##### UNIVERSIDADE FEDERAL DE JUIZ DE FORA

ICE - INSTITUTO DE CIÊNCIAS EXATAS DEPT. CIÊNCIA DE COMPUTAÇÃO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO DCC012 · ESTRUTURA DE DADOS ||

2° SEMESTRE DE 2020

**TRABALHO**

**PARTE 2**

#### Juiz de Fora 2021



##### UNIVERSIDADE FEDERAL DE JUIZ DE FORA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO DCC012 · ESTRUTURA DE DADOS ||

ORIENTADOR:

MARCELO CANIATO RENHE

COMPONENTES:

|  |  |
| --- | --- |
| NOME | MATRÍCULA |
| BEATRIZ CUNHA RODRIGUES | 201776038 |
| IVANYLSON HONÓRIO GONÇALVES | 201776002 |
| JOÃO PEDRO SEQUETO NASCIMENTO | 201776022 |
|  |  |
|  |  |

#### Juiz de Fora 2021

**Resumo**

r

Este projeto é a segunda parte da disciplina de Estrutura de Dados 2.

## Sumário

[Sumário...........................................................................................................................................4](#_Toc62908421)

[ Introdução...........................................................................................................................................5](#_Toc62908422)

[ Detalhamento das atividades......................................................................................................5](#_Toc62908422)

[ Decisões de implementação......................................................................................6](#_Toc62908423)

[ Resultados................................................................................................................. 11](#_Toc62908425)

[ Análise detalhada dos resultados obtidos............................................................. 14](#_Toc62908426)

[ Referência utilizada no desenvolvimento do trabalho.......................................... 15](#_Toc62908427)

# Introdução

# 

# Nesta fase do trabalho, foi necessário analisar um conjuntos de dados fornecido pelos arquivos “brazil\_cities\_coordinates.csv” e “brazil\_covid19\_cities\_processado.csv”, e comparar o desempenho de diferentes estruturas de dados baseados no conhecimento adquirido ao longo da disciplina. Com isso, fomos capazes de:

# Manipular o arquivo em modo texto e armazená-lo em uma quad tree;

# Implementar cada estrutura de dados;

# Analisar o desempenho das estruturas em diferentes situações;

# Apresentar os resultados obtidos.

# Detalhamento das atividades

|  |  |
| --- | --- |
| Integrantes | Funções |
| BEATRIZ CUNHA RODRIGUES | Esteve responsável pela implementação da tabela hash, do módulo de testes e contribuiu para implementação da análise das estruturas. |
| IVANYLSON HONÓRIO GONÇALVES | Esteve responsável pela implementação da árvore B, contribuiu para a implementação da árvore AVL e para análise das estruturas. |
| JOÃO PEDRO SEQUETO NASCIMENTO | Esteve responsável pela implementação do quad tree (pré processamento dos dados), pela implementação da árvore AVL e contribuiu para análise das estruturas. |

# Decisões de implementação

# Explicações das estruturas de dados e as razões para as decisões de implementação

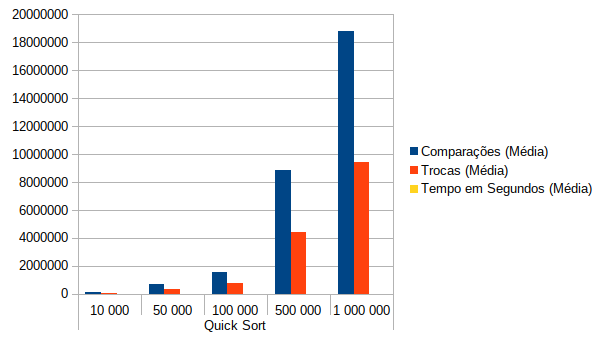
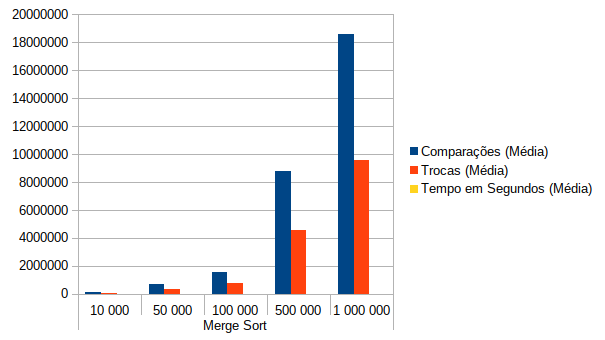
# Resultados

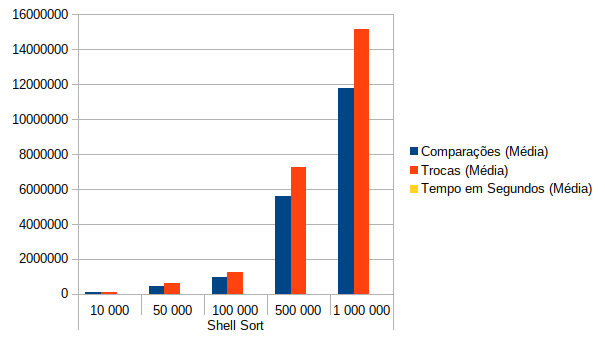
Para este relatório foram realizados testes com os algoritmos selecionados ordenando vetores de diferentes tamanhos e com ordenação aleatória. Foram utilizados vetores de 5 tamanhos diferentes: 10000, 50000, 100000, 500000 e 1000000. Cada vetor foi ordenado 5 vezes e o valor utilizado no relatório foi a média obtida das 5 interações. As métricas definidas para avaliar o resultado foram a quantidade de movimentações entre chaves, quantidade de comparações entre chaves e tempo de execução de cada algoritmo. As tabelas abaixo mostram os valores obtidos com os testes realizados para cada um dos algoritmos.

