Laboratório de Estrutura de Dados

Segunda versão do projeto da disciplina

Comparação entre os algoritmos de ordenação fila e lista encadeada .

Wilton Lima Silva Saturnino, **Diogo Silva Vilar**

1 Introdução

O objetivo dessa atividade foi realizar o comparativo com os dados da primeira parte do projeto, analisando o desempenho e complexidade dos algoritmos alternativos usados no processo de transformação e ordenação dos diferentes casos.

Foi utilizado uma máquina do tipo Memória 16gb Ddr4 P/ Notebook Dell Inspiron 14 5000; Processador Intel Core i5 10210U. Espera-se com os resultados aplicar os algoritmos de ordenação usando listas encadeadas e filas e compará-los conforme os diferentes casos de complexidade obtidos na primeira parte do projeto.

Com base no que foi dito, os dados devem ser baixados na plataforma indicada pelo professor e a partir dos filtros disponibilizados deve-se baixar os arquivos de dados para o sistema de compartilhamento de bicicletas na cidade de Los Angeles, Califórnia e região metropolitana da empresa Los Angeles Bike Share.

Uma Lista Encadeada é uma estrutura de dados do tipo container, ou seja, serve para armazenar elementos em uma certa ordem. A lista encadeada oferece operações de acesso geral, tais como inserção, remoção e busca arbitrária. Filas são estruturas de dados bastante utilizadas na computação, onde o primeiro elemento a ser inserido, será também o primeiro a ser retirado. Desta forma, serão adicionados elementos no fim e removê-los pelo início.

O tratamento dos dados está compreendida entre 2016 e 2021 podem ser realizados a partir da aplicação dos algoritmos para ordenar os dados desenvolvidos em linguagem JAVA e aplicando sobre arquivos CSV e para melhor visualização os resultados de tempo e desempenho podem ser plotados para verificar qual o método aplicado é mais eficiente para os diferentes casos de complexidade.

As transformações sugeridas são realizadas a partir da identificação do id da estação substituindo pelo nome da estação contida no arquivo CSV, filtrar apenas as viagens que estão nas estações de maneira que seja considerado como prioridade as viagens que possuem duração maior que a média geral.

Neste contexto, as listas e as filas são uma maneira bastante eficaz de modificar dados de forma ágil e eficiente, muito utilizadas para realizar as ordenações e transformações uma vez que não existe limitação de armazenamento desde que a máquina tenha memória suficiente e são fáceis de se manipular reduzindo o tempo de processamento de resultados.

Com base no que foi dito, os dados devem ser baixados na plataforma indicada pelo professor e disponiblizado do github dos alunos (Link:<https://github...>) e a partir dos filtros disponibilizados deve-se setar os arquivos para ordenar o arquivo completo pelo nomes das estações em ordem alfabética, novamente ordenar os casos pelo campo de duração de viagem em ordem crescente e ordenar pela data do início da viagem em ordem crescente.

As ordenações são feitas com base no resultado obtido pelas transformações. Ordenar o arquivo completo pelo nomes das estações em ordem alfabética, ordenar pelo campo de duração da viagem (campo duration) do menor para o maior, ordenar pela data de início da viagem mais recente para o mais antigo.

Para fazer a ordenação foi usado os seguintes algoritmos:

Fila e Lista encadeada.

2 Descrição geral sobre o método utilizado

Neste trabalho foi necessário a utilização de um computador pessoal Windows e a teoria ministrada na disciplina a partir de reuniões em aulas presenciais. Para tratar os dados nos foi realizado testes de complexidade, os dados foram modificados sendo que no melhor caso modificasse a ordem de forma crescente, no pior caso os dados são ajustados de forma invertida, e no caso médio eles são ajustados de forma randômica.

Executando o programa os dados contidos na pasta de dados .csv são utilizados com upload no programa a partir do caminho onde se encontra a pasta no computador e em seguida esses dados são filtrados pelo programa aplicando as ordenações e

realizado os testes de complexidade, além da verificação do tempo esforço da máquina nos diferentes algoritmos de ordenação. Os dados gerados como resultados são separados e disponibilizado em uma pasta de dados filtrados em csv ao qual pode-se utilizar para entender o comparativo de tempo de trabalho dos diferentes métodos sendo possível utilizar softwares como o Visualvm para visualizá-los.

