Uma comparação do cálculo da mediana de inteiros contidos em árvores AVL e Rubro-Negras

Edwardes Amaro Galhardo [1],
Cássio Martins Carlos [2],
João Augusto Arce Santana [3],
Vinícius Carvalho Lopes [4] e
Antonio Carlos de Oliveira Júnior [5]

Data de submissão: 28/2/2020. Data de aprovação: 1º/4/2020.

Resumo – Este trabalho apresenta uma abordagem para calcular a mediana das chaves, após a sua inserção em estruturas de dados do tipo árvores AVL e árvores Rubro-Negras. Por meio da linguagem C, realizam-se inserções de chaves nas duas árvores, com o número de nós variando entre 10 e 2.000.000. Além de realizar o cálculo da mediana das chaves contidas nas duas árvores, realizaram-se comparações da eficiência para obtenção das medianas através de análises das alturas e do número de rotações das árvores. A partir dos resultados encontrados, nota-se que as duas estruturas possuem uma complexidade assintótica *O(n)* para encontrar a mediana, porém, a árvore Rubro-Negra apresenta um desempenho melhor para a obtenção da mediana dos números inteiros contidos nela.
Palavras-chave: Análise assintótica. Árvores binárias. Estrutura de dados. Mediana.

A comparison of the calculation of the median of integers contained in AVL and Red-Black trees

Abstract – This paper presents an approach to calculate the median of the keys, after their insertion in data structures like AVL-Trees and Red-Black-Trees. Through by the C Language, key insertions are performed in the two trees, with the number of nodes ranging from 10 to 2,000,000. In addition to calculating the median of the keys contained in the two trees, efficiency comparisons were made to obtain the medians through analysis of the heights and number of rotations of the trees. From the results found, it was noted that the two structures have an asymptotic complexity O(n) to find the median, but the Red-Black-Tree presents a better performance to obtain the median of the integers in it.
Keywords: Asymptotic analysis. Binary trees. Data structure. Median.

Introdução

Uma das tarefas mais importantes em ciência da computação é a de encontrar itens em uma lista. Uma lista ordenada possibilita encontrar os elementos com um custo assintótico menor do que, por exemplo, por meio de uma árvore binária de busca (SZWARCFITER, 1994; CORMEN, 2009; CAPELLE, 2019).

[1] Mestrando do Programa de Pós-Graduação em Ciência da Computação do Instituto de Informática - UFG. Professor EBTT/Computação do Instituto Federal do Tocantins - IFTO.
*edwardes.galhardo@ifto.edu.br
[2] Mestrando do Programa de Pós-Graduação em Ciência da Computação do Instituto de Informática - UFG.
*cassiomartinsc@gmail.com
[3] Mestrando do Programa de Pós-Graduação em Ciência da Computação do Instituto de Informática - UFG.
*joao.arce@gmail.com
[4] Mestrando do Programa de Pós-Graduação em Ciência da Computação do Instituto de Informática - UFG.
*vinilopes03@gmail.com
[5] Professor Adjunto 4 do Instituto de Informática da Universidade Federal de Goiás - UFG.
*antoniojr@ufg.br

Árvore é uma das estruturas de dados baseada em nós e que tem sido amplamente utilizada na programação. Estruturar os dados no formato de árvore significa estabelecer uma relação hierárquica entre os nós. Para cada nó de uma árvore, é designada uma relação com os outros nós, semelhante ao padrão utilizado em árvores genealógicas. Em uma árvore binária, um nó recebe no máximo dois filhos: um filho esquerdo e um filho direito. Para inserir um novo nó na árvore, é associada uma chave ao referido nó, sendo este posicionado de modo que sua chave seja maior que as chaves de todos os outros nós em sua subárvore esquerda, e menor que as chaves de todos os demais nós em sua subárvore direita. Este procedimento recursivo é utilizado para formar a árvore binária de busca para uma lista de itens.

