Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Diogo Silva Vilar, Francisco Welio Firmino da Silva Junior

SUMÁRIO

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de LEDA que teve como objetivo compreender e pôr em prática os algoritmos de ordenação que foram estudados juntamente com os métodos de análise dos resultados gerais obtidos a partir desses algoritmos no decorrer da disciplina de estrutura de dados, com isso, o projeto na qual constitui esse relatório baseia-se em, a partir de um dataset disponibilizado pelo professor aos alunos chamado “passwords.csv”, que contém uma listagem com mais de 600 mil senhas registradas, juntamente com a informação de seus respectivos tamanhos e datas de criação, fossem realizados métodos que fizessem uma análise sob esse dataset, a fim de classificar as senhas de acordo com os requisitos das seguintes classificações: “Muito Ruim”, “Ruim”, “Fraca”, “Boa”, “Muito Boa” e “Sem Classificação”.

Isso será feito a partir do desenvolvimento dos algoritmos de ordenação Insertion Sort, Selection Sort, Merge Sort, Quick Sort, Quick Sort com Mediana 3, Counting Sort e Heap Sort, todos foram abordados em sala de aula com relação à funcionamento e custo de desempenho.

Com isso, a partir dos métodos realizados será criado um novo arquivo chamado password_classifier, que irá armazenar todas as informações do dataset “passwords.csv” contendo uma coluna com a classificação de cada senha, apresentando um filtro pelas categorias de senhas que receberam as classificações “Boa” e “Muito Boa” pelo programa.

Após a realização de todos os processos listados acima, realizou-se uma análise a respeito de cada algoritmo, observando assim quais obtiveram os melhores e os piores resultados em cada um dos casos que foram solicitados, com foco na observação do tempo que foi necessário para a finalização do programa em milissegundos.

2. Descrição geral sobre o método utilizado

Todos os testes foram feitos em uma máquina usando o compilador Visual Studio Code, com todos os programas sendo realizados na linguagem de programação JAVA, usando o ambiente de desenvolvimento integrado JDK (Java Development Kit) na versão 19.0.1, a mais atualizada até a data atual (06/11/2022).

Utilizando métodos e funções para facilitar a compilação e organização do programa, localizamos o arquivo passwords.csv na pasta local e realizamos a leitura do mesmo.

No total, a pasta possui no total 23 arquivos que irão constituir o programa:

- LerCsv.java
- README.md
- passwords_length_coutingSort_medioCaso.csv
- passwords_length_coutingSort_melhorCaso.csv
- passwords_length_coutingSort_piorCaso.csv
- passwords_length_heapSort_medioCaso.csv
- passwords_length_heapSort_melhorCaso.csv
- passwords_length_heapSort_piorCaso.csv
- passwords_length_insertionSort_medioCaso.csv
- passwords_length_insertionSort_melhorCaso.csv
- passwords_length_insertionSort_piorCaso.csv
- passwords_length_mergeSort_medioCaso.csv
- passwords_length_mergeSort_melhorCaso.csv
- passwords_length_mergeSort_piorCaso.csv
- passwords_length_quick3Sort_medioCaso.csv
- passwords_length_quick3Sort_melhorCaso.csv
- passwords_length_quick3Sort_piorCaso.csv
- passwords_length_quickSort_medioCaso.csv
- passwords_length_quickSort_melhorCaso.csv
- passwords_length_quickSort_piorCaso.csv
- passwords_length_selectionSort_medioCaso.csv

- passwords_length_selectionSort_melhorCaso.csv
- passwords_length_selectionSort_piorCaso.csv

Todos os arquivos presentes na pasta estão vazios e serão preenchidos após rodar o programa principal LerCsv.java.

2.1 Descrição geral do ambiente de testes

Todos os testes foram realizados em três máquinas com configurações diferentes, a primeira máquina com o processador Intel Core i3 6006U, 8GB de memória RAM com 2333mhz de frequência, um SSD de 120GB e gráficos Mesa Intel® HD Graphics 520 (SKL GT2) com sistema operacional Linux Ubuntu 22.04.i LTS.

