UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

TRABALHO PRÁTICO 2 - BCC203

Vinicius Gabriel Verona & Carlos Eduardo Romaniello

Professor: Dr. Guilherme Tavares de Assis

Relatório referente ao primeiro trabalho prático de EDII

Data: 08 de abril de 2021
Local: Ouro Preto – Minas Gerais – Brasil

Sumário

1 Introdução **2**

2 Desenvolvimento **3**

2.1 Decisões 3

2.2 Desafios 4

3 Resultados **5**

3.1 Testes 5

3.2 Conclusões 6

3.2.1 Comparações de chaves 6

3.2.2 Acesso a arquivos 7

3.2.3 Tempo de execução 9

3.2.4 Considerações Finais 9

1 Introdução

Este relatório, referente ao segundo trabalho prático de EDII (Estrutura de dados 2), tem como objetivo analisar o desempenho dos algoritmos e técnicas apresentadas em aula, são estes:

- Intercalação Balanceada de Vários Caminhos;
- Intercalação Balanceada de Vários Caminhos pelo método de Substituição;
- Quicksort Externo;

2 Desenvolvimento

2.1 Decisões

Durante o desenvolvimento de todos os algoritmos citados na seção anterior, foram tomadas algumas decisões para melhorar o entendimento e facilitar a implementação dos códigos.

Primeiramente, para poder automatizar a execução de todos os programas, foram utilizados alguns arquivos implementados na linguagem de programação ***Julia***[1] e *shell script* para poder executar todos os experimentos necessários. Além disso, como discutido na reunião com o professor Guilherme, foram realizadas duas cópias do arquivo "PROVAO.txt" através de um programa em *Julia*, uma ordenada crescentemente e uma decrescentemente para que servissem de arquivo de entrada dependendo do parâmetro passado pela linha de comando.

Para qualquer execução do código, o arquivo de entrada foi clonado no seu repectivo diretório para que caso fosse necessário realizar qualuer mudança, essa mudança seria refletida na cópia, mantendo o arquivo original intacto para as outras execuções.

Para os algoritmos da intercalação balanceada de vários caminhos e para o *Quick Sort externo* foi utilizado o algoritmo *Quick Sort* para realizar a ordenção dos registros dentro da memória interna. Para o algoritmo de intercalação balanceada usando o método de substituição foi utilziado o *Heap Sort* como mencionado em sala de aula. No caso dos métodos de intercalação balanceada a fita que possui a ordenação final é renomeada para "PROVAO_RESULTADO.txt".

Durante a elaboração do código para a intercalação balanceada usando o método de substituição foi utilizada uma string *"- -"* para representar a separação dos blocos dentro dos arquivos, já que os blocos gerados são de tamanhos diversos.

Por fim, optamos por substituir as tabelas contendo resultados por gráficos interativos que facilitam a análise e visualisação dos resultados. Para tal, foi utilizada a linguagem de programação *JavaScript*, e assim como os *scripts* de automatização de testes, os códigos e gráficos se encontram, juntamente a este relatório, em pastas subdivididas por tópicos e métodos.

[1] Mais informações sobre a linguagem em https://julialang.org

2.2 Desafios

Durante a implementação dos métodos citados houve algumas dificuldades. Grande parte surgiu durante a manipulação de ponteiros em arquivos textos externos.

Ao adaptarmos o código referente ao *Quick Sort Externo* apresentado em aula percebemos um comportamento anormal. Quando executado no Windows ele funciona normalmente, porém ao ser executado em Linux, o algoritmo não consegue atribuir os devidos valores, comprometendo a ordenação final. Ao rastrear o problema, percebemos que o *bug* estava acontecendo em uma atribuição de valores a uma variável; acreditamos que este erro se dá devido ao uso de uma função com comportamento diferente em cada sistema. Sendo assim, não conseguimos corrigir o problema para sistemas operacionais Linux.

3 Resultados

3.1 Testes

A fim de teste e conforme solicitado para cada método foram executados uma sequência de testes com configurações pré-definidas. As configurações foram:

- 5 quantidades diferentes de registros para cada método
  - 100 registros;
  - 1.000 registros;
  - 10.000 registros;
  - 100.000 registros;
  - 471.705 registros;
- 3 tipos de ordenação para cada quantidade de registros
  - 1 ->Ordenados crescentemente;
  - 2 ->Ordernados decrescentemente;
  - 3 ->Ordenados aleatoriamente;

Para a análise dos resultados, foram coletados os dados referentes ao acesso do algoritmo aos arquivos texto, o tempo gasto para a ordenação total e as comparações entre registros.