Portanto, árvores têm inúmeras aplicações, sendo usadas para analisar circuitos elétricos, representar a estrutura de fórmulas matemáticas, organizar informações em sistemas de banco de dados, apresentar a estrutura sintática de programas fonte em compiladores, aplicações em ciência de redes, e muitos outros (ROSEN; KRITHIVASAN, 2012).

Em ciências de redes essas arvores são aplicadas em redes *Peer-to-Peer*, por exemplo, em Jagadish et al. (2005) é proposto a construção de uma rede sobreposta *Peer-to-Peer* com base em uma estrutura de árvores, denominados no estudo de BATON (*Balanced Tree Overlay Network*), que tornam o menor custo de algumas operações como por exemplo, na rede BATON o custo para atualizar as tabelas de roteamento é $O(log\ n)$ enquanto que em redes genéricas o custo é de $O(log^2\ n)$.

Uma ordenação hierárquica de um número finito de objetos pode ser armazenada numa estrutura de dados em árvore. Com isso, a complexidade de muitas operações em árvores binárias de busca é logarítmica. Dada uma árvore binária de busca, contendo números inteiros, pretende-se encontrar a mediana das chaves dessa árvore de maneira eficiente.

Este trabalho apresenta os resultados obtidos após a implementação do cálculo da mediana das chaves de duas árvores binárias de busca, uma AVL e outra Rubro-Negra.

Materiais e métodos

O objetivo geral deste trabalho é apresentar os resultados da comparação das implementações de cálculo da mediana das chaves contidas nas estruturas de dados do tipo árvore AVL e Rubro-Negra.

Para isso, foi necessário implementar as referidas árvores, realizando o percurso *em-ordem* para encontrar a mediana. Pelo fato de as duas árvores (AVL e Rubro-Negra) garantirem alturas logarítmicas $O(log_2\ n)$, fator que mais influencia nas operações, tem-se também a necessidade de comparar com inserções de n chaves aleatórias as operações em cada árvore:

- Comparar o tempo (em milissegundos) do cálculo da mediana com N inserções;
- Comparar o número de rotações;
- Comparar as alturas; e
- Definir o custo assintótico das inserções.

Fundamentação teórica

Mediana é o valor que separa um conjunto de dados em duas partes, uma metade com os maiores valores, e a outra metade com os menores valores do conjunto. Existem duas situações:

Caso 1: Seja o conjunto A de números inteiros ordenados definido como {A} = {1,2,3, ..., n}, onde o total de elementos de A é um número **ímpar**.

$$\textbf{mediana} = \left\lceil \frac{n}{2} \right\rceil$$

Caso 2: Seja o conjunto de números inteiros ordenado definido como {A} = {1,2,3, …, n}, onde o total de elementos de A é um número **par**.

O cálculo da mediana é a média aritmética dos seguintes elementos: n / 2 somado com o elemento (n / 2) + 1.

$$\textbf{mediana} = \frac{\mathbf{n/2 + (n/2) + 1}}{\mathbf{2}}$$

Visto cada caso, para encontrar a mediana dentro de uma árvore, pode-se primeiro analisar com base no número de nós existentes na árvore, o caso mais adequado, e aplicar o referido cálculo.

Árvores binárias de busca

Em uma árvore binária de busca, cada nó contém ponteiros que guardam a chave e os filhos direito e esquerdo, podendo haver outras informações.

Nas árvores binárias, a maioria das operações tomam um tempo que é diretamente relacionado com a altura da árvore. Dessa forma, uma maneira de garantir que as operações sejam feitas próximas do melhor tempo possível é fazer um balanceamento na árvore. Balancear a árvore significa manter a árvore com a altura o mais reduzida possível para a quantidade de nós existentes (SCHEINERMAN, 2017).

Árvores AVL

Uma árvore AVL pode ser vista como uma árvore binária de busca balanceada, ou seja, a diferença entre a altura das subárvores esquerda e direita não pode ser maior que 1 (um).

Ao realizar o balanceamento, garante-se que, no pior caso, o tempo de execução de cada operação comum seja logarítmico, ou *O (log n)*. Por exemplo, a altura de uma árvore AVL com *n* nós é *O (log n)*, independentemente da ordem em que os valores são inseridos (CORMEN, 2013).

