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

TRABALHO PRÁTICO 1 - BCC203

Vinicius Gabriel Verona & Carlos Eduardo Romaniello

Professor: Dr. Guilherme Tavares de Assis

Relatório referente ao primeiro trabalho prático de EDII

Data: 1 de março de 2021
Local: Ouro Preto – Minas Gerais – Brasil

Sumário

1 Introdução

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

- Acesso Sequencial Indexado (ASI);
- Árvore Binária Externa;
- Árvore B Externa;
- Árvore B* Externa;

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 gerar números aleatórios para cada execução. Além disso, pela facilidade e rapidez na manipulação de arquivos 'txt' em *Julia*, para as execuções onde a ordenação dos registros deveria ser aleatória, foi criado um programa para criar um arquivo 'txt' contendo esses registros que foram utilizados para todos os algoritmos implementados.

Para o ASI, implementamos também métodos de ordenação, mais especificamente o *QuickSort*, com a finalidade de otimizar a inserção dos itens no arquivo binário de saida. Todos os itens foram gerados e manipulados na memória principal antes de serem escritos no arquivo binário, como indicado pelo professor Guilherme e pela tutora Cibele.

Para a Árvore Binária, os itens também foram manipulados em memória principal. Além disso, foram utilizados os algoritmos de árvore binária AVL implementados por nós enquanto cursávamos a matéria BCC202 (Estrutura de Dados 1). Durante a pesquisa do item desejado, foi lido do arquivo apenas o nó a ser analisado.

Para a Árvore B e B*, ambas também manipularam os itens em memória principal antes de serem escritas em um arquivo binário. Para a árvore B, foi utilizado o código apresentado durante uma aula dada pelo professor Guilherme, realizando apenas algumas alterações para se encaixar no contexto do trabalho. Por fim, para a árvore B*, também foi utilizado o código da árvore B com apenas algumas modificações para se adequar à teoria do método. Nelas, durante a fase de pesquisa, também foram lidas apenas as páginas a serem analisadas.

Por fim, optamos por substituir as tabelas contendo resultados por gráficos interativos, que facilitam a análise e visualisação dos resultados. Para tal, foi utilizado 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 houveram algumas dificuldades. Grande parte surgiu durante a manipulação de ponteiros em arquivos externos.

Para a árvore binária, começamos a implementar um código com manipulações totalmente externa que acabou por funcionar, porém, devido aos recursos computacionais, o tempo necessário para a entrega do trabalho e para a análise dos dados, além da indicação do professor e da tutora, optamos por reescrever o código contendo a criação da árvore em memória principal.

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 e parcialmente randômicas. As configurações foram:

- 5 quantidades diferentes de registros para cada método
  - 100 registros;
  - 1.000 registros;
  - 10.000 registros;
  - 100.000 registros;
  - 1.000.000 registros;
- 3 tipos de ordenação para cada quantidade de registros
  - 1 ->Ordenados crescentemente;
  - 2 ->Ordernados decrescentemente;
  - 3 ->Ordenados aleatoriamente;
- 10 chaves escolhidas aleatoriamente, porém pertencentes ao conjunto de registros, para cada ordenação, totalizando 150 execuções por método.

Para a análise dos resultados, foram coletados os dados referentes ao acesso do algoritmo aos arquivos binários, o tempo gasto para a criação da estrutura na memória principal & externa, para a pesquisa das chaves e a quantidade de comparações feitas durante a criação do arquivo e durante a pesquisa.

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

Observando os gráficos gerados e levando em consideração a forma de implementação dos algorítmos em memória principal e externa, 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:

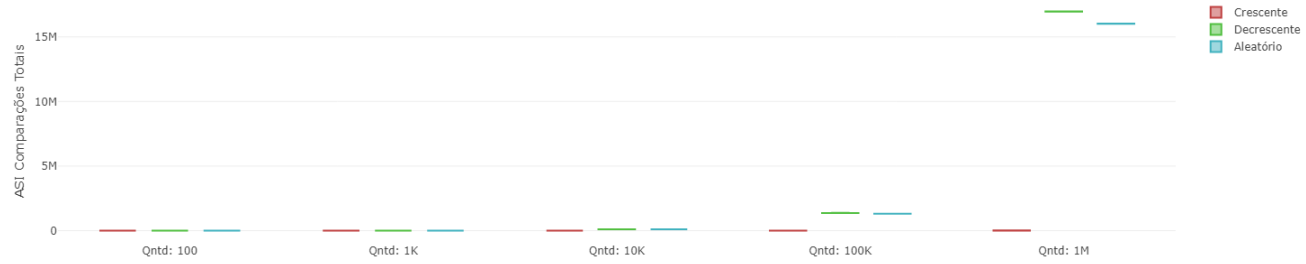

Figura 3.1: ASI. O gráfico interativo se encontra no diretório */graficos/ASI/compTotal/*

