Laboratório de Estrutura de Dados

Segunda versão do projeto da disciplina

Ordenação elementar utilizando uma estrutura de dados

Diogo Silva Vilar, Francisco Welio Firmino da Silva Junior

SUMÁRIO

1. Introdução……3

2. Descrição geral sobre o método utilizado……..4

   2.1 Descrição geral do ambiente de testes……5

3. Resultados e Análise……5

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

Isso será feito a partir do desenvolvimento da ordenação em uma estrutura de dados, sendo a escolhida a Árvore Binária. Inicialmente optou-se por fazer com lista encadeada, porém, a leitura dos dados já estava sendo bastante demorada para a quantidade de valores lidos, de tal modo que foi preferido trocar para Árvore binária.

Com isso, a partir dos métodos realizados será criado um novo arquivo chamado DateOrdenada.csv ,LengthOrdenada.csv, PasswordOrdenada.csv , que irá armazenar todas as informações do dataset “passwords.csv” contendo uma coluna com a classificação de cada senha.

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 5 pastas e 13 arquivos que irão constituir o programa:

- .vscode
  - settings.json
- arquivos
  - DateOrdenada.csv (vazio)
  - LengthOrdenada.csv (vazio)
  - PasswordOrdenada.csv (vazio)
- bin
  - Arvore.class
  - Dado.class
  - TrabalhoComArvores.class
- lib
  - opencsv-3.8.jar
- src
  - Arvore.java
  - Dado.java
  - TrabalhoComArvores.java (main)
- Projeto 2 Passwords.pdf
- README.md

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
Árvore binária		10691 ms	
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

A estrutura utilizada foi a Árvore Binária pois ela é uma estrutura não linear e tem custo O(log n) o que possibilita a leitura e armazena mais rapidamente, tendo menos custo computacional. Mostrando que em caso médio, teve um tempo relativamente rápido, tendo em vista a comparação com o Insertion e Selection Sort. Porém teve um rendimento menor que os outros algoritmos.

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. E como uma estrutura de dados com uma árvore pode interferir, promover uma leitura rápida

dos dados, e uma execução prática. Apesar da movimentação/ leitura de uma árvore ser O(log n), o tratamento de dados e a ordenação custam mais tempo de execução, em caso apenas da leitura do arquivo.csv, é previsto uma redução considerável do tempo de execução.