Análise Comparativa Entre Estruturas de Dados Baseadas em Árvores

Lucas Vinícius Silva dos Santos

Universidade Federal do Piauí, Campus Picos (UFPI)
Brasil

vinicius.lucas@ufpi.edu.br

Resumo. *Tratar grandes volumes de dados em tempo de execução tende a se tornar uma complicação para usuários finais e desenvolvedores. Para isso, as estruturas de dados são amplamente utilizadas em softwares complexos, principalmente aqueles que necessitam realizar intensas operações de busca e inserção. Esse artigo técnico apresenta um experimento comparativo entre estruturas de dados baseadas em árvores, unicamente com o proposito de medir a eficiência em realizar operações computacionais em tais estruturas.*

1. Introdução

Estruturas de dados são técnicas fundamentais para a manipulação de dados em execução de softwares. Tratar grandes volumes de dados em tempo de execução tende a se tornar uma complicação para usuários finais e desenvolvedores. Quanto mais complexo se torna um software, maior é a demanda de custo computacional que ele exige ao ambiente de execução. As estruturas de dados servem para contornar esse problema, permitindo um gerenciamento de recursos de maneira mais inteligente e eficaz, evitando que operações de buscas, inserções e manipulação tornem-se transtornos para usuários e desenvolvedores.

As estruturas de dados baseadas em árvores são estruturas amplamente utilizadas para realizar as operações citadas, permitindo a manipulação de dados de forma hierárquica. Existem diferentes estruturas de árvores indicadas para se aplicar em variadas situações, cada uma com suas vantagens e limitações e formas únicas de funcionamento. Esse artigo técnico apresenta uma análise comparativa entre duas diferentes estruturas de árvores: Binária de Busca e AVL. Para realizar tal comparação foram implementados quatro algoritmos distintos para medir o desempenho obtido ao realizar operações nessas estruturas. O desempenho foi medido através do tempo de busca dentro dessas estruturas e o número de passos necessários para encontrar determinado elemento.

O restante desse artigo técnico está organizado da seguinte forma. A Seção 2 apresenta o background com os conceitos básicos dos assuntos tratados. A Seção 3 apresenta a metodologia implementada para realizar os experimentos de comparação. A Seção 4 apresenta os resultados e comentários dos experimentos comparativos. A Seção 5 traça algumas conclusões finais e comentários do trabalho realizado.

2. Background

Essa seção apresenta os conceitos dos assuntos tratados nesse relatório, explanando brevemente sobre os conceitos relacionados a estruturas de árvores, árvores Binárias de Busca e árvores AVL.

2.1. Estruturas de Árvore

As estruturas de dados de arvores são estruturas nomeadas assim por apresentarem um aspecto representativo semelhante a árvores da vida real [Cormen et al. 2002]. Diferente das estruturas de listas encadeadas, aqui as informações não são dispostas em forma sequencial, nas árvores os dados inseridos são armazenados de forma hierárquica. Nesse tipo de estrutura temos elementos inseridos são armazenados em nós encadeados, cada nó de uma árvore binária aponta para até dois outros nós [Celes et al. 2004]. O nó que inicia a árvore e aponta para outros nós é denominado de raiz, a partir da raiz é possível chegar em qualquer outro nó da árvore. A Figura 1 exemplifica a formação de uma árvore binária.

