Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Alunos:

Dandara Gabrielle Montenegro Gonçalves.
João Victor Marinho Souza
Thiago Felipe Viana Diniz

1. Introdução

Com os estudos em sala de aula na cadeira de Estrutura de Dados juntamente com Laboratório de Estrutura de Dados sobre algoritmos de ordenação, foi apresentada pelo professor Fábio Leite a responsabilidade da criação e desenvolvimento de um projeto que visasse a análise da aplicação de oito algoritmos de ordenação (CountingSort, MargeSort, QSortMediana3, HeapSort, QuickSort, InsertionSort, SelectionSort e BubbleSort) e assim podermos ver como cada um se sai no seu pior, médio e melhor caso.

O arquivo que iremos usar é o “LA_Metro_BikeSharing_CLEANED_2016quater3-2021q3.csv” do Lost Angeles Metro Bike Share, aplicando os códigos de ordenação de Strings e Inteiro, com isso, de acordo com as exigências do projeto, criamos o relatório que tem como objetivo especificar como o nosso projeto funciona, como foi feito e quais resultados obtivemos. Com base nessa breve introdução, para que nosso projeto tenha pleno funcionamento e rode em vossa máquina, é necessário que você tenha acesso a página do GitHub onde o código do projeto está hospedado. Na página inicial existirão duas pastas principais: “src”, onde estarão todos os arquivos pertencentes ao funcionamento do projeto na linguagem JAVA e o “README.md”, que conterá a explicação de como o código virá a funcionar.

Diante a tais informações, ao final do funcionamento do nosso arquivo, o usuário virá a perceber que alguns algoritmos se sairão melhores que outros e tudo isso estará relatado no final do relatório com os levantamentos que fizemos em relação a alguns pontos específicos selecionados pelo professor da disciplina.

2. Descrição geral sobre o método utilizado

De forma resumida, baixamos a tabela cedida pelo professor do Lost Angeles Metro Bike Share do tipo.CSV onde no nosso Main ela é lida e interpretada, e cada uma das suas linhas é passada para o construtor da classe App, que nada mais é do que uma classe que transforma os dados da linha em atributos de classe que iremos manipular mais a frente. Em seguida, o que fazemos é passar uma matriz, onde cada elemento é um objeto instanciado da classe App que contém os dados que precisamos. Com os dados manipulados de forma que podemos utilizar no nosso código agora basta que passemos essa classe de Appe o parâmetro que desejamos ordenar, sendo eles : Station Name, Duration e Start Time, com isso cada um dos códigos de ordenação irá ordenar os dados e

devolver os mesmos de forma ordenada, que por fim serão passados para a classe Executa Csv.

Na classe ‘ExecutaCsv’ no ‘VerificarExistencia’ é formada o nome do arquivo para a busca, definida o nome da variável, verificado se o arquivo existe ou não, é definido um if para deletar o arquivo caso ele já exista para que possa ser criado um arquivo vazio para por fim ser retornado o caminho desse mesmo arquivo. Logo é inicializado o objeto arquivo como caminho, inicializa o objeto ‘fw’ para inicializar o construtor ‘bw’, este mesmo construtor recebe como argumento o objeto do tipo FileWriter e escreve o conteúdo neste arquivo; com a adição da quebra linha os recursos são fechados em seguida.

Ao inicializar a variável ‘linhas’ para formar as linhas do arquivo no método ‘FiltrarCSV’, um for é utilizado para adicionar o nome das colunas no arquivo e verificar se é ou não o último elemento, logo é concatenado cada variável com uma vírgula as separando. Com um else é concatenado o último elemento da linha. Em seguida é chamada a função ‘VerificarExistencia’ para criar o arquivo específico e caso já exista, este é apagado; a função Escrever é chamada para escrever no arquivo especificado o que está contido na variável ‘linhas’.

No arquivo App, a classe App define as variáveis com os nomes dos arquivos, cria matrizes para receber os dados das linhas e colunas dos arquivos especificados a fim de carregar os arquivos nas matrizes ‘MainArchive’ e ‘StatationsID’. Em seguida ocorre a primeira transformação, em um for comparamos os dados da coluna ‘Stations_id’ do arquivo CSV com o arquivo ‘LA_Metro_Bike_Share’, logo chamamos os metodo ‘AdicionaVirgula’ para concatenar os elementos das colunas linha a linha para enviar os nome do arquivo a ser gerado.

A segunda transformação cria a variavel para receber a posição da linha que contém viagens de Pasadena, contamos a posição das linhas da lista para controle, com isso, usamos novamente um for para comparação da coluna ‘stat_station’ e ‘end_station’ com as estações em ‘stations.csv’ verificando assim se são de Pasadena ou não. Para finalizar a segunda transformação chamamos a função ‘FiltrarCSV’ contida no arquivo ‘Executa.CSV’ para concatenar os elementos com vírgula os separando.

Na terceira e última transformação, calculamos a média da duração dos passeios contidos em ‘LAMetroTrips.csv’, em um for somamos todos os valores da coluna duration, em seguida dividimos pelo total de elementos para obtermos a média geral de tempo dos passeios. Com um contador zeramos a variável ‘listaPosicao’ e o contador, em outro for são filtrados os passeios com maior tempo que a média geral e no if verificamos se os valores são maiores que a média; chamamos a função ‘FiltrarCSV’ contida no ‘Executa.csv’ para concatenar os elementos com vírgula por separação. No ‘AdicionaVirgula’, a variável que vai conter a formar cada linha do arquivo é inicializada, em um for concatenamos o último elemento da linha para por fim chamarmos a função contida em ‘ExecutarCsv’ para criar o arquivo especificado e caso já exista, ele o apaga. É chamada a função contida no ‘ExecutaCsv’, Escrever para escrever no arquivo especificado o que está contido na variável linhas.

Descrição geral do ambiente de testes

As configurações da máquina que usamos para criação do projeto e submete-los aos testes foram:

Processador: Intel i3 - Décima geração;

8Gb de RAM - 2666HZ;

SSD;

Radeon RX 550;

Sistema Operacional: Windows 10.

E para o desenvolvimento, o ambiente que usamos foi:

VSCode (Visual Studio Code).

3. Resultados e Análise

Elaborar os resultados dos testes usando tabelas e gráficos