Busca e Ordenação

Algoritmos de Ordenação $O(N \log N)$

Prof. Edson Alves - UnB/FGA

2019

Sumário

QuickSort

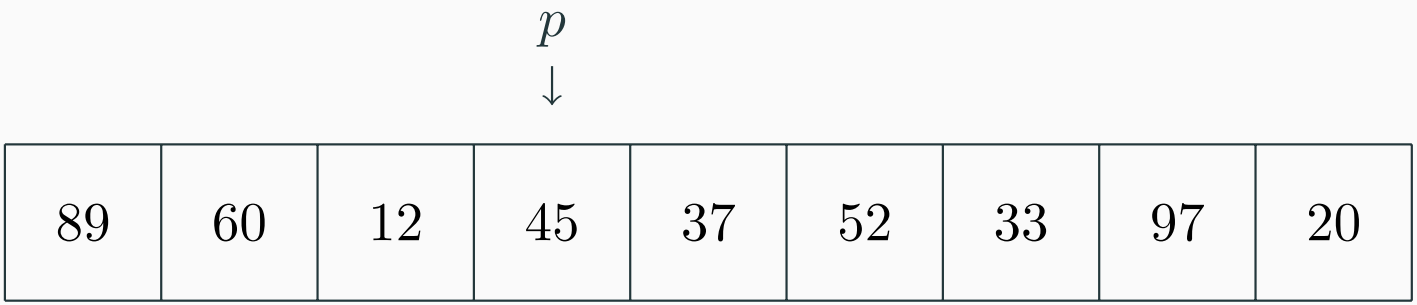
Motivação

- Embora o *MergeSort* seja um algoritmo que atinja a limite inferior $O(N \log N)$ para algoritmos de ordenação baseados em comparações, ele demanda uma memória adicional $O(N)$, não sendo portando um algoritmo *in-place*
- A ideia do *QuickSort* é aproveita a ideia da divisão do vetor em subvetores menores, como é feito no *MergeSort*
- Contudo, a ideia é que o algoritmo seja *in-place*
- Assim, a divisão do vetor não será posicional, mas sim baseada no valor de um elemento escolhido arbitrariamente, denominado pivô
- O pivô permite um rearranjo dos elementos usando a própria memória do vetor, tornando o algoritmo *in-place*
- Embora o *QuickSort* tenha complexidade média $O(N \log N)$, no pior caso ele pode se degenerar para $O(N^2)$

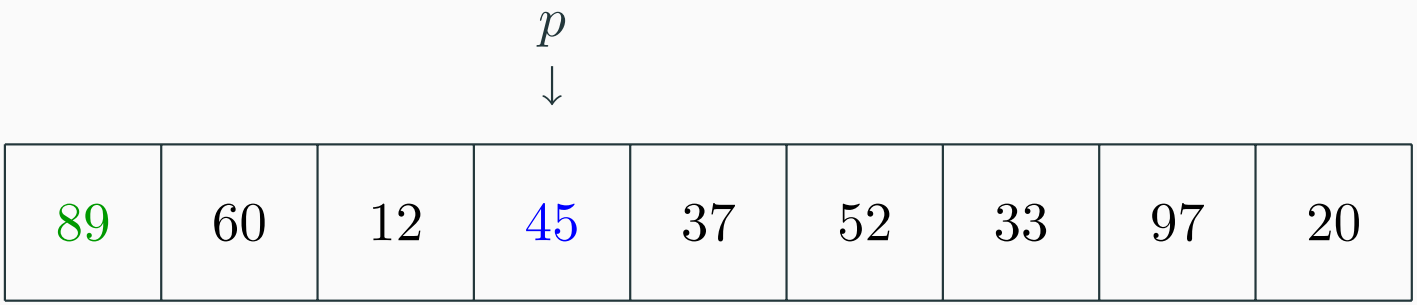
Pivoteamento

- Pivoteamento é o processo de reposicionamento dos elementos do vetor de acordo com o valor x do elemento pivô que ocupa o índice p
- Ao final do pivoteamento, todos elementos com valores menores que x estarão à esquerda do pivô, e os demais à direita
- O pivô já estará na posição correta em relação ao ordenamento global, de modo que o *QuickSort* pode prosseguir recursivamente nas duas partes separadas pelo pivô
- Para simplificar a rotina, no início do pivoteamento o pivô troca de posição com o primeiro elemento do vetor
- Ao final, o pivô se move para a posição adequada e esta posição é retornada
- Para evitar o pior caso, a escolha do pivô deve ser aleatória entre todos os índices possíveis