Vale ressaltar que apresentaremos as conclusões sem o anexo de tabelas pois julgamos os gráficos presentes mais que necessários para a visualisaçao dos dados obtidos.

3.2 Conclusões

Analisando os gráficos gerados e levando em consideração a forma de implementação dos algorítmos, observamos os seguintes comportamentos:

3.2.1 Comparações de chaves

Utilizando o critério de comparações de chaves obtivemos os seguintes resultados para cada um dos métodos e cada ordenação do arquivo de entrada:

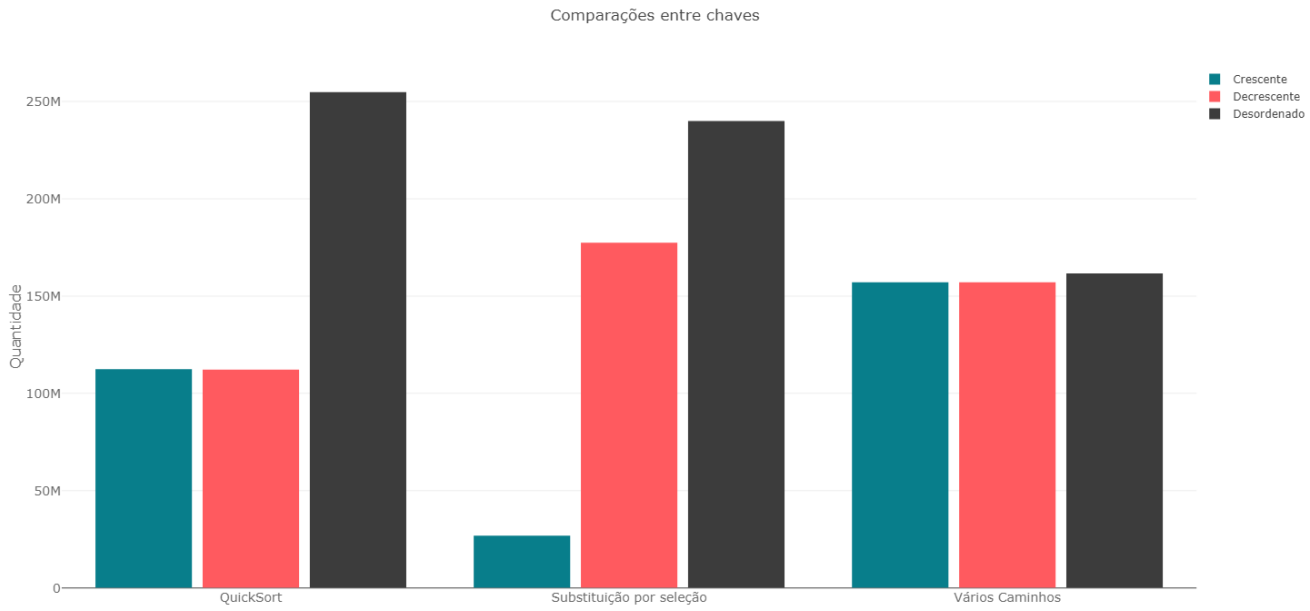

Figura 3.1: Comparações - O gráfico interativo se encontra no diretório */graficos/Comparacoes/quantidade/*

Podemos perceber que para o critério de comparações em um arquivo com um número significativo de registros, o algoritmo de Intercalação Balanceada de vários caminhos (2f) se apresenta mais estável. Para casos específicos de ordenação, o *Quick Sort Externo* foi melhor em um arquivo decrescente, a Intercalação com o método de substituição por seleção foi melhor no arquivo crescente e a Intercalação balanceada de vários caminhos (2f) no arquivo desordenado.

3.2.2 Acesso a arquivos

Utilizando o critério de acesso aos arquivos texto obtivemos os seguintes resultados para cada um dos métodos:

