Algoritmos de Ordenação - Parte 2

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Algoritmos de Ordenação

- Estudamos anteriormente alguns algoritmos de ordenação interna
- Hoje vamos ver alguns dos mais rápidos e mais usados historicamente
  - Porém relativamente complexos!

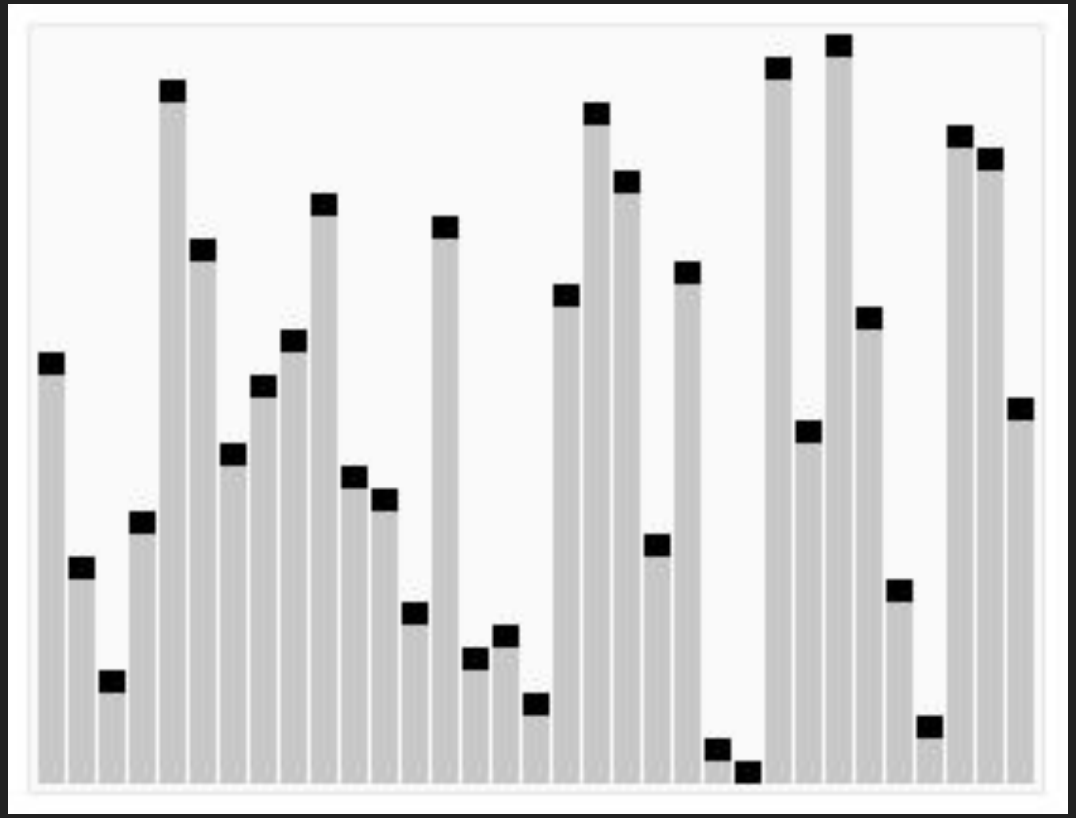
Quicksort

Quicksort

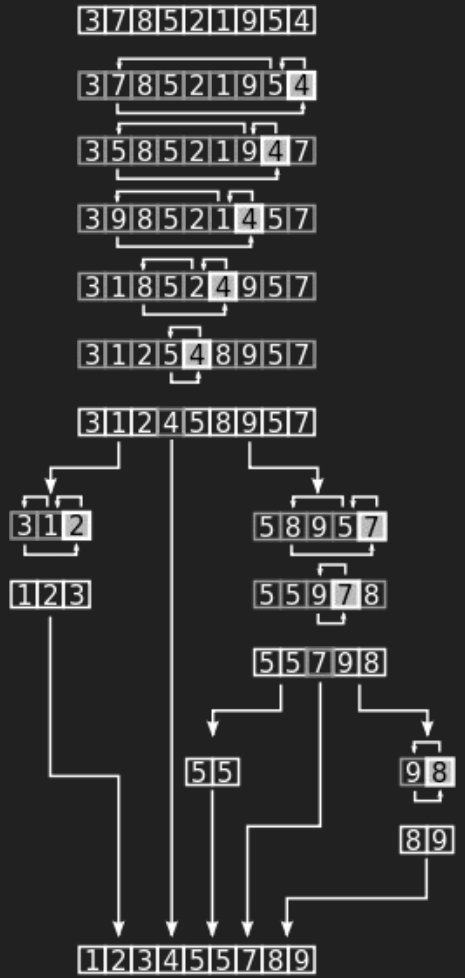
- É baseado na ordenação por divisão e conquista
  - Assim como o mergesort!
- Mas a divisão do vetor em 2 partes é o grande truque!

