Comparações entre árvores AVL e ávores Rubro Negras

Adriano M. Grams[1], **Mateus E. R. da Silveira**[1]

[1] Curso de Ciência da Computação
Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, Brazil

mateus.edival@gmail.com, adrianograms@hotmail.com

Abstract. *This work aims to do an empirical benchmarking between AVL and Red Black trees. The adopted criteria was the time of execution and the number of comparisons made to construct the tree, and to search it. The results show that AVL tree spends more time during construction, but uses almost the same time to perform a search as the Red Black, even though it has a smaller depth than the other.*

Resumo. *Este trabalho tem como objetivo comparar de forma empírica as árvores AVL e Red Black, nos quesitos de tempo de execução e número de comparações feitas, em dois cenários diferentes, o de construção da árvore, e o de busca. Os resultados mostram que a AVL possui um maior tempo de construção, porém, com tempo de busca praticamente idêntico ao da Red Black, mesmo possuindo uma profundidade menor que a mesma.*

1. Introdução

Árvores binárias de buscas são estruturas de dados que fornecem um conjunto de operações para armazenar, consultar e manipular um conjunto de dados, com o benefício de manterem uma ordenação sobre os dados. É comum que outras estruturas como dicionários e filas de prioridades sejam implementados usando árvores binárias de busca. O que torna ela vantajosa em relação a outras estruturas, é que caso ela seja uma árvore balanceada, suas operações de busca possuem $O(log_2 n)$ de complexidade assintótica no pior caso, sendo n a quantidade de elementos, [Cormen et al. 2009]. No entanto, caso a árvore se encontre deformada, ou seja, esteja na condição de uma cadeia linear de elementos, suas operações terão custo de $O(n)$ no pior caso, sendo assim, nada mais que uma lista encadeada simples.

A Figura 1 mostra dois exemplos de árvores. Enquanto a imagem **(a)** é um exemplo de uma árvore balanceada, a imagem **(b)** é um exemplo de uma árvore não balanceada. O elemento mais alto é conhecido como raiz da árvore. Cada elemento do conjunto de dados se torna um nó, e possuirá dois ponteiros para outros nós, seus ”filhos”, um para esquerda e outro para direita. A ideia é que todos os nós a esquerda de um nó possuem elementos menores que o elemento do nó em questão, e todos os nós a direita possuem elementos maiores, caso o objetivo seja manter uma ordenação crescente dos elementos. Essa propriedade é importante para que o tempo logarítmico seja atingindo durante as buscas realizadas, e por isso o momento que um elemento é inserido se torna importante para decidir se uma árvore irá ou não se degenerar [Cormen et al. 2009].

O desafio se torna mitigar o processo de degeneração da árvore durante sua construção e exclusão de elementos. Para tal existem estruturas baseadas em árvores

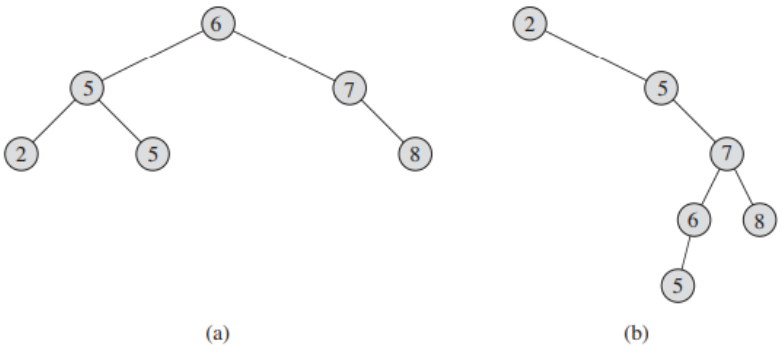

Figura 1. Dois exemplos de árvores binárias de busca [Cormen et al. 2009]

binárias de busca que estipulam um conjunto de regras a serem respeitados durante a inserção e exclusão de elementos. Duas árvores famosas são a *Red-Black*/Rubro Negra (*Red Blacks*) e AVL.

