Primeira versão do projeto da disciplina

Comparação entre os algoritmos: Selection, Insertion, Counting, Merge, quick, quick mediana de 3, heapify;

Alunos**: Simeone Aquino de Holanda, Matheus De Oliveira.**

1. Introdução

Esse relatório é referente ao projeto das disciplinas de Estrutura de dados teórica e laboratório. Com os nossos resultados obtido da execução dos algoritmos em um conjunto de dados grandes, foi possível perceber uma grande vantagem de tempo em alguns deles, como por exemplo os algoritmos que utiliza o método de divisão e conquista junto a recursividade para a ordenar, que foram os que tiveram os melhores resultados em milissegundos, enquanto que os algoritmos elementares tiveram uma performasse extremamente demorada.

2. Descrição Geral sobre o método utilizado

No projeto foi usado uma estratégia de criação de objetos que consistem em dividir os algoritmos exigidos em 3 tipos de objetos sendo eles Linear, Elementares e Recursivos, dessa forma foi criado 3 classes baseados nesses tipos e colocadas na pasta Entities, onde cada classe guarda seus respectivos algoritmos para serem chamados na classe principal nomeada de application.java dentro da pasta programa. Além disso, temos uma classe feita unicamente para trabalhar com os arquivos .csv , na classe GerarCsv(que também está na pasta Entities), é possível notar os métodos readFile e writeInfile que são responsáveis pela leitura e escrita de dados “.csv”.

Sobre o cálculo em milissegundos foi feito da seguinte forma, com o uso do método currentTimeMillis() é possível calcular o período de tempo de execução em partes do código, desse modo fazemos a chamada dele antes e depois do método de ordenação e com uma subtração do tempo final pelo inicial temos o tempo calculado em milissegundos que o determinado algoritmo levou para realizar a operação.

O programa é simples e rápido e tem como base o arquivo covid.csv que é onde é feito a leitura dos dados para se trabalhar as ordenações, todos os resultados são gerados na pasta Results que está na pasta src junto a Entites e programa, podendo ser visualizado todos os datasets ordenado pela coluna de dados requisitados.

Descrição geral do ambiente de teste

Para a execução do projeto é aconselhável o uso do sistema operacional Windows, onde foi feito todos os casos de testes. Sobre o processamento é possível ver na saída do arquivo application.java que o quick sort é um dos que

mais demora quando a ordenação de cidades ocorre chegando a levar em média 10 segundos para ser processado, acreditamos que é devido a sua forma de implementação que não facilidade na ordenação de strings. Abaixo foi feito uma tabela comparativa entre os algoritmos demostrando o pior, médio e melhor caso. Para os casos médios consideramos as próprias colunas de mortes, casos e cidades. Contudo os casos piores foram invertidos o sinal dos algoritmos para ordenaram decrescente e para os melhores casos usamos o csv que os algoritmos de ordenação geraram no final dessa forma calculamos os melhores casos com colunas já ordenadas.

3. Resultados Da análise

TABELA DE COMPARATIVA DO TEMPO DE ORDENAÇÃO DOS ALGORITOMO

Tabela de City:

Tabela de ordenação de cidades							
Casos	**Insertion**	**Selection**	**Couting**	**Merge**	**Quick**	**Mediana de 3**	**Heapy**
Pior	10709ms	33745 ms	xxxx	179ms	10002ms	560ms	409ms
Médio	4654ms	13127ms	xxxx	231ms	804ms	424ms	457ms
Melhor	1105ms	12444 ms	xxxx	158ms	170ms	496ms	354ms

Tabela de Casos:

Tabela de ordenação de casos							
Casos	**Insertion**	**Selection**	**Couting**	**Merge**	**Quick**	**Mediana de 3**	**Heapy**
Pior	831ms	1016ms	177ms	50ms	558 ms	30ms	18ms
Médio	119ms	920ms	170ms	16ms	22ms	26ms	16ms
Melhor	109ms	655 ms	14ms	45ms	18ms	26ms	13ms

Tabela de Mortes:

Tabela de ordenação de Mortes							
Casos	**Insertion**	**Selection**	**Couting**	**Merge**	**Quick**	**Mediana de 3**	**Heapy**
Pior	719ms	795ms	146ms	56ms	123 ms	28ms	12 ms
Médio	460ms	655ms	20ms	66ms	21 ms	20ms	12ms
Melhor	104ms	598ms	11ms	45ms	19 ms	19ms	12 ms

Vale lembrar que esses foram os resultados para uma execução feita em nossas maquinas locais, a execução em outros computadores pode tanto demorar quanto ser mais rápido, dependendo da memória, processador dentre outros equipamentos do pc.