**FACULDADE DE TECNOLOGIA DE GUARATINGUETÁ**

**Algoritmos de Ordenação**

**Gabriel Ramos Galvão**

Relatório Técnico-científico apresentado à Faculdade de Tecnologia de Guaratinguetá, como requisito parcial para conclusão do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

**Guaratinguetá - SP**

**2022**

RAMOS, Gabriel.  **RELATÓRIO TÉCNICO-CIENTÍFICO:** Algoritmos de Ordenação. Guaratinguetá, 2022. 10 páginas. Relatório Técnico-científico. Curso de Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Guaratinguetá, 2022.

RAMOS, Gabriel. **ALGORITMOS DE ORDENAÇÃO.** 2022.10 páginas. Relatório Técnico-científico, Curso de Análise e Desenvolvimento de Sistemas. Faculdade de Tecnologia de Guaratinguetá, 2022.

**Resumo**

Este documento visa apresentar, explicar e comparar diferentes métodos de organização com base em algoritmos.

**Abstract**

This document aims to present, explain and compare different methods of organization based on algorithms.

**Sumário**

1. **Contextualização...........................................................................1**
2. **Algoritmos.....................................................................................2**
   1. **Bubble Sort...........................................................................2**
   2. **Selection Sort.......................................................................3**
   3. **Insertion Sort........................................................................4**
   4. **Merge Sort............................................................................5**
   5. **Quick Sort.............................................................................6**
3. **Conclusão......................................................................................7**
4. **Referências....................................................................................8**

**1. Contextualização**

A primeira linguagem de programação nasceu no início dos anos 50, seu nome era FORTRAN. Olhando pelos olhos de hoje, o FORTRAN se parece extremamente rudimentar. Porém, ele foi uma grande inovação durante seu tempo.

Com o passar dos anos, as linguagens foram evoluindo: nasceram COBOL, C, Java e dezenas de outras. Os algoritmos de computador, que eram executados dentro das linguagens, também evoluíram muito com o passar do tempo: nasceram as linguagens orientadas a objeto, tal qual o já mencionado Java e o C++.

Esse relatório tem o propósito de explicar e comparar alguns dos algoritmos de organização que existem dentro do mundo da computação, e que são ensinados na matéria de Estrutura de Dados.

**2. Algoritmos**

**2.1 Bubble Sort**

Dentre todos os métodos de organização, a organização por bolhas se encontra sendo a menos eficiente de todas, principalmente para listas longas e de complexidade avançada.

Ele funciona examinando cada conjunto de elementos adjacentes, indo da esquerda para a direita. Para efetuar a organização e trocar a posição dos elementos, ele os verifica como valores booleanos no caso de estarem fora de ordem.

Sua desvantagem se dá ao fato de que ele guarda o valor de um elemento e percorre toda a extensão do vetor, o comparando com todos os outros elementos, o tornando assim o mais ineficiente de todos quando aplicado em listas longas.

*\* Organizou 50000 elementos em 4189 milissegundos;*

*\* Organizou 50000 elementos em 4208 milissegundos;*

*\* Organizou 50000 elementos em 3980 milissegundos.*

*\**

*\* Organizou 75000 elementos em 9522 milissegundos;*

*\* Organizou 75000 elementos em 9682 milissegundos;*

*\* Organizou 75000 elementos em 9212 milissegundos.*

*\**

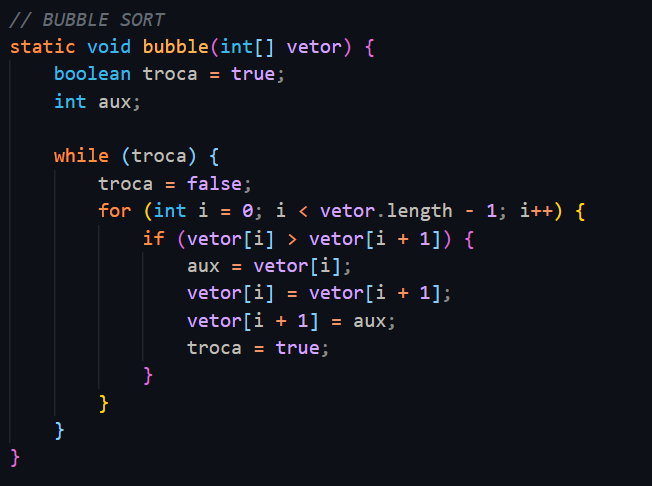
*\* Organizou 100000 elementos em 17412 milissegundos;*

