Universidade Federal do Maranhão
Aluno: João Mateus Silva Lopes
Disciplina: Estrutura de Dados II
Professor: João Mateus Silva Lopes

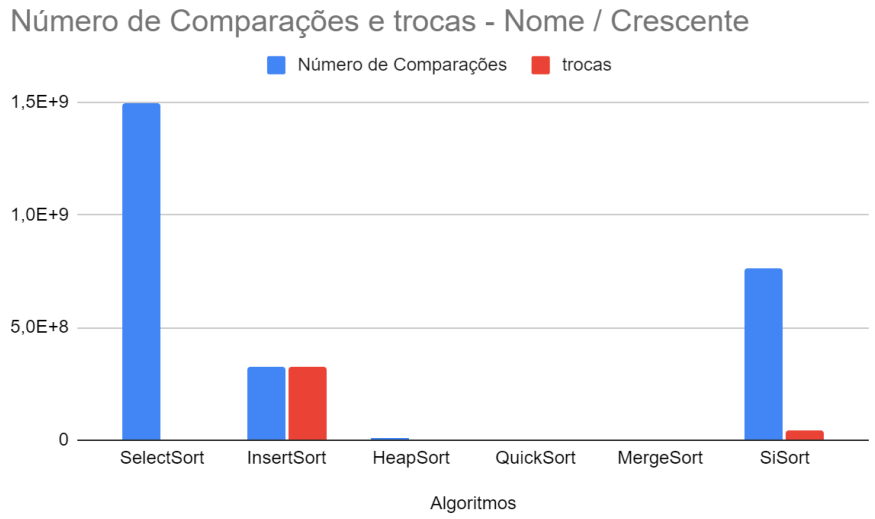
1. Resultados dos testes

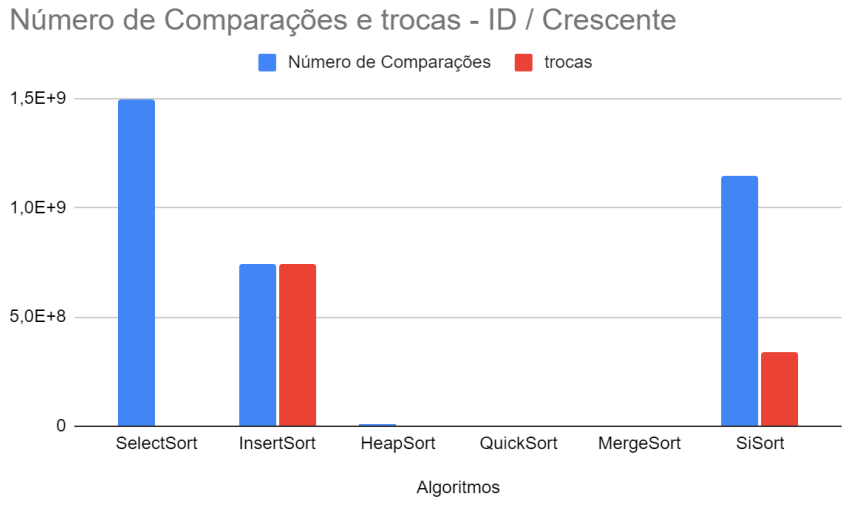
Nome / Crescente			
Algoritmos	**Número de Comparações**	**trocas**	**Tempo(ms)**
SelectSort	1.491.994.408	141999	172361
InsertSort	327.258.782	327.400.781	72.265
HeapSort	7.258.233	2.348.411	124
QuickSort	2.377.264	1.046.633	121
MergeSort	4.871.712	5.155.711	93
SiSort	765.038.945	41.293.858	74294
MergeSortJDK	-	-	60

Nome / Decrescente			
Algoritmos	**Número de Comparações**	**trocas**	**Tempo(ms)**
SelectSort	1.491.994.408	141999	154094
InsertSort	326.445.698	326.587.697	69939
HeapSort	6.818.988	2.201.996	93
QuickSort	2.376.564	1.046.283	77
MergeSort	5.155.711	4.871.712	62
SiSort	762.111.749	44.221.054	73077
MergeSortJDK			

