Laboratório de Estrutura de Dados

Versão Final do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Grupo: Password project

Nome dos alunos

João Pedro Miranda Cariry
Joanderson Pereira de Souza
Fenando de Paiva Almeida Ferreira

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de Laboratório de Estrutura de Dados (LEDA) que consiste na comparação de alguns algoritmos de ordenação para isso foi desenvolvido um algoritmo em linguagem Java e utilizado os dados do arquivo passwords.csv, contendo a listagem de mais de 600 mil senhas, baixado através do link.

(https://drive.google.com/file/d/1-8WPvcqCEf7dAnuRTxrdCBln81o_9X4S/view).

A partir das informações desse arquivo o algoritmo gerou a classificação, a filtragem e a ordenação para comparação do desempenho dos algoritmos de ordenação. Este arquivo é dividido em colunas com as seguintes categorias:

- Password - contém as senhas de usuários
- Length - contém a quantidade de caracteres para cada respectiva senha
- Data - contém a data em que a senha foi criada na sequencia ano mês dia e hora.

Classificação

Para a classificação, foi usado como entrada de dados o arquivo passwords.csv. Após classificar as senhas, o algoritmo gera um novo arquivo com o nome de password_classifier e com a coluna (class) referente a cada classe para a respectiva senha. São classificadas de acordo com suas características seguindo as regras abaixo:

- Muito Ruim: tamanho da string menor que 5, e só um tipo de caractere, por exemplo: só letra (letras minúsculas ou maiúsculas), só número ou só caractere especial.

- Ruim: tamanho da string menor ou igual a 5, e só um tipo de caracteres, por exemplo: letra (letras minúsculas ou maiúsculas), número ou caractere especial.

- Fraca: tamanho da string menor ou igual a 6, e só dois tipos de caracteres, por exemplo: letra (letras minúsculas e maiúsculas), número ou caractere especial.

- Boa: tamanho da string menor ou igual a 7, e só todos de caracteres, por exemplo: letra (letras minúsculas e maiúsculas), número e caractere especial.

- Muito Boa: tamanho da string maior que 8, e só todos os tipos de caracteres, por exemplo: letra (letras minúsculas e maiúsculas), número ou caractere especial

- Sem Classificação: Senhas que não se qualificam com nenhuma das classificações acima. Nos casos específicos das senhas com mais de um tipo, fica a critério dos alunos escolher a qual o tipo classificar.

Abaixo segue a lista dos algoritmos de ordenação utilizados para comparação.

- Bubblesort
- Selectionsort
- Insertionsort
- Mergesort
- Radixsort
- Quicksort Mediana de 3
- Countingsort
- Heapsort
- Radixsort

Transformações

Para as transformações, foi utilizado como entrada dos dados o arquivo com o nome password_classifier.csv.

1. Transformar data para o formato a seguir DD/MM/AAAA (dia / mês / ano)
   - Gerar um arquivo chamado **passwords_formated_data.csv**
2. Filtrar senha pela categoria Boa e Muito Boa.
   - Gerar um arquivo chamado **passwords_classifier.csv**

2. Descrição geral sobre o método utilizado

Ordenações

Para as ordenações dos dados, foi utilizado como entrada de dados o arquivo passwords_formated_data.csv. E para ser feita a análise dos algoritmos foram gerados um arquivo diferente para cada algoritmo e para cada caso (melhor, médio e pior).

1. Ordenar o arquivo completo de senhas pelo campo length em ordem decrescente por exemplo.

- *passwords_length_insertionSort_medioCaso.csv*
- *passwords_length_insertionSort_piorCaso.csv*
- *passwords_length_insertionSort_melhorCaso.csv*

2. Ordenar o arquivo completo de senhas por mês, (OBS: deve-se obter o mês a partir da data já formatada) da coluna data, de forma crescente, por exemplo.

- *passwords_data_month_insertionSort_medioCaso.csv*
- *passwords_month_data_insertionSort_piorCaso.csv*
- *passwords_data_month_insertionSort_melhorCaso.csv*

3. Ordenar o arquivo completo de senhas pela coluna data de forma crescente. por exemplo.

- *passwords_data_insertionSort_medioCaso.csv*
- *passwords_month_insertionSort_piorCaso.csv*
- *passwords_data_insertionSort_melhorCaso.csv*

Foi utilizado no código uma função para medir tempo de execução de cada algoritmo comparando assim a eficiência de cada algoritmo de ordenação