A condição de equilíbrio da árvore AVL, também conhecida como fator de equilíbrio do nó, representa uma informação adicional armazenada para cada nó. Isso é combinado com uma técnica que restaura eficientemente a condição de equilíbrio da árvore, chamada *rotação*.

Árvores Rubro-Negras

Uma árvore Rubro-Negra é um tipo de árvore binária de busca balanceada, uma estrutura de dados usada para implementar vetores associativos. É uma estrutura de dados que tem um bom pior caso de tempo de execução para suas operações e é eficiente na prática. Por exemplo, pode-se buscar, inserir e remover em tempo *O(log n)*, onde *n* é o número total de elementos da árvore.

De maneira simplificada, uma árvore Rubro-Negra é uma árvore de busca binária que insere e remove de forma inteligente, para assegurar que a árvore permaneça aproximadamente balanceada (CAPELLE, 2019).

Para assegurar seu balanceamento existem algumas condições que devem ser respeitadas, tais como:

- O nó raiz é preto;
- Se um nó é vermelho, os seus filhos devem ser pretos; e
- Todo caminho nulo contém o mesmo número de nós pretos.

Além disso, o que difere esta árvore da AVL é que existem alguns casos em que não há a necessidade de realizar rotações, apenas mudar a cor do nó. Essa cor é uma informação adicional que se encontra em cada nó.

Resultados e discussões

Para resolver o cálculo da mediana das chaves das árvores, foi utilizado o seguinte algoritmo:

Algoritmo 1 – Calcular Mediana

```
1: MAX ← 2.000.000
2: chavesOrdenadas [MAX]
3: pos ← 0
4:
5: EmOrdem (Arvore r) {
6:      if (r=NULL) {
7:        EmOrdem (r –> esq)
8:        chavesOrdenadas [pos] ← r–> chave
9:        pos←pos+1
10:       EmOrdem (r –> dir)
11:    }
12: }
13:
14: Mediana, (qtd, chaves [ ]) {
15:       valorMediana ← 0
16:       if (qtd % 2 = 0)
17:          valorMediana ← (chaves [qtd/2] + chaves [qtd/2 +1]) /2
18:       else
19:          valorMediana ← chaves [qtd/2 +1]
20:       return valorMediana
21:  }
22:
23: Main ( ) {
24:       EmOrdem ( r )
25:       Mediana (MAX, chavesOrdenadas)
26: }
```

Primeiramente, são armazenadas em um vetor as chaves da árvore. Para isso, efetua-se o percurso *em-ordem* para garantir que as chaves sejam armazenadas ordenadamente. Após isso, de acordo com o número de chaves, é efetuado o cálculo da mediana com base no mais adequado dos dois casos demonstrados anteriormente. A partir desse algoritmo, foi possível comparar:

- Tempo de cálculo da mediana levando em conta as operações de inserção;
- Número de rotações; e
- Altura das árvores.

As configurações da máquina para a realização dos testes e o *software* utilizados foram:

- Processador AMD Quad-Core A12-9720P 3.60GHz;
- RAM DDR4 com capacidade de armazenamento de 8 Gigabytes, frequência de 1600 MHz;
- HD 5400 RPM com capacidade de armazenamento de 1 Tb;
- Dev-C++ version 5.5.2.

As implementações ocorreram em linguagem C utilizando as bibliotecas:

```
#include $<stdio.h>
#include $<stdlib.h>
#include $<time.h>
```

As Tabelas 1 e 2 contêm os números de nós, alturas, número de rotações e o tempo em milissegundos para encontrar a mediana levando em consideração as inserções das chaves nas árvores, obtidas a partir dos testes conduzidos durante este trabalho.

A Tabela 1 apresenta os valores da árvore AVL, e a Tabela 2 apresenta os dados da árvore Rubro-Negra.