1.1. *Red Blacks*

Árvores rubro-negras são árvores binárias de buscas que além de possuírem um conjunto de regras, cada nó possui um pouco mais de informação, no caso, sua cor [Cormen et al. 2009]. Isso faz com que elas se aproximem do estado de árvore balanceada durante toda sua vida útil. O conjunto de regras que elas devem seguir é:

1. Todo nó é ou preto ou vermelho;
2. A raiz é preta;
3. Toda folha é preta;
4. Se um nó é vermelho, então seus filhos são pretos;
5. Para cada nó, todos os caminhos dele para as folhas abaixo possuem a mesmo número de nós pretos.

A regra **3** traz um novo conceito, a de folha. Folhas são os nós mais profundos de uma árvore, e neste caso carregam consigo o elemento **Nil** ou **Null**, dependendo da implementação. A figura 2 mostra o exemplo de *Red Black* seguindo todas as regras.

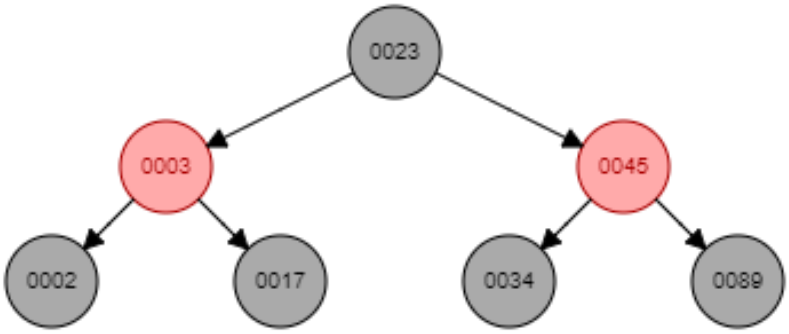

Figura 2. Exemplo de uma árvore rubro-negra [USFCA 2021b]

O resultado das operações de inserção e exclusão podem colocar a *Red Black* em um estado inválido onde a regras estão violadas, já que ambas as operações modificam diretamente a árvore. Para retornar a árvore para um estado válido, respeitando as propriedades, é necessário empregar a técnica conhecida como rotação, que modificam quais elementos são apontados pelos nós, a fim de assegurarem que as regras não estão sendo infringidas. Existem dois tipo de rotação, a esquerda e a direita. Como ambas apenas modificam ponteiros, é uma operação com custo $O(1)$ [Cormen et al. 2009].

1.2. Árvores AVL

Árvores AVL também buscam manter sua estrutura balanceada. Diferentemente das *Red Black*, elas mantêm isso por meio da altura. Para cada nó presente na árvore, a altura das subárvores esquerdas e direitas devem diferir em apenas 1. A altura de cada nó é então armazenado dentro dele. Para que a árvore se mantenha balanceada, ela devem empregar a operação de rotação, assim como as rubro-negras, assim respeitando a regra da altura imposta. A figura 3 apresenta uma árvore AVL, onde podemos ver um atributo associado para cada nó, sua altura [Cormen et al. 2009].

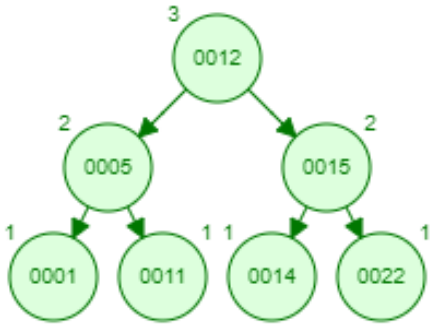

Figura 3. Exemplo de uma árvore AVL [USFCA 2021a]

1.3. Objetivo

Aplicar e corroborar conceitos adquiridos com relação aos métodos de balanceamento de árvores e seu impacto no custo de sua manipulação em termos de tempo e número de comparações, implementando as seguintes estratégias de construção de árvores binárias:

- Árvore AVL;
- Árvore Rubro Negra.

O objetivo também se estende em avaliar o comportamento dos métodos perante um conjunto de testes com diferentes características.

