Algoritmos e Estruturas de Dados

Quicksort

Prof. Mauri Ferrandin

Quicksort

- Proposto por Hoare em 1960 e publicado em 1962
- É o algoritmo de ordenação interna mais rápido que se conhece para uma ampla variedade de situações
- Provavelmente é o mais utilizado

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 rearranjado por meio da escolha arbitrária de um pivô p
- O vetor v particionado em dois:
  - ***Partição esquerda: chaves ≤ p***
  - ***Partição direita: chaves ≥ p***

Particionamento

Algoritmo

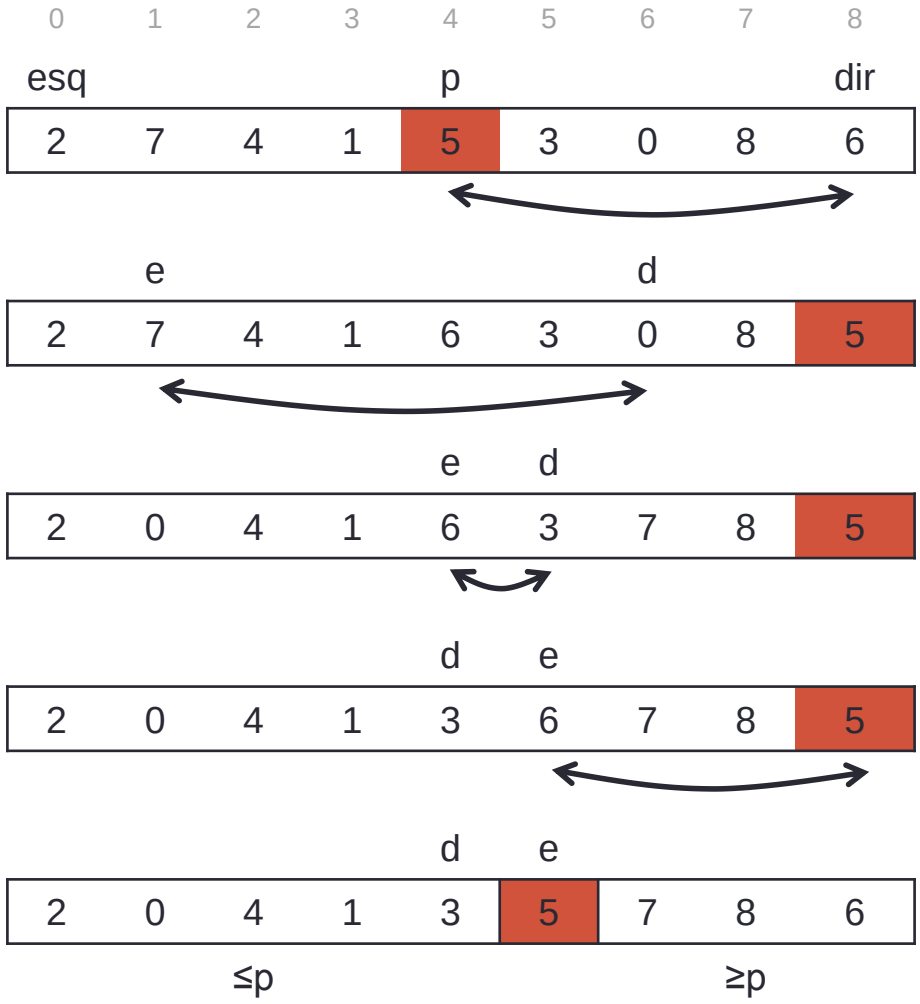
- Escolha arbitrariamente o pivô **p**
- Troque a posição do pivô com a do elemento mais a direita (v[**dir**])
- Percorra a partir da esquerda até que v[**e**] ≥ **p**
- Percorra a partir da direita até que v[**d**] ≤ **p**
- Troque v[**e**] com v[**d**]
- Repita os passos anteriores até que **e** e **d** se cruzem (**d** < **e**)
- Troque v[**e**] com v[**dir**]

