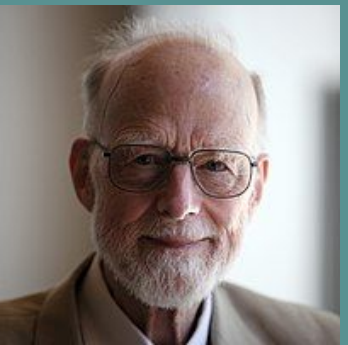
Quick Sort

Odolir D. dos Santos Junior
Luiz Eduardo Borgert Coelho

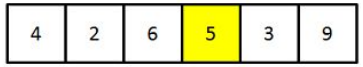
O que é?

- Método de ordenação muito rápido e eficiente.
- Utiliza a técnica “dividir para conquistar”
- Rearranja as chaves de modo com que as “menores” precedem as “maiores”



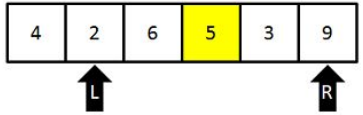
Charles Antony Richard Hoare
1960 (Universidade de Moscou)

Step 1
Determine pivot

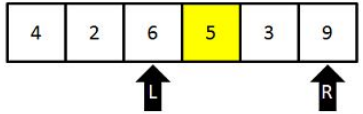


Step 2
Start pointers at left and right

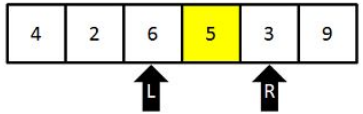
Step 3
Since 4 < 5, shift left pointer



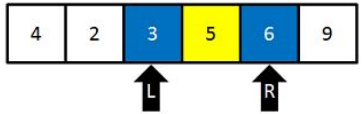
Step 4
Since 2 < 5, shift left pointer
Since 6 > 5, stop



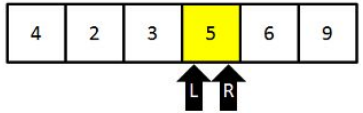
Step 5
Since 9 > 5, shift right pointer
Since 3 < 5, stop



