Laboratório de Estrutura de Dados

Segunda versão do projeto da disciplina

Implementação de 3 Estruturas de Dados na Primeira versão do projeto

Arthur Pereira da Silva; Antônio Wenicios Ademar da Silva

1. Introdução

O objetivo da segunda etapa do projeto da disciplina de Laboratório de Estrutura de Dados foi a implementação de pelo menos 3 estruturas de dados na primeira versão do projeto. O presente relatório busca evidenciar em que parte foram feitas as implementações, bem como as mudanças promovidas em detrimento da aplicação das estruturas.

As transformações pedidas são: substituição do id das estações pelo nome (campo station_name), que está contido no segundo dataset (station_csv); filtragem das viagens que estão nas estações de Pasadena; bem como filtragem das viagens que possuem duração maior que a média geral.

Já as ordenações são as seguintes: ordenar o arquivo pelo nome das estações, em ordem alfabética; ordenar pelo campo de duração da viagem, do menor para o maior; e ordenar pela data de início da viagem, da mais recente para a mais antiga.

As principais estruturas de dados implementadas foram as listas encadeadas e as filas. As listas e as filas são capazes de modificar dados de forma ágil e eficiente, por isso viabilizou-se a sua implementação nas transformações e ordenações, uma vez que não existe limitação de armazenamento, a depender da máquina. Além disso, viabilizou-se a implementação da estrutura arraylist.

2. Descrição geral sobre o método utilizado

O Objetivando-se tratar os dados, foram feitas modificações no dataset considerado para as ordenações, sendo que, no melhor caso, ocorre a modificação dos dados para serem dispostos de forma crescente; no pior caso, os dados são invertidos (ordem decrescente); e no médio caso eles são dispersos, ou seja, dispostos de forma randômica.

Ao executar o programa principal (App), primeiramente, são carregadas todas as transformações, e, logo em seguida, todas as ordenações são carregadas.

Dependendo das configurações de uma máquina, pode-se obter diferentes interpretações para analisar a eficiência de um algoritmo. Por esse motivo, tendo como base a máquina descrita abaixo, as análises realizadas devem ser consideradas para casos de testes semelhantes.

Descrição geral do ambiente de testes:

O código foi executado em diferentes máquinas, mas, para fins de análise, foram consideradas apenas uma. Segue, abaixo, as configurações:

- Processador Intel(R) Core(™) i5-3470 CPU @ 3.20GHz
- Memória RAM instalada 16,0 GB (utilizável: 15,9 GB)
- Sistema Operacional: Windows 11 / 64 bits
- IDE Intellij

3. Análises e justificativas

3.1 Tranformações

3.1.1 Transformação 1

Para a transformação 1 , foi implementado um hashmap com as informações das estações do arquivo “stations.csv”. Posteriormente, é implementada uma fila com os dados do arquivo principal “LA_Metro_BikeSharing_CLEANED_2016quater3-2021q3.csv”. Dessa forma, em cada valor da fila é feita uma varredura no hashmap para obter o nome das estações, gerando um arquivo CSV transformado.

3.1.2 Transformação 2

Essa transformação é responsável por filtrar apenas as estações que estam na região de Pasadena. Esse algoritmo faz a remoção das linhas que não contêm os nomes das estações de Pasadena.

Inicialmente, é implementado um ArrayList para armazenar o nome das estações. O algoritmo lê o arquivo “pivot_stations.csv” e passa o nome das estações para esse ArrayList. Depois o arquivo “LAMetroTrips.csv” é lido, fazendo-se a verficação de cada linha. Se a linha contém o nome de uma estação de Pasadena armazenada no ArrayList, então faz-se a inserção na fila. Os dados da fila são escritos no arquivo “LAMetroTrips_F1.csv”.

3.1.3 Transformação 3

Nesta transformação, é feita uma filtragem das viagens que duram mais que a média geral. O algoritmo implementado utiliza-se de um construtor para calcular contagem das viagens e a soma total, obtendo-se a média. Através de uma função, filtra-se todas as viagens que possuem média maior que o geral, que passam a ser inseridas em uma fila, para depois serem escritas no arquivo CSV.

3.2 Ordenações

3.2.1 Ordenação 1

Essa classe das ordenações é responsável por ordenar o arquivo pelo nome das estações em ordem alfabética. Para tanto, cada linha do arquivo CSV é lida e inserida na lista encadeada. Posteriormente, é feita a ordenação da lista, que é passada para um outro arquivo CSV.

3.2.2 Ordenação 2

Na ordenação 2, promove-se uma ordenação pela duração da viagem, do menor para o maior valor. O arquivo passado é o LAMetroTrips.csv. Foi feita uma implementação da lista encadeada, onde cada registro do arquivo é passado para um nó, que posteriormente é adicionado na lista encadeada. A ordenação é feita de forma automática, por isso a escolha por essa estrutura. Por fim, os dados são escritos no CSV.

3.2.3 Ordenação 3

Na ordenação 3, é feita uma ordenação pela data da viagem, da mais recente para a mais antiga. Assim, como nas demais, nessa ordenação é realizada a implementação da lista encadeada, passando-se o arquivo para ser lido e inserindo-se os registros na lista, em que é feita a ordenação automática. Por fim, a lista, é escrita em um arquivo CSV.

4. Resultados

Pode-se notar que a implementação das listas gerou um demanda maior de tempo para a realização das ordenações, algo que poderia ser realizado mais facilmente com a implementação dos algoritmos da versão inicial do projeto. O uso das listas encadeadas seria melhor aplicado para a questão do armazenamento de valores, pois diferentemente da versão inicial do projeto não teríamos tanta preocupação com o limite de armazenamento.