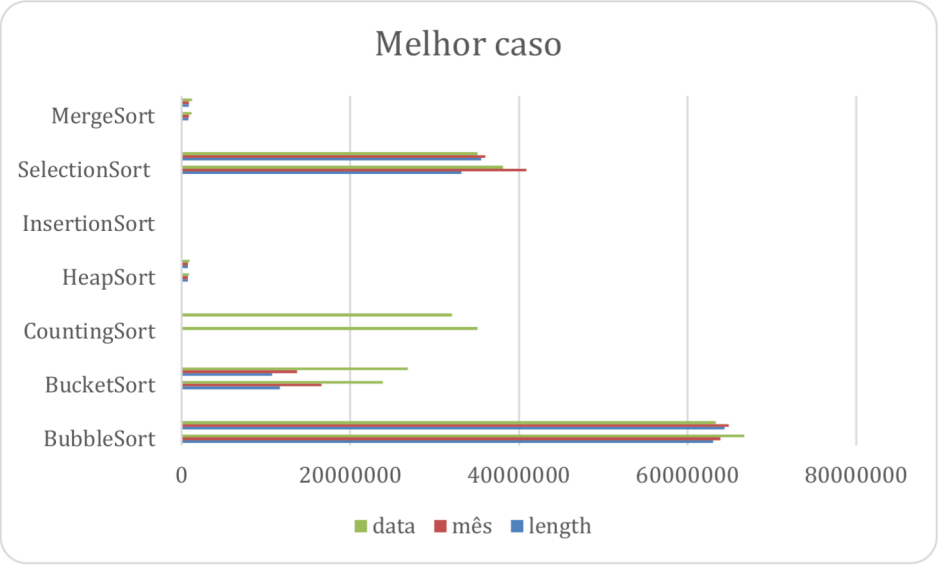
Descrição geral do ambiente de testes

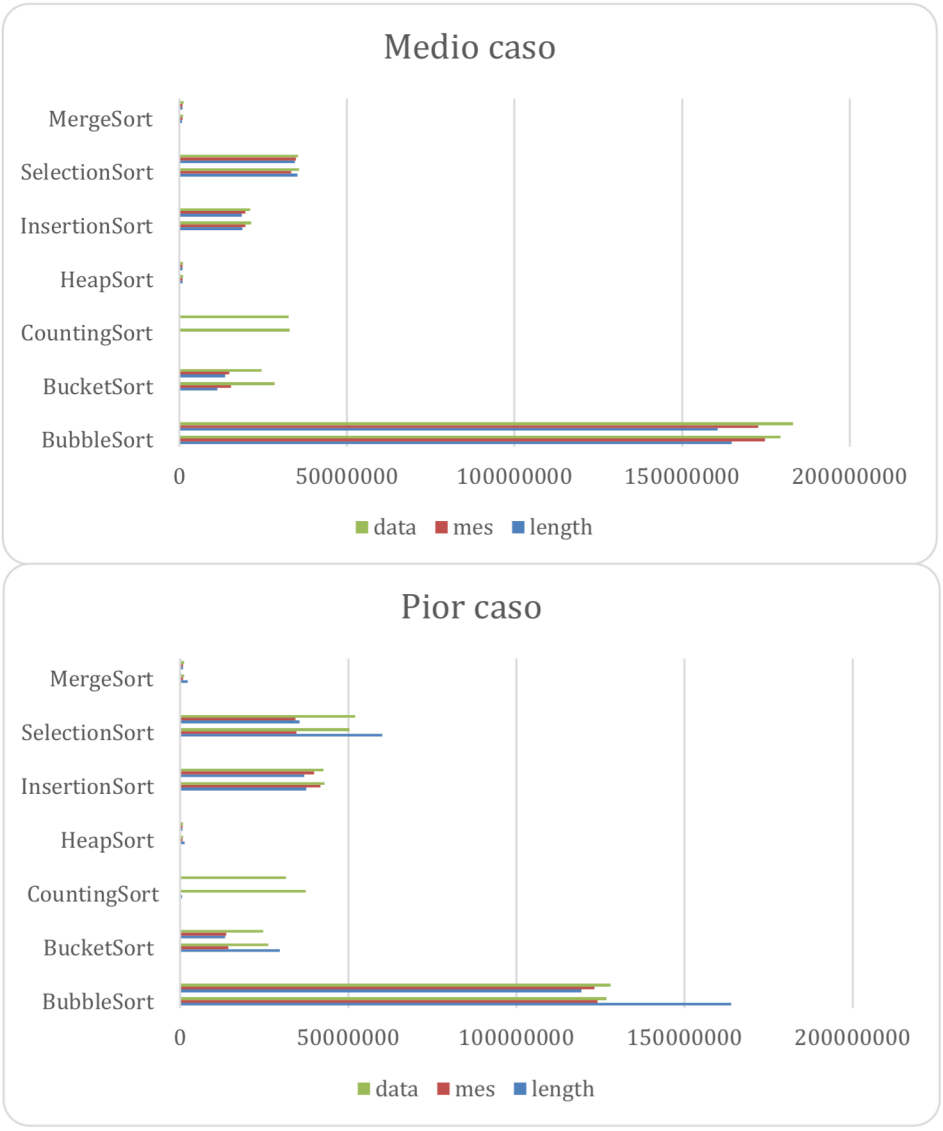
As configurações da máquina usada para criar o projeto e para submetê-lo aos testes foram:

- Sistema operacional Windows 10 de 64 bits
- Processador AMD Ryzen 2600 Six-Core Processor
- 16,0GB de memória RAM DDR4
- Editor de código-fonte Visual Code.

3. Resultados e Análise

Abaixo segue o resultado de todos os dados obtidos, em cada ordenação de caso. Medindo o tempo de execução (em nanosegundos). Então a partir dos dados coletados foram produzidos gráficos utilizando o Excel.





Ao analisar os gráficos acima de acordo com cada caso e método de ordenação é possível visualizar que alguns algoritmos que fazem menos comparações como o mergesort e o heapsort que se comportaram de forma eficiente, ao ordenar valores em string ou numéricos. Alguns algoritmos não tiveram um desempenho tão bom como é o caso do Bubble sort que se comportou mal em todos os casos.

O insertion sort teve ótimo desempenho ao ordenar no melhor caso, porem nos outros casos ele sofreu um aumento significativo no seu tempo de execução. O counting sort foi implementado apenas em dados que podiam ser considerados como inteiro, como

é o caso do tamanho das senhas e o número referente ao mês, já na ordenação das datas o desempenho não foi tão bom.

Quais os algoritmos mais eficientes e por qual motivo?

	Melhor Caso	Médio Caso	Pior Caso
Bublesort			
Selectionsort			
Insertionsort			
Mergesort			
Quicksort Mediana de 3			
Countingsort			
Heapsort			
Radixsort			

Qual é a sua análise geral sobre os resultados?