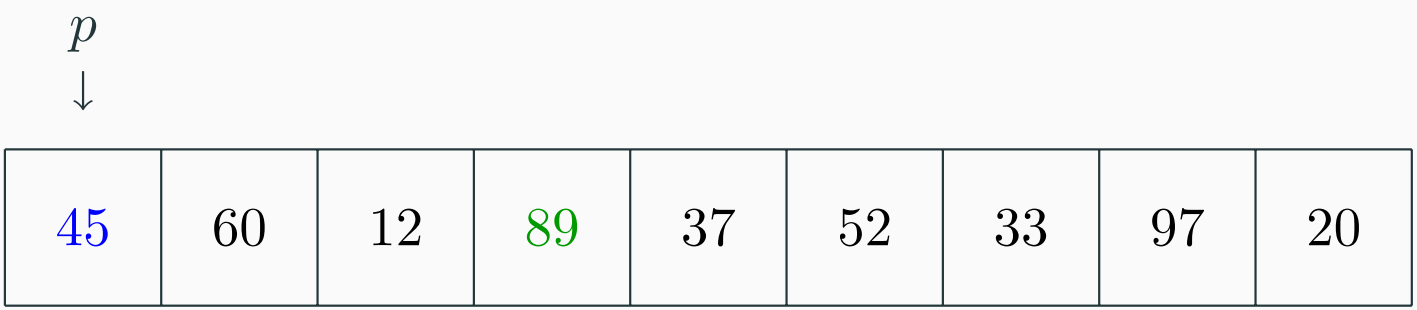
Visualização da rotina de pivoteamento



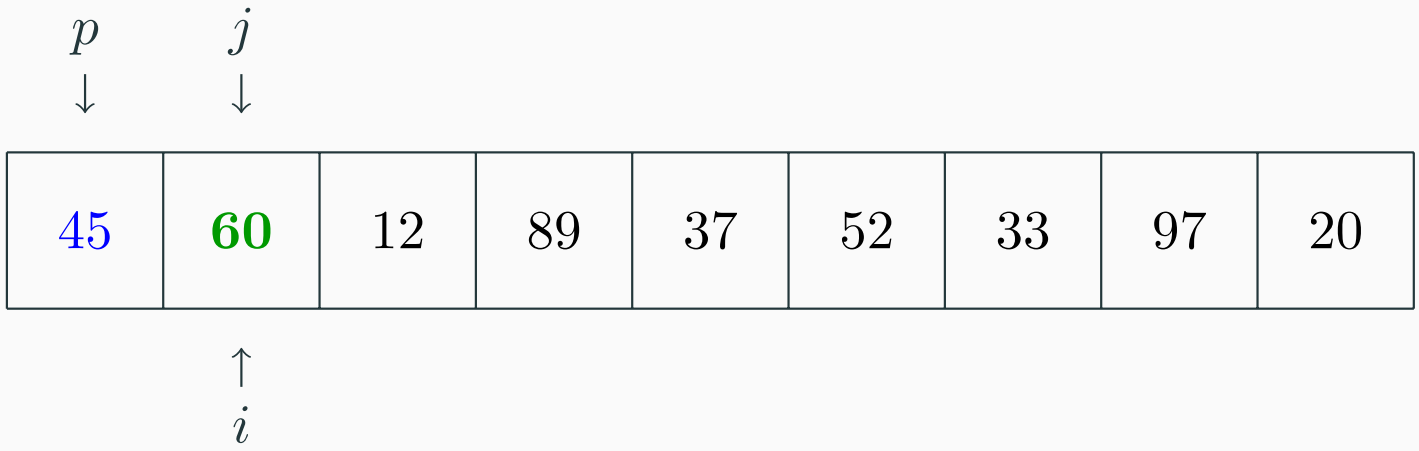
Visualização da rotina de pivoteamento



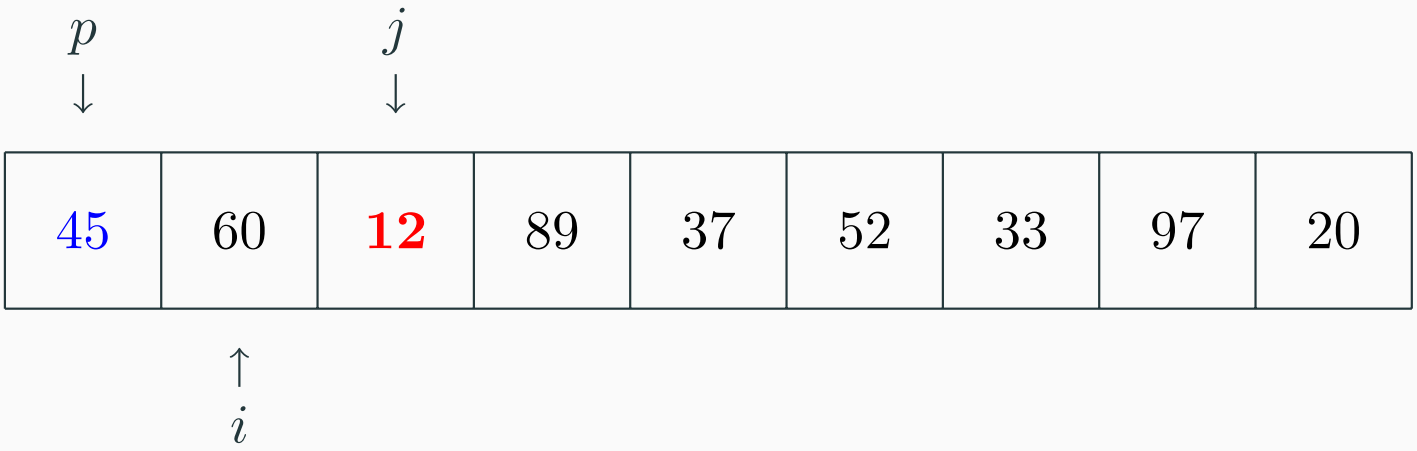
Visualização da rotina de pivoteamento



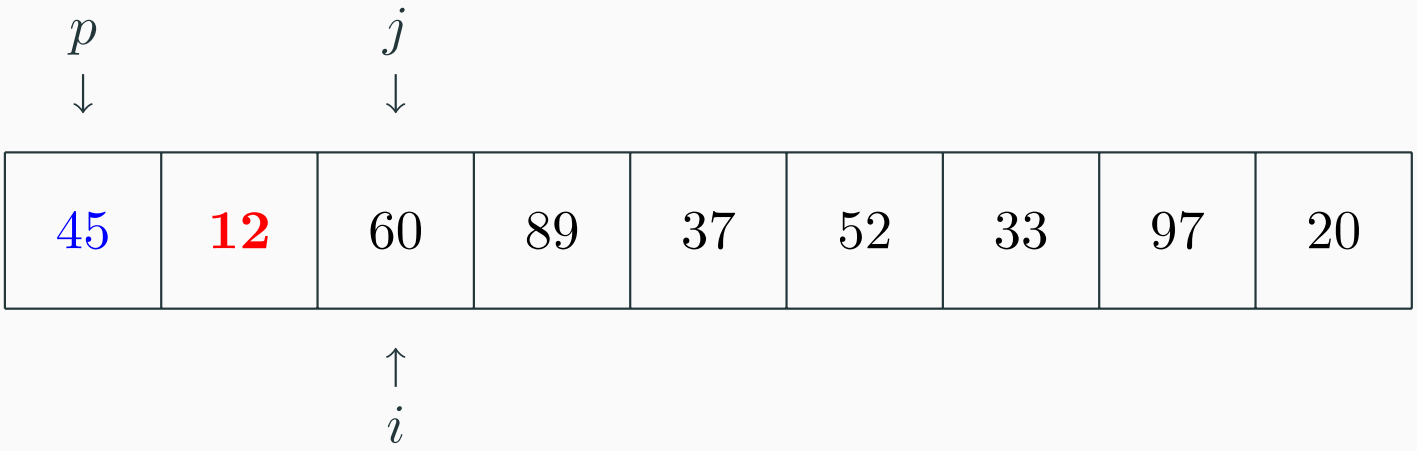
Visualização da rotina de pivoteamento