Particionamento

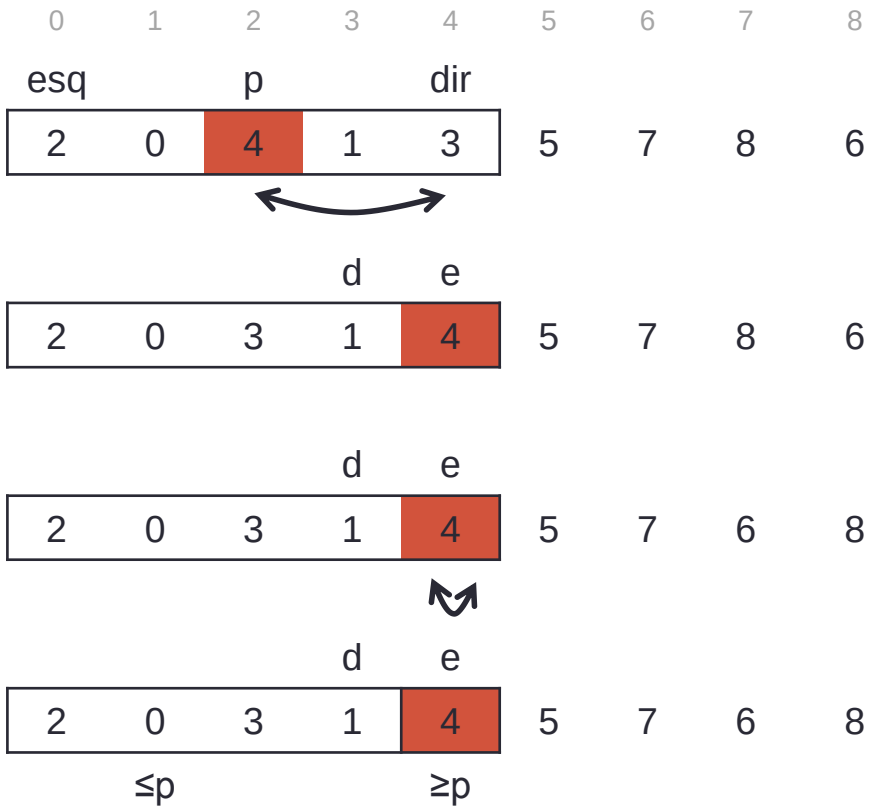
Ao final do particionamento

- 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

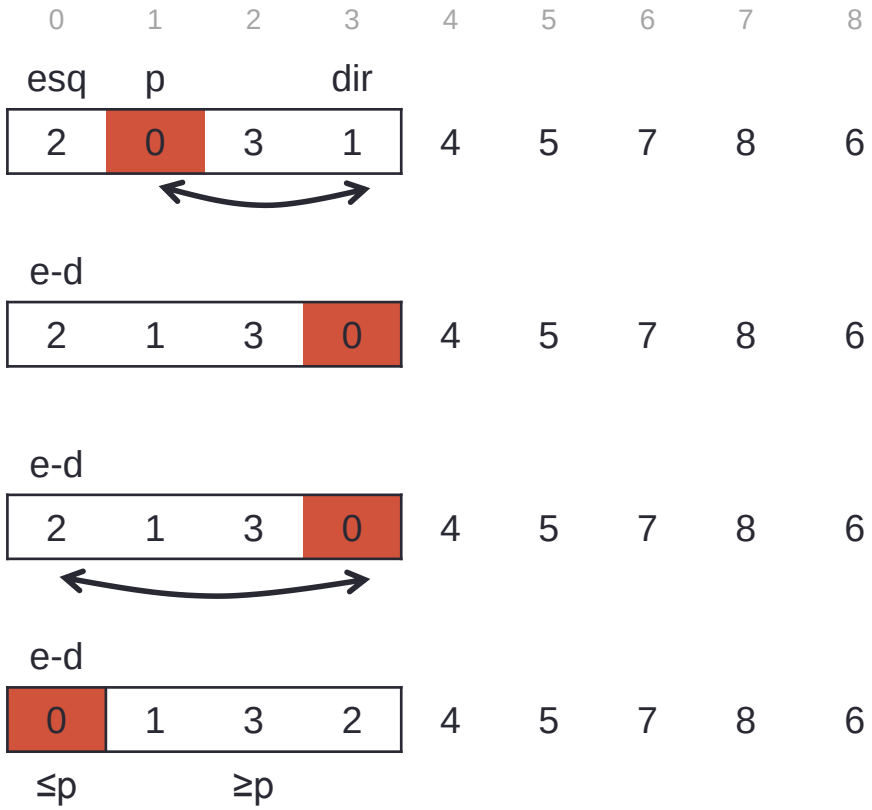
- Selecionando o pivô como v[(d+e)/2]



