

Classificação de dados com base em chaves de classificação

Marcus Pereira

Nathan Ferreira

Alysson Machado

Iago Miguel

Período 2020.0

Marcus Salerno

Técnicas de Programação

Objetivos

- ❖ Implementação dos algoritmos de classificação **SelectionSort** e **MergeSort**;
- ❖ Determinar a **eficiência dos algoritmos** de classificação;
- ❖ Aplicar os métodos de ordenação em um exemplo com lista de Vector;

Ordem de Apresentação

- ❖ Alysson Machado (Introdução);
- ❖ Nathan Ferreira (Ordenação por Seleção);
- ❖ Iago Miguel(Ordenação por Intercalação);
- ❖ Marcus Pereira (Comparação da Eficiência dos algoritmos);
- ❖ Alysson Machado (Demonstração da Eficiência);
- ❖ Alysson Machado (Aplicação usando Lista de Vector);

Acesse os Materiais

- ❖ Acesse pelo site:

<https://alyssonmach.github.io/chaves-de-classificacao/>

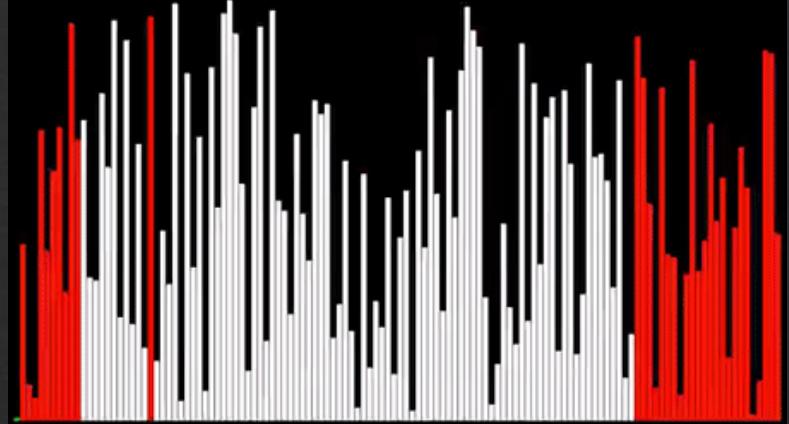


Introdução ao tema...



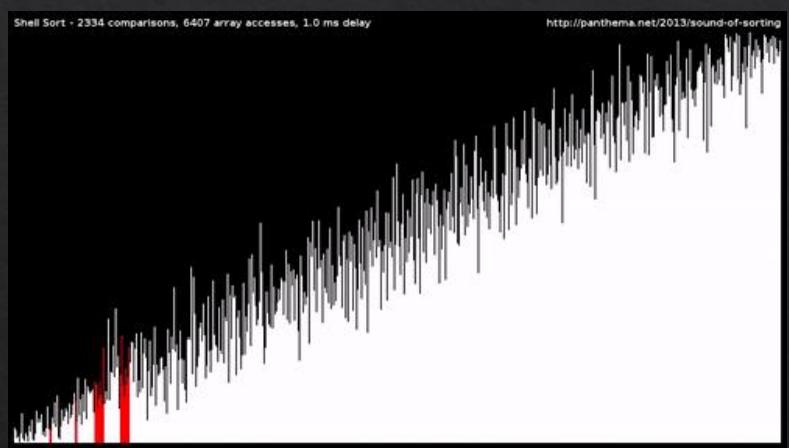
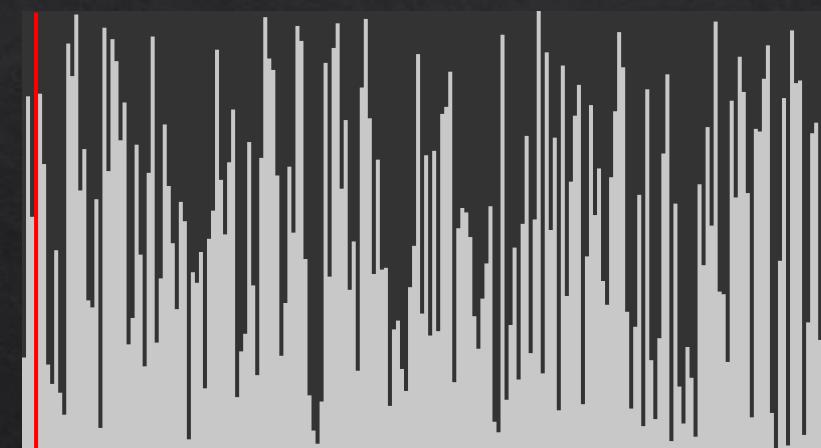
Selection Sort - 134 comparisons, 265 array accesses, 0.50 ms delay

