

Algoritmos e Estruturas de Dados

Quicksort

Prof. Maiquel de Brito
maiquel.b@ufsc.br

Quicksort

Proposto por Hoare em 1960 e publicado em 1962

Apresenta bom desempenho em várias situações

Algoritmo

Dividir para conquistar:

Particionar o conjunto de N itens em problemas menores e ordenar as várias partes independentemente

- Uma vez efetuada a partição, cada uma das partes pode ser ordenada pelo mesmo algoritmo (de forma recursiva)
- A parte mais delicada do quicksort é o processo de partição
- O vetor v é reorganizado por meio da escolha arbitrária de um pivô p
- Os elementos do vetor v são reposicionados de forma que:
 - **Partição esquerda: $chaves \leq p$**
 - **Partição direita: $chaves \geq p$**

Particionamento

Algoritmo

- Escolha arbitrariamente o pivô **p**
- Percorra a partir da esquerda encontrar $v[\mathbf{e}] \geq \mathbf{p}$
- Percorra a partir da direita encontrar $v[\mathbf{d}] \leq \mathbf{p}$
- Troque $v[\mathbf{e}]$ com $v[\mathbf{d}]$
- Repita os passos anteriores até que **e** e **d** se cruzem ($\mathbf{d} < \mathbf{e}$)

Particionamento


Ao final do particionamento

- Pelo menos um elemento (pivô) está em sua posição final
- Elementos na partição esquerda são menores
- Elementos na partição da direita são maiores que o pivô



- Selecionando o pivô como $v[(\text{esq}+\text{dir})/2] = 4$ ($\text{esq} = 0$, $\text{dir} = 4$)

| | esq | | | | p | | | | dir |
|---------|-----|---|---|---|---|---|---|---|-----|
| | | e | | | | | d | | |
| Passo 1 | 2 | 7 | 4 | 1 | 5 | 3 | 0 | 8 | 6 |
| | | | | | | | | | |
| | | d | | | | | e | | |
| Passo 2 | 2 | 0 | 4 | 1 | 5 | 3 | 7 | 8 | 6 |
| | | | | | | | | | |
| | | | | | e | d | | | |
| Passo 3 | 2 | 0 | 4 | 1 | 5 | 3 | 7 | 8 | 6 |
| | | | | | | | | | |
| | | | | | d | e | | | |
| Passo 4 | 2 | 0 | 4 | 1 | 3 | 5 | 7 | 8 | 6 |
| | | | | | | | | | |



- Selecionando o pivô como $v[(\text{esq}+\text{dir})/2] = 4$ ($\text{esq} = 0$, $\text{dir} = 4$)

| | esq | | p | | dir | | | | |
|---------|------------|---|---|---|------------|---|---|---|---|
| | | | e | | d | | | | |
| Passo 4 | 2 | 0 | 4 | 1 | 3 | 5 | 7 | 8 | 6 |
| | | |  | | | | | | |
| | | | d | | e | | | | |
| Passo 5 | 2 | 0 | 3 | 1 | 4 | 5 | 7 | 8 | 6 |
| | | |  | | | | | | |

- Selecionando o pivô como $v[(\text{esq}+\text{dir})/2] = 3$ ($\text{esq} = 1$, $\text{dir} = 3$)

| | | esq | p | dir | | | | | |
|---------|---|-----|---|-----|---|---|---|---|---|
| | | | e | d | | | | | |
| Passo 8 | 0 | 2 | 3 | 1 | 4 | 5 | 7 | 8 | 6 |
| | | |  | | | | | | |
| | | | d | e | | | | | |
| Passo 9 | 0 | 2 | 1 | 3 | 4 | 5 | 7 | 8 | 6 |
| | | |  | | | | | | |