Quicksort

- Ao invés de dividir na metade, escolhemos um pivô!
- Realizamos algumas trocas iniciais para garantir que ele os números à esquerda sejam menores e os à direita sejam maiores
- E aí repetimos recursivamente o mesmo para cada parte!
  - Sempre ordenando depois de chamar o particionamento



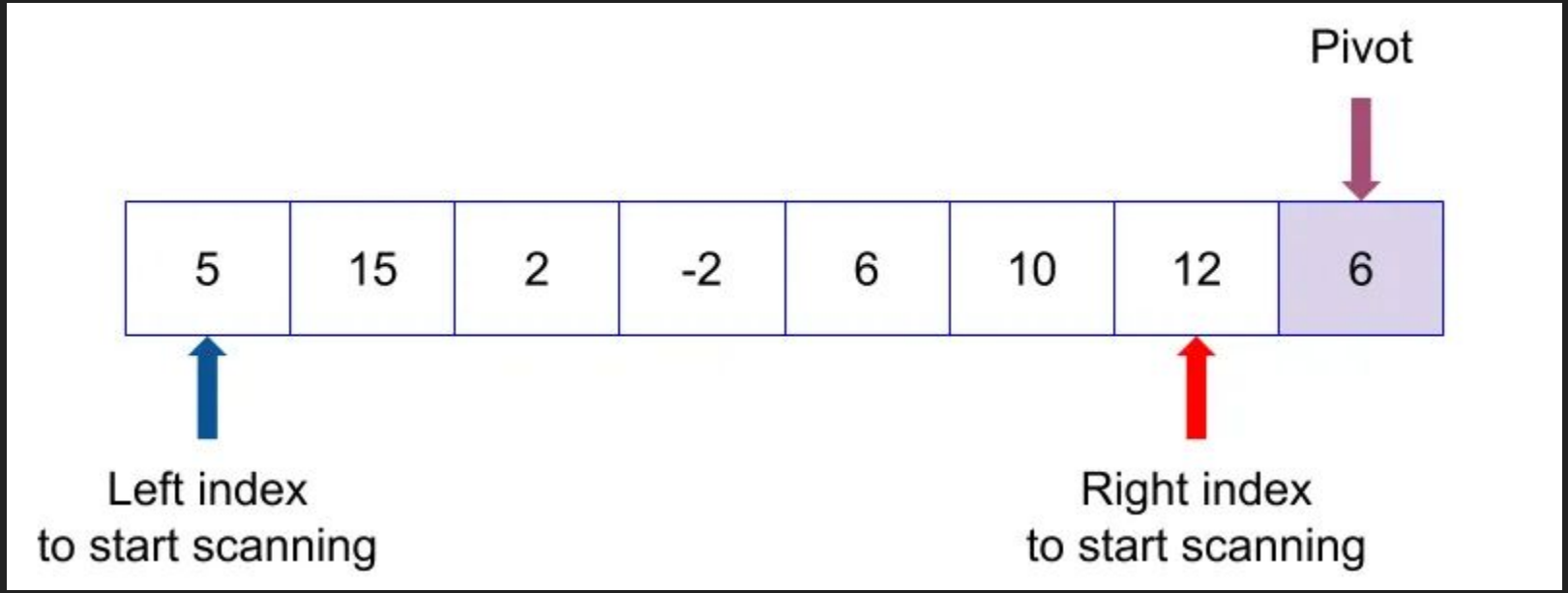
https://upload.wikimedia.org/wikipedia/commons/6/6a/Sorting_quicksort_anim.gif