<http://panthema.net/2013/sound-of-sorting>



Shell Sort - 2334 comparisons, 6407 array accesses, 1.0 ms delay

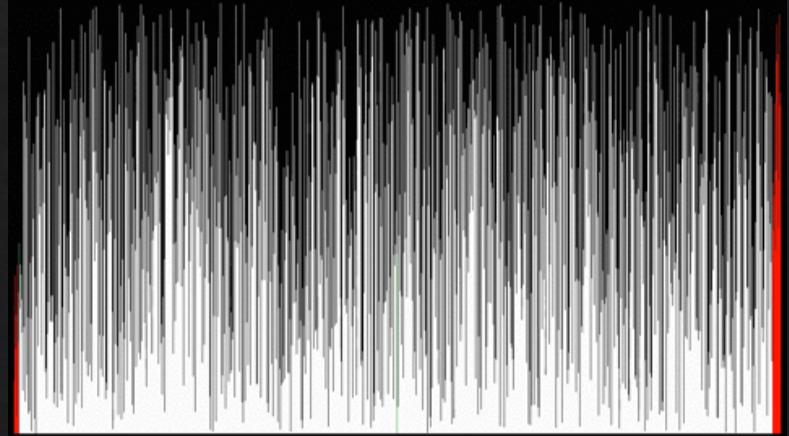
<http://panthema.net/2013/sound-of-sorting>



MAKE GIFS AT GFSOUP.COM

Quick Sort (LR ptrs) - 22 comparisons, 34 array accesses, 1.00 ms delay

<http://panthema.net/2013/sound-of-sorting>



SelectionSort

Métodos simples de ordenação

[por seleção]

Selection sort (por seleção)

- ❖ Primeiro ele encontra o menor elemento e admite-o como primeiro;
- ❖ Compara um elemento (n) com o primeiro elemento, sempre;
- ❖ Quando ocorrer de o primeiro elemento estiver desordenado em relação ao elemento N, há a troca.



Selection sort (por seleção)

Vantagens	Desvantagens
Método simples	Existem métodos mais eficazes em certos casos
Aconselhado para grandes listas	-
Terá sempre o mesmo custo, independente de ordenação	Isso implica que há possibilidade de desperdício de processamento

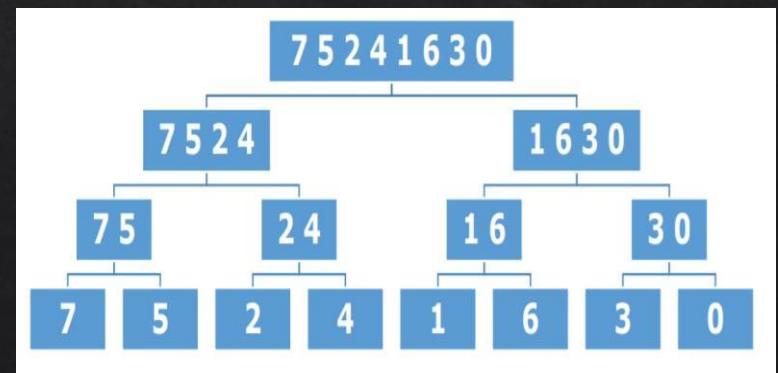
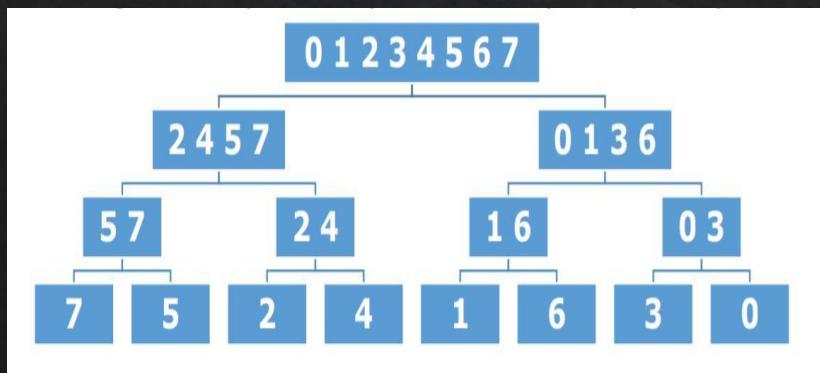
MergeSort