Tabela 1 – Resultados dos testes com árvore AVL

AVL			
Nº de nós	**Altura**	**Rotações**	**Tempo (ms)**
10	3	1	0
50	6	28	1
1.000	11	669	2
10.000	15	6.880	11
100.000	19	70.091	164
1.000.000	23	701.377	1.852
2.000.000	24	1.407.540	4.645

Fonte: Dados gerados pelos autores

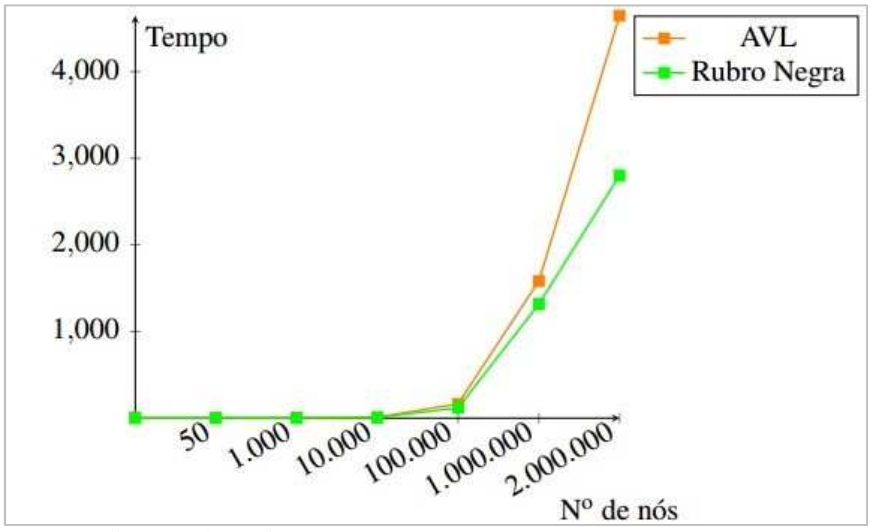
Tabela 2 – Resultados dos testes com árvore Rubro-Negra

Rubro-Negra			
Nº de nós	**Altura**	**Rotações**	**Tempo (*ms*)**
10	3	1	0
50	6	25	1
1.000	11	584	1
10.000	15	5.792	6
100.000	19	58.318	118
1.000.000	25	588.966	1.316
2.000.000	27	1.176.676	2.800

Fonte: Dados gerados pelos autores

De maneira geral, observa-se que, à medida que o número de nós inseridos nas árvores aumenta, o número de rotações, a altura e o tempo também aumentam, conforme o esperado. Também se identifica que, para quantidades pequenas de nós, a diferença das alturas em ambas as árvores é pouco expressiva e, conforme o valor aumenta, essa diferença se torna mais evidente. Além disso, confirma-se que as duas árvores apresentaram alturas logarítmicas em todos os testes.