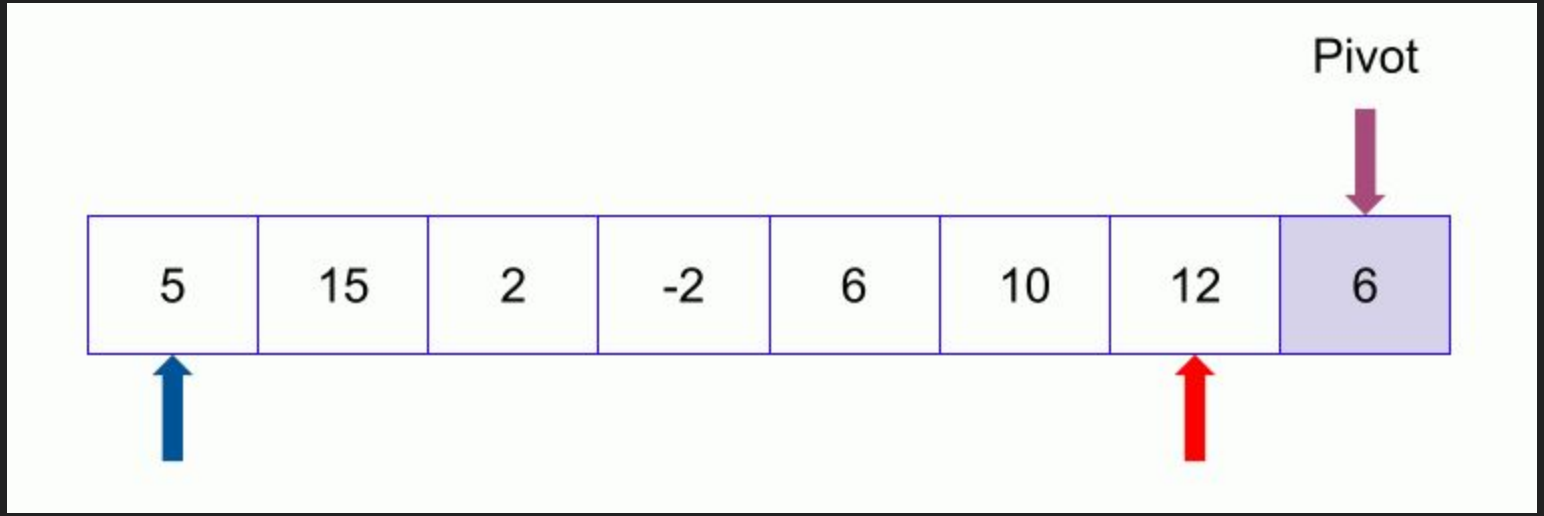
Exemplo de Quicksort usando o pivô como o último elemento.
Gera resultado O(n²) caso o vetor já esteja ordenado