Métodos simples de ordenação

[por intercalação]

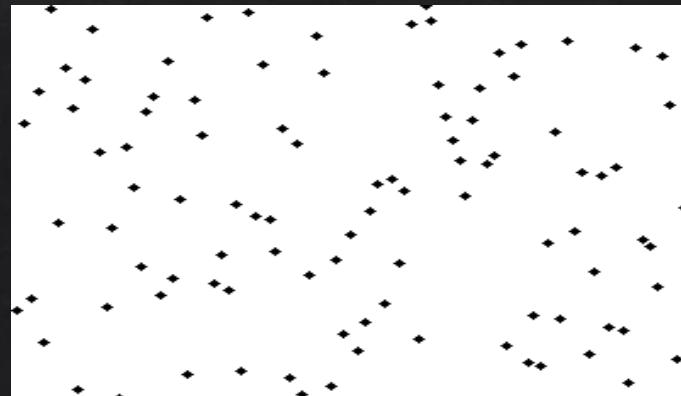
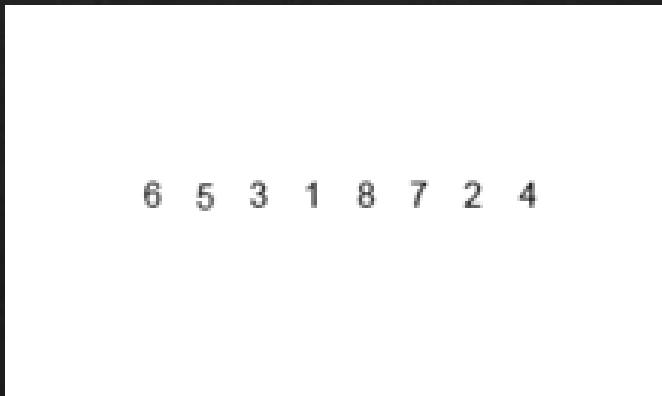
Merge Sort (por intercalação)

- ◊ Utiliza a estratégia “dividir para conquistar” para resolver problemas;
- ◊ Divide o vetor em pedaços menores, resolve cada pedaço e depois junta (merge) os resultados. O vetor será dividido em duas partes iguais, que serão cada uma divididas em duas partes, e assim por diante até ficar um ou dois elementos cuja ordenação é trivial;
- ◊ Para juntar as partes novamente, porém, na ordem correta, os dois elementos de cada parte são separados e depois de comparados o menor é retirado de sua parte. Logo em seguida os menores entre os restantes são, também, separados e assim sucessivamente até juntar todos.



Merge Sort (por intercalação)