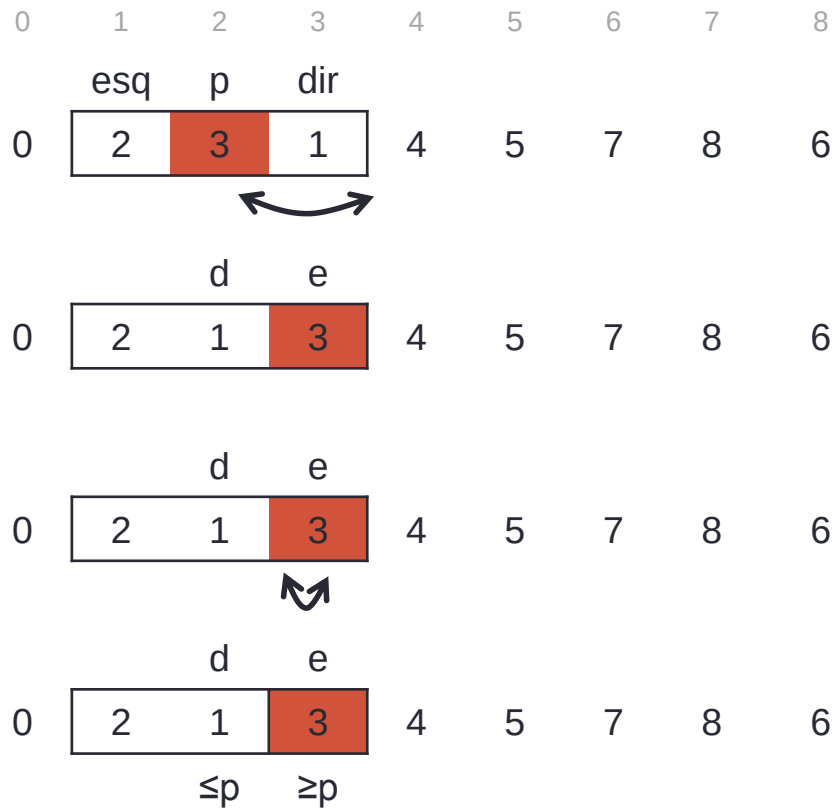
- Selecionando o pivô como v[(d+e)/2]



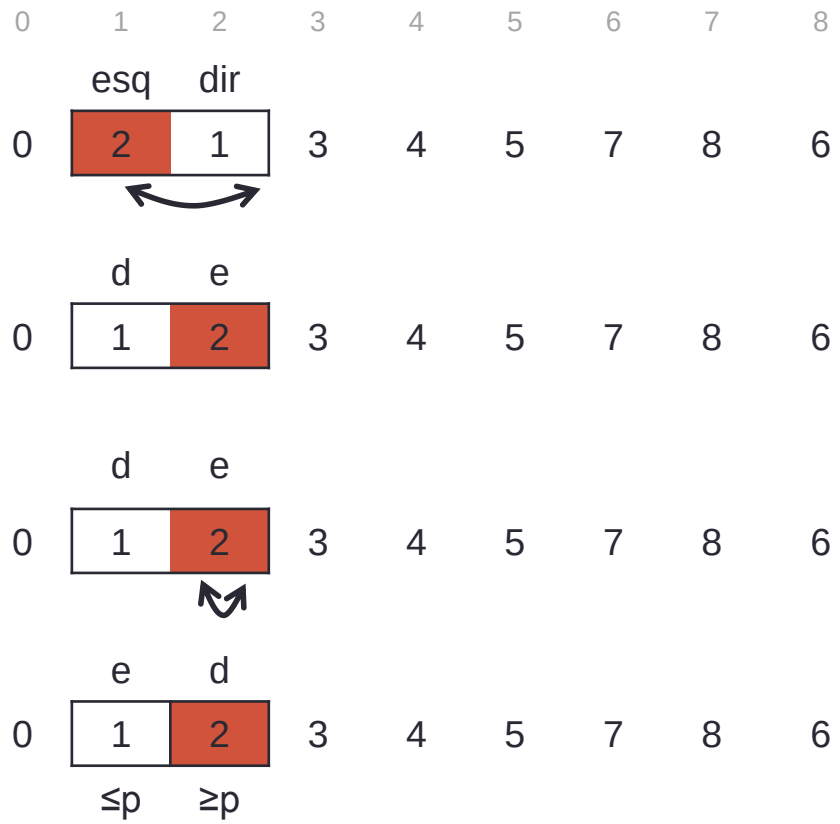
- Selecionando o pivô como v[(d+e)/2]