Fonte:
https://en.wikipedia.org/wiki/File:Quicksort-diagram.svg

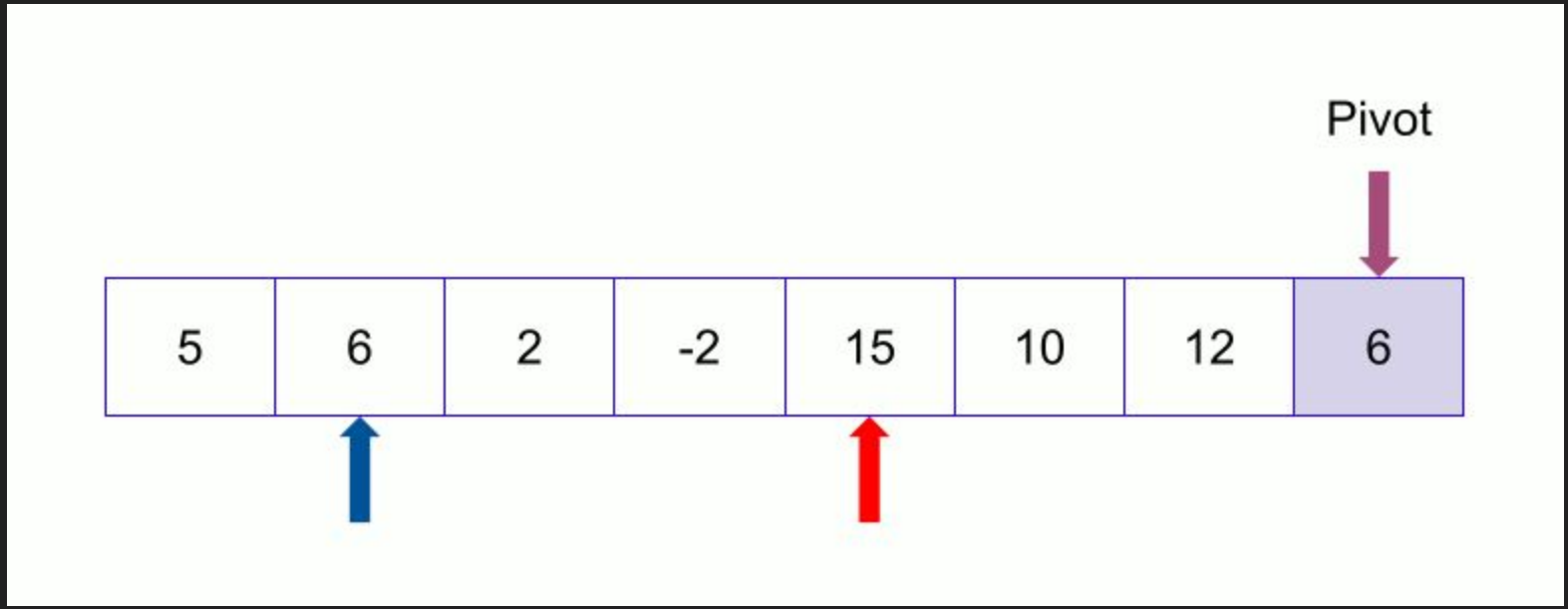
Passo a passo - 1 - Divisão



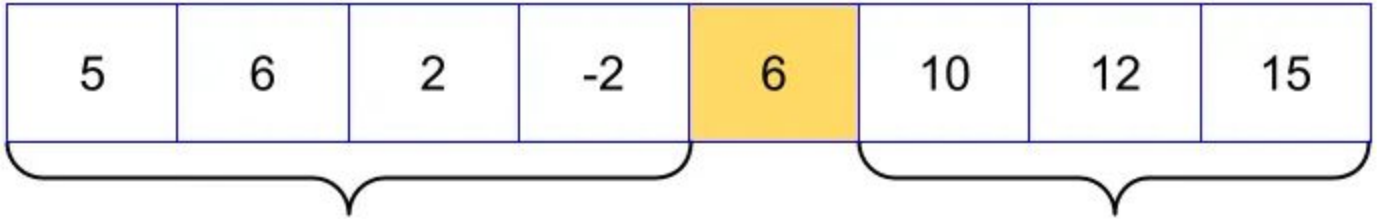
Fonte: https://blog.shahadmahmud.com/quicksort/



Fonte: https://blog.shahadmahmud.com/quicksort/



Fonte: https://blog.shahadmahmud.com/quicksort/



Fonte: https://blog.shahadmahmud.com/quicksort/

Passo a passo - 2 - Conquista

Fonte: https://blog.shahadmahmud.com/quicksort/

Quicksort

➔ O código!