2. Materiais e Métodos

Todos os experimentos foram executados em um computador *Desktop*, com sistema operacional Windows 10, CPU i5-10400, 16GB de RAM e com um SSD de 240GB de capacidade de armazenamento. Os algoritmos de árvore AVL e *Red Black* usados advém do

site GeeksforGeeks e foram implementados na linguagem C++ [GeeksforGeeks 2021a] [GeeksforGeeks 2021b].

Esse trabalho avalia ambas as árvores em duas ocasiões. Durante sua **construção** e **consulta**. Isso verá o quão bem a estrutura balanceia seus nós para manter o tempo das operações de inserção e consulta em tempo logarítmico. Os critérios a serem avaliados são:

1. Durante a construção das árvores:
   (a) Tempo cronológico gasto na construção das estruturas;
   (b) Número de comparações entre chaves realizadas para construir cada uma das estruturas;
   (c) Profundidade da árvore gerada.
2. Durante a consulta às árvores:
   (a) Tempo cronológico gasto na consulta de todos os elementos presentes no arquivo dentro das duas estruturas construídas anteriormente;
   (b) Número de comparações realizadas entre chaves para consultar todos os elementos dos arquivos de consulta nas árvores formadas anteriormente.

Para ambos os casos foram fornecidos arquivos de entradas com diferentes tamanhos. Tanto para construção e consulta foram executados 5 vezes e retirado a média dos tempos de execução.

3. Resultados

Os resultados foram divididos em duas partes, uma para a construção das árvores, e outra para as buscas nelas.

3.1. Construção

Em relação a construção das árvores, os resultados obtidos dizem que em questão de tempo de execução, é possível observar pela Figura 4, que a árvore AVL teve um tempo de execução maior que o da *Red Black*, com uma diferença de **20%** para o arquivo de 250 mil elementos.

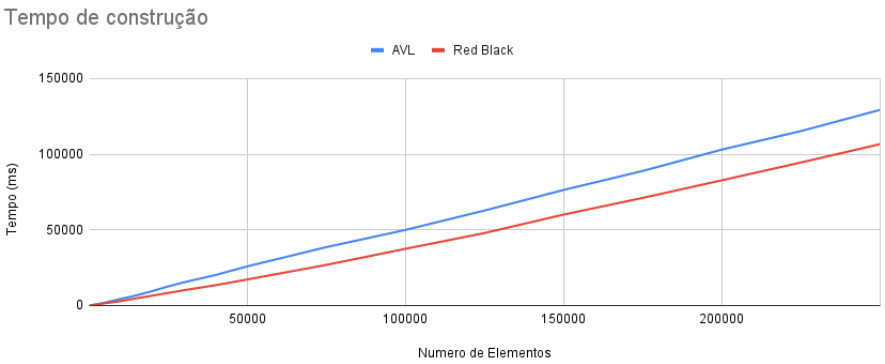

Figura 4. Tempo de construção

Em questão de número de comparações, o resultado foi o inverso em relação ao tempo, como pode ser observado na Figura 5, o número de comparações feitas pela AVL

foi menor que o da *Red Black*, isso devido ao fato que a AVL mesmo tendo um custo maior de construção como um todo, o processo de busca pelo local onde será inserido o novo nó possui menos comparações que a *Red Black*, pois a AVL é uma árvore mais balanceada.