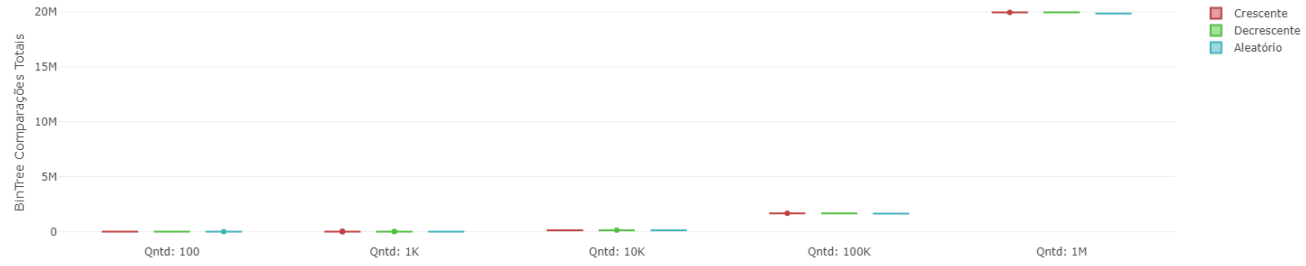

Figura 3.2: Árvore Bin. O gráfico interativo se encontra no diretório */graficos/BinTree/compTotal/*

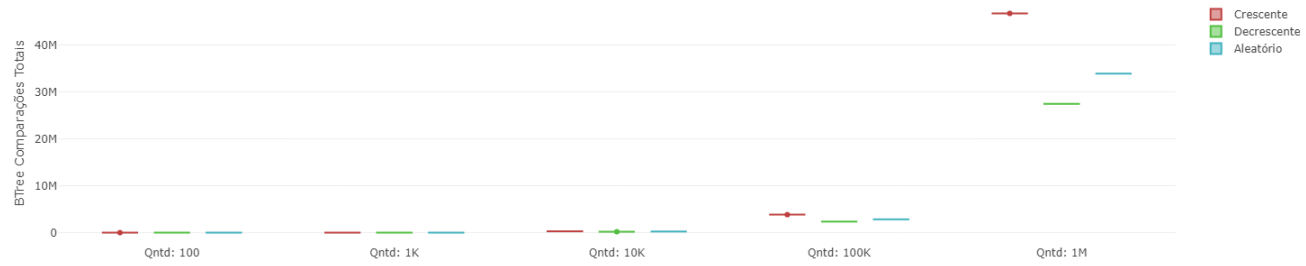

Figura 3.3: Árvore B. O gráfico interativo se encontra no diretório */graficos/BTree/compTotal/*

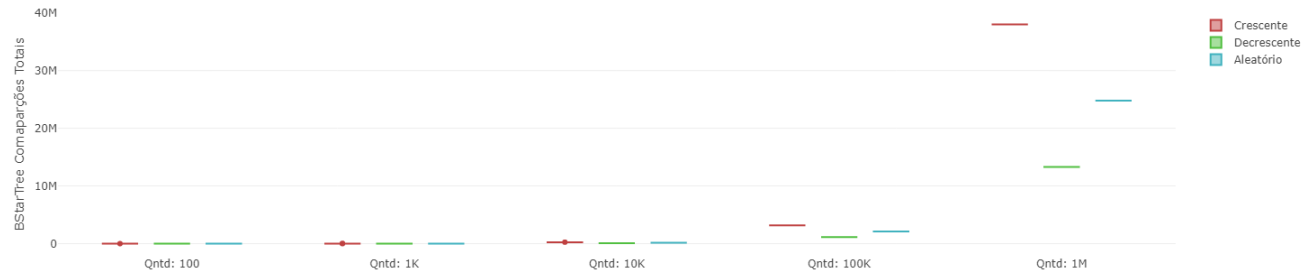

Figura 3.4: Árvore B*. O gráfico interativo se encontra no diretório */graficos/BStarTree/compTotal/*

Podemos perceber que para o critério de comparações, da criação do arquivo até a pesquisa, o Acesso Sequencial Indexado é melhor, o que é previsto, pois suas comparações são feitas apenas no *QuickSort*, durante a criação do arquivo binário, e na localização da página do registro.