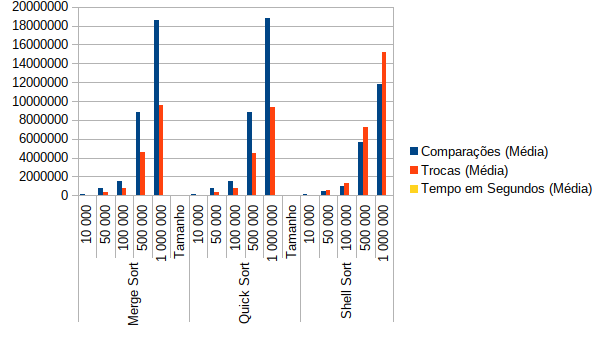
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tamanho | Comparações (Média) | Trocas (Média) | Tempo em Segundos (Média) |
| Merge Sort | 10 000 | 120509 | 63865 | 0.2 |
| 50 000 | 718487 | 376210 | 1.6 |
| 100 000 | 1537208 | 802468 | 3.2 |
| 500 000 | 8803616 | 4550147 | 16.8 |
| 1 000 000 | 18607749 | 9600914 | 34.8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tamanho | Comparações (Média) | Trocas (Média) | Tempo em Segundos (Média) |
| Quick Sort | 10 000 | 120798 | 60291 | 0.07 |
| 50 000 | 732538 | 366037 | 0.4 |
| 100 000 | 1549702 | 774531 | 0.9 |
| 500 000 | 8876459 | 4437546 | 5.1 |
| 1 000 000 | 18827631 | 9412910 | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tamanho | Comparações (Média) | Trocas (Média) | Tempo em Segundos (Média) |
| Shell Sort | 10 000 | 102536 | 75243 | 0.06 |
| 50 000 | 455719 | 599792 | 0.4 |
| 100 000 | 967146 | 1271607 | 0.7 |
| 500 000 | 5601426 | 7270872 | 4.8 |
| 1 000 000 | 11804265 | 15177925 | 9.5 |

****

****

****

# Análise detalhada dos resultados obtidos

Após a realização e cálculo da média das métricas selecionadas para a análise dos algoritmos, conseguimos observar algumas relações entre as operações de comparação e trocas com o tempo de execução e obter algumas conclusões.

O primeiro ponto que foi observado foi o número de comparações e trocas entre o algoritmo do Mergesort e Quicksort. Foi possível verificar que o número de ambas operações foi similar nos dois algoritmos, porém ocorreu uma diferença no tempo de execução, onde o Mergesort levou um tempo bem acima do que o Quicksort. Os dois algoritmos utilizam a divisão do vetor para a ordenação, de acordo com o paradigma de divisão e conquista, porém o quicksort não utiliza um vetor auxiliar para realizar intercalação dos elementos e também realiza menos operações de trocas entre os elementos, o que pode determinar a redução no tempo de execução.

A próxima análise foi sobre o desempenho do algoritmo Shellsort nesse experimento. Foi possível verificar que o algoritmo demonstrou um tempo de execução melhor que o Mergesort e próximo do Quicksort. Analisando os dados obtidos com a execução do Shellsort, percebemos que o algoritmo realiza um número maior de operações de trocas, porém ocorre uma redução significativa do número de comparações, o que pode ser o motivo da redução de tempo em relação aos demais algoritmos.

Por fim, nesse experimento foi possível verificar uma similaridade de desempenho entre o Quicksort e o Shellsort para o grupo de dados selecionados. Ambos demonstraram um tempo médio de execução bem próximo entre si, e ambos foram mais eficientes do que o Mergesort para a ordenação dos registros.

# Referência utilizada no desenvolvimento do trabalho

• Souza, J. É. G., Ricarte, J. V. G., Lima, N. C. A., Algoritmos de Ordenação: Um estudo comparativo. In: ENCONTRO DE COMPUTAÇÃO DO OESTE POTIGUAR. 2017, Pau de Ferro, p. 163-173. <Acessado em Jan de 2021>

• Drozdek, A. (2002) Estrutura de dados e algoritmos em C++, Thomson Learning, São Paulo <Acessado em Jan de 2021>

• https://blog.pantuza.com/artigos/o-algoritmo-de-ordenacao-quicksort <Acessado em Jan de 2021>

• https://www.geeksforgeeks.org/quick-sort/ <Acessado em Jan de 2021>

• http://www.cplusplus.com/reference/algorithm/merge/ <Acessado em Jan de 2021>

• https://www.geeksforgeeks.org/shellsort/ <Acessado em Jan de 2021>

• https://pt.wikipedia.org/wiki/Shell\_sort <Acessado em Jan de 2021>

• https://pt.wikipedia.org/wiki/Quicksort <Acessado em Jan de 2021>

• https://pt.wikipedia.org/wiki/MergeSort <Acessado em Jan de 2021>