*\* Organizou 100000 elementos em 17449 milissegundos;*

*\* Organizou 100000 elementos em 17267 milissegundos.*

Esse método não é ineficiente quando se trata de uma organização de complexidade O(n), mas ele se demonstra extremamente ineficaz e demorado em complexidades O(n²). Sua principal vantagem é seu fácil entendimento e implementação.

Recomendo apenas em listas pequenas.



**2.2 Selection Sort**

A organização por seleção funciona passando o menor valor do vetor para a primeira posição (isso depende da ordem requerida, pode ser que seja o maior valor primeiro), e depois o segundo menor para a segunda posição, e assim ele segue.

Ele se irá escolher um número a partir do primeiro e irá compará-lo com todos a sua direita. Assim que encontrar um número menor, ele o escolherá para comparar com os outros. Caso não encontre um número menor, ele o passará para a primeira posição e escolherá o primeiro número a sua direita para comparar com o resto da lista. Esse processo irá se repetir até que a lista esteja ordenada.

*\* Organizou 50000 elementos em 539 milissegundos;*

*\* Organizou 50000 elementos em 590 milissegundos;*

*\* Organizou 50000 elementos em 539 milissegundos.*

*\* Organizou 75000 elementos em 1208 milissegundos;*

*\* Organizou 75000 elementos em 1218 milissegundos;*

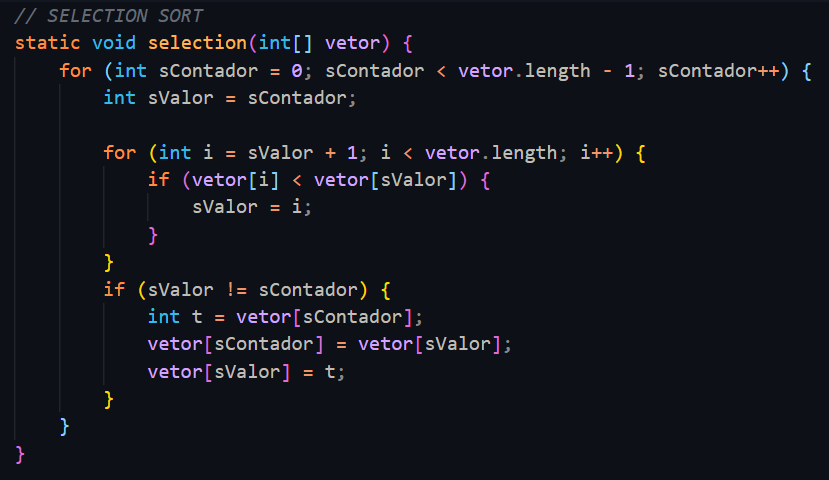
*\* Organizou 75000 elementos em 1201 milissegundos;*

*\* Organizou 100000 elementos em 2141 milissegundos;*

*\* Organizou 100000 elementos em 2117 milissegundos;*

*\* Organizou 100000 elementos em 2135 milissegundos;*

É um método focado em ordenar vetores mais complexos, por ter dois laços de repetição, não sendo o ideal para ordenar vetores de complexidade O(n).



**2.3 Insertion Sort**

O algoritmo de ordenação por inserção é, provavelmente, o melhor de todos para a organização de listas pequenas. Seu funcionamento é simples e se assemelha muito à maneira usada para ordenar cartas em jogos de baralho.

Seu procedimento de organização se baseia em percorrer o vetor da esquerda para a direita. A medida que ele avança, vai deixando os elementos à sua esquerda cada vez mais ordenados.