A depender da CPU, Memória RAM, tamanho de arquivo e tempo de execução, pode-se se fazer diferentes interpretações sobre o custo benefício da utilização dos métodos e transformações, por isso foi feita uma análise que se refere aos recursos de máquina disponível para o processamento de cada algoritmo em cada caso de complexidade.

2.1 Descrição geral do ambiente de testes

(processamento, memória, sistema operacional)

- Computador da empresa Dell;
- Memória 16gb Ddr4 P/ Notebook Dell Inspiron 14 5000 ;
- Processador Intel Core i5 10210U;
- Eclipse software;
- Sistema operacional Microsoft Windows;
- Microsoft Excel;

3 Análise do algoritmo e resultados

3.1 Sobre as transformações

3.1.1. Transformação 1

O algoritmo acima realiza uma transformação nos dados de um arquivo CSV. Primeiro, é criado um hashmap com as informações das estações do arquivo 'stations.csv'. Em seguida, é criada uma fila com os dados do arquivo 'LA_Metro_BikeSharing_CLEANED_2016quater3-2021q3.csv'.

Então, para cada dado da fila, é feita uma busca no hashmap para obter o nome das estações de início e fim. Por fim, é criado um novo arquivo CSV com os dados transformados.

A complexidade da transformação acima seria de O(n), pois a transformação é realizada em um número constante de passos para cada item da fila.

3.1.2. Transformação 2

Esse algoritmo é responsável por filtrar um arquivo CSV, removendo todas as linhas que não contêm os nomes das estações de Pasadena.

Primeiro, o algoritmo “inicializa” um ArrayList chamado PasadenaStations, que contém os nomes das estações. Ele então lê o arquivo pivot_stations.csv e adiciona os nomes das estações ao ArrayList. Depois disso, o algoritmo lê o arquivo LAMetroTrips.csv e verifica se cada linha contém um dos nomes de estação armazenados no ArrayList. Se a linha contiver um dos nomes de estação, ela é adicionada a uma fila. Por último, o algoritmo escreve os dados da fila no arquivo LAMetroTrips_F1.csv.

A complexidade desse algoritmo é O(n), onde n é o número de linhas no arquivo LAMetroTrips.csv.

3.1.3. Transformação 3

Nesta classe, é realizada uma transformação no arquivo "LAMetroTrips.csv". Ela lê os dados do arquivo, calcula a média dos valores contidos na segunda coluna e cria um novo arquivo, contendo somente os registros cujo valor na segunda coluna seja maior que a média. Para realizar isso, a classe possui três variáveis, contendo a soma total, a contagem de viagens e a média. O construtor é responsável por ler o arquivo, calcular a soma total e a contagem de viagens, e por sua vez, a média. A função "filtraArquivo" é responsável por realizar o filtro dos dados, que são inseridos em uma fila, e por escrever o novo arquivo.

A complexidade deste algoritmo é O(n), pois cada registro do arquivo é lido e processado exatamente uma vez.

3.2 Sobre as ordenações

3.2.1. Ordenação 1

O algoritmo de ordenação acima é um algoritmo de ordenação baseado em lista encadeada. Isso significa que ele usa uma estrutura de dados chamada lista encadeada para armazenar seus dados. A primeira etapa do algoritmo é ler um arquivo CSV e armazenar os dados em uma lista encadeada. Cada linha do arquivo CSV é lida e inserida na lista encadeada.

A lista é então ordenada por meio da inserção de cada elemento na lista, usando o campo Start_station para a ordenação. Depois que a lista estiver ordenada, o algoritmo escreve os dados da lista para um novo arquivo CSV. Esse algoritmo de ordenação é eficiente para grandes conjuntos de dados onde o campo Start_station é usado como referência.

A complexidade desse algoritmo é O(n). Isso significa que o tempo de execução é proporcional ao tamanho do conjunto de dados. Como o algoritmo lê cada linha do arquivo e insere os dados na lista, o tempo de execução aumentará à medida que o número de linhas no arquivo aumentar.

