Laboratório 06 – Quicksort e seu pivô

Mateus Resende Ottoni

Link para o código desenvolvido: AEDS2/Laboratório/LAB06/Quicksort.java at main · Mateus-Resende-Ottoni/AEDS2 (github.com)

Resultado:

```
------------------------

Array organizado: Aleatório _ 100 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

------------------------

------------------------

Array organizado: Aleatório _ 1000 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

------------------------

------------------------

Array organizado: Aleatório _ 10000 elementos
Tempos para ordenar:
FirstPivot: 0,0070s.
LastPivot: 0,0000s.
RandomPivot: 0,0020s.
MedianOfThreePivot: 0,0000s.

------------------------

------------------------

Array organizado: Aleatório _ 17500 elementos
Tempos para ordenar:
FirstPivot: 0,0050s.
LastPivot: 0,0010s.
RandomPivot: 0,0050s.
MedianOfThreePivot: 0,0060s.
```

Array organizado: Organizado _ 100 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Organizado _ 1000 elementos
Tempos para ordenar:
FirstPivot: 0,0040s.
LastPivot: 0,0010s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Organizado _ 10000 elementos
Tempos para ordenar:
FirstPivot: 0,0350s.
LastPivot: 0,0510s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Organizado _ 17500 elementos
Tempos para ordenar:
FirstPivot: 0,0850s.
LastPivot: 0,1250s.
RandomPivot: 0,0050s.
MedianOfThreePivot: 0,0000s.

Array organizado: Semi-organizado _ 100 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.

LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

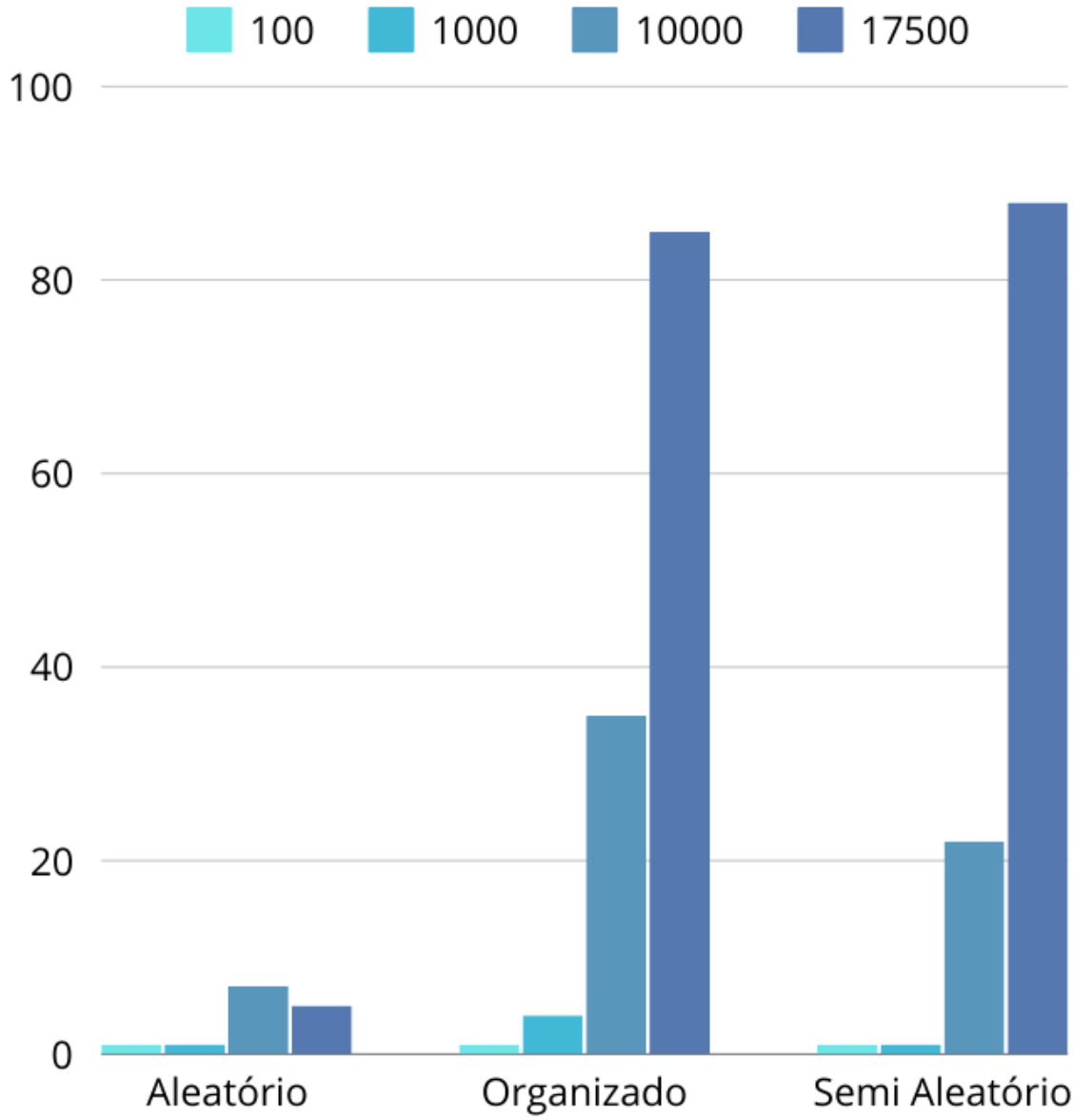
Array organizado: Semi-organizado _ 1000 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Semi-organizado _ 10000 elementos
Tempos para ordenar:
FirstPivot: 0,0220s.
LastPivot: 0,0500s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

