Laboratório de Estrutura de Dados

Projeto de Comparação entre os algoritmos de ordenação elementar

Universidade Estadual da Paraíba – UEPB

Nome do Aluno - Luanderlandy Fellipe da Silva

Matricula – 191080527

Curso – Ciência da Computação

Professor - Fabio Luiz Leite Junior

Período Letivo - 2020.2

Sumário

1. Introdução

Este relatório corresponde a descrição e amostragem dos resultados obtidos no Projeto 1 da disciplina de Laboratório de Estrutura de Dados, que teve como objetivo aplicar e compreender as técnicas de análise e ordenação dos algoritmos que foram estudados ao longo da disciplina.

1.1 Objetivos

O projeto consiste em organizar uma planilha de dados sobre os casos da Covid-19 no Brasil por meio dos seus números de casos, óbitos e nome das cidades. Os algoritmos de ordenação utilizados para esses 3 campos de dados foram: Selection Sort, Insertion Sort, Counting Sort, Merge Sort, Quick Sort, Quick Sort por Mediana de 3, e Heap Sort.

As ordenações foram feitas considerando 3 arquivos de dados:

- **Médio Caso:** O arquivo original, fornecido para o programa (caso.csv) com seus dados não ordenados.
- **Pior Caso:** O arquivo caso.csv com suas linhas invertidas (do fim para o começo.)
- **Melhor Caso:** O arquivo que já passou por uma ordenação previa dos dados.

O programa também registra o tempo de execução de cada um dos métodos de ordenação e armazena em uma tabela para ser feito uma comparação desses valores.

1.2 Resumo dos Resultados

Por meio dos resultados obtidos no projeto, conseguimos notar diferenças nas velocidades de execução dos algoritmos bem significante, mostrando que alguns são melhores e mais rápidos quando se trata de grandes quantidades de dados, como também comparar as diferenças entre as ordenações quando feitas entre o médio, pior e melhor caso.

2. Descrição geral sobre o método utilizado

2.1 Descrição geral do ambiente de testes

- Processador: Intel core i5-10210U 1.60GHz
- Memória RAM: 8Gb
- Armazenamento: SSD KINGSTON 512Gb
- SO: Windows 10 Pro (64 bits)

2.2 Descrição das ferramentas/classes

O programa foi dividido entre 5 classes principais:

- **AlgoritmosOrdenacao.java:** Classe contendo todos os métodos de ordenação estudados e criados nas aulas, composta por seus métodos auxiliares de Arrays e Strings, que tem a principal função de aplicar todos os métodos de ordenação nos dados de casos confirmados, óbitos e nome das cidades.
- **ContadorExecucao.java:** Objeto contador do tempo de execução dos algoritmos, que funciona como um timer do inicio ao fim de cada algoritmo de ordenação e que registra em uma tabela (no console e em um arquivo.txt) os tempos de execuções efetuados pelos algoritmos para serem comparados.
- **SortAndWrite.java:** Classe com finalidade de ser um organizar os métodos de ordenação para a classe que chamará eles (CovidDataControler) utilizar com facilidade. Ela é encarregada de ordenar os valores e escrever os arquivos.csv dos casos, óbitos e cidades.
- **CovidDataControler.java:** Classe que trata de receber os dados do arquivo.csv, armazenar nos seus respectivos arrays de valores (casos, óbitos, cidades, informacoesGerais), executando os métodos de ordenação e criando

os arquivos.csv que servirão de base na ordenação dos outros métodos (Pior e Melhor Caso).

- **Main.java:** Classe Principal (contendo o método Main) que inicializa todas as outras classes e chama os algoritmos de ordenação para o Médio Caso (padrão), Pior Caso (Arquivo invertido), Melhor Caso (Arquivo ordenado).

2.3 Passo a passo da Execução:

O programa é executado a partir da classe **Main.java**, com o arquivo .csv (caso.csv) contido na pasta principal do projeto. Os códigos fontes foram armazenados no diretório **Projeto_Covid19/src**, e contêm as 5 classes principais utilizadas. Quando compilado e executado, os arquivos binários (java bytecode) vão pra pasta (criada durante a execução) **Projeto_Covid19/bin** e começa fazendo a ordenação pelo arquivo original, sendo isso o primeiro método (Médio Caso).