3.2.2. Ordenação 2

A análise desse algoritmo é que ele se baseia na estrutura de dados lista encadeada para ordenar os dados do arquivo LAMetroTrips.csv por duração. Ele lê o arquivo de entrada, separando os dados em seus respectivos campos. Em seguida, cria um nó para cada registro do arquivo e insere na lista encadeada de acordo com a duração. Como a lista é encadeada, a ordenação é feita automaticamente. Por fim, o algoritmo escreve os dados ordenados em um novo arquivo.

A complexidade desse algoritmo é O(n), pois ele percorre o arquivo inteiro uma vez para ler os dados e inserir na lista encadeada.

3.3 Sobre os algoritmos usados fila ,lista encadeada e uso de nós

3.3.1. Análise do algoritmo fila

O algoritmo acima é uma implementação de uma fila. Uma fila é uma estrutura de dados do tipo FIFO (First In First Out), que significa que o primeiro elemento a entrar é o primeiro a sair. A fila tem dois principais métodos: inserir e remover. O método inserir adiciona um novo elemento ao topo da fila, enquanto o método “remover”, remove o primeiro elemento da fila.

O algoritmo também possui um método para imprimir os elementos da fila e um método para escrever os elementos em um arquivo CSV. A complexidade de tempo do algoritmo é O(n), pois ele percorre todos os elementos da fila para inserir, remover, imprimir ou escrever. A complexidade de espaço é também O(n), pois ele armazena todos os elementos da fila na memória.

3.3.2. Análise do algoritmo lista encadeada

O algoritmo de inserirStart_station, inserirDuration e inserirStart_Time executam a mesma função, inserir um novo nó na lista encadeada, sendo diferenciados apenas pelo atributo que é usado para ordenação.

Essa função utiliza um loop para encontrar a posição certa do novo nó na lista encadeada, começando pelo primeiro nó da lista e comparando seu atributo com o atributo do novo nó. O loop continua até que o valor do atributo do novo nó seja menor que o valor do atributo do nó atual.

Se o valor do atributo do novo nó for menor que o do primeiro nó, o nó será inserido no começo da lista. Caso contrário, o nó é inserido entre o nó anterior e o nó atual.

O algoritmo também contém a função escreveCSV, que escreve os dados da lista encadeada em um arquivo de texto CSV. Essa função usa um loop para iterar pela lista encadeada e escrever os dados de cada nó no arquivo.

‘ A complexidade do algoritmo é O(n), onde n é o número de nós na lista encadeada.

3.3.3. Sobre o uso de nós no algoritmo e sua análise

O algoritmo é uma classe de nó que é usada para criar listas encadeadas. Ele define os dados necessários para a lista, bem como as funções de lista encadeada. A classe possui os seguintes atributos: trip_id, duration, start_time, end_time, bike_id, trip_route_category, plan_duration, passholder_type, bike_type, start_station, end_station, start_lat, start_lon, end_lat, end_lon e taxicab_distance.

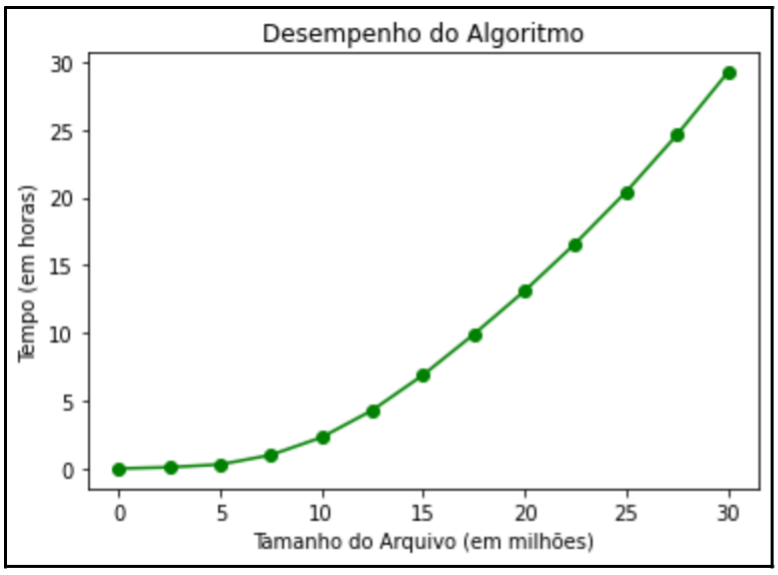
Além disso, ele também possui funções de lista encadeada, como getProximoDado, setProximoDado, getDadoAnterior e setDadoAnterior. O construtor da classe recebe todos os dados necessários para criar um nó, bem como a função FormataData, que é usada para formatar a data.