*\* Organizou 50000 elementos em 537 milissegundos;*

*\* Organizou 50000 elementos em 533 milissegundos;*

*\* Organizou 50000 elementos em 537 milissegundos.*

*\**

*\* Organizou 75000 elementos em 1216 milissegundos;*

*\* Organizou 75000 elementos em 1198 milissegundos;*

*\* Organizou 75000 elementos em 1209 milissegundos.*

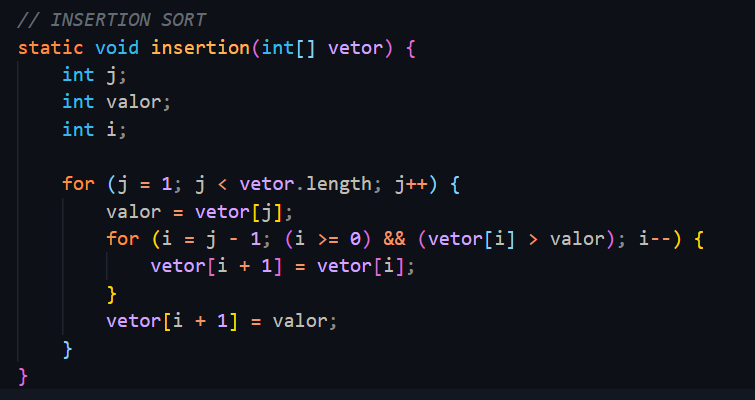
*\**

*\* Organizou 100000 elementos em 2136 milissegundos;*

*\* Organizou 100000 elementos em 2133 milissegundos;*

*\* Organizou 100000 elementos em 2139 milissegundos.*

Com certeza, um método muito importante de saber, devido à sua eficiência e seu meio simples de funcionamento. Porém, ele não é adequado para grandes volumes de elementos. Ainda assim, seu desempenho se mostra muito superior à organização por bolha.



**2.4 Merge Sort**

A organização por junção é denominada de “organização por divisão e conquista”, pois seu funcionamento se baseia em dividir o vetor em dois, organizá-las separadamente e depois mesclá-las em um único vetor.

*\* Organizou 50000 elementos em 10 milissegundos;*

*\* Organizou 50000 elementos em 10 milissegundos;*

*\* Organizou 50000 elementos em 10 milissegundos.*

*\**

*\* Organizou 75000 elementos em 15 milissegundos;*

*\* Organizou 75000 elementos em 18 milissegundos;*

*\* Organizou 75000 elementos em 15 milissegundos.*

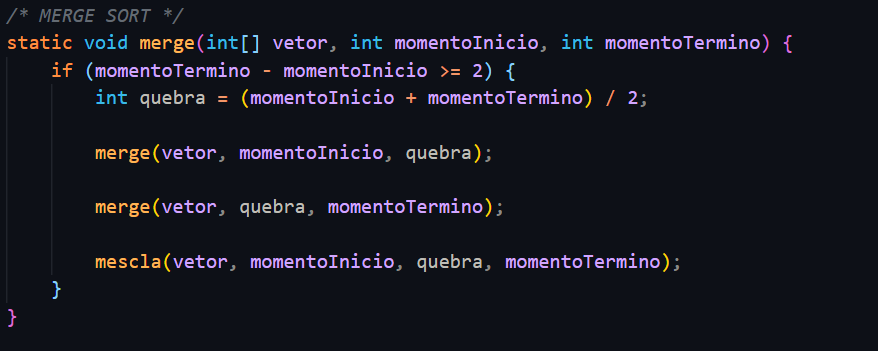
*\**

*\* Organizou 100000 elementos em 19 milissegundos;*

*\* Organizou 100000 elementos em 19 milissegundos;*

*\* Organizou 100000 elementos em 20 milissegundos.*

É um método de complexidade O(n log n), logo, não é o método ideal para organizar quantidades pequenas de elementos, mas é um dos mais eficientes para organizar listas longas. Porém, um empecilho é que ele necessita de memória adicional para organizar os subvetores.





**2.5 Quick Sort**

A organização rápida é um método de organização é conhecido por ser o mais eficiente e, também, por funcionar na maneira de dividir e conquistar.

Nele, se atribui um elemento à posição de pivô, e é a partir dele que a lista será organizada, deixando todos os números menores que ele à esquerda e todos os números maiores que ele à direita.