O objeto Main envia para **CovidDataControler** que tratará de receber os dados que serão ordenados (pelos métodos em **AlgoritmosOrdenacao**) do arquivo (casos, óbitos e cidades) assim como os outros dados da planilha (outrasInfos) que servirão para fazer a escrita desses valores posteriormente em **SortAndWrite.** Os dados são recebidos e o Main executa a ordenação de todos esses dados e armazena os seus respectivos .csv em **Projeto_Covid19/DadosOrdenados/MedioCaso,** assim exibindo no final das ordenações a tabela de valores do calculo do Medio Caso (obtido pelo **ContadorExecução).**

Após o cálculo e exibição dos tempos de execução do médio caso, é feito o mesmo cálculo para o PiorCaso (o objeto **CovidDataControler** trata de fazer a criação do arquivo invertido para o piorCaso) assim como o mesmo é feito no cálculo do melhorCaso (só que com o arquivo ordenado previamente através do método quickSort).

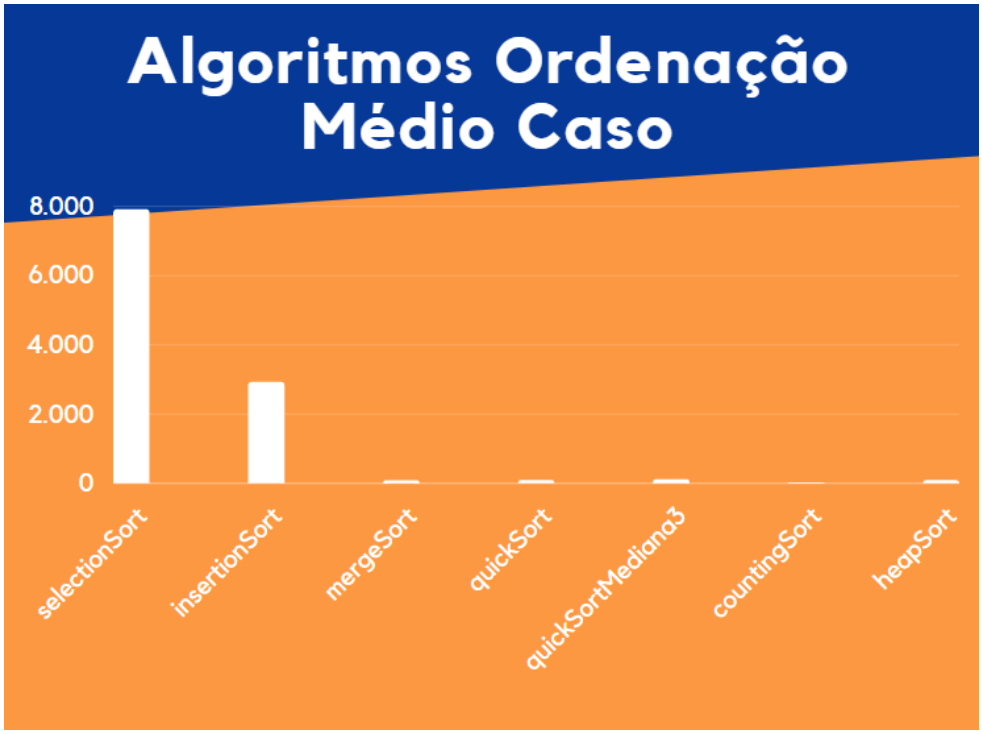
Ao final de cada execução desses 3 métodos, é obtido os valores das 3 tabelas dos tempos de execução e ela é armazenada em uma grande tabela que pode ser encontrada em **Projeto_Covid19/DadosOrdenados/timesExecution.txt.**