Figura 1 – Tempo (*ms*) gasto para o cálculo da mediana com *n* inserções

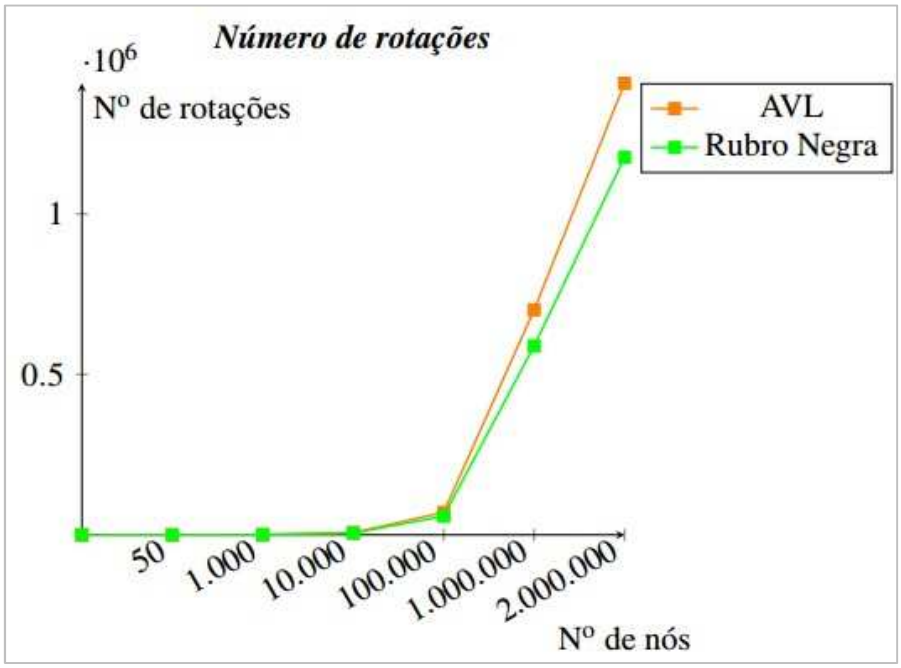


Fonte: Dados gerados pelos autores

A Figura 1 representa a comparação dos tempos em milissegundos para o cálculo da mediana com *n* inserções nas árvores. O número de chaves varia de 10 a 2.000.000 de valores aleatórios e não repetidos. Cada linha representa uma das duas estruturas analisadas e pode-se observar que, em se tratando do tempo, existe uma diferença significativa entre as duas estruturas. Para as duas estruturas, o tempo assintótico de inserção é $O(log_2 n)$, porém, a árvore

AVL apresenta um tempo maior para efetuar as inserções das chaves, tendo um aumento aproximado de 66% se comparado com o da árvore Rubro-Negra.

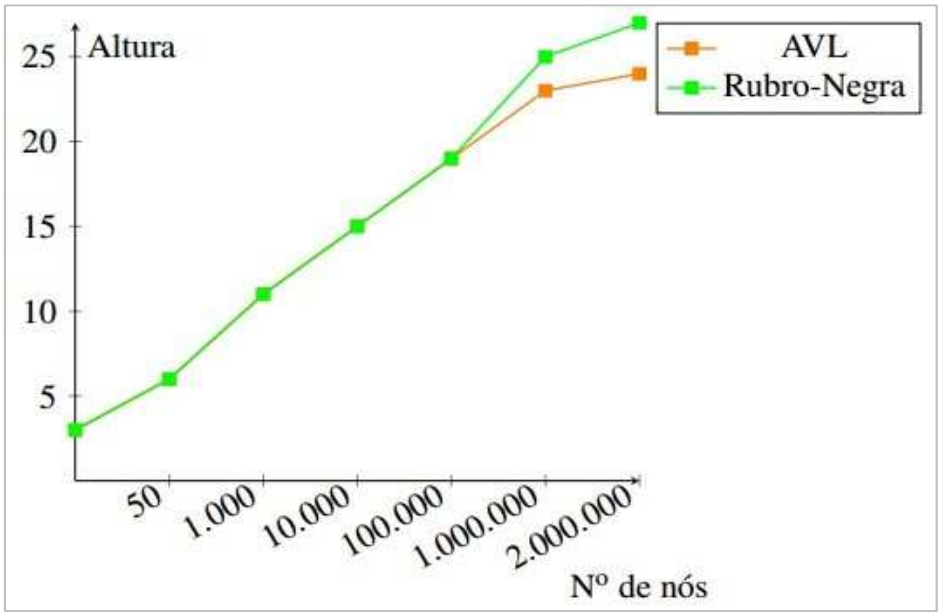
Figura 2 – Comparação do número de rotações



Fonte: Dados gerados pelos autores

A Figura 2 apresenta os dados das comparações dos números de rotações das duas árvores. No gráfico identifica-se uma diferença entre as duas estruturas, a árvore Rubro-Negra faz menos rotações se comparada com a AVL. Isso se deve ao fato de que a Rubro-Negra, para o seu balanceamento, além de utilizar rotações, pode apenas trocar a cor dos nós, fazendo com que o número de rotações seja menor do que o da AVL, que, por sua vez, só faz o balanceamento rotacionando seus nós. Vale ressaltar que isso também vai influenciar o tempo das inserções, pois a cada novo nó inserido na árvore, é necessário balancear toda a árvore novamente.