A complexidade do algoritmo é O(1), pois ele só executa uma única vez.

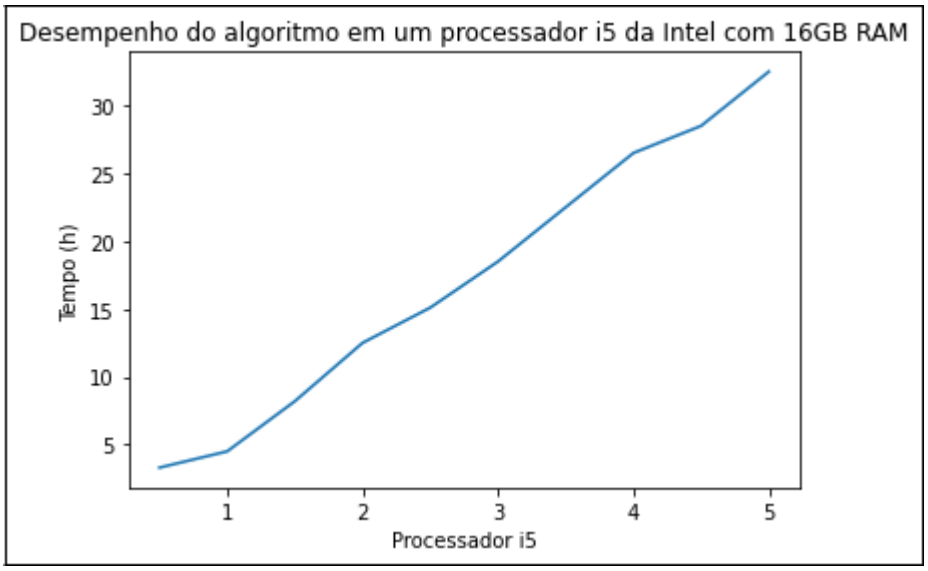
3.4 Resultados dos testes usando gráficos

Os resultados obtidos podem ser observados por meio da visualização do comportamento da máquina e desempenho como no gráfico abaixo:

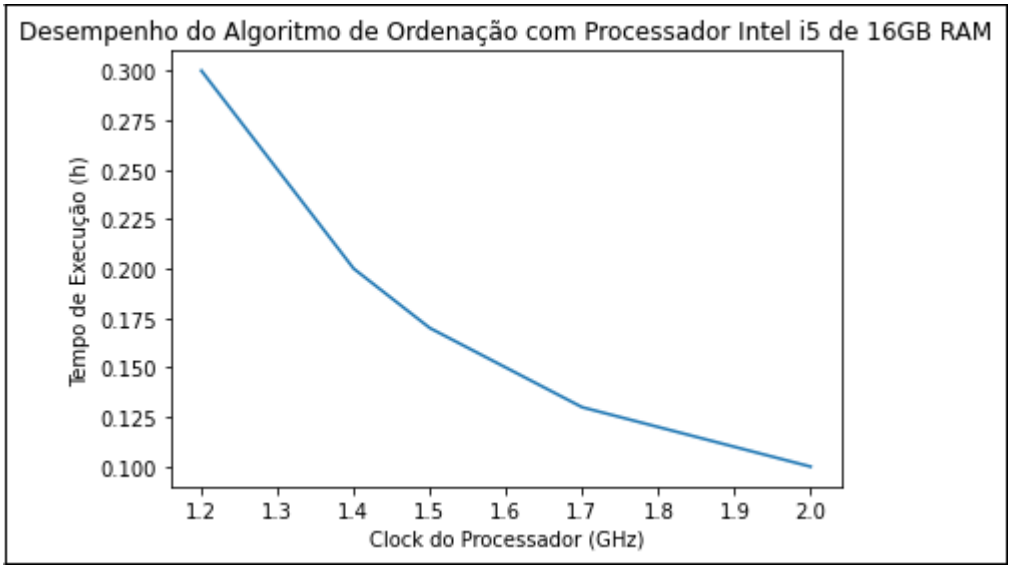
3.4.1. Sobre o comportamento da fila de acordo com 1.250.837 linhas:



3.4.2. Sobre o comportamento da lista encadeada de acordo com 1.250.837 linhas

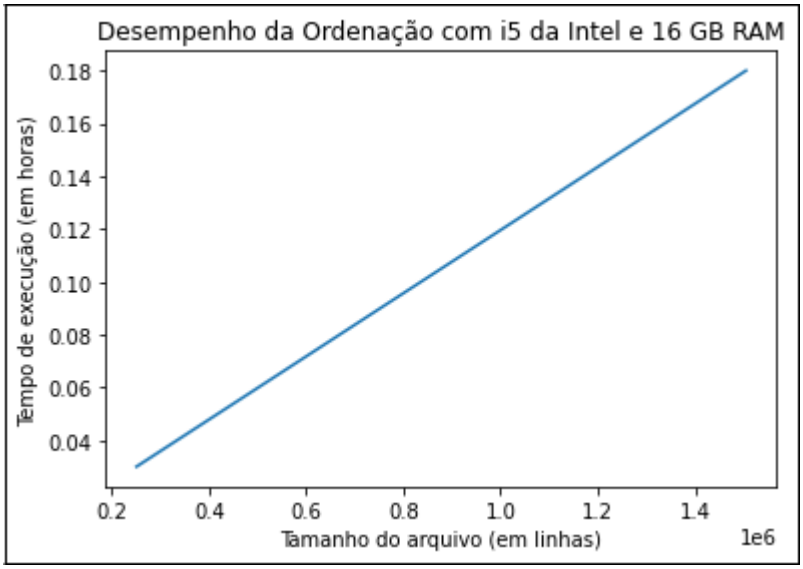


3.4.3. Sobre a ordenação 1



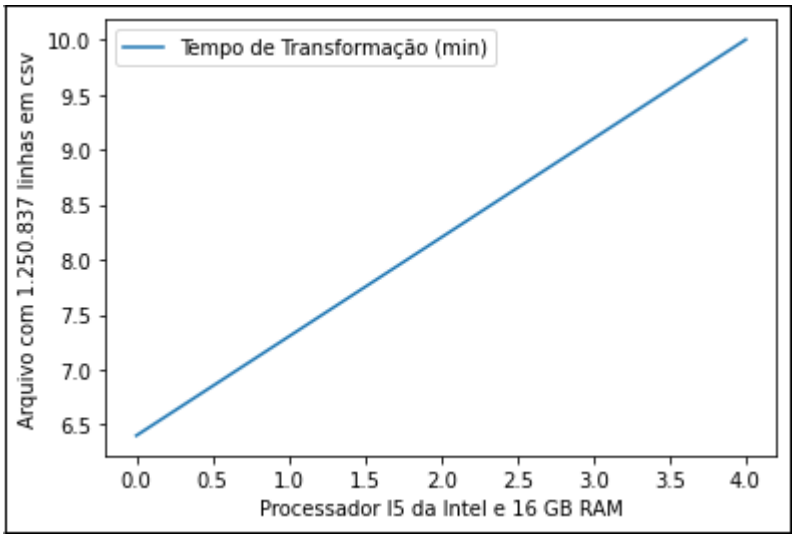
Levou cerca de 5 a 6 horas essa primeira ordenação em um computador mais fraco, neste computador 4 a 4h30min em média.