*\* Organizou 50000 elementos em 26 milissegundos;*

*\* Organizou 50000 elementos em 21 milissegundos;*

*\* Organizou 50000 elementos em 27 milissegundos.*

*\**

*\* Organizou 75000 elementos em 35 milissegundos;*

*\* Organizou 75000 elementos em 51 milissegundos;*

*\* Organizou 75000 elementos em 36 milissegundos.*

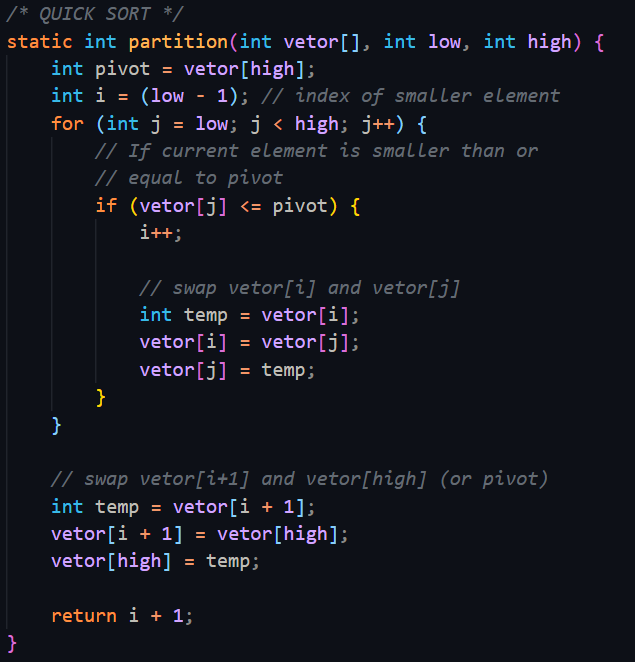
*\**

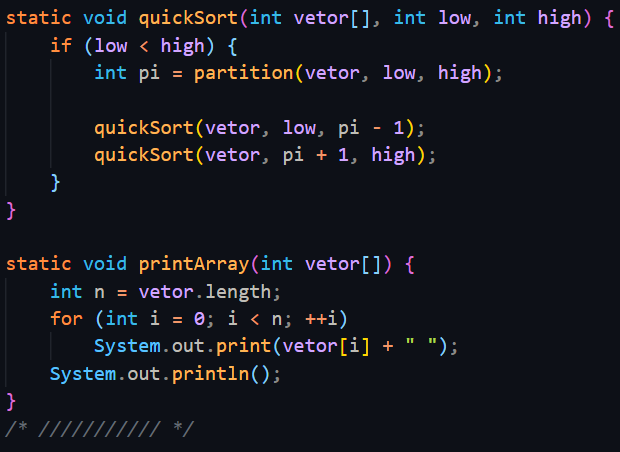
*\* Organizou 100000 elementos em 59 milissegundos;*

*\* Organizou 100000 elementos em 56 milissegundos;*

*\* Organizou 100000 elementos em 57 milissegundos.*

É um método de complexidade O(n log n), portanto é ideal para organizar quantidades grandes de elementos.





**3. Conclusão**

De acordo com os dados apresentados durante a pesquisa, podemos chegar na conclusão de que os algoritmos *bubble, insertion e selection* não são ideais para organizar grandes quantidades de elementos, sendo o *bubble* o menos otimizado para tal e ambos os outros tendo praticamente o mesmo desempenho.

Os outros dois algoritmos, o *quick* e o *merge,* se mostram ser extremamente eficientes na tarefa de organizar vetores com grandes quantidades de elementos, sendo o *merge* mais eficiente que o *quick*, mas com o empecilho de necessitar memória extra para a resolução dos subvetores.

Logo, chega-se à conclusão de que a ordenação por junção é a ideal para grandes quantidades de elementos e que, para quantidades menores, a opção ideal é a ordenação por inserção, por ter um desempenho minimante superior à ordenação por seleção.

**4. Referências**

<https://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28261>

<https://www.geeksforgeeks.org/insertion-sort/>

<https://joaoarthurbm.github.io/eda/posts/merge-sort/>

<https://github.com/gabriel-raamos/Algoritmos-de-Ordena-o>