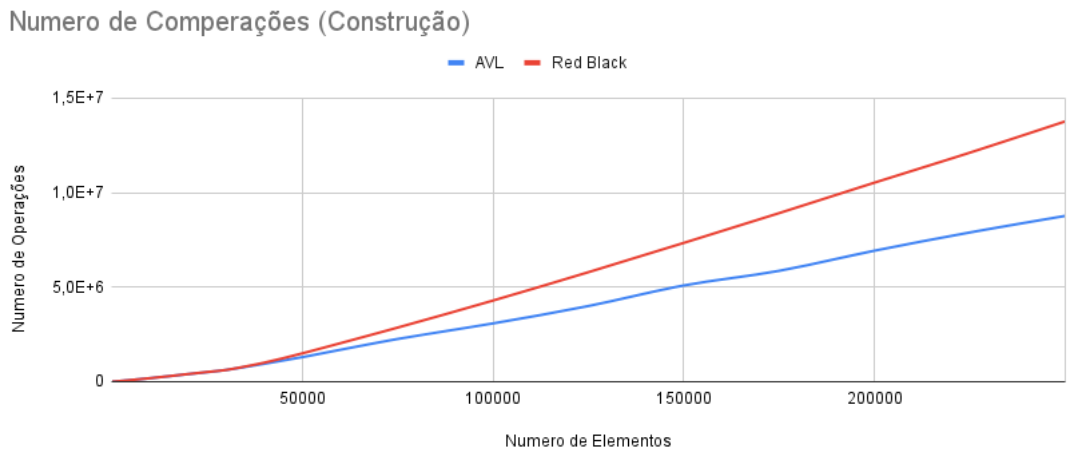

Figura 5. Número de comparações na construção

Para a profundidade das árvores geradas, como pode se observar na Figura 6, os resultados obtidos mostram que a AVL e *Red Black* se mantiveram praticamente empatadas em questão de profundidade até o conjunto de 40 mil, a partir dele, a Red Black teve um salto em sua profundidade, tendo o dobro da profundidade da AVL para o caso de 250 mil, o que corrobora com o fato dela possuir mais comparações na construção da árvore.

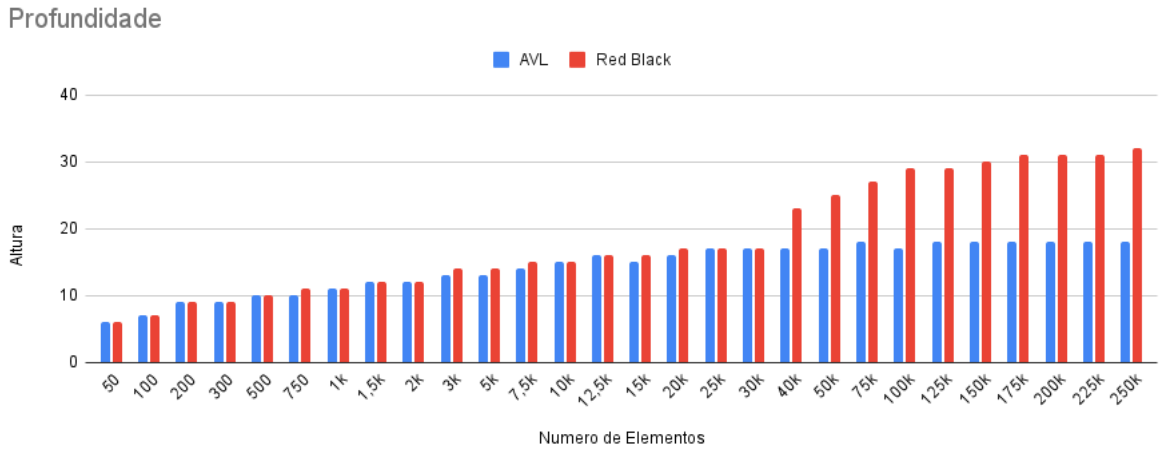

Figura 6. Profundidade

Também é interessante notar que, o valor da profundidade da AVL se refere a miníma profundidade possível para o conjunto de elementos testado, por exemplo, para 250 mil a profundidade é 18, ou seja, o máximo que essa árvore pode comportar de elementos é 2^{18} ou 262.144, se a árvore possuísse uma profundidade 17, não seria possível inserir todos os 250 mil valores.

3.2. Busca

Para as operações de buscas, em relação ao tempo de execução, como pode ser visto na Figura 7, as duas árvores tiveram um resultado muito semelhante, com um pequena vantagem para a AVL. Isso ocorreu possivelmente devido ao conjunto de busca ser o mesmo que o conjunto de construção apenas embaralhado de forma diferente, fazendo com que as buscas não precisem descer até as folhas para achar esses valores, e fazendo com que não haja buscas em vão por valores que não estão na árvore.

