Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Identificação do aluno,

Denis William Muniz de Souza - Matrícula: 212080296

João Victor de Araújo Silva - Matrícula: 212080270

Rodrigo Loiola de Farias - Matrícula: 212080032

1. Introdução

Este relatório apresenta os resultados do projeto desenvolvido na disciplina de LEDA, que teve como foco a análise de dados do sistema de compartilhamento de bicicletas de Metrô de Los Angeles e área metropolitana, conhecido como Lost Angeles Metro Bike Share. O conjunto de dados utilizado para o projeto abrange o período de 2016 a 2021 e contém informações gerais, como duração da viagem, tipo de bicicleta, estação inicial, horário de início e fim da locação, entre outros.

Para realizar as análises e transformações nos dados, foi utilizado o Java como linguagem de programação. O projeto exigiu a implementação de soluções divididas em dois grupos: transformações e ordenações. Cada solução exigiu a geração de um arquivo no formato CSV com o resultado esperado para cada grupo.

No grupo de transformações, foram realizadas mudanças no conjunto de dados com o objetivo de facilitar a referência das estações pelo nome, a aplicação de filtros por localidade e a identificação de viagens com duração maior do que a média geral. Essas transformações foram importantes para facilitar a análise dos dados e extrair informações úteis para o projeto.

No grupo de ordenações, foram aplicados algoritmos para ordenar os dados de acordo com critérios específicos, como a duração da viagem ou o tipo de bicicleta utilizada. Essas ordenações foram fundamentais para identificar padrões e tendências nos dados e permitiram a criação de gráficos e visualizações que ajudaram a entender melhor o comportamento dos usuários do sistema de compartilhamento de bicicletas.

Em resumo, o projeto da disciplina de LEDA focou na análise de dados do sistema de compartilhamento de bicicletas de Metrô de Los Angeles e área metropolitana, utilizando a linguagem de programação Java para implementar soluções de transformação e ordenação nos dados. As transformações foram voltadas para a referência das estações pelo nome, filtros por localidade e identificação de viagens com duração maior que a

média geral. As ordenações permitiram identificar padrões e tendências nos dados e auxiliaram na criação de gráficos e visualizações.

Em relação as soluções de ordenação, envolvem a aplicação de ordenação envolvendo o melhor, médio e pior caso. Nas seguintes situações:

- Ordenação pelo nome das estações em ordem alfabética.
- Ordenação pela duração da viagem (Do menor para o maior).
- Ordenação pela data de início da viagem, do recente para o mais antigo.

Os algoritmos usados, foram

- Insertion Sort
- Selection Sort
- Merge Sort
- Quick Sort e Quick Sort com mediana de 3
- Counting Sort
- Heap Sort

Este trabalho foi dividido em três seções. Na primeira seção, foi apresentada uma visão geral do trabalho. A segunda seção, denominada "Descrição Geral", descreve os testes realizados e como a ferramenta foi implementada. Já a terceira e última seção, intitulada "Resultados e Análise", consiste na evidenciação da comparação dos testes realizados.

Ao avaliar os resultados obtidos neste projeto, foi possível inferir que o algoritmo de ordenação mais eficiente em todos os tipos de ordenação é o Merge Sort, enquanto o pior desempenho foi apresentado pelo Selection Sort. Entretanto, o Counting Sort apresentou uma execução mais rápida que o Merge Sort em determinadas situações.

Os resultados obtidos podem ser utilizados para orientar a escolha do algoritmo de ordenação mais adequado para determinado conjunto de dados. Além disso, a análise

realizada permite identificar quais algoritmos apresentam melhor desempenho em diferentes cenários, o que pode ser útil na otimização de processos que envolvem grandes volumes de dados.

2. Descrição geral sobre o método utilizado

Para otimizar a entrega, utilizamos uma amostra de 13.000 linhas para os testes de ordenação e não houve necessidade de ajuste no conjunto de dados para os testes de transformação.