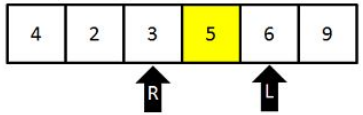
Step 6
Swap values at pointers

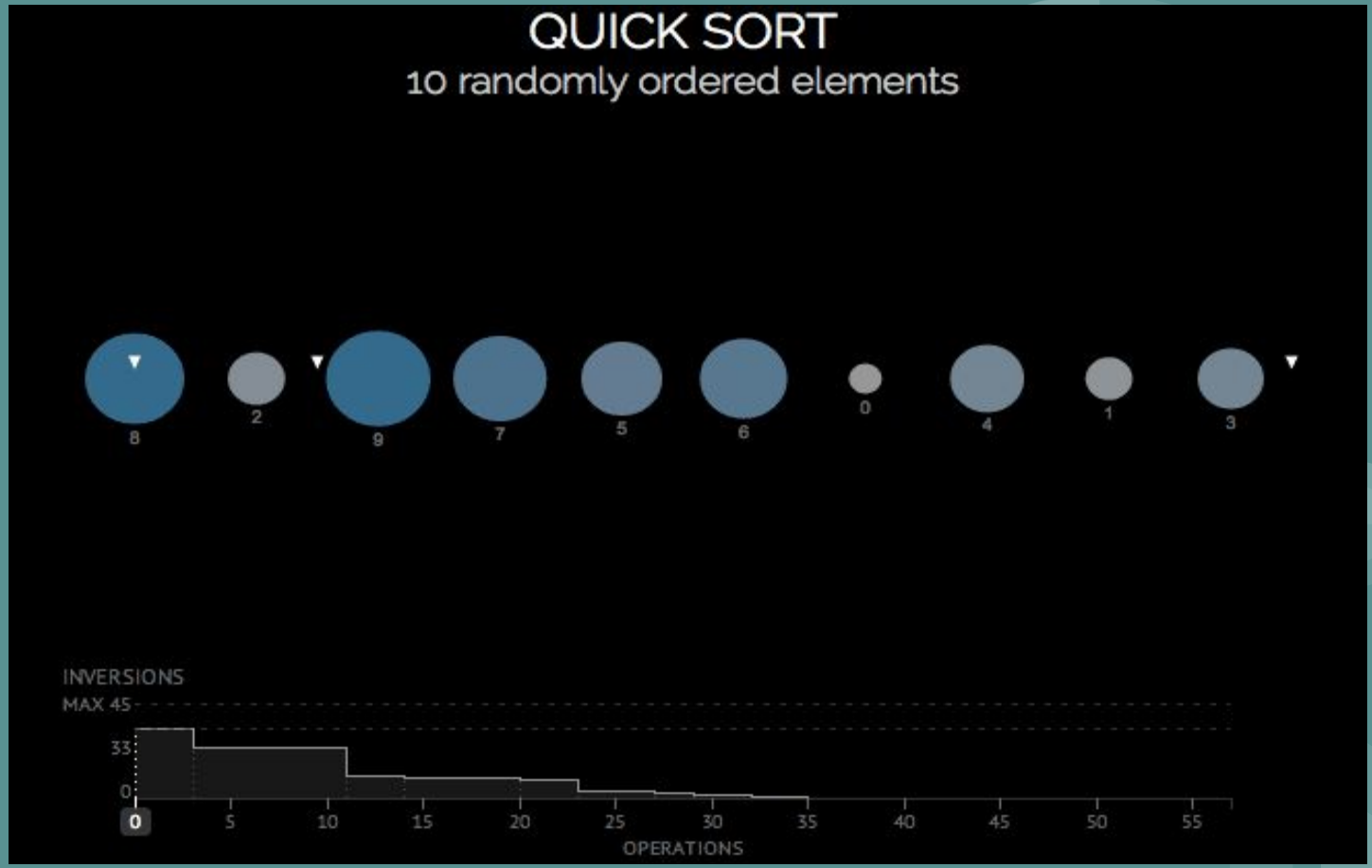


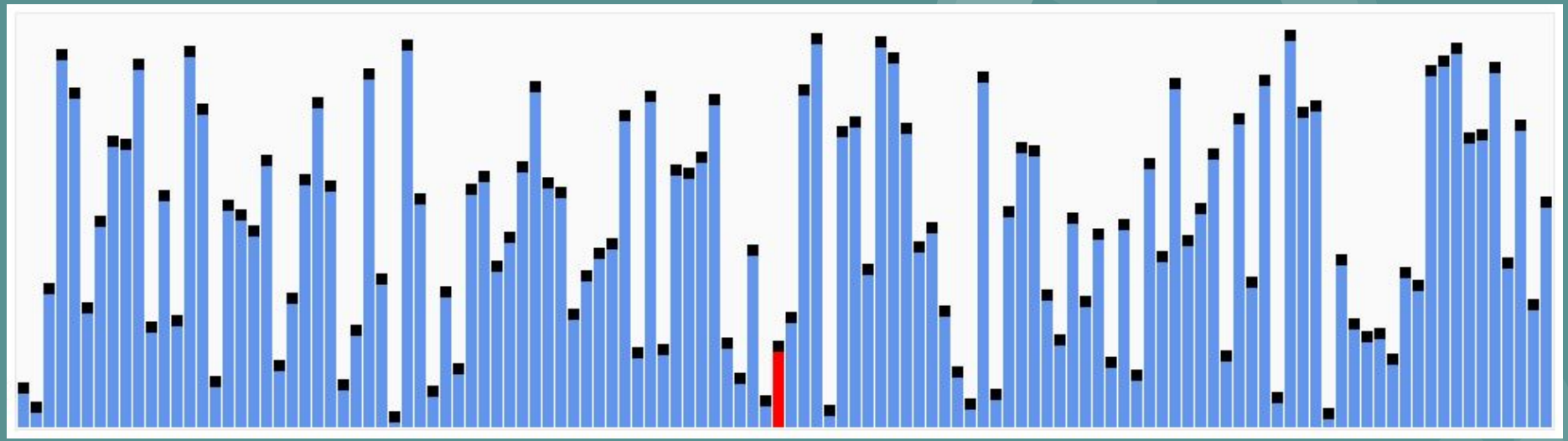
Step 7
Move pointers one more step



Step 8
Since 5 == 5,
move pointers one more step
Stop




QUICK SORT
10 randomly ordered elements
8
2
9
7
5
6
0
4
1
3
INVERSIONS
MAX 45
33
0
0
5
10
15
20
25
30
35
40
45
50
55
OPERATIONS



