Relatório de Estrutura de Dados II - Parte 2

Felipe Reis Gonçalves[1], Otávio Augusto Ferreira Rodrigues[2], and Rafael Braga Ladeira Dutra[3]

[1]201335053
[2]201276030
[3]201235054

[1]Juiz de Fora - MG - Brasil

[2]Universidade Federal de Juiz de Fora - UFJF

feliperg15@yahoo.com.br

otavioaufegues@gmail.com

dutra.rafaelbraga@gmail.com

Repositório do trabalho no GitHub: https://github.com/FelipeReis1/trabalhoED2pt2.

1. Instruções

Para executar o programa, navegue até a pasta dist, dentro do diretório do trabalho, e execute o comando java -jar TrabalhoED2pt2.jar caminhodiretorio, sendo caminhodiretorio o caminho onde se encontram os arquivos csv, exemplo: "\home\user\Desktop\", colocando a \ no final do diretório.

2. Atividades realizadas por cada membro

- Felipe Reis Gonçalves: Módulo de testes, QuadTree, BTree, args e execução na linha de comando.
- Otávio Augusto Ferreira Rodrigues: Classe de números primos, Tabela Hash, BTree, args e execução na linha de comando.
- Rafael Braga Ladeira Dutra: Classe DadosCovid com os carregamentos, Árvore AVL, BTree, args e execução na linha de comando.

3. Implementações

A estruturação das nossas classes foi feita de forma que o Main apenas execute o nosso módulo de testes, instanciando toda e qualquer estrutura de árvore ou tabela hash que foram criadas em arquivos separados para uma melhor organização. Implementamos um pré-carregamento de ambos os datasets com a classe DadosCovid no início da execução do programa. Além disso, criamos uma classe para fazer uma peneira de números primos, que é utilizada dimensionar de forma otimizada a tabela hash. Estruturamos também todas as classes dos nós em arquivos separados, estas armazenam os dados necessários de cada uma das árvores ou tabela hash. Em cada estrutura, uma função "Save"foi implementada para gerar um arquivo com os dados de saída requisitados.

O nosso menu de testes foi criado utilizando a função switch, onde no primeiro caso, é feita a execução das etapas do trabalho até a realizações dos teste, todos os dados dos arquivos importados são carregados nas referentes estruturas (QuadTree, Tabela Hash, Árvore AVL e Árvore B). Antes de executar os testes nas estruturas o usuário informa um código de cidade onde serão contabilizados os casos de Covid-19 registrados e duas coordenadas para uma busca de número de casos por região. Todo resultado é salvo em um arquivo de texto.

Cada uma das opções seguintes no menu se referem a um teste de uma das estruturas implementados no trabalho, ao seleciona-las será requisitado um valor para o número de registros a serem inseridos na estrutura de dados. São selecionados então registros aleatórios do dataset e a população é feita. Caso o número de registros inseridos seja menor que 20, o resultado será impresso no console e se for maior que 20, o resultado será salvo em um arquivo de texto com o nome referente à estrutura do caso. Para isso, estruturas de exibição dos resultados foram criadas para cada uma das árvores e tabela hash, estruturas estas que verificam o tamanho da entrada da chave.

Para a Etapa 1 do trabalho criamos a classe QuadTree e NoQuad para representar a estrutura da árvore, como dito anteriormente os datasets foram carregados em Arraylists no início da execução, com estrutura de Nó compatíveis com os Nós da árvore, assim para a inserção dos registros dentro da Quadtree utilizamos os dados armazenados na classe DadosCovid no método carregamentoQuadTree().

A estapa 2 que consiste em armazenar os registros do dataset brazil_covid19_cities_processado.csv em uma tabela hash, foi implementada em uma classe tabelaHash, para que essa estrutura fosse eficiente os fatores primordiais foram o seu onde o dimensionamento, multiplicando o número de registros de entrada por 1.5 e escolhendo o valor primo mais aproximado, gerando um om fator de carga, e criando uma bos função de espalhamento pra diminuir as colisões. Para tratar as colisões, buscamos sempre o próximo índice de valor primo para que houvesse um melhor espalhamento dos valores inseridos. Tivemos problemas de muitas colisões ao tentar implementar este tratamento com hash duplo, verificamos que o número de colisões era duas vezes maior do que o método adotado após.