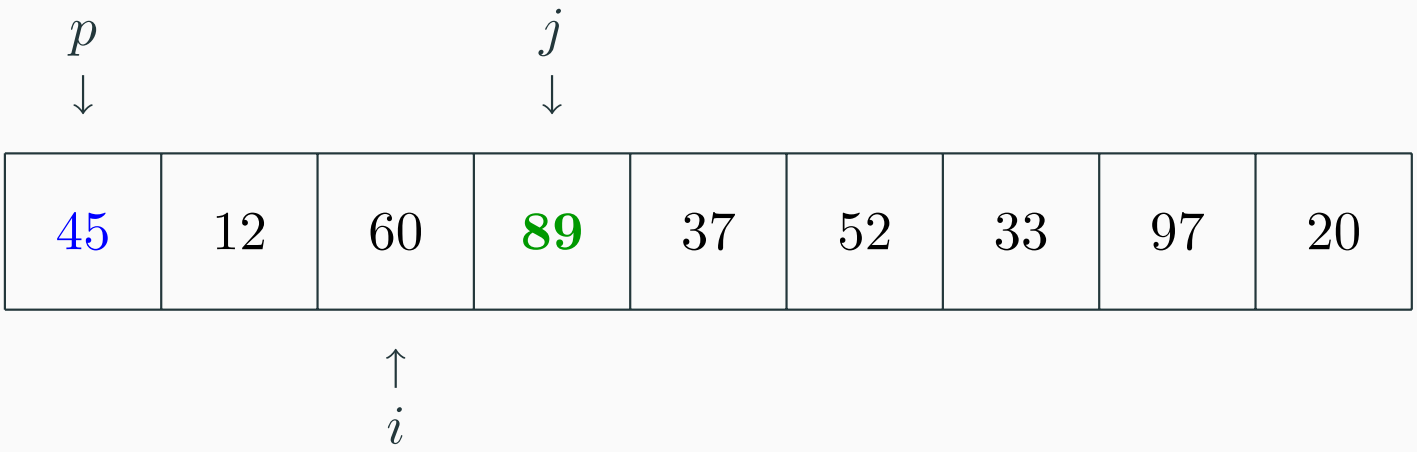
Visualização da rotina de pivoteamento



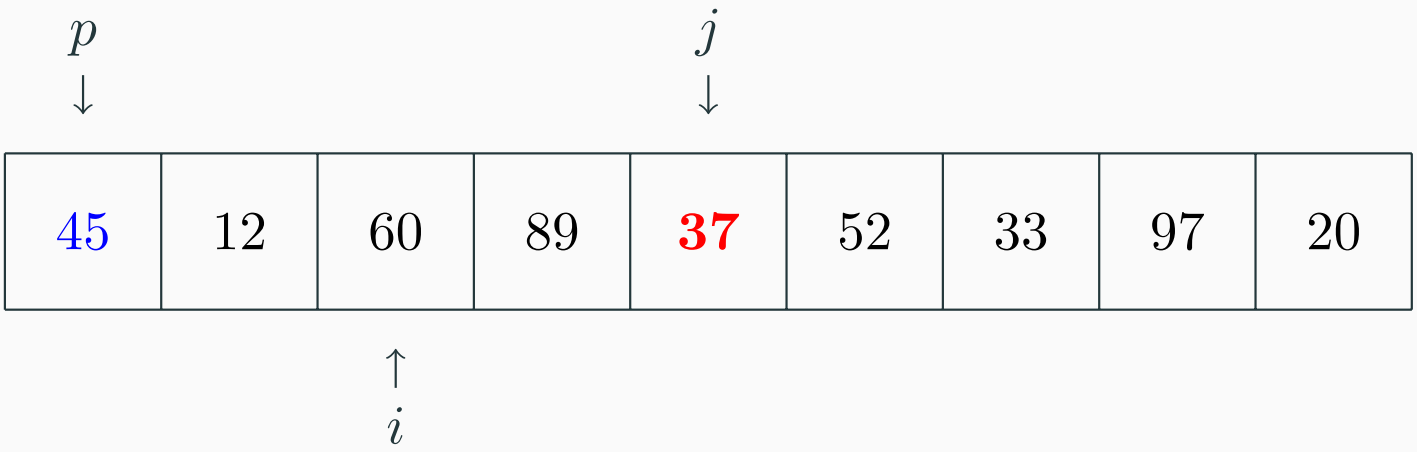
Visualização da rotina de pivoteamento



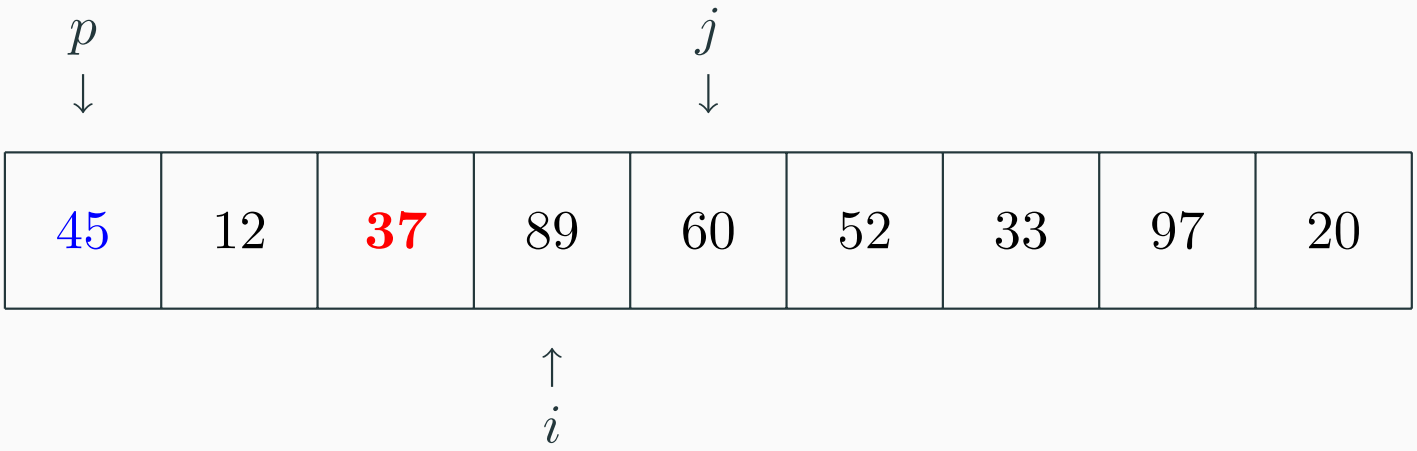
Visualização da rotina de pivoteamento



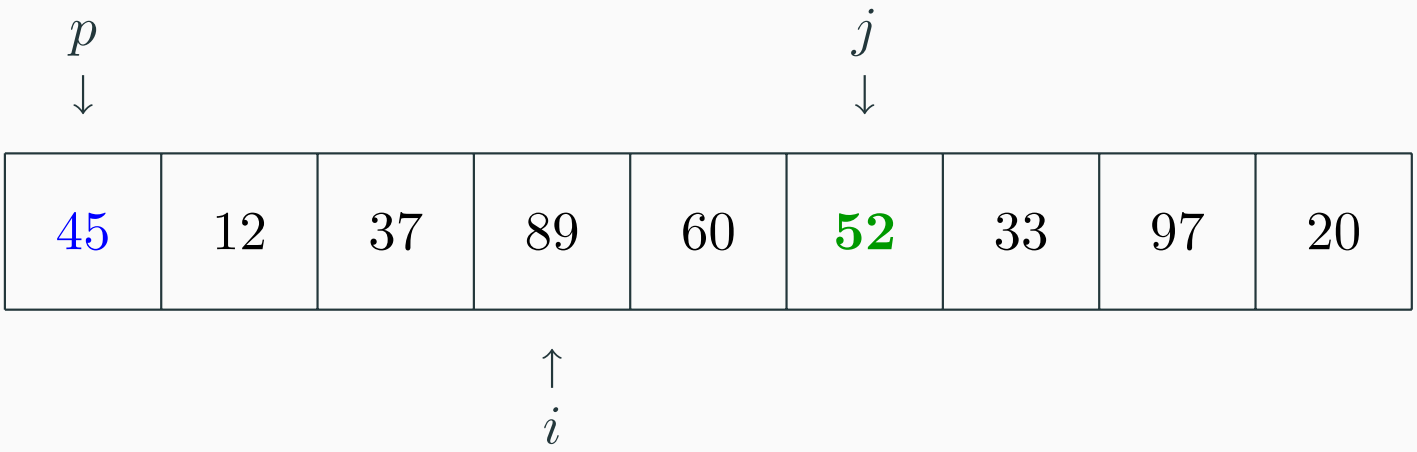
Visualização da rotina de pivoteamento