Na análise das transformações, comparamos o uso de CPU, memória RAM, tamanho do arquivo gerado e tempo de execução. Já na análise dos algoritmos de ordenação, foram realizadas duas modalidades de testes.

Na primeira modalidade, avaliamos o desempenho de cada algoritmo de ordenação em diferentes casos (médio, melhor e pior), levando em consideração o uso de recursos da máquina, como o nome da estação, a duração e a data inicial. O tempo de execução foi medido em segundos.

Na segunda modalidade, comparamos todos os algoritmos de ordenação em relação ao tipo de caso, buscando identificar qual deles apresenta o melhor desempenho em termos de tempo de execução.

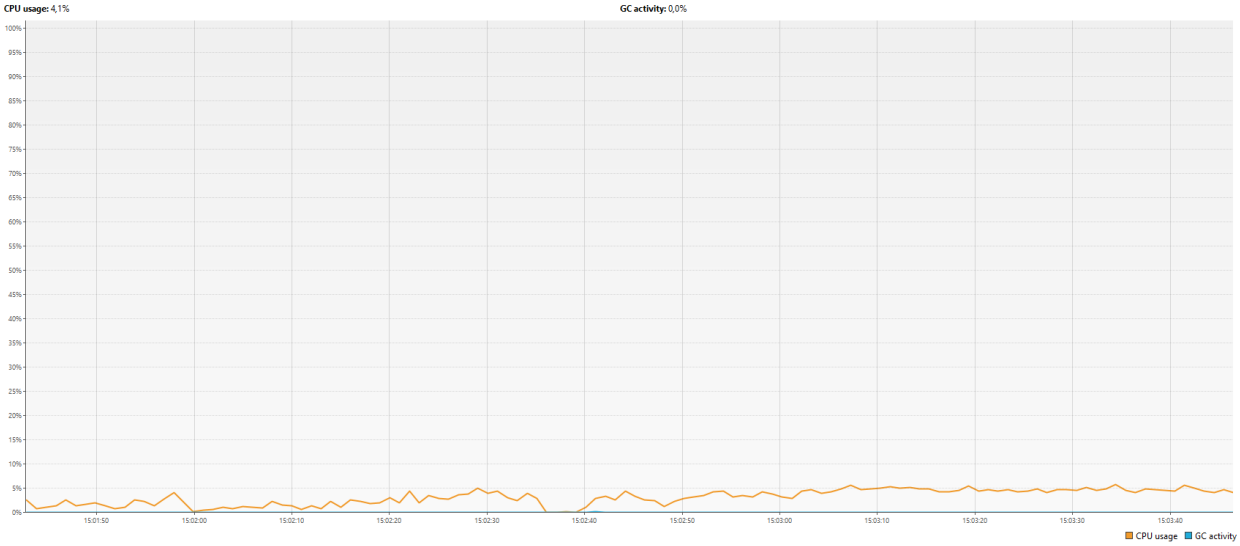
Descrição geral do ambiente de testes

O teste foram realizados em um desktop ou computador de mesa com o um AMD Ryzen 7 5700G com clock de 3.8GHz e com 8 núcleos CPU e 16 threads, memória RAM de 32GBytes (2 x 16GBytes) com 3600MHz e sistema operacional Windows 11 Pro de 64Bits versão 22H2.