3. Resultados e Análise

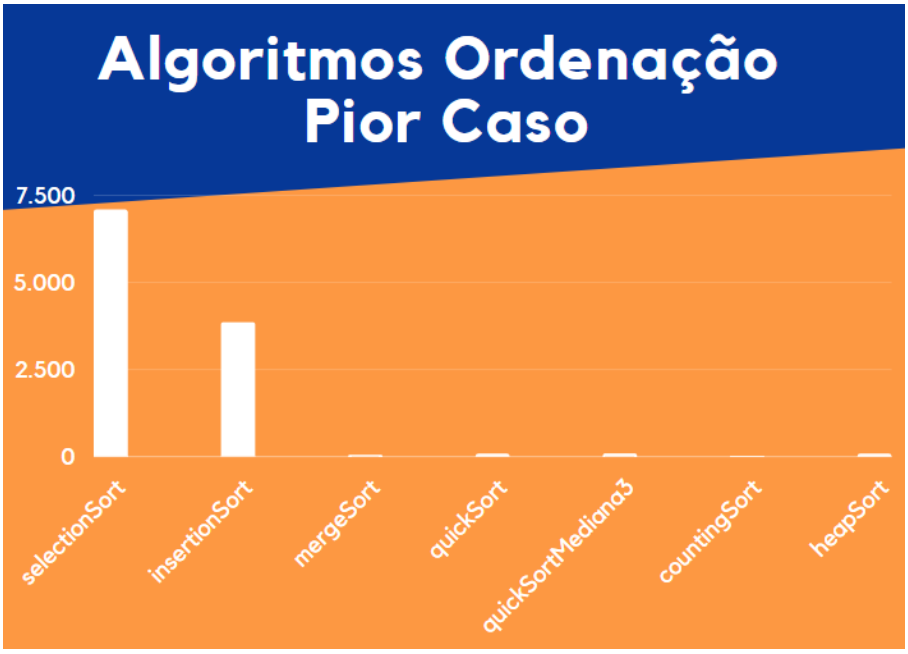
3.1 Gráficos Comparativos

Algoritmos X Tempos de Execução (Millisegundos)

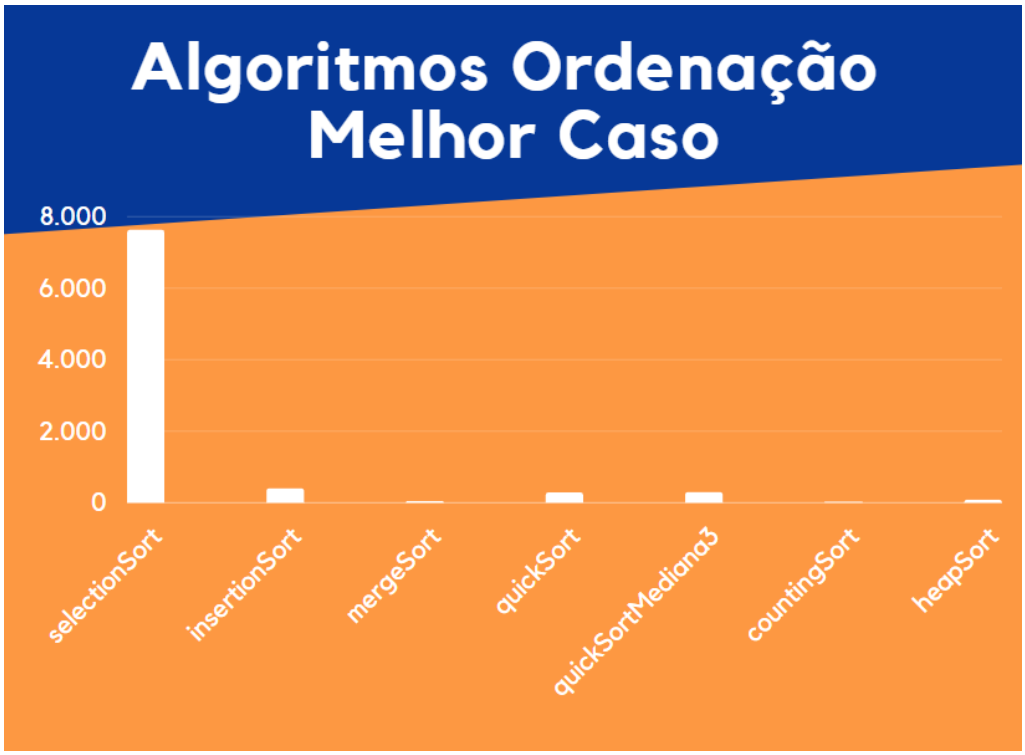
Histograma tempos de execuções (miliseg) dos algoritmos no Médio Caso