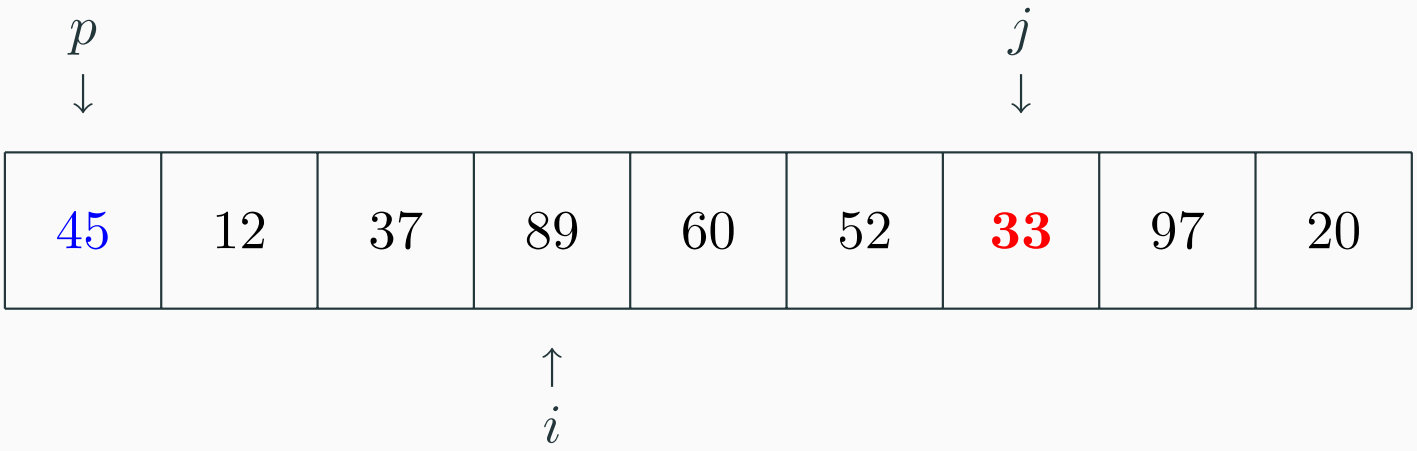
Visualização da rotina de pivoteamento



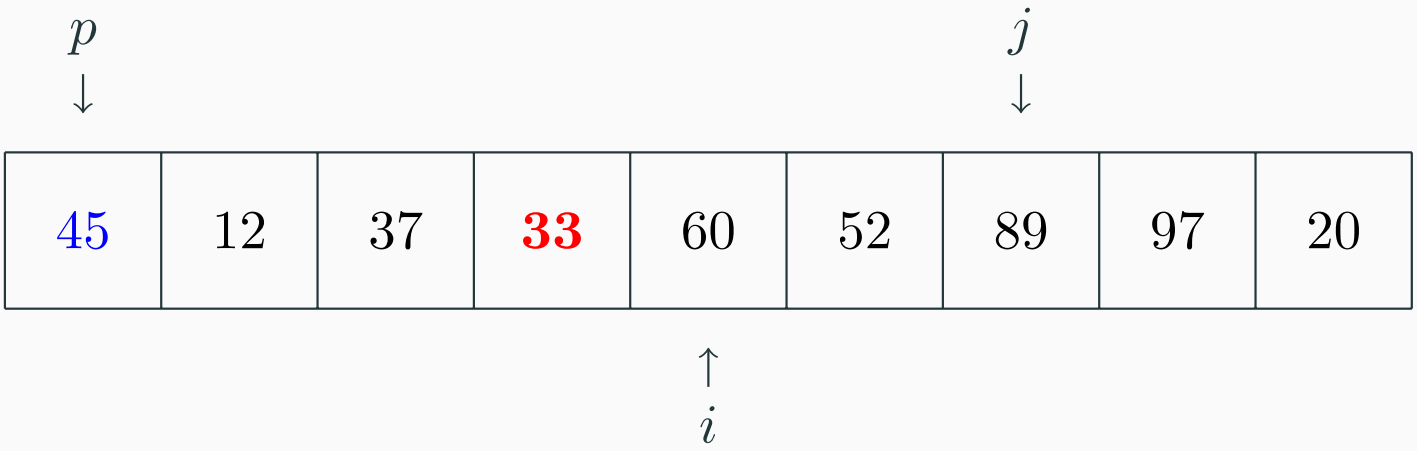
Visualização da rotina de pivoteamento



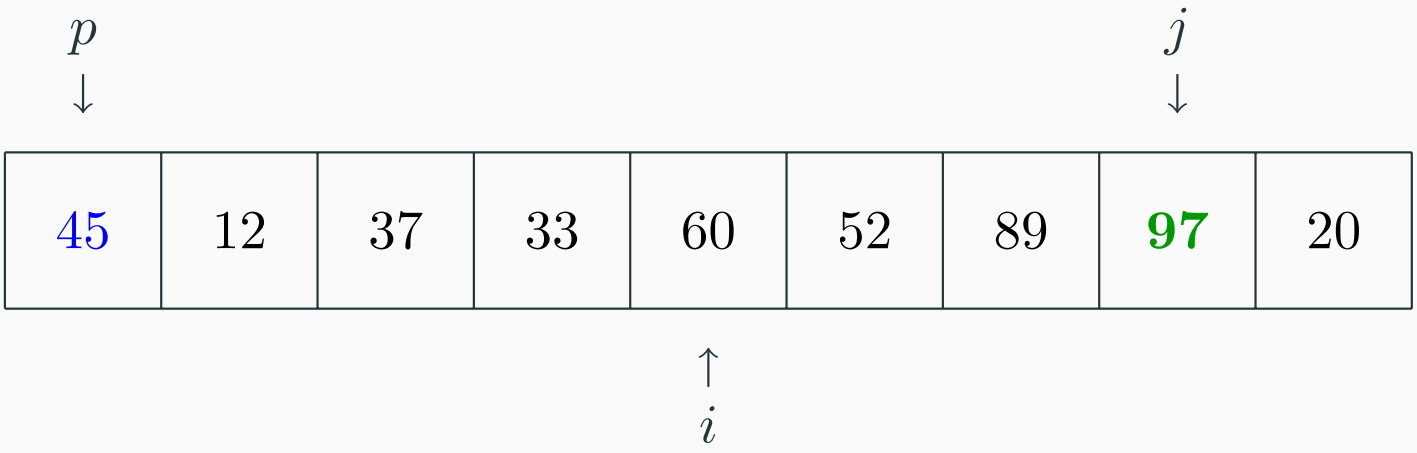
Visualização da rotina de pivoteamento



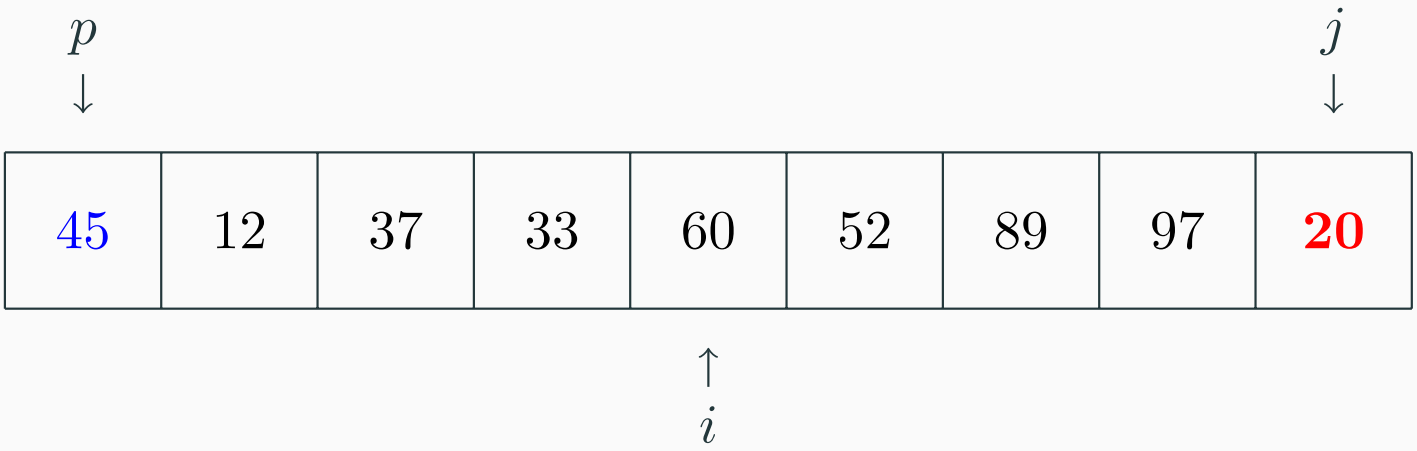
Visualização da rotina de pivoteamento