No entanto, analisando os gráficos de cada etapa separadamente, vemos que os gráficos de árvores executam menos comparações durante a fase de pesquisa (os arquivos individuais estão presentes no diretório $graficos/metodo-de-pesquisa/operacao/etapa$).

3.2.2 Acesso a arquivos

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

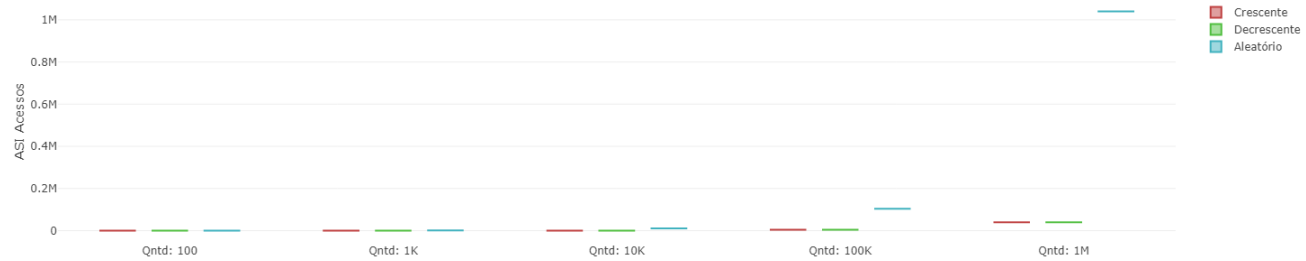

Figura 3.5: ASI. O gráfico interativo se encontra no diretório */graficos/ASI/acessos/*

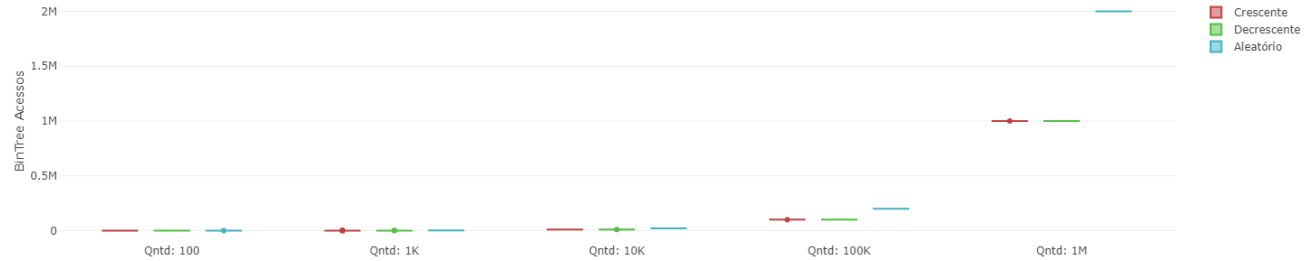

Figura 3.6: Árvore Bin. O gráfico interativo se encontra no diretório */graficos/BinTree/acessos/*

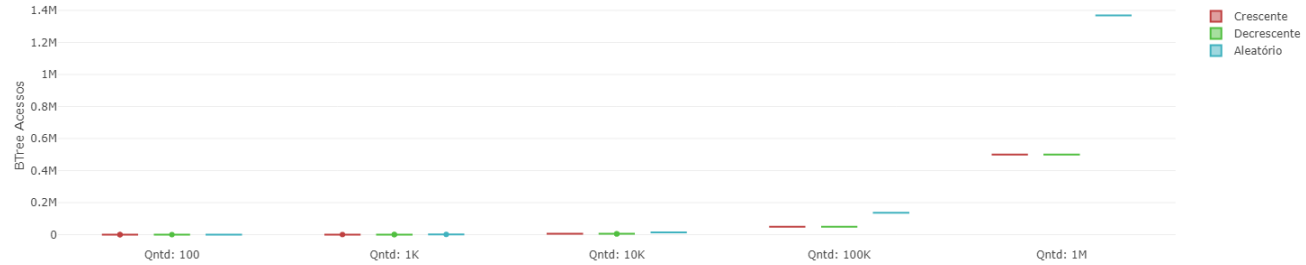

Figura 3.7: Árvore B. O gráfico interativo se encontra no diretório */graficos/BTree/acessos/*