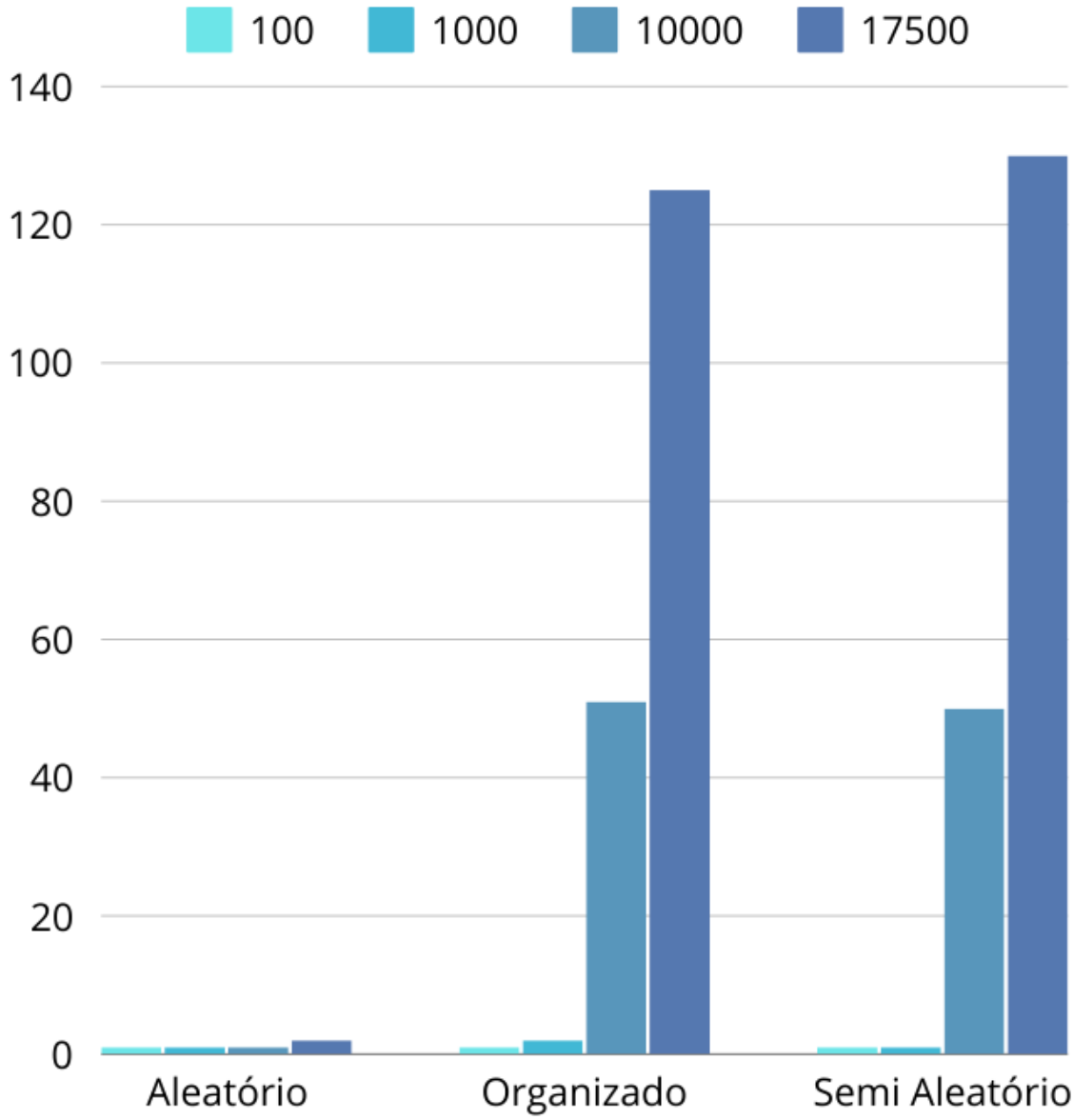
Array organizado: Semi-organizado _ 17500 elementos
Tempos para ordenar:
FirstPivot: 0,0880s.
LastPivot: 0,1300s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Sumário em gráficos (medido em milissegundos, por cada tamanho do conjunto):