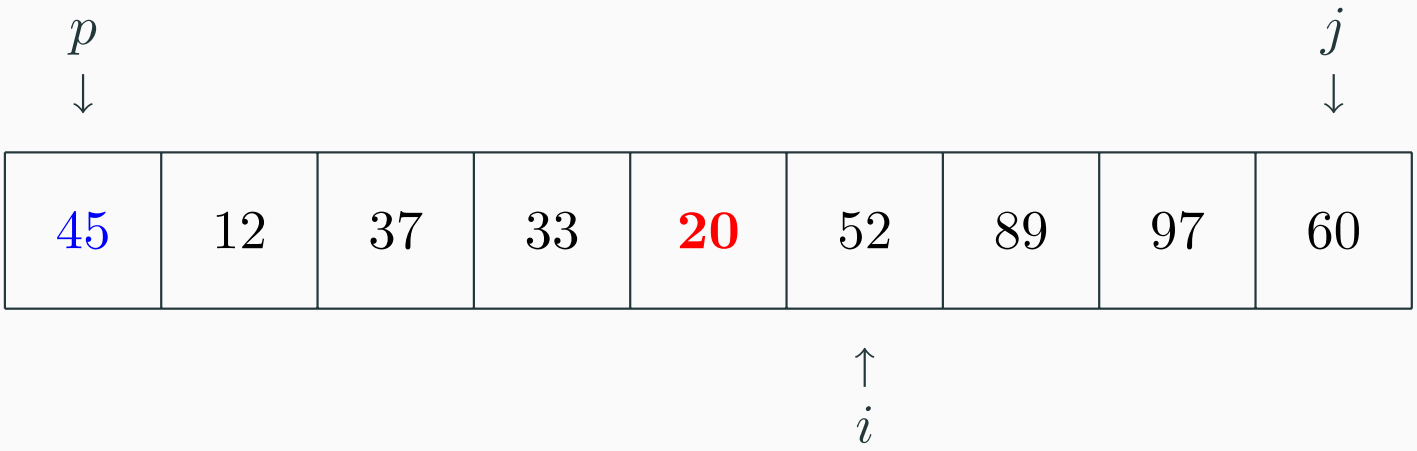
Visualização da rotina de pivoteamento



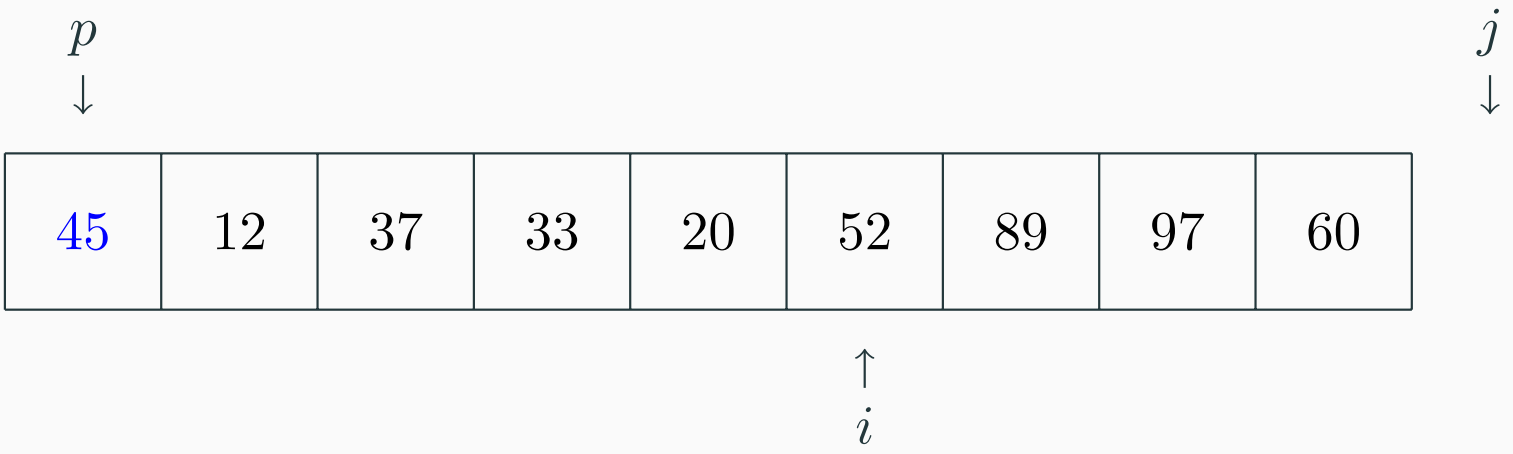
Visualização da rotina de pivoteamento



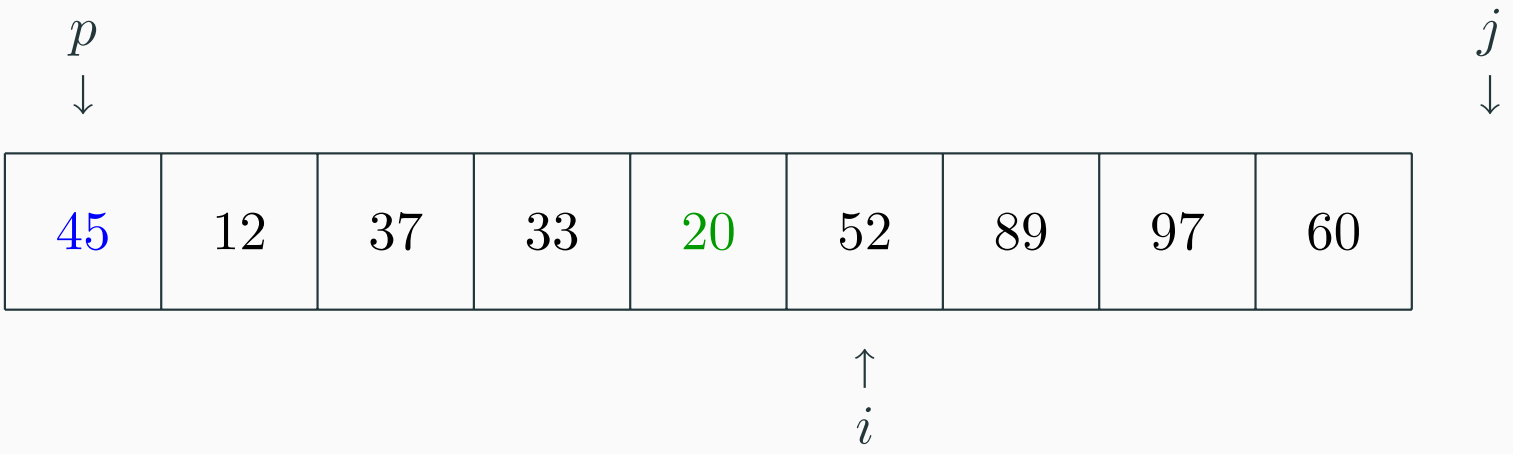
Visualização da rotina de pivoteamento



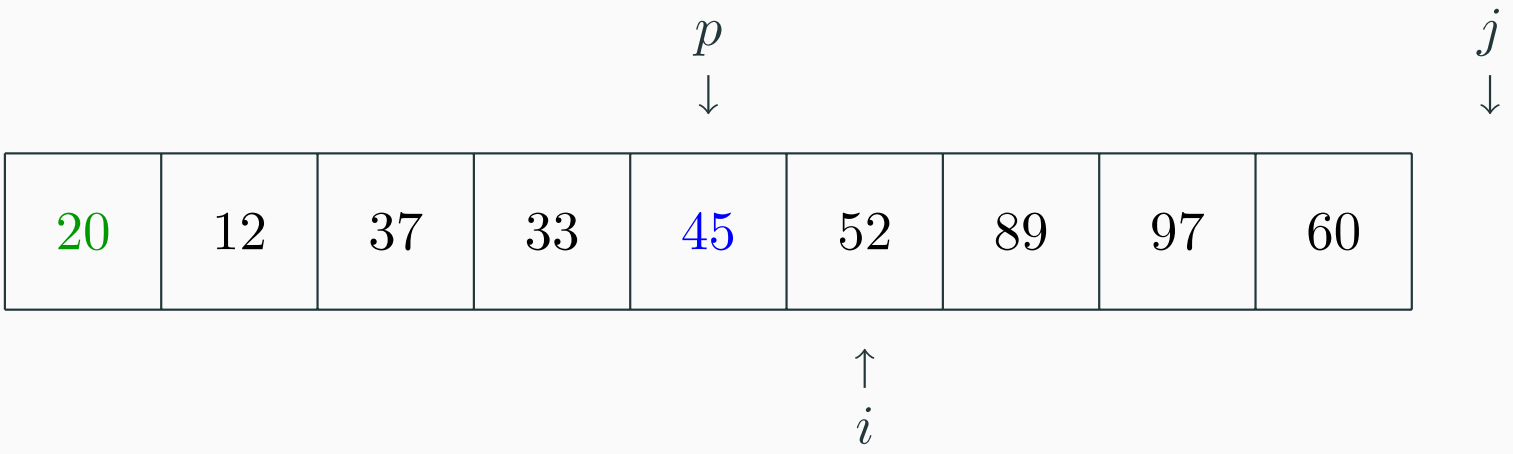
Visualização da rotina de pivoteamento