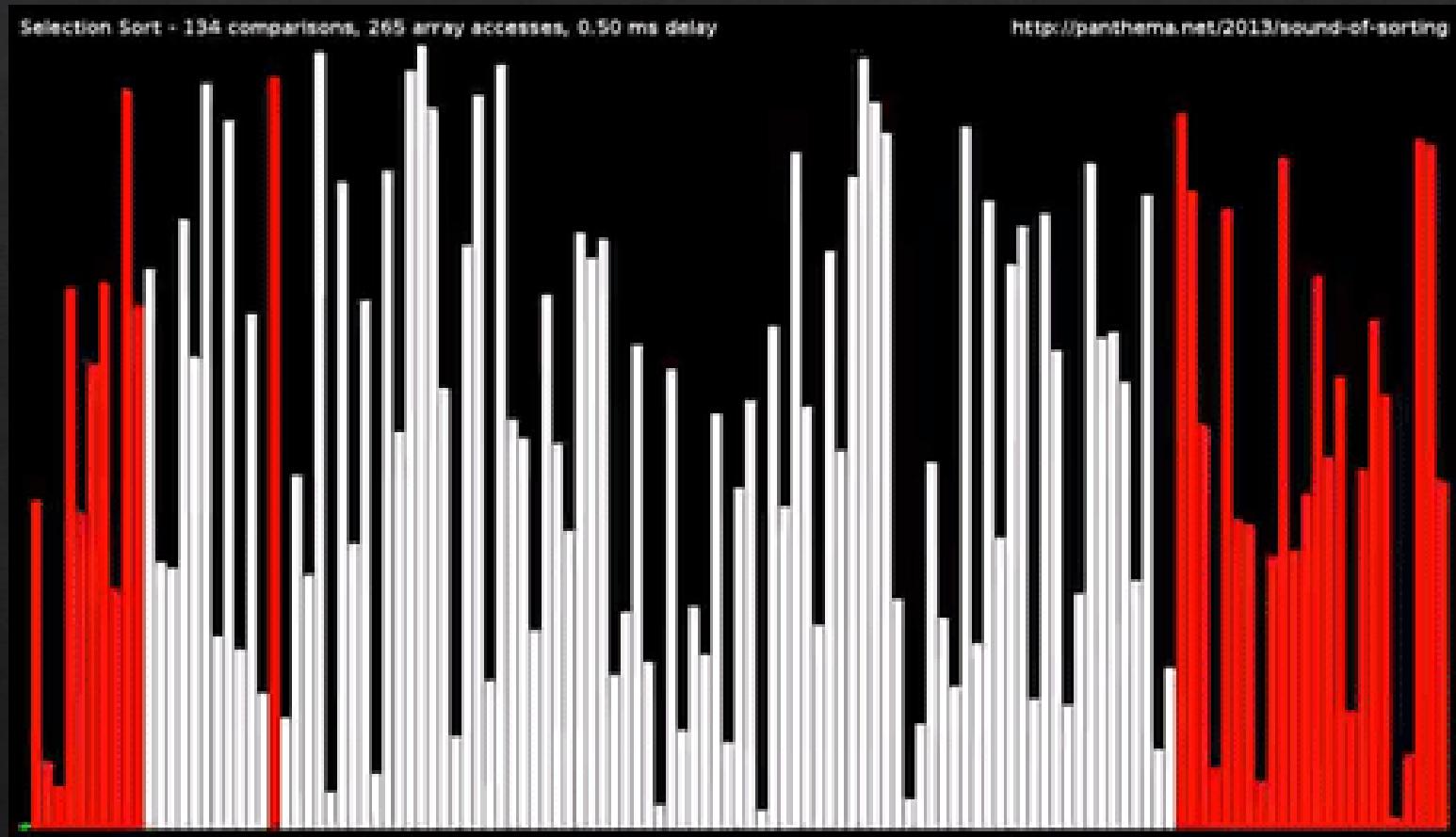
Vantagens	Desvantagens
Método de ordenação com tempo	Utiliza mais memória para poder ordenar (vetor auxiliar)
Pode ser adaptado para ordenação de arquivos externos (memória secundária)	O MergeSort é estável: não altera a ordem de dados iguais



Eficiência do SelectionSort

- ❖ O algoritmo de classificação por seleção itera $n - 1$ vezes, colocando a cada passagem o menor elemento restante em sua posição classificada.
- ❖ Localizar o menor elemento restante requer $n - 1$ comparações durante a primeira iteração, $n - 2$ durante a segunda iteração e, então, $n - 3, \dots, 3, 2, 1$.
- ❖ Isso resulta em um total de $n(n - 1) / 2$ ou $(n^2 - n)/2$ comparações. Na notação O , os menores termos são eliminados e as constantes são ignoradas, deixando um O final de $O(n^2)$.