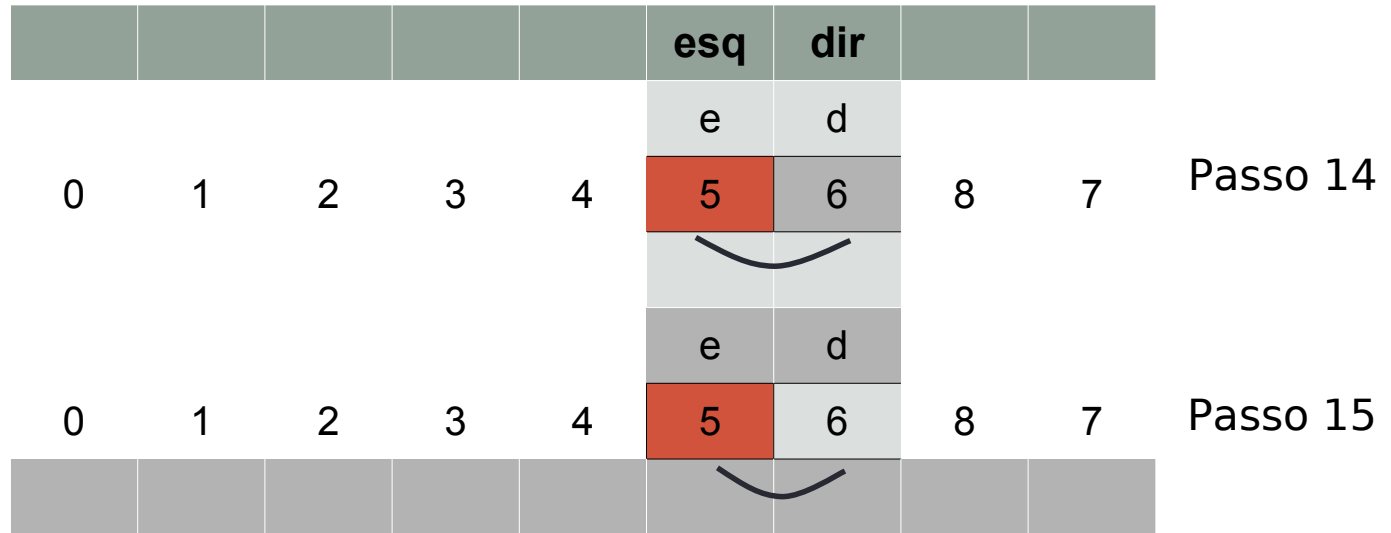
■ Seleccionando o pivô como $v[(\text{esq}+\text{dir})/2] = 2$ ($\text{esq} = 1$, $\text{dir} = 2$)

| | | esq | dir | | | | | | |
|----------|---|---|------------|---|---|---|---|---|---|
| | | e | d | | | | | | |
| Passo 10 | 0 | 2 | 1 | 3 | 4 | 5 | 7 | 8 | 6 |
| | |  | | | | | | | |
| | | d | e | | | | | | |
| Passo 11 | 0 | 1 | 2 | 3 | 4 | 5 | 7 | 8 | 6 |
| | |  | | | | | | | |