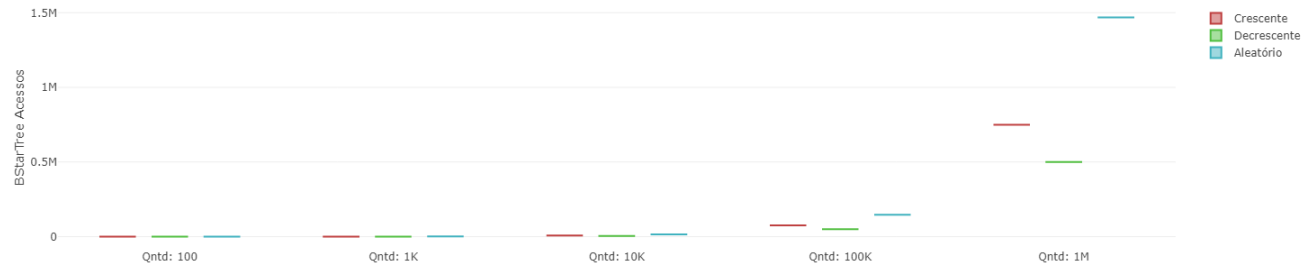

Figura 3.8: Árvore B*. O gráfico interativo se encontra no diretório */graficos/BStarTree/acessos/*

Analisando os gráficos de acesso ao arquivo externo, percebemos que os piores casos ocorrem durante a ordenação aleatória dos itens, isso ocorre devido a necessidade de se ler as chaves do arquivo txt mencionado nas sessões anteriores.

Porém, ao analisar os acessos somente durante a fase de pesquisa, percebemos que o método ASI é muito pior do que os métodos de árvore, que em um arquivo de 1 milhão de registros executam cerca de 40 acessos ao arquivo, enquanto o ASI acessa em torno de 40 mil vezes.

3.2.3 Tempo de execução

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

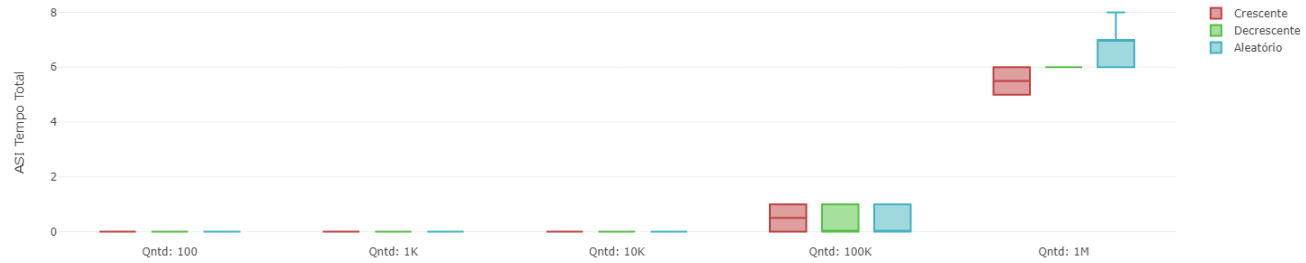

Figura 3.9: ASI. O gráfico interativo se encontra no diretório */graficos/ASI/tempoTotal/*

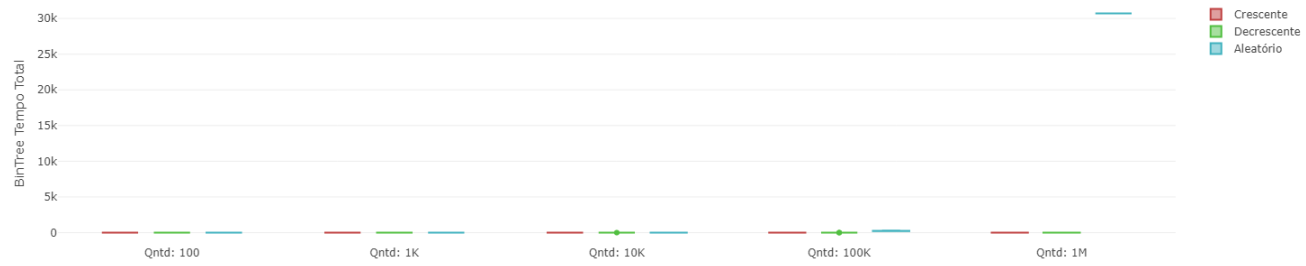

Figura 3.10: Árvore Bin. O gráfico interativo se encontra no diretório */graficos/BinTree/tempoTotal/*

