGO EM C

```c
//quicksort recursivo
void quickSort(int vetor[], int inicio, int fim){
    if(inicio < fim){
        int pivo = particionar(vetor, inicio, fim);
        quickSort(vetor, inicio, pivo - 1);
        quickSort(vetor, pivo + 1, fim);
    }

}
```

ONDE ESTÁ A RECURSIVIDADE?

🔁 Na função quickSort() ocorrem duas chamadas dela mesma:

- Para a parte esquerda:
  - quickSort(vetor, inicio, pivo – 1);
- Para a parte direita:
  - quickSort(vetor, pivo + 1, fim);

🧠 A função se repete até que o vetor esteja totalmente ordenado.

VANTAGENS E DESVANTAGENS

✅ Vantagens:

- Muito rápido na maioria dos casos.
- Simples de entender visualmente.
- Eficiente em médias e grandes quantidades de dados.

❌ Desvantagens:

- No pior caso (vetor já ordenado), pode ter desempenho ruim.
- Pode consumir muita pilha (recursividade profunda).

CONCLUSÃO

Em resumo, o Quicksort é uma opção popular para ordenação, mas é importante considerar a possibilidade de desempenho de pior caso e a necessidade de ordenação estável ao escolher um algoritmo de ordenação