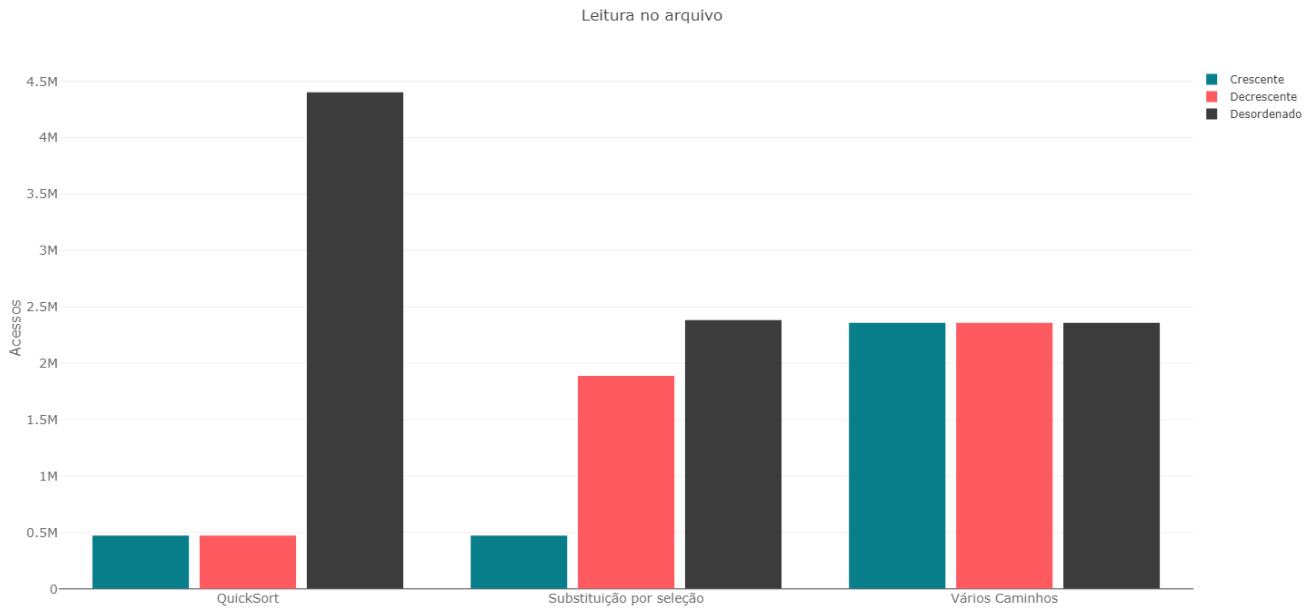

Figura 3.2: Leituras - O gráfico interativo se encontra no diretório */graficos/Leitura/quantidade/*

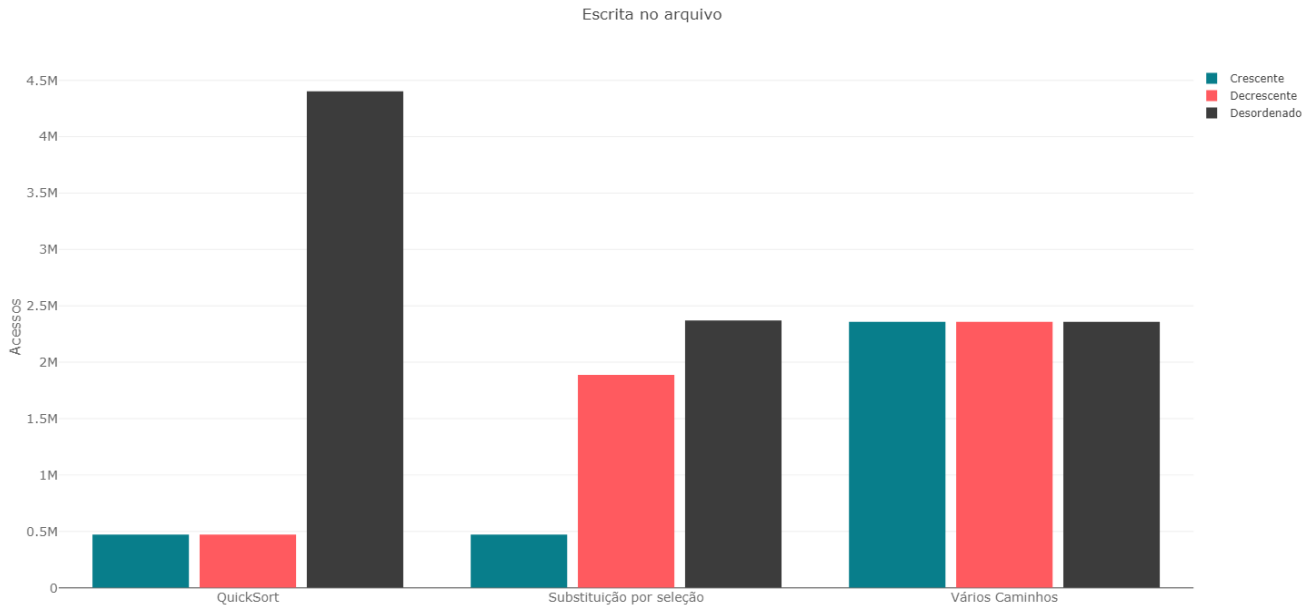

Figura 3.3: Escrita - O gráfico interativo se encontra no diretório */graficos/Escrita/quantidade/*