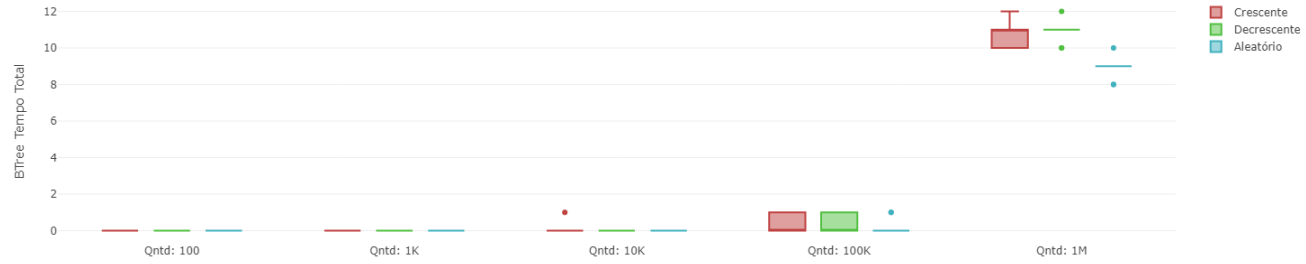

Figura 3.11: Árvore B. O gráfico interativo se encontra no diretório */graficos/BTree/tempoTotal/*

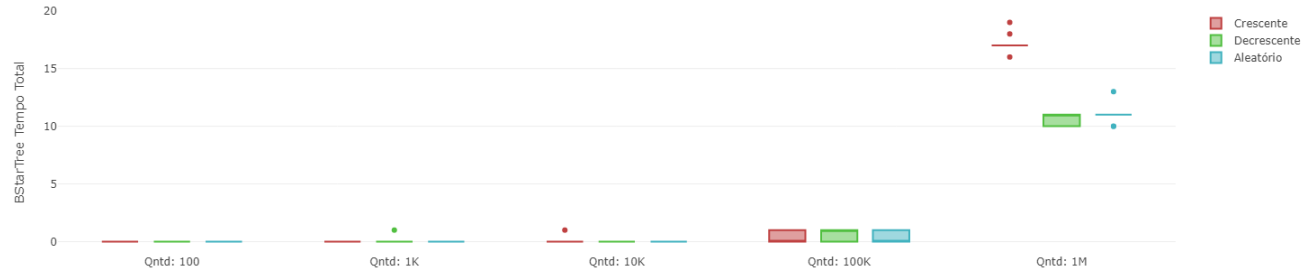

Figura 3.12: Árvore B*. O gráfico interativo se encontra no diretório */graficos/BStarTree/tempoTotal/*

Analisando o tempo de execução de todos os métodos englobando todas as fases, percebemos um caso anormal na árvore binária, onde para gerar a árvore com as chaves sendo

inseridas aleatoriamente e realizar a pesquisa, o algoritmo levou mais de 30 mil segundos (8,3 horas), que mesmo após várias revisões do algorítmo não tenha sido encontrado nenhum erro de implementação, esse é o único caso absurdo.

Apesar disso, todos os outro métodos realizam sua função em um tempo satisfatório de no máximo 20 segundos, independentemente da quantidade de registros e da ordenação. Nese critério percebe-se que o ASI levou vantagem, levando no máximo 9 segundos.

Para os casos particulares de pesquisa, não obtivemos conclusões pois o tempo retornado é em média 0 segundos em todos os casos.

3.2.4 Considerações Finais

Após analisar todos os dados conjuntos e individuais de cada critério, ainda que próximos, os melhores resultados foram o da árvores B e da árvore B*, mesmo que o ASI tenha levando vantagem em alguns critérios. Elas se saíram melhor no quesito geral pois estão sempre apresentando bons resultados independente do critério.

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.