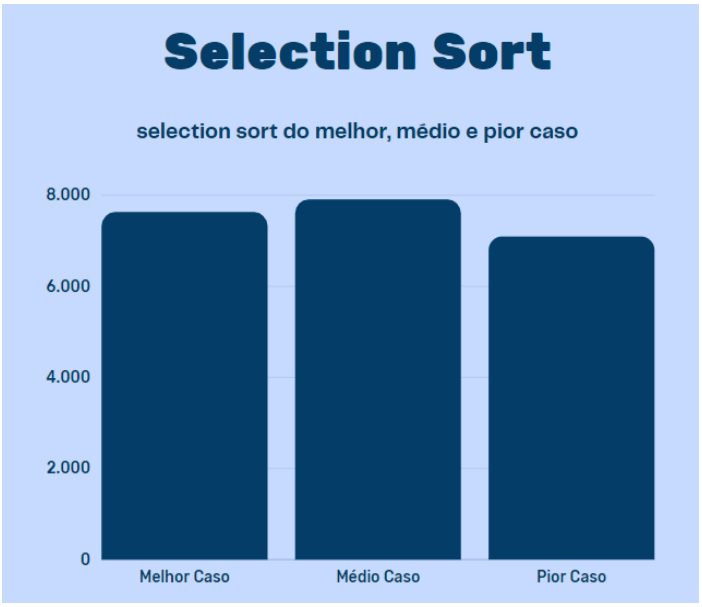


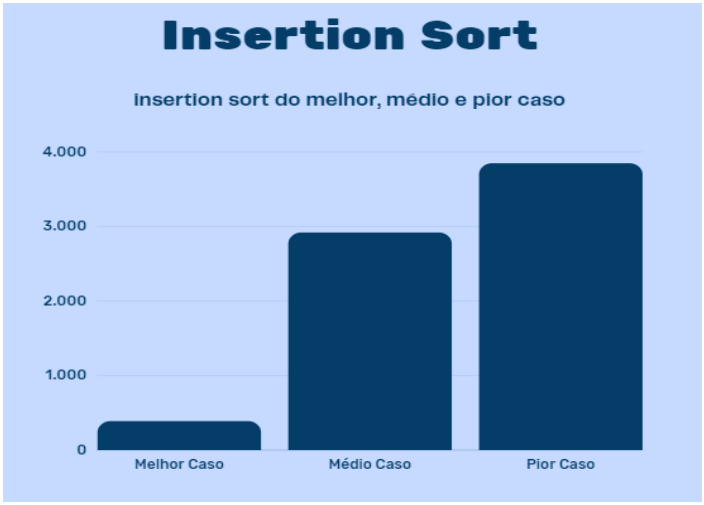
Histograma tempos de execuções (miliseg) dos algoritmos no Pior Caso