Sobre as implementações das estruturas de árvores balanceadas (Árvore AVL e Árvore B) da estapa 3, na inserção de ambas passamos os campos necessários de acordo com a especificação do trabalho, sendo um deles, a própria tabela hash. A partir disso, conseguimos acessar as informações necessárias dentro dos nós por um método de busca da classe TabelaHash (buscaHash()), que retorna o item da tabela com a chave inserida na árvore. Com isso também resolvemos o problema da diferença no tamanho do códigos de Cidade entre os datasets. Utilizamos para ordenação o par código da cidade e data. Para data, fizemos uso da biblioteca Date do Java para executar operações que verificavam se a data era anterior ou posterior, facilitando assim a ordenação dos dados.

Problemas menores no código foram resolvidos através de análises linha por linha por cada um dos membros, e também usando do recurso de debugging na IDE, pois muitas vezes algumas variáveis das estruturas estavam armazenando valores incorretos gerando problemas como IndexOutOfBounds, dessa forma conseguíamos verificar exatamente em qual parte do código algo estava faltando ou estava além do necessário.

4. Resultados

A seguir, iremos apresentar as tabelas com os resultados para cada N valores, apresentando o total de uma cidade, total de uma região, tempo de inserção, tempo de busca, tempo de busca de uma região e número de comparações. Na última linha, apresentamos as médias desses valores.

Após as tabelas, mostraremos gráficos para as médias de comparações e para as médias de tempo de inserção de cada uma das árvores (AVL, B ordem 20, B ordem 200).

Tabelas:

- Árvore AVL:
  10.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	872	33017	124.0	0.0	9.0	469703
2	2358	33043	90.0	0.0	6.0	950574
3	5592	39808	67.0	0.0	3.0	1427119
4	1246	32248	61.0	0.0	4.0	1903414
5	0	33115	72.0	0.0	4.0	2380600
Médias			82.8	0.0	5.2	1426282.0

50.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	10364	178436	149.0	0.0	26.0	4198532
2	14896	182204	135.0	0.0	25.0	6066827
3	9912	183832	138.0	0.0	24.0	8007305
4	16938	180738	141.0	0.0	21.0	9810763
5	21934	188713	134.0	0.0	38.0	11718499
Médias			139.0	0.0	26.8	7960385.2

100.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	20284	352252	249.0	0.0	56.0	15313164
2	18985	341876	244.0	0.0	49.0	18565520
3	26305	353076	242.0	0.0	67.0	22439668
4	21422	352892	256.0	0.0	60.0	26081893
5	33715	364667	242.0	0.0	69.0	29818154
Médias			246.6	0.0	60.2	2.24436798E7

500.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	129858	1789337	1357.0	0.0	332.0	47834101
2	126611	1774819	1311.0	0.0	319.0	66073312
3	146150	1810026	1308.0	0.0	352.0	85719389
4	142371	1793166	1319.0	0.0	309.0	103839134
5	124697	1763461	1376.0	0.0	340.0	123109549
Médias			1334.2	0.0	330.4	8.5315097E7

1.000.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	279031	3579519	2878.0	0.0	641.0	158321467
2	276172	3581583	2979.0	0.0	687.0	195937310
3	264311	3577433	3012.0	0.0	638.0	2312246585
4	283192	3610878	2938.0	0.0	299.0	265746975
5	251474	3573004	2902.0	0.0	648.0	300863585
Médias			2941.8	0.0	582.6	2.304231844E8