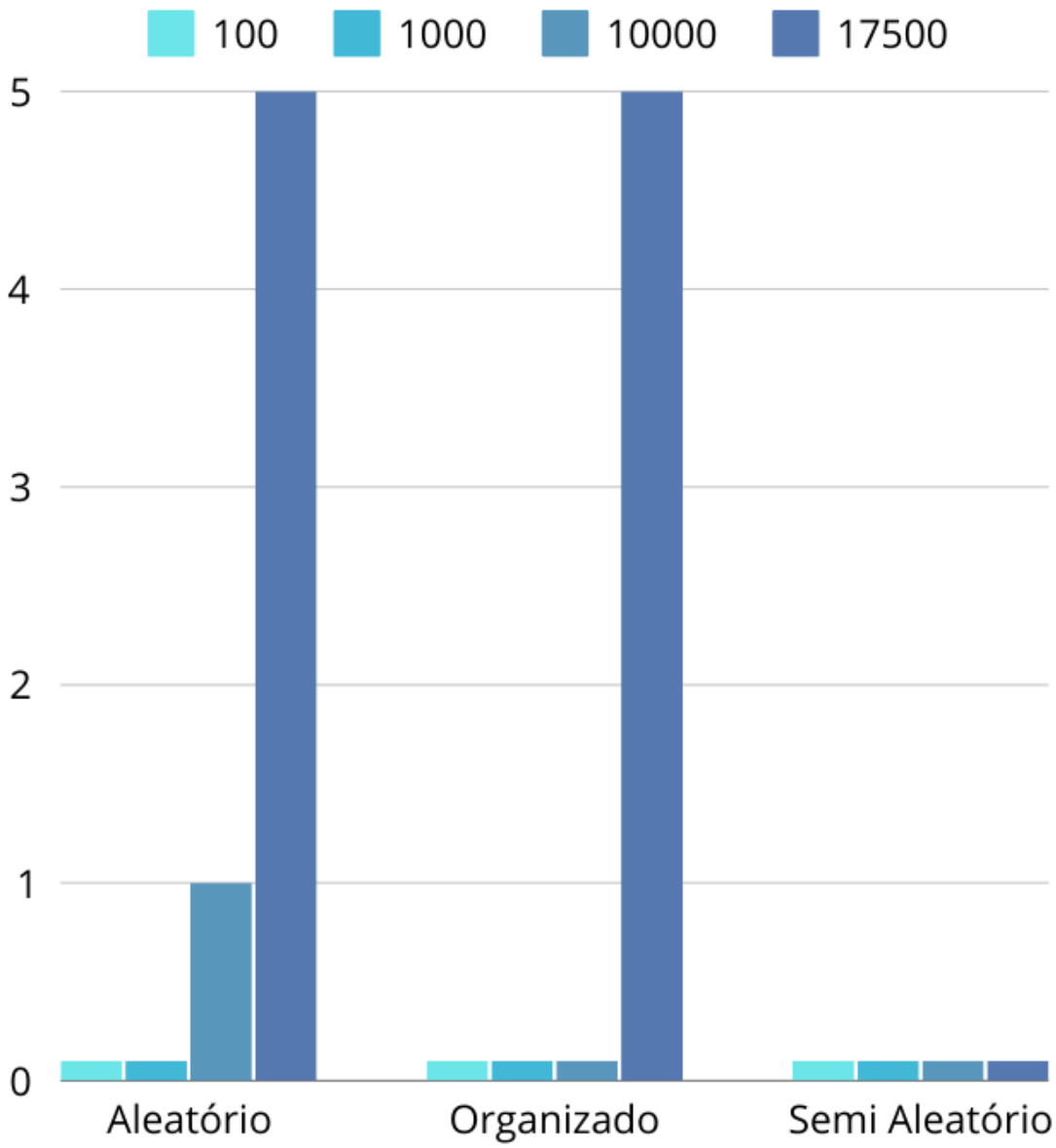
FirstPivot: Esse método utiliza do primeiro elemento do conjunto/subconjunto como sendo o pivô para a organização