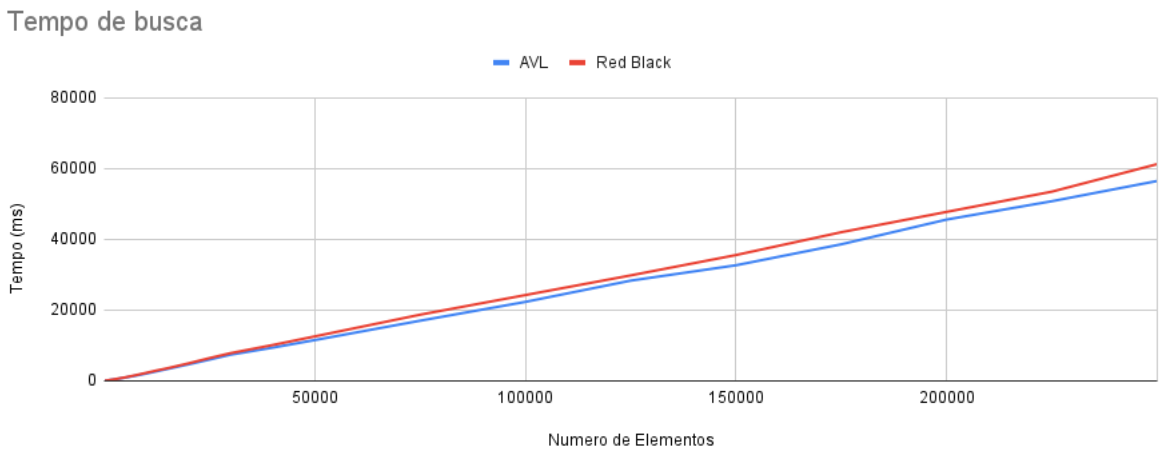

Figura 7. Tempo de busca

Por último, temos o número de comparações, que, semelhante ao caso do tempo de execução para as buscas, foi muito parecido entre ambas as árvores comparadas.

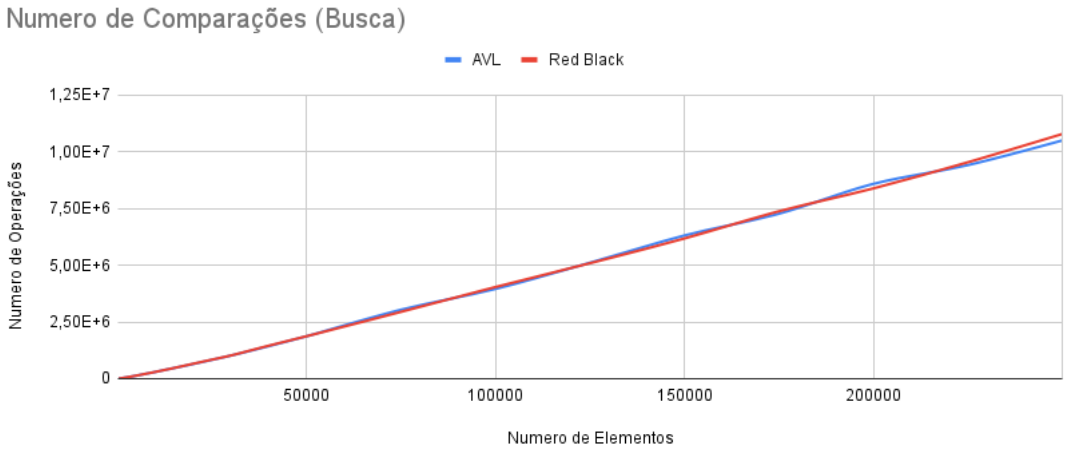

Figura 8. Número de comparações na busca

No Apêndice A está presente as Tabelas 1 e 2, contendo os resultados completos dos experimento realizados.