ID / crescente			
Algoritmos	**Número de Comparações**	**trocas**	**Tempo(ms)**
SelectSort	1.491.994.408	141999	34622
InsertSort	742.140.798	742.282.797	27820
HeapSort	7.258.233	2.446.568	54
QuickSort	1.405.122	560.562	53
MergeSort	5.155.711	4.871.712	60
SiSort	1.145.674.501	339.341.698	24665

ID / Decrescente			
Algoritmos	**Número de Comparações**	**trocas**	**Tempo(ms)**
SelectSort	1.491.994.408	141999	36950
InsertSort	749.847.602	749.989.601	30223
HeapSort	7.553.616	2.446.872	51
QuickSort	1.407.654	561.828	51
MergeSort	5.155.711	4.871.712	49
SiSort	1.152.847.882	346.515.079	25885





2. Análise dos Resultados

A ordenação de elementos é um dos problemas mais básicos e importantes na computação. Esse problema consiste em tornar uma sequência de elementos e rearranjá-los de acordo com uma ordem específica. A ordem pode ser crescente, decrescente, sendo elementos como texto, números, datas, entre outros.

Neste trabalho é feito uma análise sobre os resultados obtidos na aplicação de algoritmos de ordenação: SelectSort, InsertSort, HeapSort, QuickSort, MergeSort e SIsort. Esses algoritmos são utilizados para a ordenação de um arquivo CSV com 142 mil itens, com cada item apresentado 6 atributos. Os itens se apresentam em uma ordem aleatória em todos os seus campos, sendo assim impossível testar os melhores e piores casos de cada algoritmo.

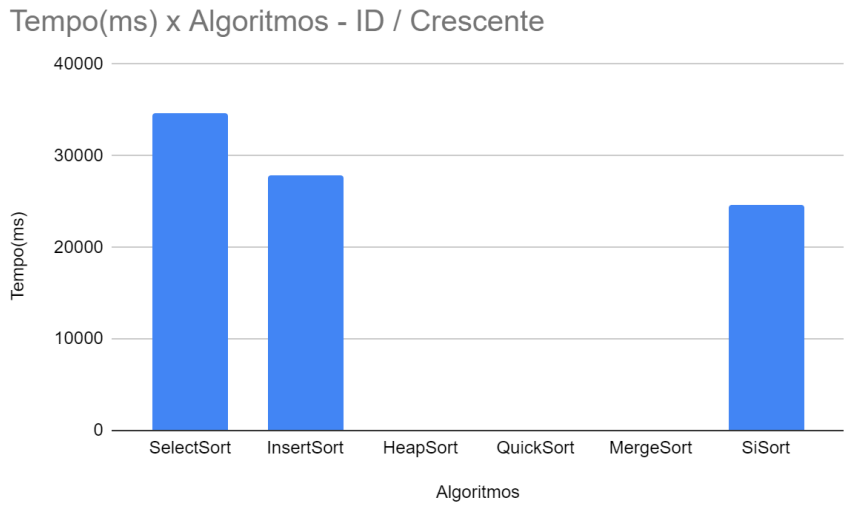
Para uma primeira análise, é possível perceber que os resultados obtidos estão próximos dos resultados obtidos com a análise da complexidade assintótica, considerando o caso médio das complexidades assintóticas de casa algoritmo. Usando o algoritmo HeapSort, como exemplo para uma análise sobre os resultados obtidos, a quantidade de operações é dada por n log(n), sendo N a quantidade de itens, temos a quantidade de operações é dada por: 142000*log(142000) = 1.012.428, diferente do resultado obtido no teste feito no qual apresenta o valor de 2.446.872 operações. Logo é possível perceber que nlog(n) contém o fator C que multiplica a função que determina as operações realizadas, sendo o K próximo de 2,4.