- Árvore B (Ordem 20):
  10.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	652	33903	89.0	0.0	55.0	1343947
2	3789	36413	71.0	0.0	24.0	2676993
3	0	35520	61.0	0.0	11.0	4076675
4	4630	39124	62.0	0.0	6.0	5367885
5	9042	36951	77.0	0.0	6.0	6726682
Médias			72.0	0.0	20.4	4038436.4

50.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	18672	200565	135.0	0.0	83.0	13707480
2	16834	169961	148.0	0.0	85.0	21197180
3	11499	171107	132.0	0.0	82.0	28431724
4	13674	187896	140.0	0.0	92.0	35417278
5	8875	172894	136.0	0.0	70.0	42342147
Médias			138.2	0.0	82.4	2.82191618E7

100.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	23948	354676	254.0	0.0	171.0	56182819
2	15387	338962	254.0	0.0	187.0	69737833
3	17133	353209	252.0	0.0	201.0	83201356
4	23227	353492	249.0	0.0	163.0	96543767
5	14928	359487	239.0	0.0	166.0	110059929
Médias			249.6	0.0	177.6	8.31451408E7

500.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	144315	1809040	1564.0	0.0	929.0	170356879
2	148395	1793852	1332.0	0.0	840.0	235577553
3	122245	1760734	1278.0	0.0	842.0	300743914
4	117998	1789137	1332.0	0.0	906.0	370354679
5	129467	1758548	1264.0	0.0	882.0	435769160
Médias			1354.0	0.0	879.8	3.02565837E8

1.000.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	272082	3589703	2813.0	2.0	1440.0	550723389
2	275628	3569863	2835.0	0.0	1440.0	666023749
3	252649	3568752	2828.0	1.0	1495.0	786213602
4	260233	3577261	2821.0	0.0	1541.0	901115111
5	241854	3520314	2841.0	0.0	1536.0	1019968040
Médias			2827.6	0.6	1490.4	7.848087782E8

- Árvore B (Ordem 200)
  10.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	1897	39520	80.0	0.0	18.0	1025037702
2	3720	35712	74.0	1.0	14.0	1030034363
3	184	34954	85.0	0.0	18.0	103506221
4	3318	33037	74.0	0.0	15.0	1040164352
5	628	28835	73.0	0.0	15.0	1045171814
Médias			77.2	0.2	16.0	1.0350940604E9

50.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	16183	183094	242.0	0.0	242.0	1068761720
2	12364	174060	234.0	0.0	238.0	1092191187
3	9182	182803	236.0	0.0	232.0	1115407589
4	28785	212150	248.0	0.0	241.0	1139668015
5	6997	162213	232.0	0.0	266.0	1163543507
Médias			238.4	0.0	243.8	1.1156144036E9

100.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	19294	345838	555.0	4.0	936.0	1243875531
2	28151	368007	543.0	0.0	894.0	1321039647
3	34324	356208	548.0	0.0	900.0	1399045449
4	24036	350219	534.0	0.0	969.0	1481046841
5	19188	342885	537.0	0.0	902.0	1561116734
Médias			543.4	0.8	920.2	1.4012248404E9