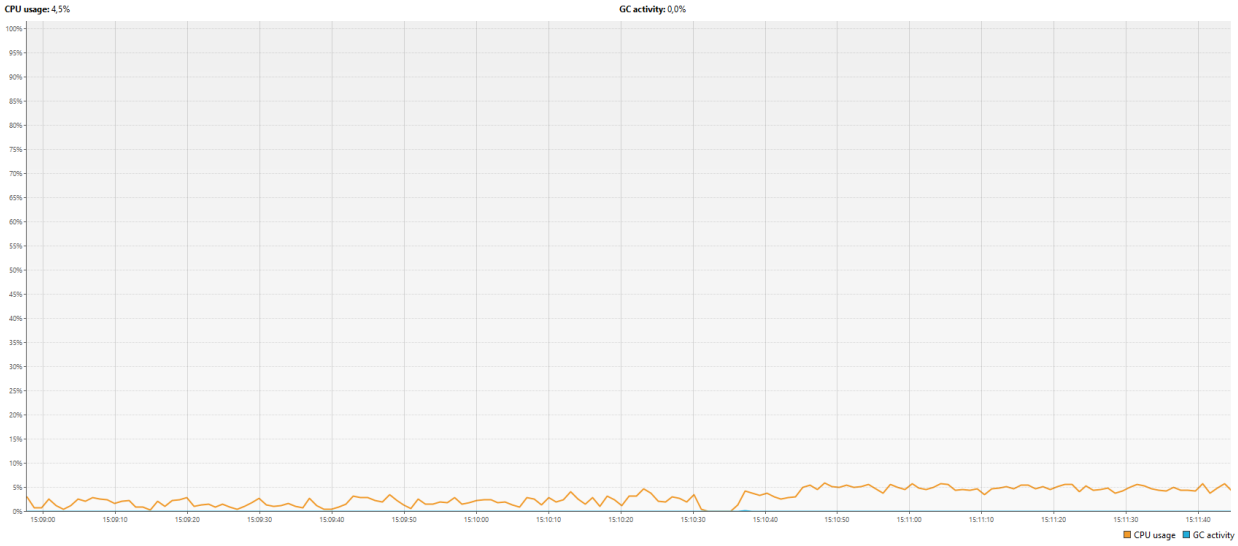
3. Resultados e Análise

Gráficos CPU

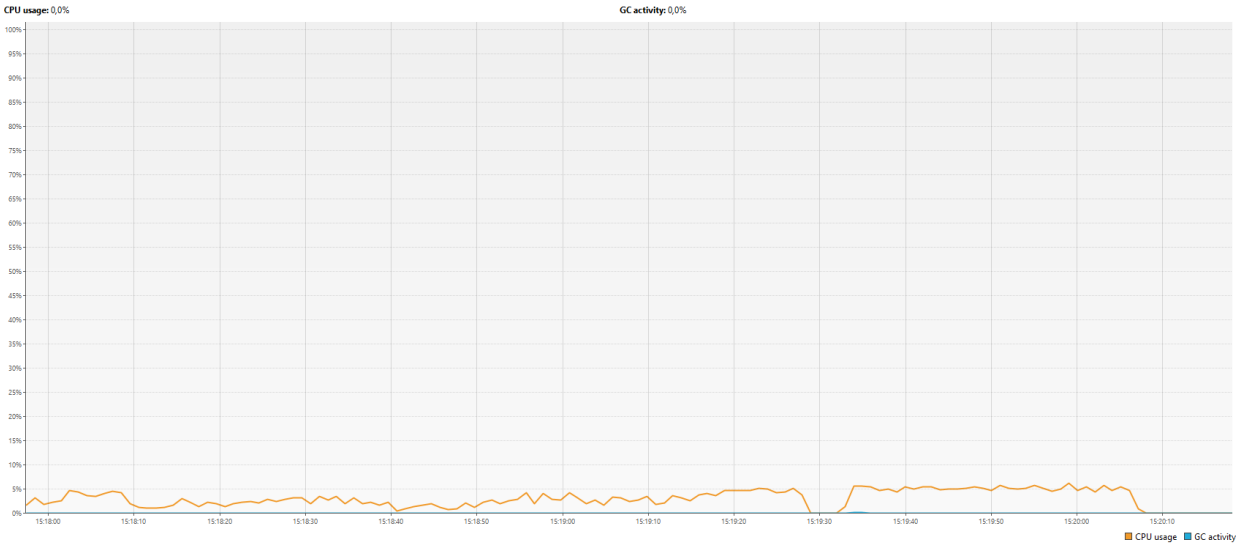
PRIMEIRA ORDENAÇÃO



SEGUNDAORDENAÇÃO

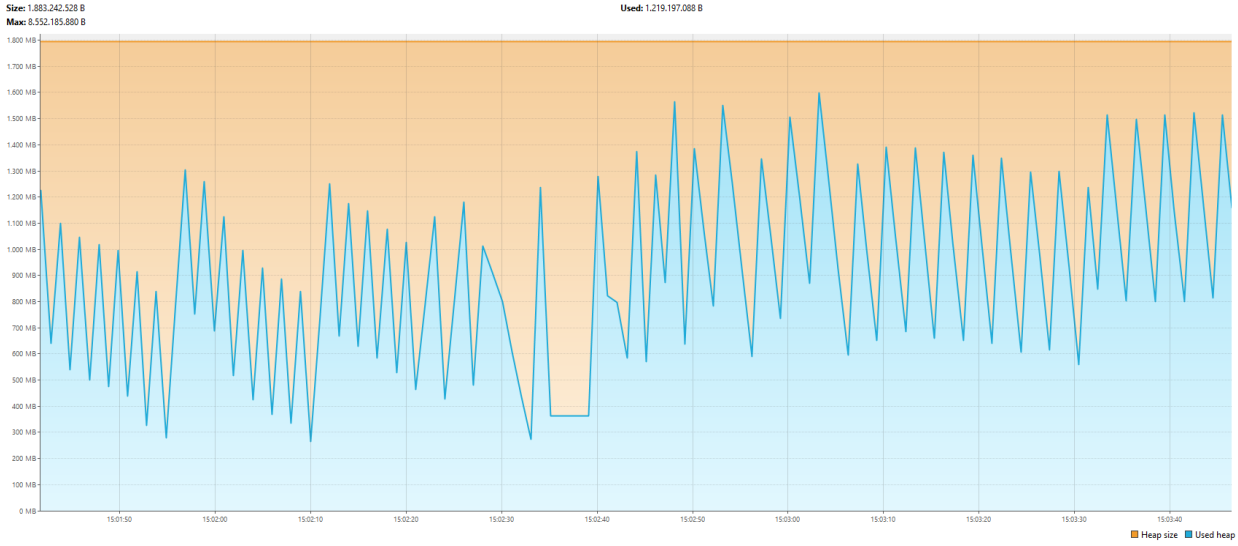


TERCEIRA ORDENAÇÃO

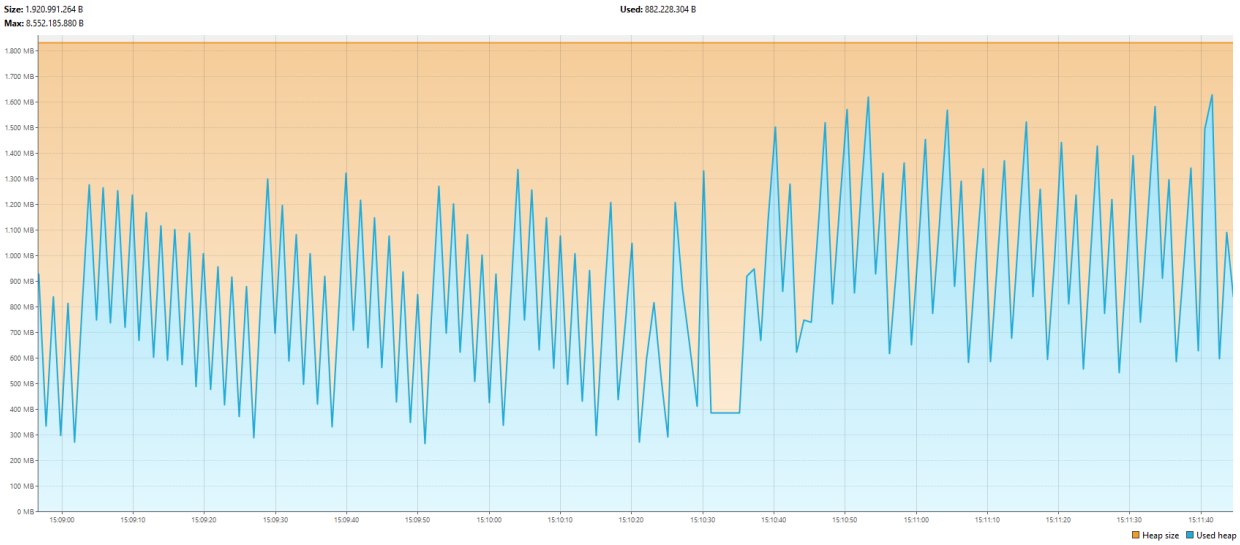


Gráficos Memória

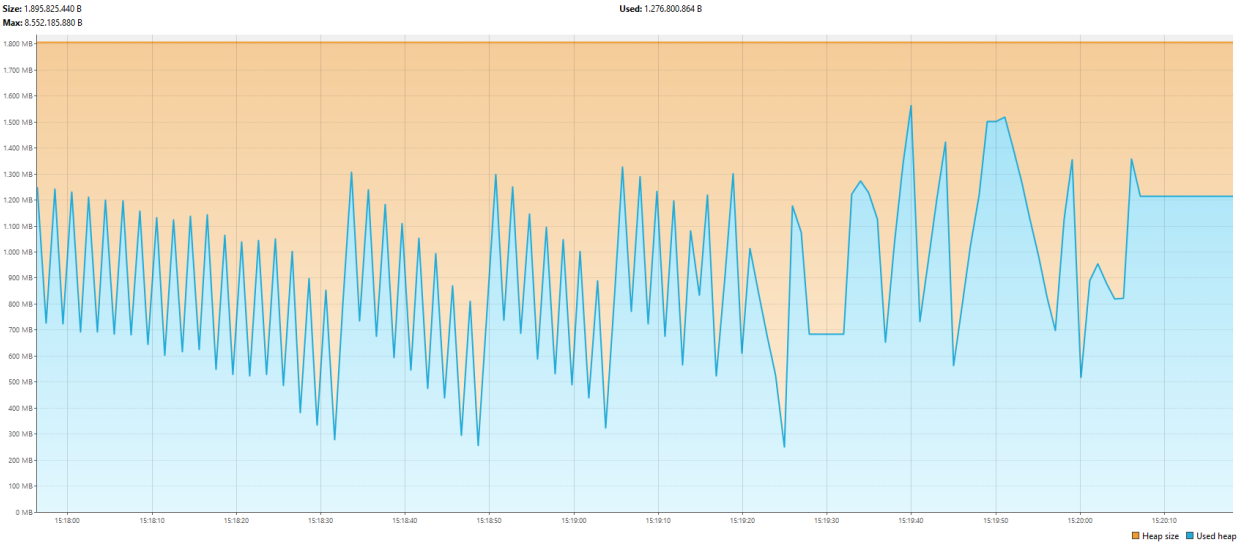
PRIMEIRA ORDENAÇÃO



SEGUNDAORDENAÇÃO



TERCEIRA ORDENAÇÃO



Com base nos dados coletados, podemos concluir que o algoritmo Selection Sort apresentou o pior desempenho em todos os tipos de ordenação e em todos os casos testados. Por outro lado, o Merge Sort mostrou ser o melhor algoritmo em termos de

tempo de execução, conseguindo ter um bom desempenho em todos os tipos de ordenação e casos de teste.

Em relação à ordenação por Duração (Duration) e data (Date), o Counting Sort apresentou o melhor desempenho dentre todos os algoritmos testados. No entanto, é importante salientar que não foi possível executar esse algoritmo na ordenação por estação (Station), o que nos impede de afirmar com certeza que ele seria o melhor nesse tipo de ordenação.

Sugere-se, para futuras análises, a realização de mais testes e experimentos com diferentes tipos de dados e tamanhos de entrada, a fim de validar os resultados obtidos e identificar possíveis limitações dos algoritmos testados. Além disso, seria interessante explorar outras métricas de desempenho, como consumo de memória e utilização de CPU, para uma avaliação mais completa do desempenho dos algoritmos.