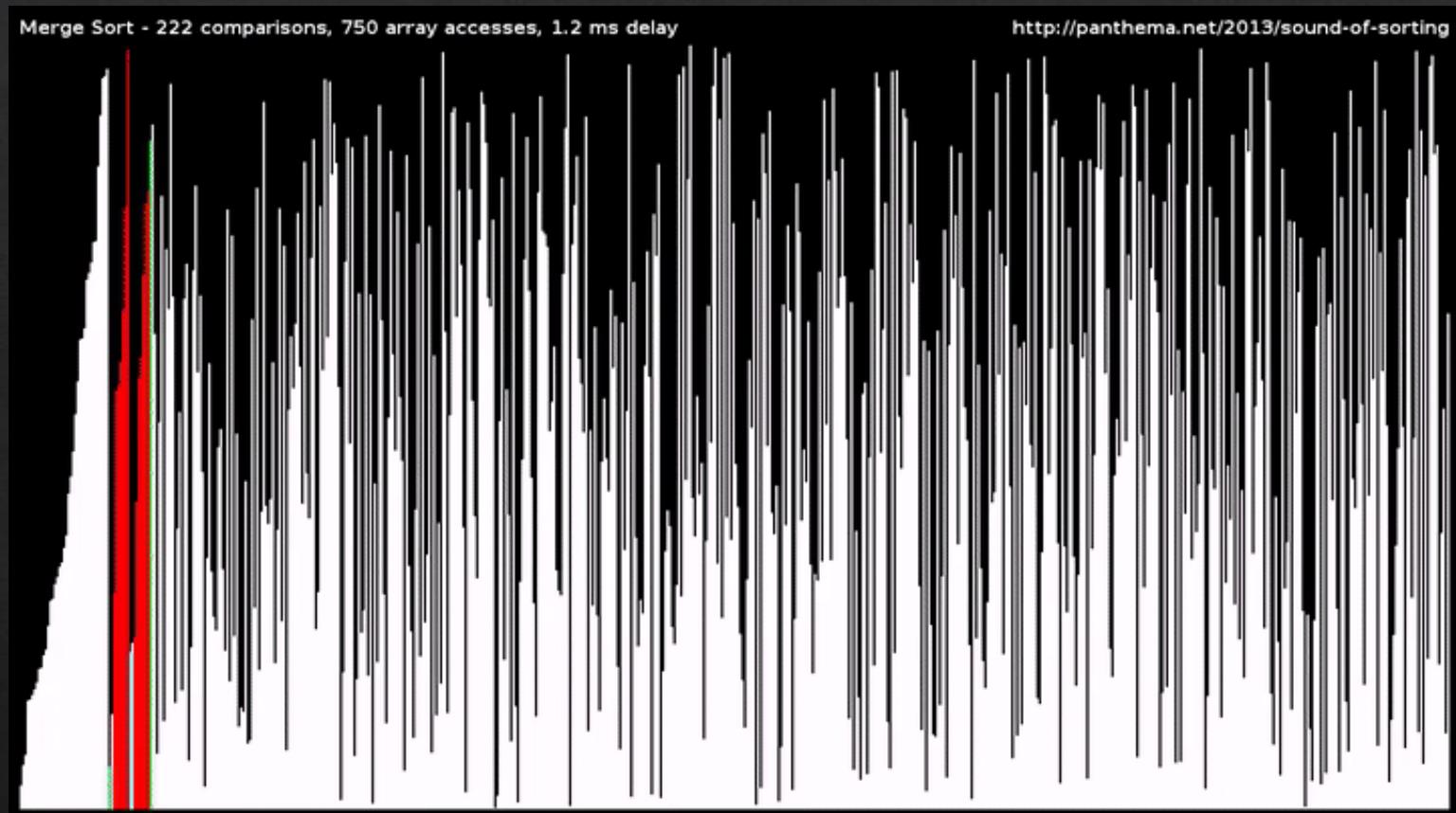
Eficiência do SelectionSort



Eficiência do MergeSort

- ❖ A classificação por intercalação é um algoritmo muito mais eficiente que a classificação por seleção;
- ❖ Considere a primeira chamada (não recursiva) à função sortSubVector. Isso resulta em duas chamadas recursivas à função sortSubVector em que cada subvetor apresenta aproximadamente a metade do tamanho do vetor original e uma única chamada à função merge;
- ❖ Cada nível divide o tamanho dos vetores pela metade, portanto dobrar o tamanho do vetor exige mais um nível. Quadruplicar o tamanho do vetor exige mais dois níveis. Esse padrão é logarítmico e resulta em $\log_2 n$ níveis. Isso resulta em uma eficiência total de $O(n \log n)$.

Eficiência do MergeSort



SelectionSort X MergeSort

- ❖ Numa comparação direta, o MergeSort se faz mais eficiente diante do SelectionSort, levando em consideração o padrão logarítmico do método de intercalação em relação ao padrão quadrático do método por seleção;
- ❖ Dessa forma, vemos que o Merge, diante de listas grandes, engloba comparações entre elementos de maneira mais eficiente e rápida e, diante disso, ele faz-se mais interessante no uso.
- ❖ Entretanto, cada caso deve ser analisado de forma singular, pois existem situações em que o SelectionSort seria melhor de utilizar, levando em consideração tamanho de lista, vetores, quantidade de gasto de memória, etc.

Aplicações e demonstração da eficiência

Obrigado!!!