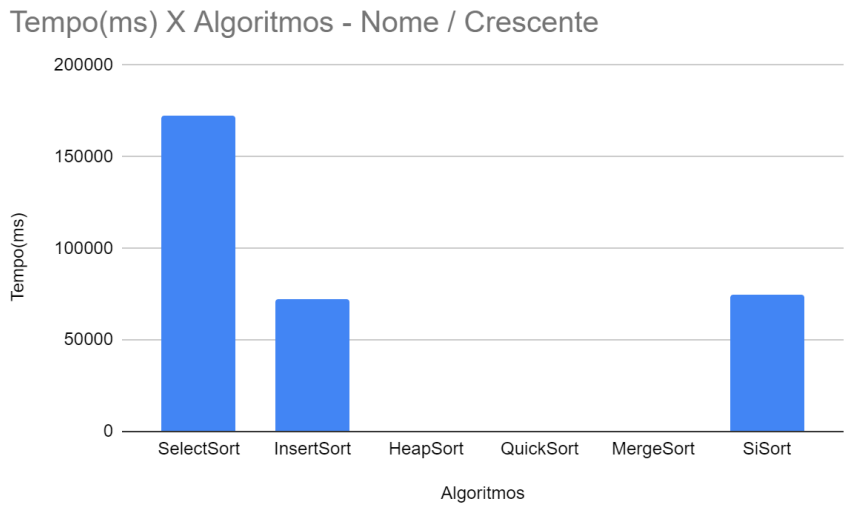
Outro fator pertinente, é o tipo de dado usado para ordenar, pois como obtido pelos testes realizados, o tipo de dado nome “String” apresenta os maiores tempos de execução. Essa influência do tipo de dados em cima do tempo de execução é mostrado nas tabelas: Nome / Crescente e ID / Decrescente, onde a tabela onde contem o nome apresenta a média de tempo de execução bem maior que a tabela onde é ordenado pelo ID(int).

ID / crescente			
Algoritmos	**Número de Comparações**	**trocas**	**Tempo(ms)**
SelectSort	1.491.994.408	141999	34622
InsertSort	742.140.798	742.282.797	27820

Nome / Crescente			
Algoritmos	**Número de Comparações**	**trocas**	**Tempo(ms)**
SelectSort	1.491.994.408	141999	172361
InsertSort	327.258.782	327.400.781	72.265

Nas tabelas apresentadas é perceptível a diferença do tempo de execução, onde o fator **dado** é o valor que se altera, fazendo com que o tempo de execução aumente.





O gráfico deixa mais evidente que o algoritmo selectSort tem o crescimento de forma bem maior, pois o mesmo é O(n^2) para médio, melhor e pior caso, sendo assim o crescimento sempre vai superar os dos demais algoritmos.

Outro ponto a se analisar é a utilização de algoritmos como heapSort, MergeSort e QuickSort para ordenação de grandes quantidades de itens, pois, como é possível verificar nos testes realizados, os algoritmos citados tem tempo de execução insignificantes comparados ao outros algoritmos devido ao crescimento nlog(n) da maioria deles o que favorece esses métodos de ordenação para grandes quantidades de itens.

3. Referências

Ordenar um ArrayList pelo atributo. Disponível em: <https://www.devmedia.com.br/forum/ordenar-um-arraylist-pelo-atributo/568549> . Acesso em: 1 maio. 2023.

CASTIGLIONI, M. Strategy — padrões de projeto em java - CollabCode - medium. Disponível em: <https://medium.com/collabcode/strategy-padr%C3%B5es-de-projeto-em-java-43889a3afc5a>. Acesso em: 1 maio. 2023.

BRUNET, J. A. Estruturas de dados e algoritmos. Disponível em: <https://joaoarthurbm.github.io/eda/posts/merge-sort/>. Acesso em: 1 maio. 2023

Heap sort. Disponível em: <https://www.geeksforgeeks.org/heap-sort/>. Acesso em: 1 maio. 2023.