Quicksort

- O quicksort é um dos algoritmos mais usados para ordenação por ter um tempo médio de O(n log n)
- Porém, é válido lembrar que o pior caso é $O(n^2)$
  - E isso depende principalmente do pivô e do vetor a ser ordenado
  - No caso do pivô no final (ou começo), por exemplo, o pior caso é do vetor ordenado!
    - Ou com todos os elementos idênticos

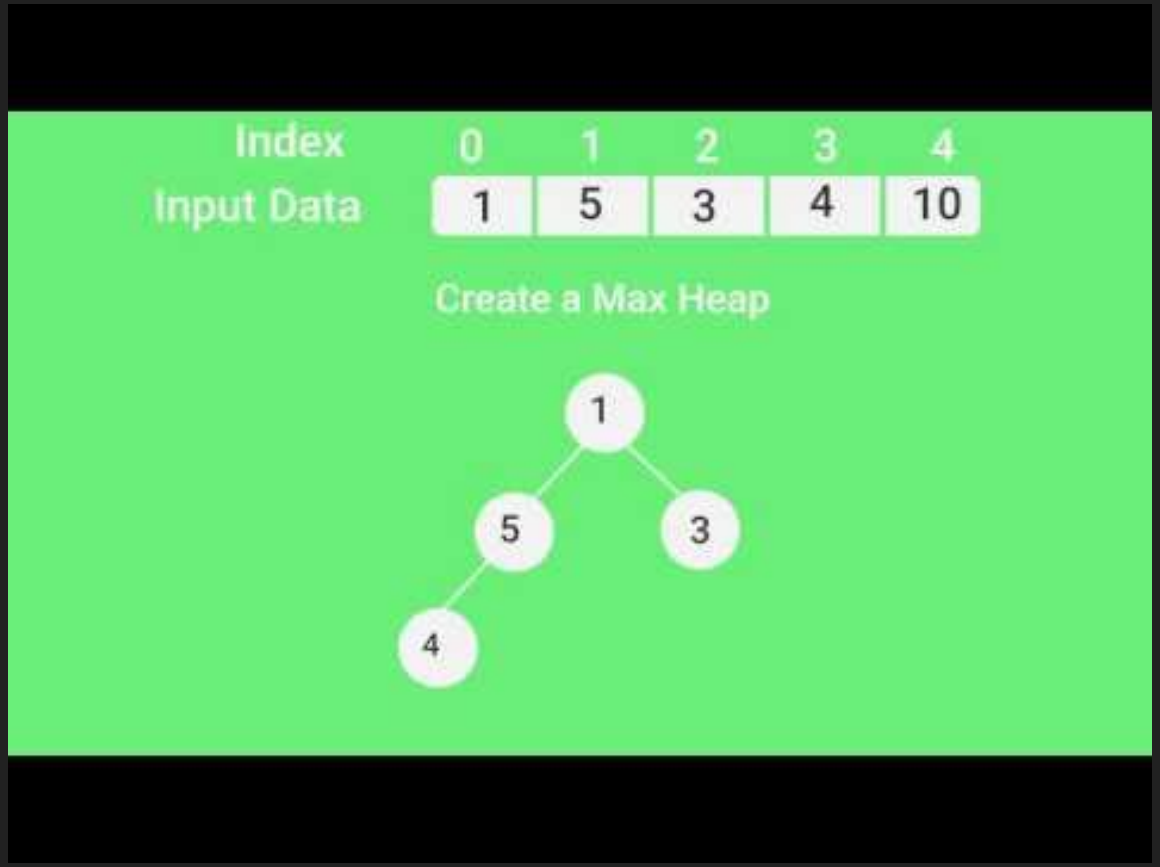
Quicksort

- Existem algumas soluções relativamente simples para boas escolhas de pivô
  - Escolher um pivô aleatório
  - Escolher o pivô do meio
  - Escolher a mediana entre o primeiro, meio e último elementos
    - Esta funciona ainda melhor para partições grandes