Analisando os gráficos de acesso ao arquivo externo, percebemos que o *Quick Sort Externo* apresenta os melhores resultados com exceção do caso em que o arquivo está desordenado. No geral, o algoritmo de intercalação balanceada com substituição por seleção apresentou melhores resultados, pois sua média foi menor que a dos outros algoritmos.

3.2.3 Tempo de execução

Utilizando o critério de tempo de execução obtivemos os seguintes resultados para cada um dos métodos:

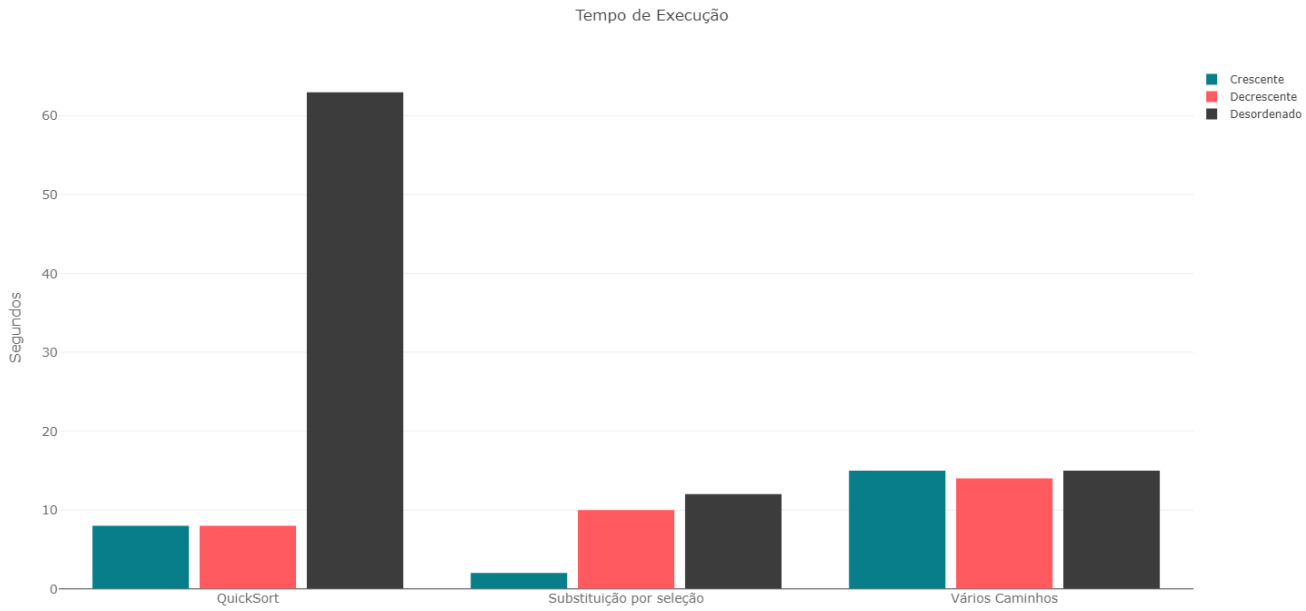

Figura 3.4: Tempo - O gráfico interativo se encontra no diretório */graficos/Tempo/quantidade/*

Analisando o tempo de execução de todos os métodos, o algoritmo de substituição por seleção foi o mais rápido, levando em média 8 segundos para ordenar todos os registros. O *Quick Sort Externo* levou em média 26,3 segundos e o algoritmo de intercalação balanceada de vários caminhos levou 14,6 segundos.

3.2.4 Considerações Finais

Os experimentos foram executados em um computador com processador Intel i7-9750H 2.6Ghz e 8GB de memória RAM. Após analisar todos os dados conjuntos e individuais de cada critério, os melhores resultados foram o da intercalação com o método de substituição por seleção.

Na pasta raiz do trabalho, estará presente um arquivo *README.md* onde estarão presentes instruções para execução do arquivo e utilização dos scripts caso necessário. Haverá também uma explicação da estruturação das pastas. Vale lembrar que o *Quick Sort Externo* executará apenas em um sistema Windows.