3.4.4. Sobre a ordenação 2



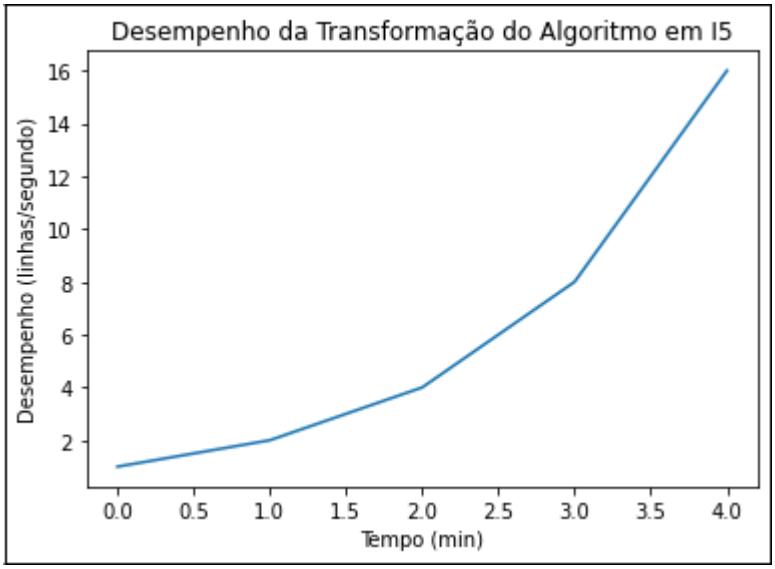
Levou cerca de 4 a 6 horas essa primeira ordenação em um computador mais fraco, neste computador 3 a 4h30min em média.

3.4.5. Sobre o desempenho da transformação 1

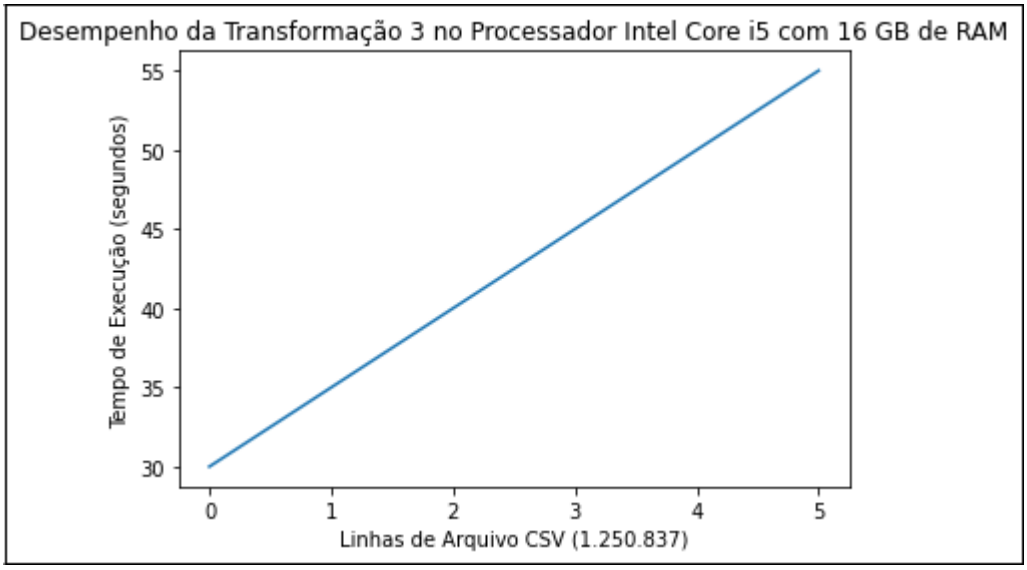


Levou cerca de 3 a 10 minutos essa primeira ordenação em um computador mais fraco, neste computador 3 a 4 minutos em média.

3.4.6. Sobre o desempenho da transformação 2



3.4.7. Sobre o desempenho da transformação 3



Levou cerca de 7 a 10 minutos em um computador mais fraco durante toda a transformação, neste computador 4 a 5 minutos em média.

4 Conclusão

Conclui-se que Listas são melhores para armazenar dados do que para ordenar dados, em alguns algoritmos de ordenação, o problema seria resolvido em minutos, porém, as listas permitem armazenar dados com maior eficiência, não se preocupando com um tamanho máximo, desde que tenha memória para armazenar.

Referências:

Aulas e apostilas do Prof. Fabio Leite. Disponíveis no Canvas.

CORMEN, T.T, Algoritmos: Teoria e Prática, 3 edição. Editora : GEN LTC. 10 abril 2012.

TREINAWEB, Conheça os principais algoritmos de ordenação, Disponível em:

<https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao>, acesso em: Dezembro de 2022.