Código em Algoritmo

```
procedimento QuickSort(X[], IniVet, FimVet)
var
   i, j, pivo, aux
início
   i <- IniVet
   j <- FimVet
   pivo <- X[(IniVet + FimVet) div 2]
      enquanto(i < j)
            enquanto (X[i] <= pivo) faça
                i <- i + 1
            fimEnquanto
            enquanto (X[j] > pivo) faça
                j <- j - 1
            fimEnquanto
            se (i < j) então
                  aux  <- X[i]
                  X[i] <- X[j]
                  X[j] <- aux
            fimSe
            i <- i + 1
            j <- j - 1
      fimEnquanto
            se ( j > IniVet) então
                  QuickSort(X, IniVet, j)
            fimSe
            se ( i < FimVet) então
                  QuickSort(X, j+1, FimVet)
            fimse
fimprocedimento
```

Código em C++

Análise do algoritmo

- Melhor caso
  $C(n) \approx n \log n = O(n \log n)$
  - quando cada partição divide o arquivo em partes iguais.

- Caso médio*
  $C(n) \approx 1,39\, n \log n = O(n \log n)$

*Sedgewick,Flajolet (1996, p. 17)

Análise do algoritmo

- Pior caso:
  $C(n) = \frac{n(n-1)}{2} = O(n^2)$

  - procedimento é chamado n vezes, eliminando um item por chamada

  - Para evitar o pior caso, são escolhidos três elementos quaisquer, e o pivô é a média dos três.

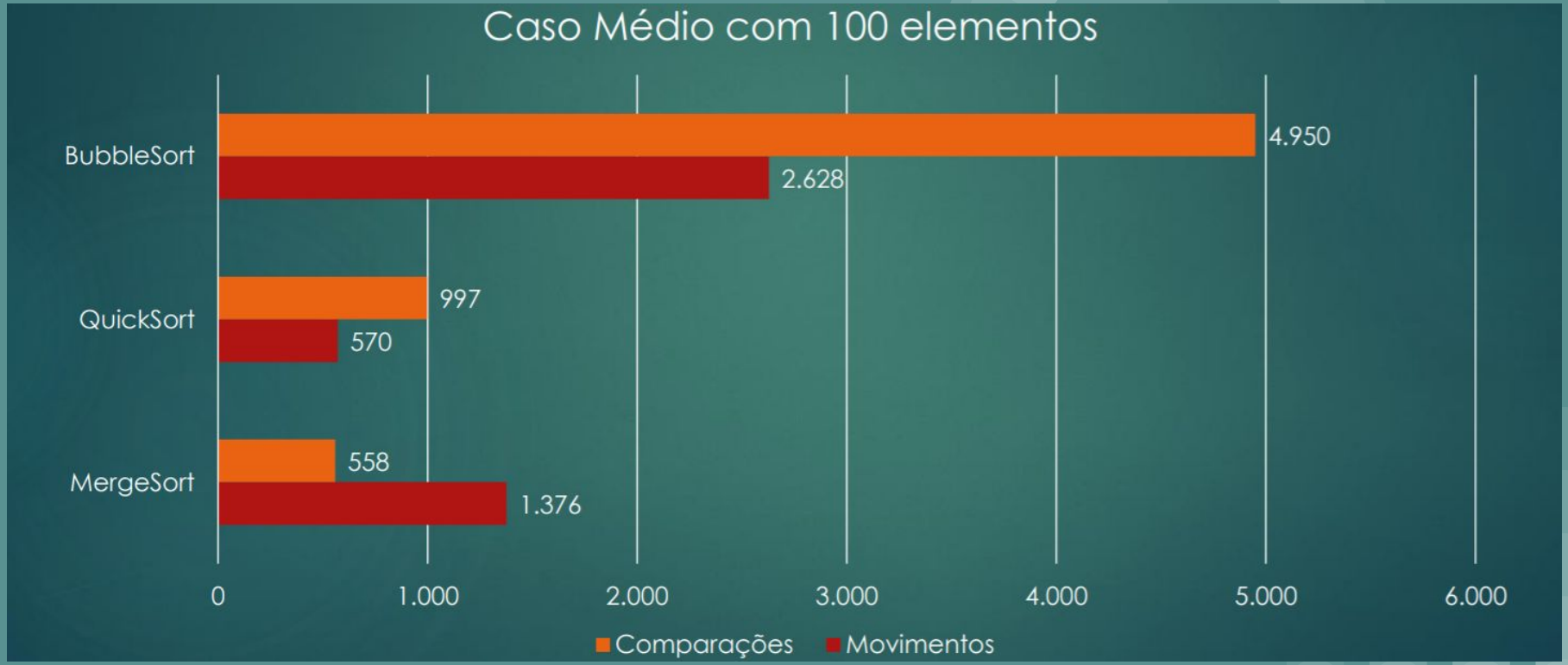
Porque utilizar?