Heapsort

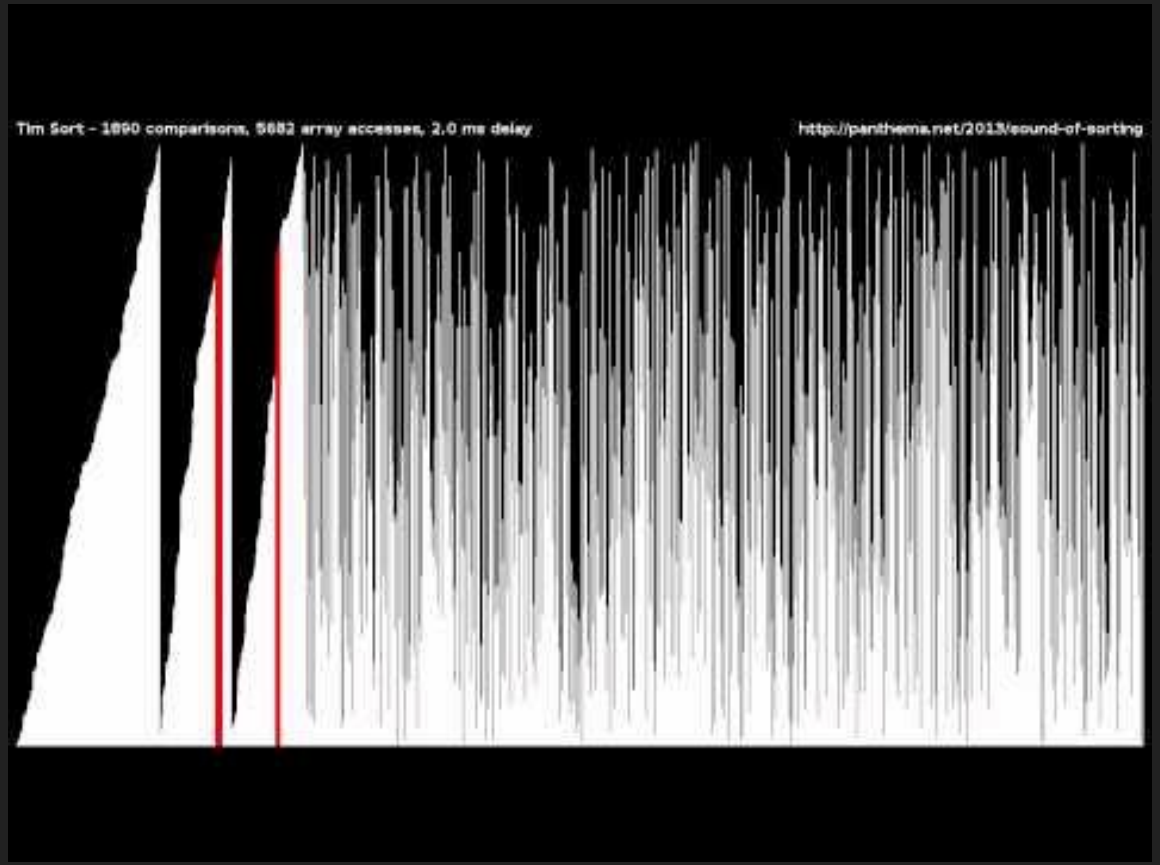
Heapsort

- Desempenho é sempre O(n log n)
- Não é estável
- Sua premissa é ordenar por árvore binária, mas usando apenas um vetor que *representa* a árvore



Fonte: https://www.geeksforgeeks.org/heap-sort/

Timsort



Fonte: https://www.youtube.com/watch?v=CEiWS0cDYgE

Timsort

- ➔ É o algoritmo padrão de ordenação de Python, criado por Tim Peters em 2002
- ➔ Também é o algoritmo para ordenação de vetores não primitivos em Java SE7 do Android, GNU Octave no V8, Swift e Rust
- ➔ É um algoritmo híbrido!

Timsort

- ➔ O Timsort é baseado na teoria de que boa parte dos vetores de dados do mundo real já tem sub-vetores ordenados
- ➔ Assim sendo, ele divide o vetor em sub-vetores de um tamanho pré-determinado, ordena estes sub-vetores com o Insertionsort e, após a ordenação dos sub-vetores, eles são combinados via Mergesort em pares, aumentando o tamanho dos pares até o vetor inteiro ser combinado

Referências

Referências

- ZIVIANI, N. Projeto de Algoritmos. 2º edição, Thomson, 2004.
- https://en.wikipedia.org/wiki/Quicksort
- https://blog.shahadmahmud.com/quicksort/
- https://en.wikipedia.org/wiki/Heapsort
- https://ide.geeksforgeeks.org/rFO7Lm
- https://en.wikipedia.org/wiki/Timsort
- https://www.geeksforgeeks.org/timsort/