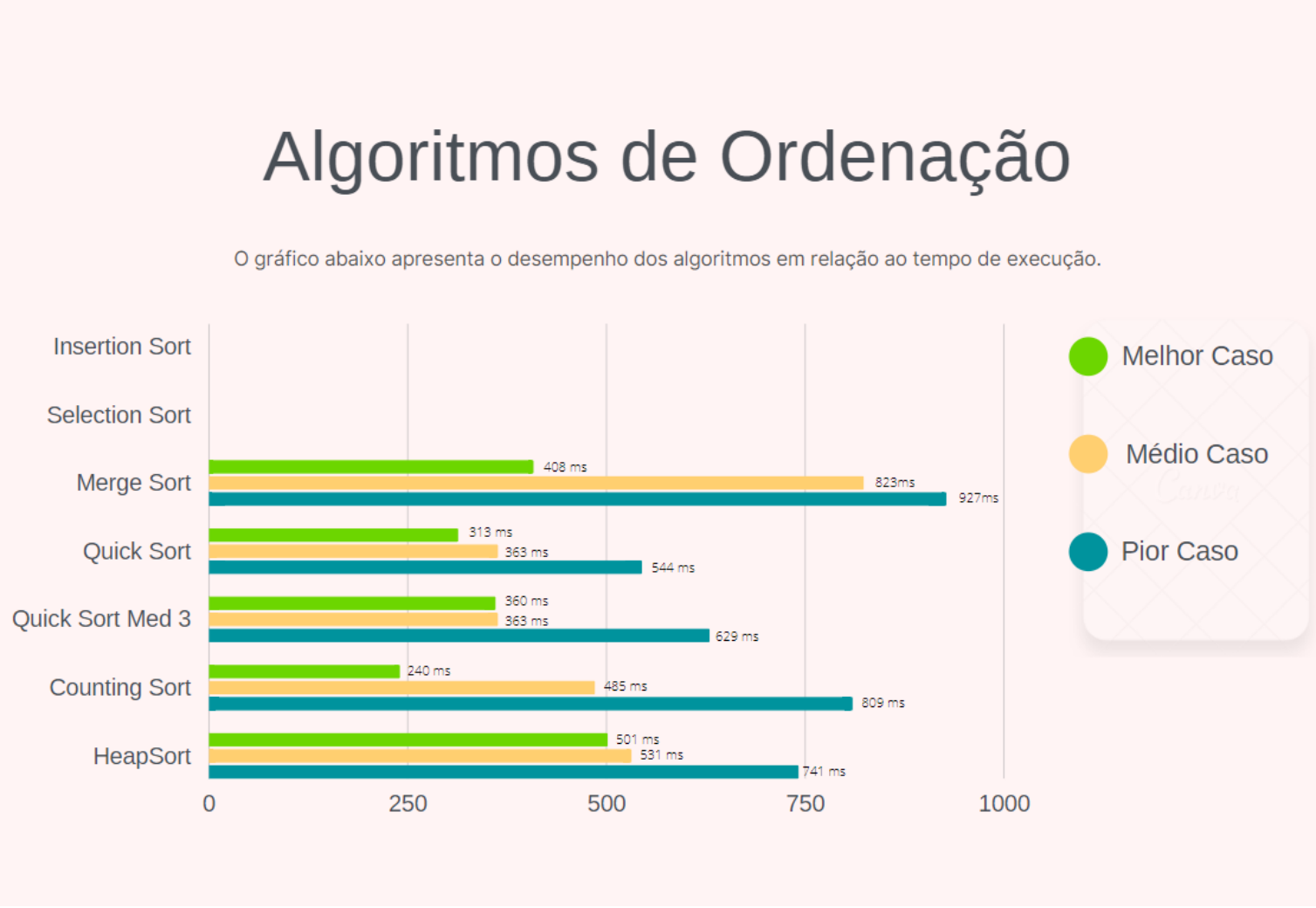
A segunda máquina utilizada para os testes possui um processador AMD Ryzen 5 3600, com 8x2 GB de memória RAM com 2666mhz de frequência, um HD de 1TB e um SSD de 256GB, placa de vídeo dedicada RTX 3080 com sistema operacional Windows 11.