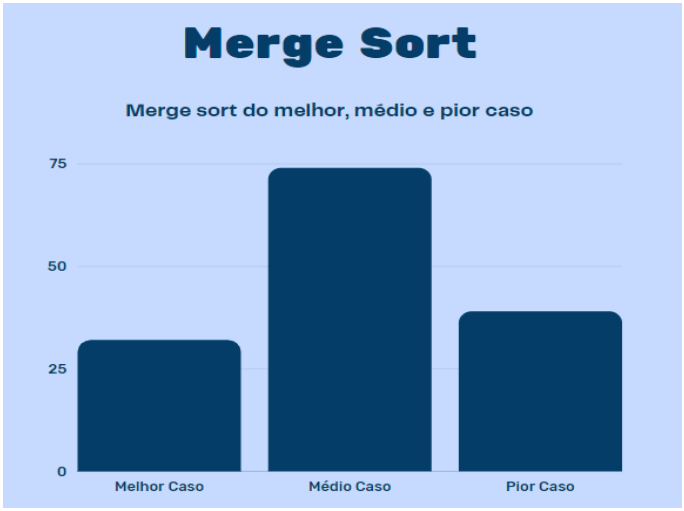


Histograma tempos de execuções (miliseg) dos algoritmos no Melhor Caso

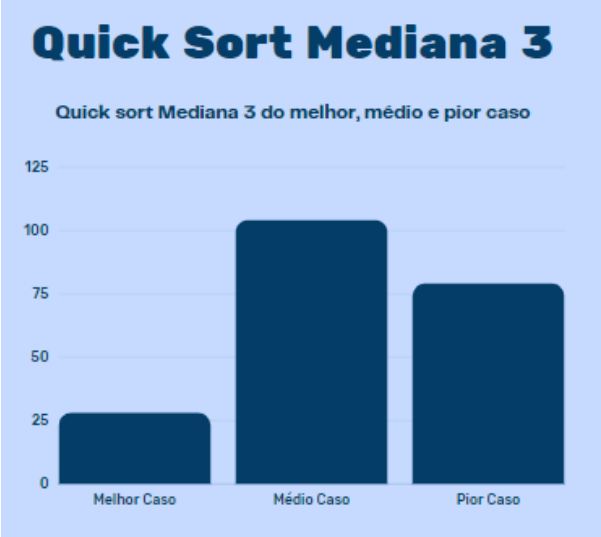


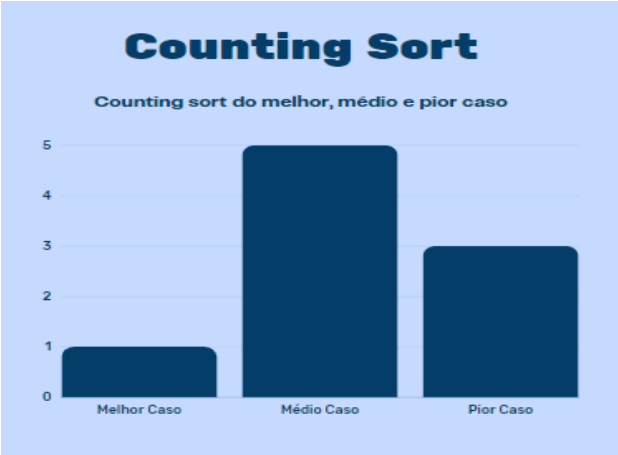

Selection Sort
selection sort do melhor, médio e pior caso
8.000
6.000
4.000
2.000
0
Melhor Caso
Médio Caso
Pior Caso

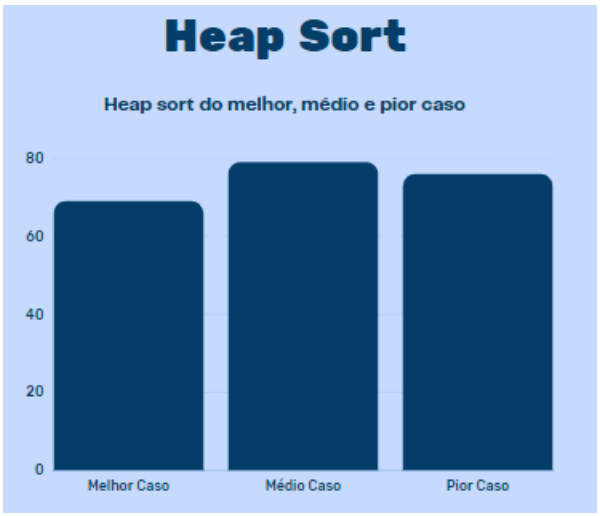

Insertion Sort
insertion sort do melhor, médio e pior caso
4.000
3.000
2.000
1.000
0
Melhor Caso
Médio Caso
Pior Caso


Merge Sort
Merge sort do melhor, médio e pior caso
75
50
25
0
Melhor Caso
Médio Caso
Pior Caso


Quick Sort
Quick sort do melhor, médio e pior caso
300
200
100
0
Melhor Caso
Médio Caso
Pior Caso


Quick Sort Mediana 3
Quick sort Mediana 3 do melhor, médio e pior caso
125
100
75
50
25
0
Melhor Caso
Médio Caso
Pior Caso


Counting Sort
Counting sort do melhor, médio e pior caso
5
4
3
2
1
0
Melhor Caso
Médio Caso
Pior Caso



4. Considerações Finais

Qual é a sua análise geral sobre os resultados?

Com a análise do código, das tabelas e dos histogramas, pode-se perceber que alguns algoritmos funcionaram como esperado, tendo os melhores resultados no Melhor Caso, porém nem sempre isso ocorreu, e isso nós como programador precisamos analisar por que ocorre e o motivo pelo qual esses resultados podem ter sido obtidos dessa maneira.

Quais os algoritmos mais eficientes e por qual motivo?

Algoritmos como Insertion Sort, Selection Sort e Merge Sort obtiveram os resultados mais negativos se comparados com o tempo de execução deles. Isso se dá pela forma como a lógica deles é feita, possuindo as vezes varias iterações em loops que possibilitam sua ordenação porem vimos que não é a melhor maneira de ser ordenado quando se trata de dados muito grandes. Já algoritmos como o Quick Sort, em especial o Quick Sort por mediana de 3 que possui um valor de pivô mais lógico, funcionaram de forma melhor por conta de como a instabilidade e lógica deles é feita, assim como o Heap Sort que possuiu ótimos resultados por se trabalhar com heaps que são uma ótima estrutura de dados para ordenação muito grandes como as que foram feitas.