- É extremamente eficiente para ordenar arquivos de dados.
- Necessita de apenas uma pequena pilha como memória auxiliar.
- Requer cerca de n log n comparações em média para ordenar n itens.

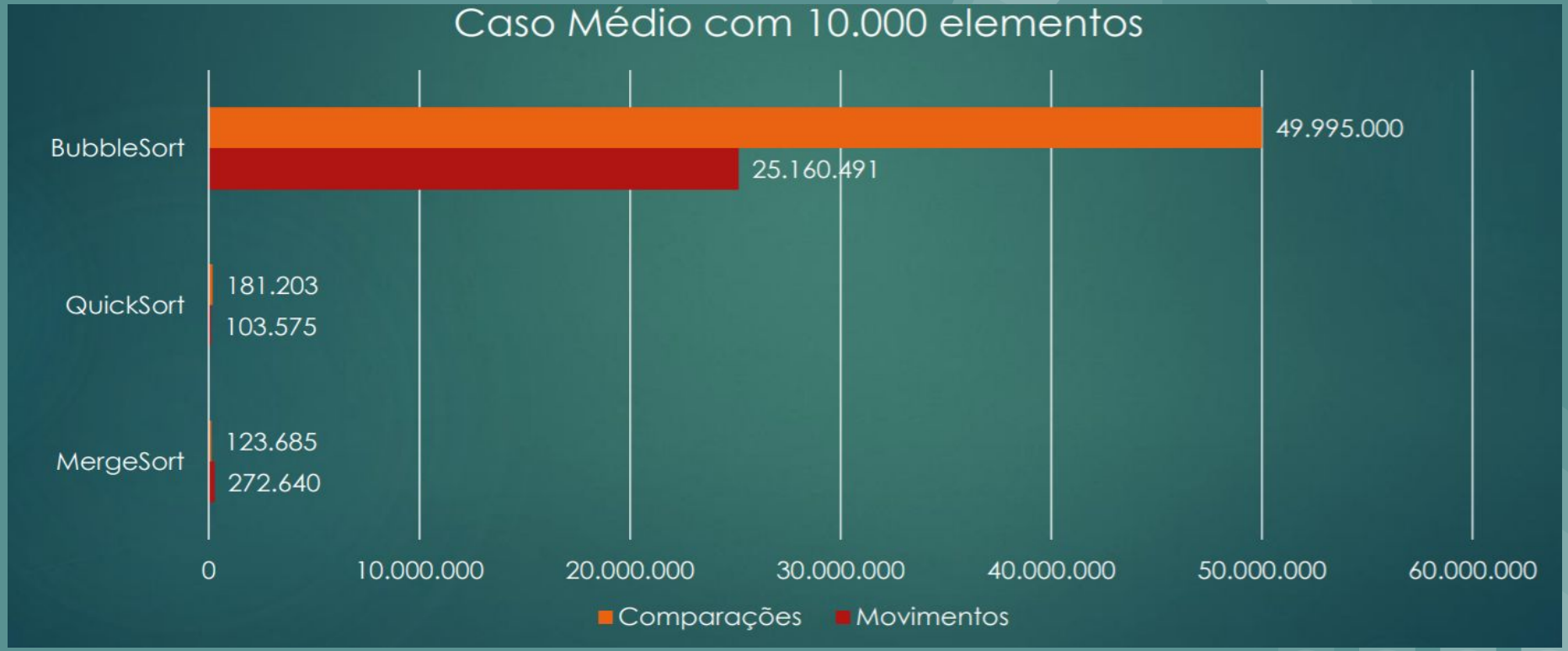
Porque não utilizar?

- Tem um pior caso $O(n^2)$ comparações.
- Um pequeno engano pode levar a efeitos inesperados para algumas entradas de dados.
- Não é estável

Comparações e movimentos



Comparações e movimentos



* Utilizado para testes um Processador Pentium D 2.8 Ghz , 1GB RAM, C++