- Selecionando o pivô como v[(d+e)/2]

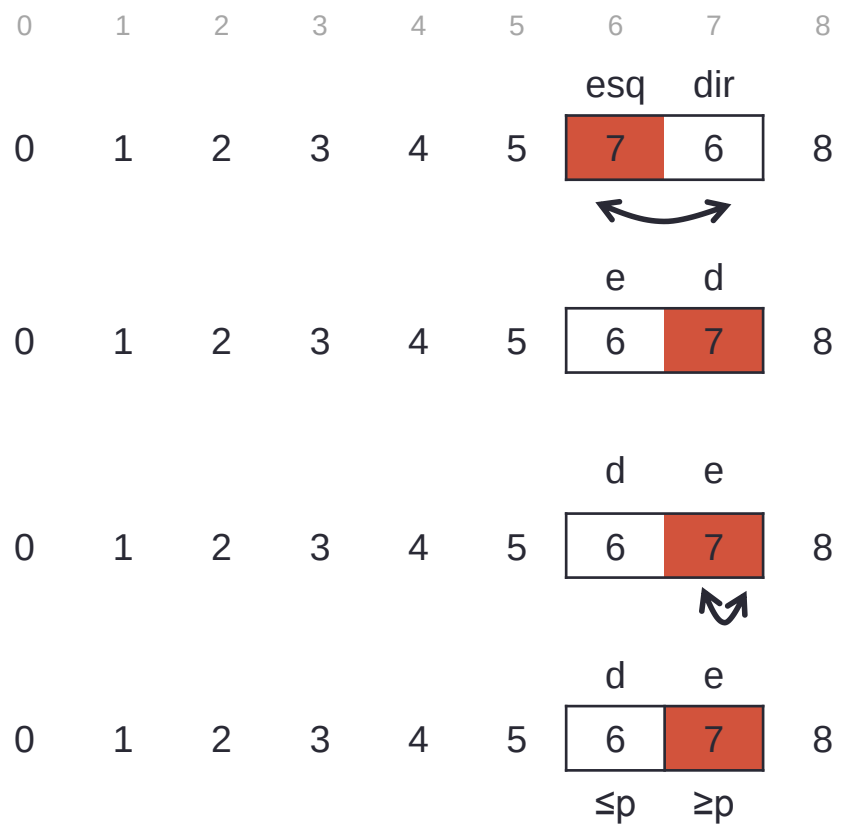


- Selecionando o pivô como v[(d+e)/2]



- Selecionando o pivô como v[(d+e)/2]

- Selecionando o pivô como v[(d+e)/2]



Implementação

```
void quicksort(int *v, int left, int right){
        if (left < right){
                int pivo = (left + right)/2, i = left, j = right -1;
                troca(&v[pivo],&v[right]);
                pivo = right;
                while (i < j){
                        while (i < right - 1 && v[i] < v[pivo])
                                i++;
                        while (j > 0 && v[j] > v[pivo])
                                j--;
                        if (i < j)
                                troca(&v[i++],&v[j--]);
                }
                if (i == j && v[i] < v[pivo])
                        i++;
                troca(&v[i], &v[pivo]);
                quicksort(v, left, i - 1);
                quicksort(v, i + 1, right);
        }
}
```

Ordenação por Partição (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 logN)
- *caso médio*: pivô corta vetor arbitrariamente O(N logN)

Escolha do pivô

- um dos elementos extremos do vector:
  - má escolha: $O(N^2)$ se vetor ordenado
- elemento aleatório:
  - envolve uso de mais uma função pesada
- mediana de três elementos (extremos do vetor e ponto médio)
  - recomendado

Vantagens e desvantagens

- Vantagens
  - Melhor opção para ordenar vetores grandes
  - Muito rápido por que o laço interno é simples
- Desvantagens
  - Não é estável (não é conhecida uma forma eficiente para tornar o quicksort estável)
  - Pior caso é quadrático