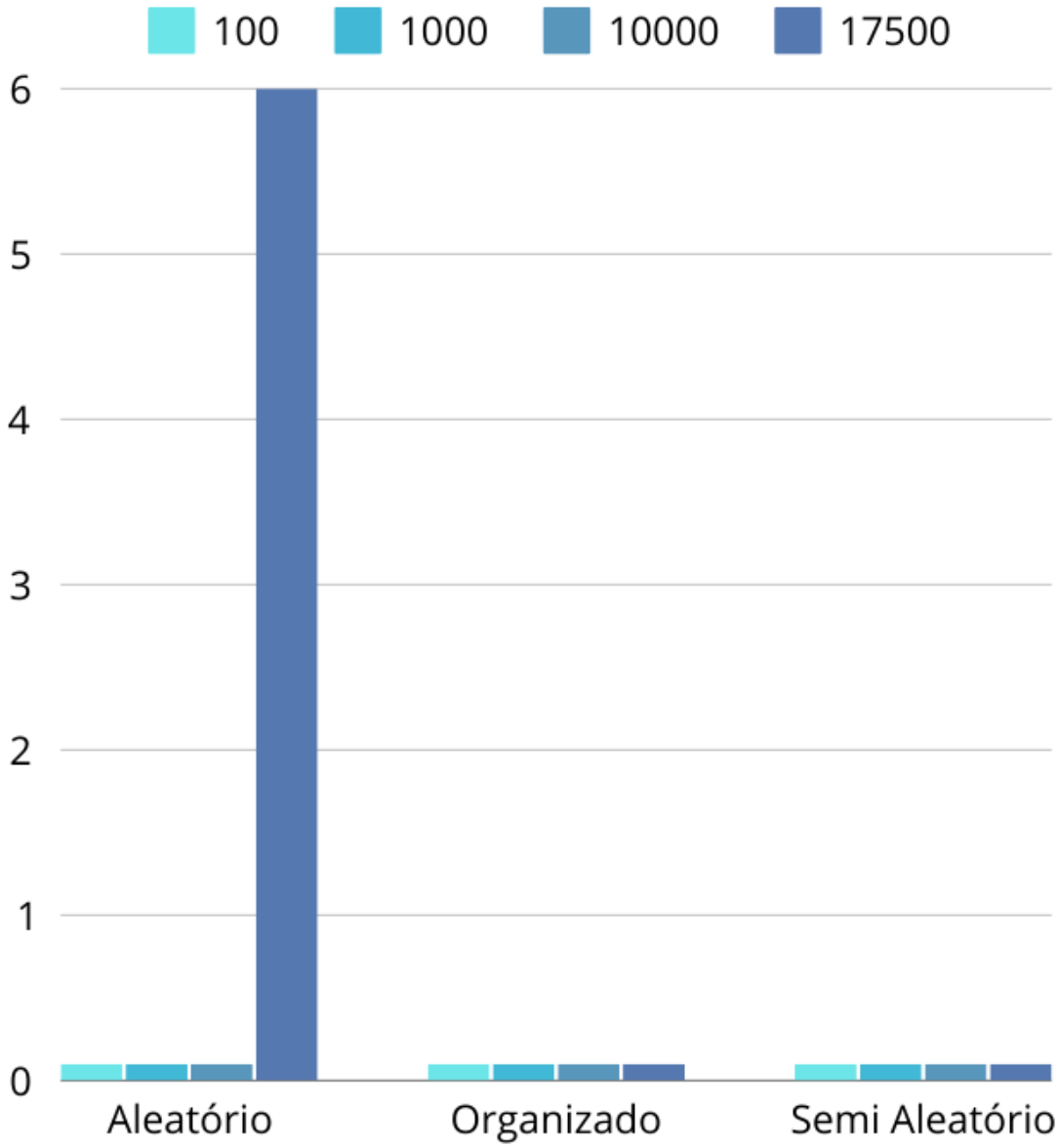
LastPivot: Esse método utiliza do último elemento do conjunto/subconjunto como sendo o pivô para a organização