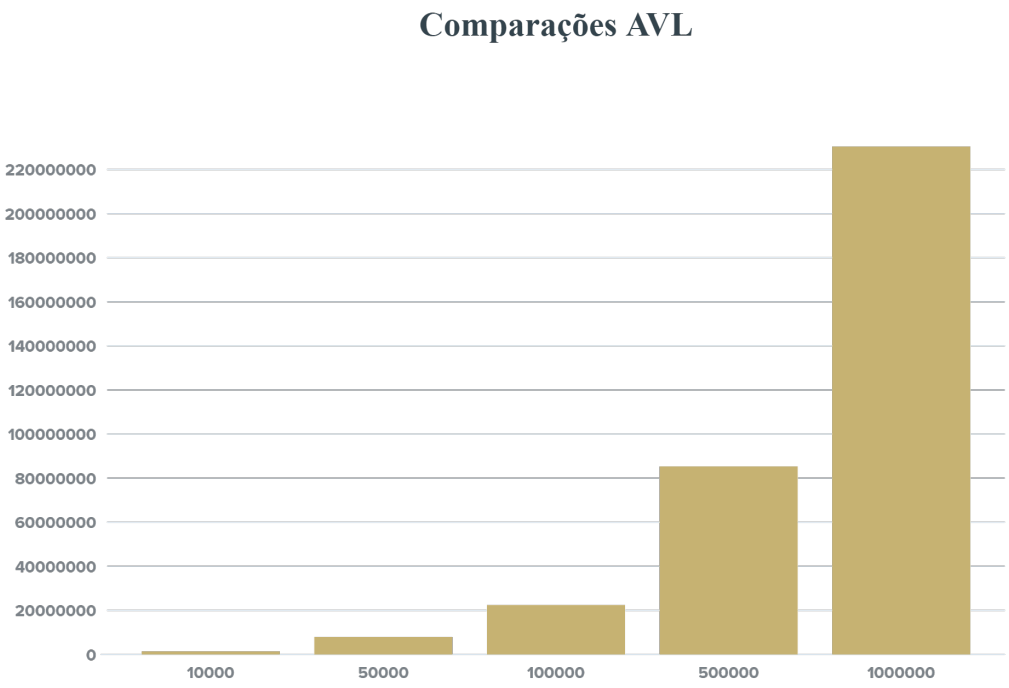
500.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	106204	1773879	4208.0	1.0	3119.0	1824416268
2	125855	1784672	4678.0	0.0	3681.0	2073484413
3	133499	1791258	5795.0	0.0	3910.0	2322200452
4	122361	1749633	4880.0	0.0	3607.0	2579187918
5	133180	1794235	4805.0	3.0	3283.0	2841465748
Médias			4873.2	0.8	3520.0	2.3281509598E9

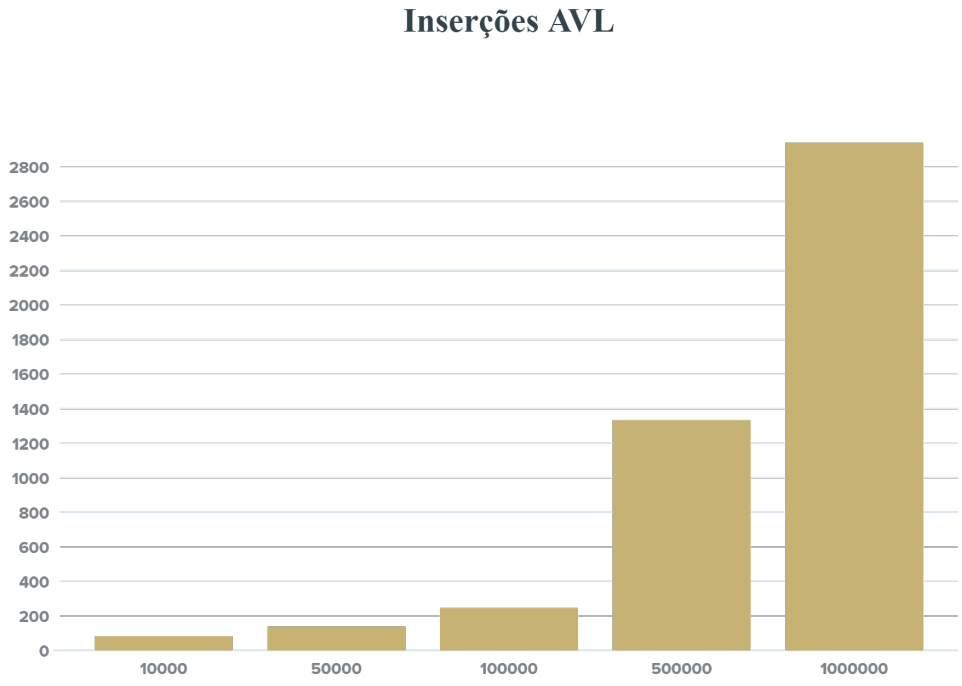
1.000.000 valores:

Rodadas	Tot./Cid.	Tot./Reg	Inser.	Busca	Busca/reg.	Num. Comp.
1	250105	3536817	10068.0	3.0	5502.0	3403389586
2	274870	3582129	13759.0	1.0	6750.0	3948593387
3	249947	3564603	10002.0	2.0	6443.0	4494082653
4	249590	3562098	9782.0	2.0	6419.0	5043004347
5	231675	3561184	9731.0	3.0	6150.0	5573947695
Médias			10668.4	2.2	6252.8	4.4926035336E9

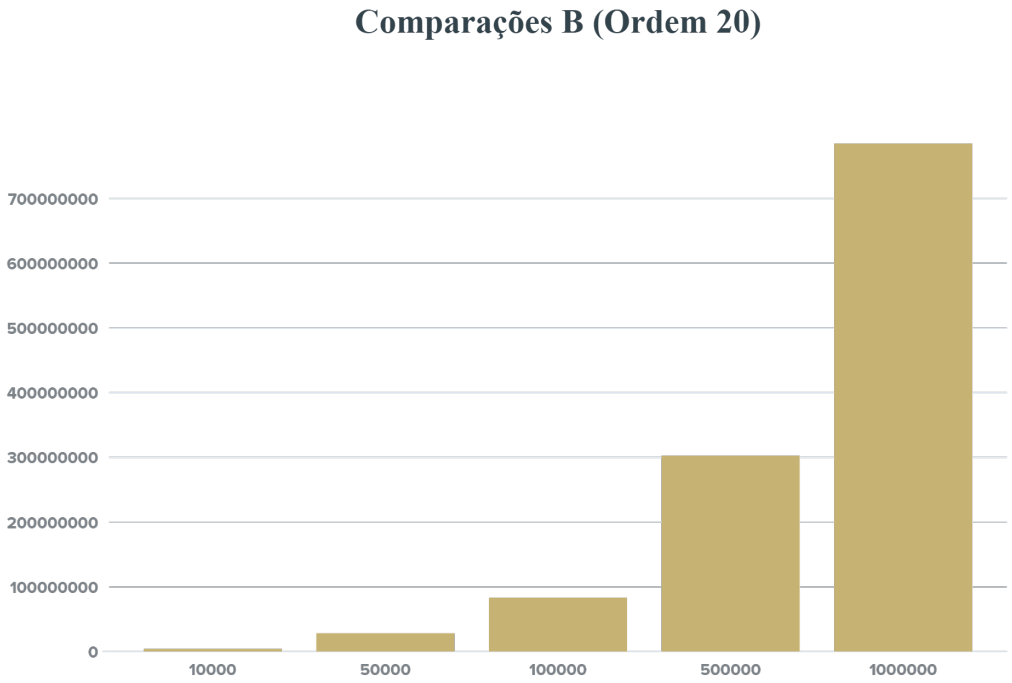
Gráficos:



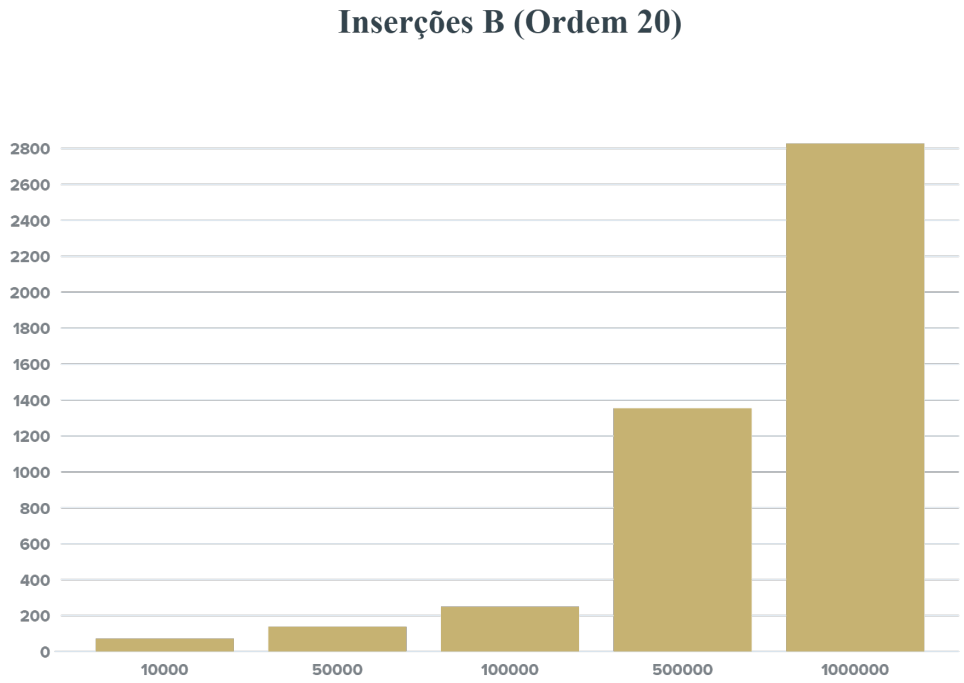
Número de comparações da árvore AVL x Número de registros



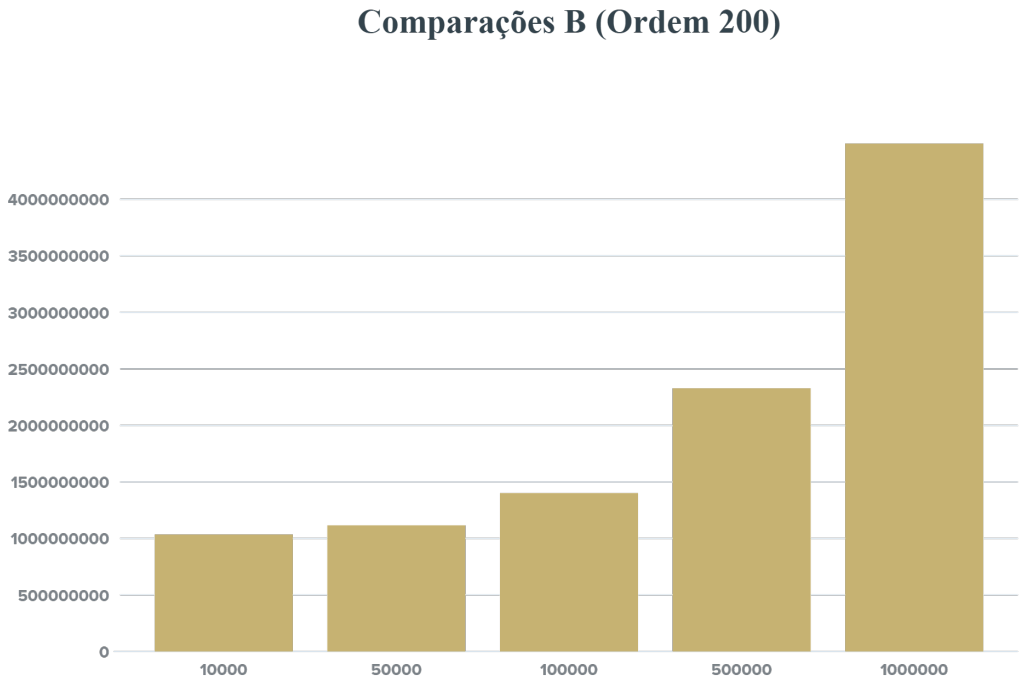
Tempo de inserção (ms) da árvore AVL x Número de registros