A terceira máquina utilizada possui o processador Intel I5 9300H, 8x2GB de memória RAM com 2666mhz de frequencia, um HD de 1TB e um SSD de 120GB com sistema operacional Windows 10.

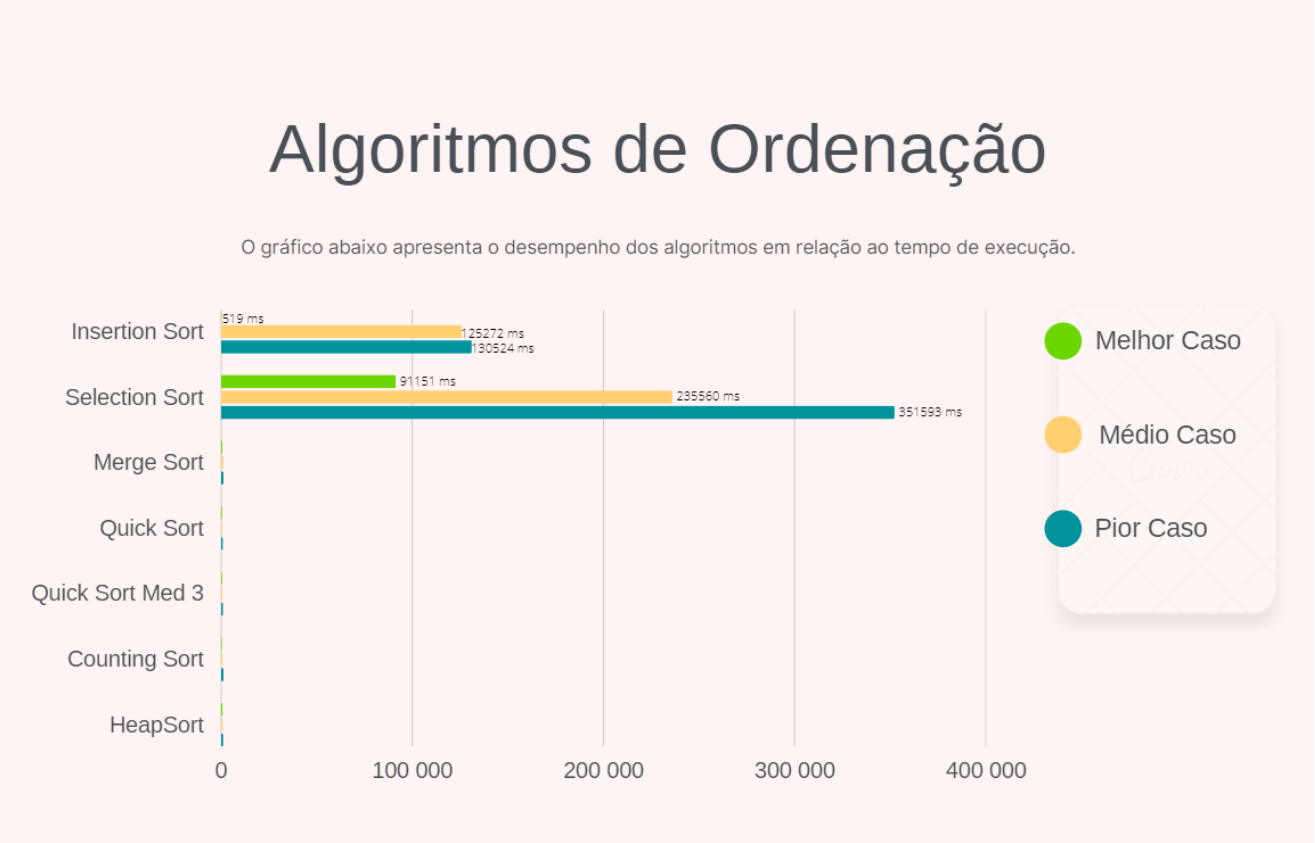
3. Resultados e Análise

Análise Geral ordenando de acordo com índice, senha, tamanho e data:

	Pior Caso	Caso Médio	Melhor Caso
Insertion Sort	130524	125272	519
Selection Sort	351593	235560	91151
Merge Sort	927	823	408
Quick Sort	544	363	313
Quicksort Med 3	629	363	360
Counting Sort	809	485	240
Heapsort	741	531	501



Os algoritmos do Selection e Insertion Sort por serem os mais lentos, quebraram o gráfico de barras como demonstrado na imagem abaixo:



3.1 Algoritmos mais eficientes

A partir da tabela e dos gráficos acima, percebe-se que o algoritmo do Counting Sort foi o que teve o melhor caso dentre todos os algoritmos com 240 milissegundos de execução, mas não muito atrás ficou o Quick Sort, Quick Sort de Mediana 3 e o Heap Sort, com valores próximos sendo estes 313, 360 e 501 milissegundos respectivamente, porém os algoritmos de Insertion Sort, Selection Sort e Merge Sort foram os que obtiveram pior desempenho sendo o Merge Sort o mais próximo dos algoritmos com melhor desempenho.

Diante das informações acima percebemos que os algoritmos Heap, Quick Sort, Quick Sort com Mediana 3 e Counting Sort se mostraram bastante eficientes, isso ocorre devido a complexidade algorítmica, que faz com que estes fossem os algoritmos que obtivessem resultados bastante satisfatórios, tendo como destaque o Quick Sort, pois este se saiu com mais desempenho que os outros, por ter a menor variância nos valores de tempo de resposta, não tendo muita diferença entre os casos, já que o melhor caso durou 313 ms, o caso médio 363 ms e o pior caso 544 ms, fazendo com que esse seja o algoritmo mais eficiente.

3.2 Análise geral sobre os resultados

Como já esperado, os algoritmos Insertion Sort e Selection Sort foram os que tiveram os piores desempenhos e resultados nos testes realizados, chegando a obter valores absurdos quando comparados aos outros métodos de ordenação como o Heap Sort, Counting Sort, Quick Sort e Quick Sort de Mediana de 3, porém o Merge Sort também trouxe um resultado inesperado, já que foi um dos menos rápidos, porém não chega nem perto dos valores do Selection E insertion Sort, isso ocorre devido a sua complexidade quadrática e acaba fazendo com que para o caso do dataset passwords seja inviável o uso desses algoritmos, sendo recomendado apenas para o uso com um

dataset pouco vasto e com menos informações, considerando que são algoritmos mais simples e de rápida implementação.

Enquanto isso os algoritmos de Counting, Heap, Quick de mediana 3 e Quick Sort trouxeram resultados mais otimizados e satisfatórios para o nosso caso, realizando todo o processo em questão de segundos, fazendo com que o ganho de desempenho seja exponencialmente vantajoso em comparação com os demais.

4. Conclusão

Este trabalho mostrou os pontos fortes e fracos dos algoritmos, o tempo de execução e como o nível de complexidade deles interferem no desempenho do programa. O Insertion Sort e o Selection Sort, foram os algoritmos que de longe mais demoraram para realizar a ordenação de 669880 dados do arquivo password.csv tendo em vista que são algoritmos $O(n^2)$. Os mais rápidos foram o Counting Sort que tem complexidade O(k+n), QuickSort normal e com mediana de 3 ,com complexidade O(n log n) e Heapsort também com complexidade O(n log n). Apesar do MergeSort também possuir complexidade O(n log n), ele não apresentou uma velocidade maior ficando por último dos algoritmos com mais desempenho, mas na frente o dos algoritmos com piores desempenho .