4. Conclusão

Com isso, concluímos que a árvore AVL possui um custo de construção maior que a *Red Black*, porém, ela possui uma profundidade bem menor em compensação, especialmente para conjunto maiores. Infelizmente não foi possível visualizar as vantagens da AVL para esse conjunto de busca, mesmo com sua profundidade menor, isso não foi possível de se observar no tempo de execução nem no número de comparações, porém, no número de comparações para encontrar as posição de inserção dos nós, é possível ver onde a AVL se sobressai em relação a *Red Black*, cenários onde as buscas tendem a chegar as folhas, e com buscas por valores que não estão na árvore, são os momentos onde a AVL possivelmente mostra um vantagem de desempenho em relação a *Red Black*.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

GeeksforGeeks (2021a). Avl tree — set 1 (insertion).

GeeksforGeeks (2021b). C program for red black tree insertion.

USFCA (2021a). Avl tree.

USFCA (2021b). Red/black tree.

Appendices

Apêndice A

Tabela 1. Resultados AVL

AVL					
Arquivo	**Construção (ms)**	**Comparações (construção)**	**Busca (ms)**	**Comparações (busca)**	**Profundidade**
50	24,6	356	14,6	556	6
100	35	795	19,6	1354	7
200	62	1968	33,2	3224	9
300	104,6	3095	48,8	5212	9
500	159,2	5772	78,4	9546	10
750	243,8	9279	117	15301	10
1000	333,4	13163	156,2	21429	11
1500	517	20893	257,8	34444	12
2000	710,6	29041	336,8	47946	12
3000	1133,6	46397	520	76412	13
5000	1948,6	83348	919,6	137000	13
7500	3124,4	133011	1462,6	219078	14
10000	4386,4	181397	2049,4	298677	15
12500	5509,2	235920	2699,8	388312	16
15000	6730,4	287322	3343,2	472544	15
20000	9567,8	397622	4641	647979	16
25000	12629,2	501512	5993,4	833221	17
30000	15345,2	618037	7384,6	1020734	17
40000	20175,6	943098	9378	1433800	17
50000	25869,2	1295734	11514,8	1869096	17
75000	38587,4	2257159	16992,2	3039809	18
100000	49943,8	3080119	22313,4	3955187	17
125000	62768,6	3999571	28321,2	5110263	18
150000	76369,8	5087902	32665	6310371	18
175000	88927,2	5861368	38595	7269287	18
200000	103019	6926734	45584,4	8583597	18
225000	115251	7890332	50773	9410449	18
250000	129273	8766054	56491	10497963	18

Tabela 2. Resultados Red Black

Red Black					
Arquivo	**Construção (ms)**	**Comparações (construção)**	**Busca (ms)**	**Comparações (busca)**	**Profundidade**
50	25	363	14,2	555	6
100	32,2	825	20	1346	7
200	55,2	1962	38,6	3235	9
300	80,8	3128	52	5120	9
500	127,2	5830	84,2	9480	10
750	190,4	9329	126	15409	11
1000	249,8	13218	166,6	21570	11
1500	399,8	20977	283,8	34561	12
2000	554,8	29333	358,2	48222	12
3000	796,4	46583	561,2	76936	14
5000	1471	83713	981,8	137373	14
7500	2135	132805	1599	218361	15
10000	2919	181803	2257,6	300238	15
12500	3829	235234	2931,6	387240	16
15000	4726,8	288513	3546,8	474377	16
20000	6532	400934	4934,4	659672	17
25000	8303,4	509695	6422	841082	17
30000	10116,2	618230	7819,6	1023684	17
40000	13423,8	1002996	10117	1455407	23
50000	17206	1496292	12557	1871583	25
75000	26844,8	2859193	18692,6	2945612	27
100000	37468,4	4288888	24243,2	4039468	29
125000	47894	5789234	29802,8	5088489	29
150000	60089,6	7337261	35554,8	6178640	30
175000	71112,2	8912339	42010,2	7376422	31
200000	82738,8	10524469	47750,4	8383882	31
225000	94498	12110753	53474,8	9536469	31
250000	106632	13764520	61253,4	10782453	32