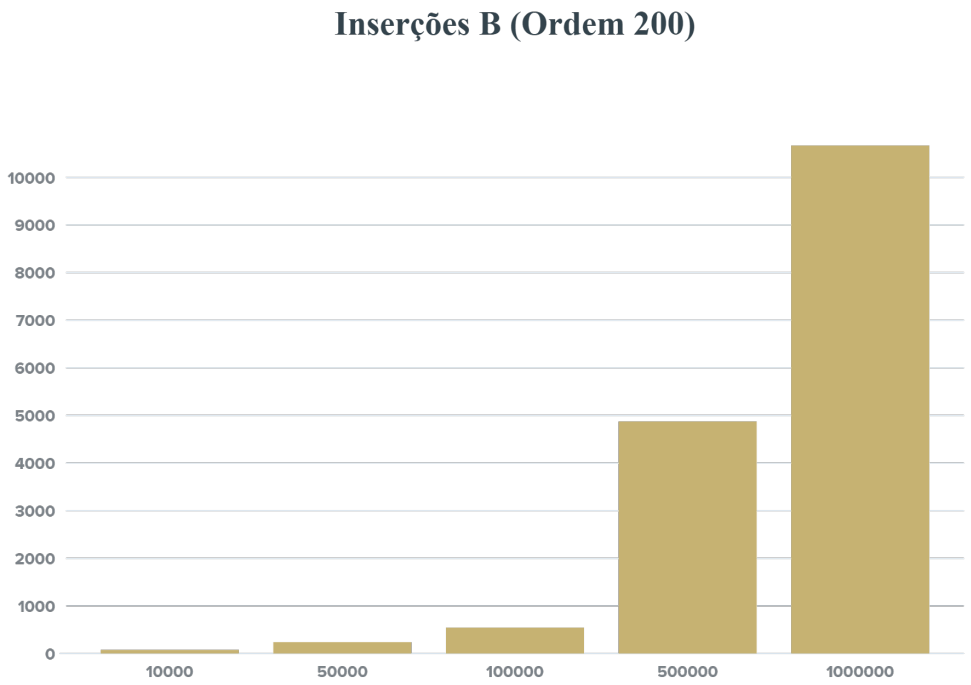
Número de comparações da árvore B ordem 20 x Número de registros



Tempo de inserção (ms) da árvore B ordem 20 x Número de registros



Número de comparações da árvore B ordem 200 x Número de registros



Tempo de inserção (ms) da árvore B ordem 200 x Número de registros

5. Conclusões

Analisando as estruturas implementadas, podemos verificar que na árvore AVL, como os dados de entrada são aleatórios, a ordem de busca se manteve em O(log n) para qualquer tamanho de entrada, pois nem todos os nós precisam ser acessados durante este processo.

Observamos que o número de comparações da árvore B de ordem 200 é bem maior que o das outras árvores, porém esse número aumenta menos com o aumento dos números de registros em relação às outras. Mesmo a de ordem 200 tendo um número maior de comparações do que a de ordem 20 com 1 milhão de registros, acreditamos que se o número de registros fosse maior, a de ordem 20 iria fazer mais comparações que a outra eventualmente.

Apesar do que observamos com o número de comparações entre as árvores B, o tempo de inserções das duas se inicia parecido quando N = 10.000, porém cresce mais na de ordem 200. Isso se deve ao fato de que a operação de split da de ordem 200 fica mais custosa, pois precisa redistribuir um maior número de registros em novos nós.

6. Referências

- https://jimkang.com/quadtreevis/
- https://en.wikipedia.org/wiki/Quadtree
- https://www.ufjf.br/jairo_souza/files/2012/11/CFLP_0028.pdf
- https://www.ufjf.br/jairo_souza/files/2009/12/5-Indexa%c3%a7%c3%a3o-Arvore-AVL.pdf
- https://www.guj.com.br/t/arvore-avl-resolvido/58272
- https://www.programiz.com/dsa/b-tree
- https://www.ufjf.br/jairo_souza/files/2012/11/5-Indexa%c3%a7%c3%a3o-Arvore_B.pdf