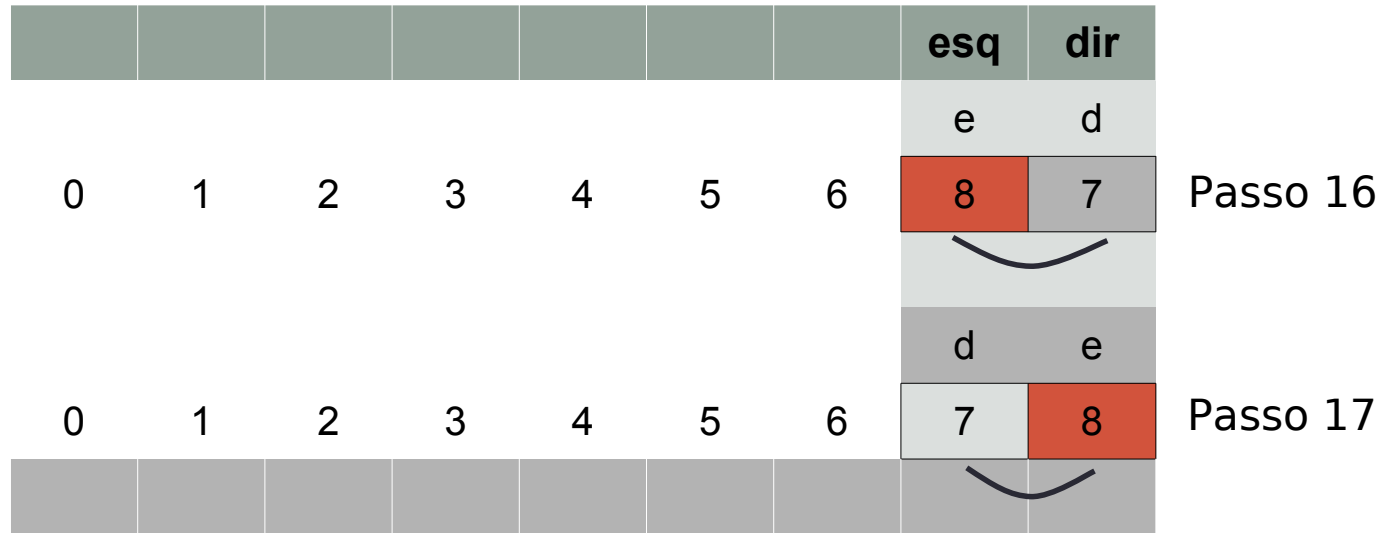
- Selecionando o pivô como $v[(\text{esq}+\text{dir})/2] = 7$ ($\text{esq} = 5$, $\text{dir} = 8$)

| | | | | | esq | p | | dir | |
|---|---|---|---|---|-----|---|---|-----|----------|
| | | | | | | e | | d | |
| 0 | 1 | 2 | 3 | 4 | 5 | 7 | 8 | 6 | Passo 12 |
| | | | | | | | | | |
| | | | | | | d | | e | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 7 | Passo 13 |
| | | | | | | | | | |

- Selecionando o pivô como $v[(\text{esq}+\text{dir})/2] = 5$ ($\text{esq} = 5$, $\text{dir} = 6$)



■ Seleccionando o pivô como $v[(\text{esq}+\text{dir})/2] = 8$ ($\text{esq} = 7$, $\text{dir} = 8$)



Implementação

```
void quicksort(int *v, int left, int right) {
    int i = left, j = right;
    int tmp;
    int pivot = v[(left + right) / 2];

    while (i <= j) {
        while (v[i] < pivot)
            i++;
        while (v[j] > pivot)
            j--;
        if (i <= j) {
            tmp = v[i];
            v[i] = v[j];
            v[j] = tmp;
            i++;    j--;
        }
    }
    if (left < j)
        quicksort(v, left, j);
    if (i < right)
        quicksort(v, i, right);
}
```

Eficiência do QuickSort

A eficiência do processo de ordenação depende de como a partição divide os dados.

Escolha pivô determina eficiência

- *pior caso*: pivô é o maior ou menor elemento $O(N^2)$
- *melhor caso*: pivô é o elemento médio $O(N \log N)$
- *caso médio*: pivô corta vetor arbitrariamente $O(N \log N)$

Escolha do pivô

- um dos elementos extremos do vetor:
 má escolha: $O(N^2)$ se vetor ordenado
- elemento aleatório: boa escolha
- mediana de três elementos (extremos do vetor e ponto médio): boa escolha

Vantagens e desvantagens

Vantagens

- Melhor caso é $O(n \cdot \log^n)$ e o caso médio tende a isso
- Boa probabilidade de evitar o pior caso, dependendo da escolha do pivô
(escolha aleatória tende a ser uma boa opção)
- Não requer memória extra porque não faz cópias do vetor
(isso não é capturado pela análise assintótica)

Desvantagens

- Pior caso é $O(n^2)$;
- Desempenho pode ser ruim em grandes volumes de dados: o quicksort faz muito acesso à memória, que pode ser lento no caso de memória virtual;
(isso não é capturado pela análise assintótica)