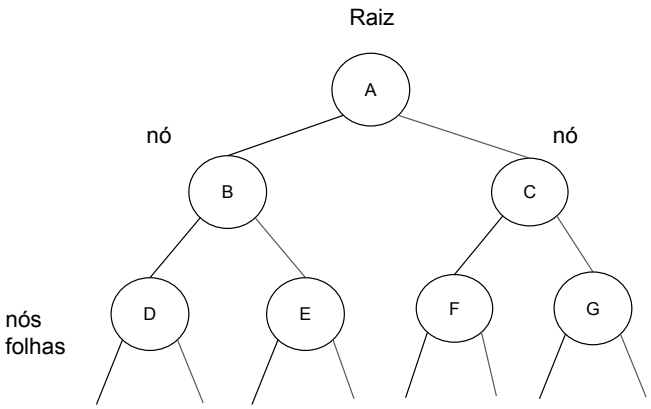

Figura 1. Exemplificação de Árvore Binária

2.2. Árvore Binária de Busca

A Árvore Binária de busca é uma estrutura de dados que adota a mecânica de hierarquias entre os elementos inseridos. Por se tratar de uma árvore binária, cada nó possui no máximo dois filhos. O fator diferencial dessa estrutura está na forma como os elementos são inseridos e organizados dentro da árvore. A Figura 2 exemplifica a formação de uma árvore Binária de Busca. Durante a inserção de um novo elemento, o valor do novo elemento é comparado o valor do nó atual, para então decidir a ”direção”onde esse novo elemento será inserido. É adotado como um padrão de programação que valores maiores ao nó atual são inseridos a direita do nó, enquanto que valores menores são inseridos a esquerda. Dessa forma, a árvore Binária de Busca segue esse padrão em que todos os valores menores que a raiz estão a esquerda, e todos os valores maiores que a raiz estão inseridos a direta. A árvore Binária de Busca proporciona rapidez e eficiência no tempo de busca de um dado, é uma diferença significativa na diminuição de tempo e custo computacional em comparação a estruturas de listas encadeadas.

2.3. Árvore AVL

Uma árvore AVL nada mais é uma estrutura de dados com árvore de forma balanceada. O termo balanceada significa dizer que a diferença na altura das subárvores esquerda e direita não é maior que 1 e nem menor que -1. Dessa forma, garantindo uma maior estabilidade para realizar operações dentro da árvore. A Figura 2 exemplifica a formação de uma árvore AVL. Uma árvore balanceada garante um ganho de performance para as diferentes operações que serão realizadas na estrutura de dados, tais como busca, inserção e remoção.