Figura 3 – Comparações das alturas das árvores



Fonte: Dados gerados pelos autores

A Figura 3 representa a análise da altura das duas árvores durante a inserção das chaves. Somente quando o número de nós possui uma grande quantidade é que é possível visualizar uma diferença significativa nas alturas das duas estruturas. Neste caso, a árvore AVL demonstrou uma vantagem sobre a árvore Rubro-Negra, pois quando o número de nós foi muito grande, a AVL obteve uma altura menor.

Considerações finais

O trabalho foi proposto com o objetivo de apresentar uma abordagem para calcular a mediana de valores inteiros contidos em árvores AVL e Rubro-Negra. Utilizaram-se inserções em ambas as árvores (implementadas em C) com variância entre 10 e 2.000.000 do número de nós para calcular a mediana.

A partir dos testes, foi realizada uma comparação e notou-se que tanto a AVL quanto a Rubro-Negra tem altura logarítmica $O(log_2\ n)$, confirmando o que a literatura menciona (CORMEN, 2013). No entanto, à medida que aumenta o número de nós, as alturas começam a divergir. Assim a AVL começa a ter uma altura menor (Figura 3) e, em algumas operações como as de busca, que não exige balanceamento ou rebalanceamento, terá uma vantagem sobre a Rubro-Negra.

Em contrapartida, a árvore Rubro-Negra, para operações de inserção, possui um desempenho superior à AVL (Figuras 1 e 2), pois, para corrigir o fator de balanceamento, em muitos casos, é necessária apenas uma mudança de cor, que é uma operação de custo menor do que uma rotação. Por isso, inserções e remoções de nós exigem mais rotações nas árvores AVL e recursões a partir da raiz e de volta para a raiz.

Para o problema proposto, de calcular a mediana, um fator importante é o número de nós, pois, quanto maior esse número, mais evidente é a diferença da altura das duas estruturas. Sabendo-se que, a título de comparação, foi utilizado o mesmo número de nós em cada teste nas duas estruturas, a operação mais importante aqui, para definir qual das duas estruturas apresenta um maior desempenho, foi a inserção, operação esta que é diretamente relacionada com a altura das árvores. Logo, a árvore mais indicada seria a Rubro-Negra, já que, para inserções, ela possui menos rotações e assim um menor tempo de execução em comparação com a árvore AVL (Tabelas 1 e 2).

É importante entender que a diferença encontrada durante os testes feitos nesta pesquisa é pouco significativa e em muitos casos a diferença no uso das duas estruturas não terá um impacto muito grande. Também é importante destacar que, para implementações com números de nós exorbitantes, essa diferença pode atingir um nível significativo, mas em casos assim pode-se fazer necessário estudar a possibilidade de uso de outras estruturas de dados que possam ser mais viáveis para o objetivo em questão.

Outro ponto relevante que se deve ter em mente quando se está analisando os resultados aqui demonstrados é que a implementação do algoritmo implica nos resultados encontrados. Outro algoritmo poderia apresentar resultados diferentes dos aqui apresentados, tornando essa diferença mais ou menos evidente.

Uma sugestão de trabalho futuro, para que se possa encontrar resultados mais concisos sobre qual das estruturas tem melhor desempenho em determinadas atividades, é comparar também operações de remoção e busca nas duas árvores.

Referências

CAPELLE, Márcia R. Estrutura de Dados e Projeto de Algoritmos: EDPA.2019.1270 slides

CORMEN, Thomas H. How to Describe and Evaluate Computer Algorithms. 2013.

CORMEN, Thomas H. et al. Introduction to algorithms. MIT press, 2009.

JAGADISH, Hosagrahar V.; OOI, Beng Chin; VU, Quang Hieu. Baton: A balanced tree structure for peer-to-peer networks. In: Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, 2005. p. 661-672.

ROSEN, Kenneth H.; KRITHIVASAN, Kamala. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.

SCHEINERMAN, Edward R. Matemática Discreta: Uma introdução. Thomson Pioneira, 2017. (Tradução da 3ª ed. norte-americana)

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. Estruturas de Dados e seus Algoritmos. Livros Técnicos e Científicos, 1994.