Visualização da rotina de pivoteamento



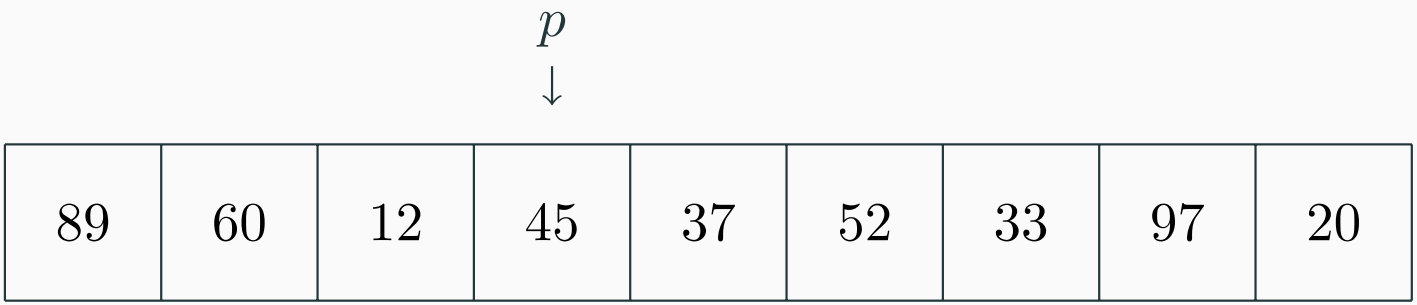
Visualização da rotina de pivoteamento



Implementação da rotina de pivoteamento

```
5  template<typename RandIt>
6  RandIt partitioning(RandIt first, RandIt last)
7  {
8      auto N = last - first;
9      RandIt p = first + (rand() % N);    // slide: RandIt p = first + 3;
10
11     swap(*first, *p);
12     p = first;
13     RandIt i = first + 1;
14
15     for (RandIt j = first + 1; j < last; ++j)
16         if (*j < *p)
17         {
18             swap(*j, *i);
19             ++i;
20         }
21
22     swap(*p, *(--i));
23
24     return i;
25 }
```

Visualização do *quicksort*



Referências

1. **DROZDEK**, Adam. *Algoritmos e Estruturas de Dados em C++*, 2002.
2. **ROUGHGARDEN**, Tim. *Algorithms Illuminated (Part 1): The Basics*, LLC, 2018.
3. **KERNIGHAN**, Bryan; **RITCHIE**, Dennis. *The C Programming Language*, 1978.
4. **STROUSTROUP**, Bjarne. *The C++ Programming Language*, 2013.