RandomPivot: Esse método utiliza de um elemento aleatório do conjunto/subconjunto como sendo o pivô para a organização



MedianOfThreePivot: Esse método utiliza do valor mediano entre o valor do início, o valor do meio, e o valor do fim do conjunto/subconjunto como sendo o pivô para a organização



Análise para cada caso:

Conjunto aleatório:

Notavelmente nesse caso, a diferença da execução de cada método de seleção de pivô é mínima, com o método *LastPivot* tendo tido o melhor resultado, porém, isso pode ser atribuído puramente a uma questão de "sorte", já que a organização aleatória evita o resultado visto nos demais casos do método em que o pior caso ocorre consistentemente (o que igualmente se aplica ao *FirstPivot*). Atribuindo a lógica dos métodos, o que demonstraria o resultado mais consistente seria *MedianOfThreePivot*

já que, pela sua natureza, ele evita a ocorrência do pior caso, apesar que o custo de achar a o elemento mediano entre o primeiro, o último e o do meio deve ser considerado.

Conjunto organizado:

Nesse caso, os que demonstraram melhor desempenho foram *RandomPivot* e *MedianOfThreePivot*, com esse último tendo o melhor caso possível. Para o *RandomPivot*, sua escolha aleatória do pivô torna improvável a seleção dos piores casos (como ocorre com o *FirstPivot* e *LastPivot*, que em um conjunto organizado sempre farão a pior escolha), o tornando eficiente, já o *MedianOfThreePivot* sempre selecionará o elemento do meio do conjunto/subconjunto, que será o elemento mediano (já que está organizado), o que garante a ocorrência do melhor caso, com o menor número possível de trocas.

Conjunto semi-organizado:

Para esse caso, a lógica é quase idêntica à do conjunto organizado, com o *MedianOfThreePivot* sendo o método de maior eficácia pela maior probabilidade de encontrar o elemento mediano do conjunto/subconjunto.