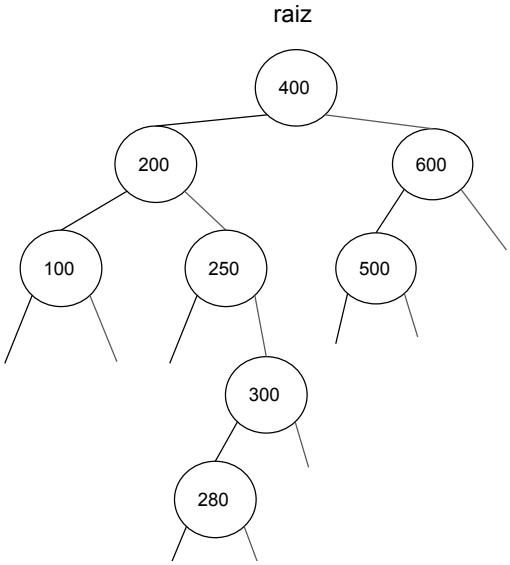

Figura 2. Exemplificação de Árvore Binária de Busca

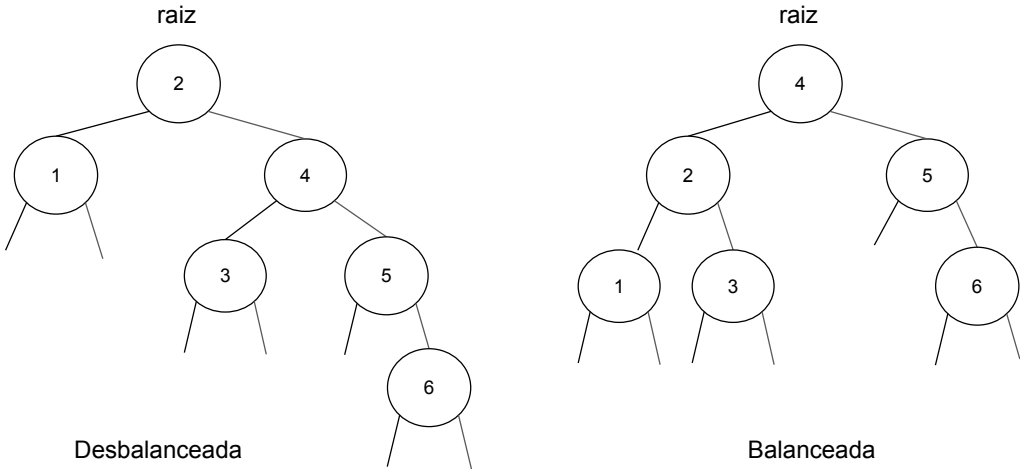

Figura 3. Exemplificação de Árvore AVL

3. Metodologia

Essa seção apresenta a metodologia e ambiente utilizados para realizar a análise e comparação das estruturas de dados citadas. Para o ambiente de testes, foi utilizado um único computador para a execução de algoritmos e medição de dados. As configurações de hardware do dispositivo são apresentadas na Tabela 1.

Tabela 1. Configurações de Hardware

Dispositivo	**Processador**	**Memória RAM**	**Sistema Operacional**
Notebook Acer	Intel i5 de 1.8 GHz	8GB	Ubuntu Linux 18.04 LTS

O experimento consistiu em submeter o dispositivo a execução de quatro algoritmos de estruturas de dados baseados em árvores Binárias de Busca e árvores AVL. Dois algoritmos eram voltados a inserção e busca de valores inteiros, enquanto que os

outros dois algoritmos restantes tinham como objetivo inserção e busca em uma árvore de strings.

3.1. Funcionalidade Utilizadas

Algumas das funções comuns a todos os algoritmos implementados:

1. **criaNo:** Recebe a raiz da árvore e um valor a ser inserido, alocando um espaço de memória necessário para criar um novo nó;
2. **inserir:** Dada a raiz e um nó insere o nó em seu devido lugar e balanceia a árvore se necessário;
3. **Mostrar:** Recebe a raiz da árvore e percorre-a mostrando o valor contido em cada nó;
4. **CalculaAlturas:** Recebe a raiz da árvore e percorre-a recalculando e atualizando a altura de cada nó;
5. **busca:** Recebe a raiz da árvore e um valor a ser buscado, mostrando o caminho percorrido na tentativa de encontrar o elemento;
6. **rotacaoDir / rotacaoEsq / rotacaoDirEsq / rotacaoEsqDir:** Funções utilizadas pelos algoritmos de árvore AVL para rotacionar e balancear a árvore;
7. **maior:** Recebe dois ponteiros referente aos nós esquerdo e direito e retornam a maior altura entre eles;
8. **fatorBalanceamento:** Recebe um nó e calcula se esse nó se encontra balanceado ou não;
9. **liberar:** Responsável pela exclusão de toda a árvore e liberação de memória.

4. Resultados da Execução do Programa:

Essa seção apresenta os resultados obtidos no experimento que teve como objetivo comparar a eficácia na utilização nas estruturas de dados de árvores apresentadas. Nos dois experimentos realizados foram utilizadas como métricas: a variação no tempo de busca e quantidades de passos necessários até alcançar o elemento buscado dentro da árvore.

4.1. Programa de Busca de Inteiros

A Figura 4 apresenta o impacto da altura da árvore e o tipo de árvore utilizada, mostrando a diferença na variação do tempo de busca nas situações apresentadas. Em ambas as alturas de 15 e 20, as árvores AVL e Binária de Busca apresentaram medianas de tempo de busca relativamente próximas, em contrapartida é mostrado uma diferença significativa na varição de tempo das estruturas. O gráfico evidência uma maior variância para o tempo de busca na árvore Binária de Busca em comparação a AVL em ambas as alturas.

O fato da árvore AVL possuir menos variações de tempo do que a árvore Binária de Busca se deve ao graças a AVL se mais estável graças ao seu balanceamento. A árvore AVL, como explicado nas seções anteriores, segue um padrão de balanceamento de ramificações, tornando os tempos de busca de elementos mais próximos entre si. Diferente do que acontece na árvore Binária de Busca, onde a inserção de elementos não possui um balanceamento para regular suas ramificações. Elementos dentro da árvore Binária tendem a ter profundidades maiores a medida que seus valores vão se distanciando da raiz, causando uma maior variação nos tempos de busca.

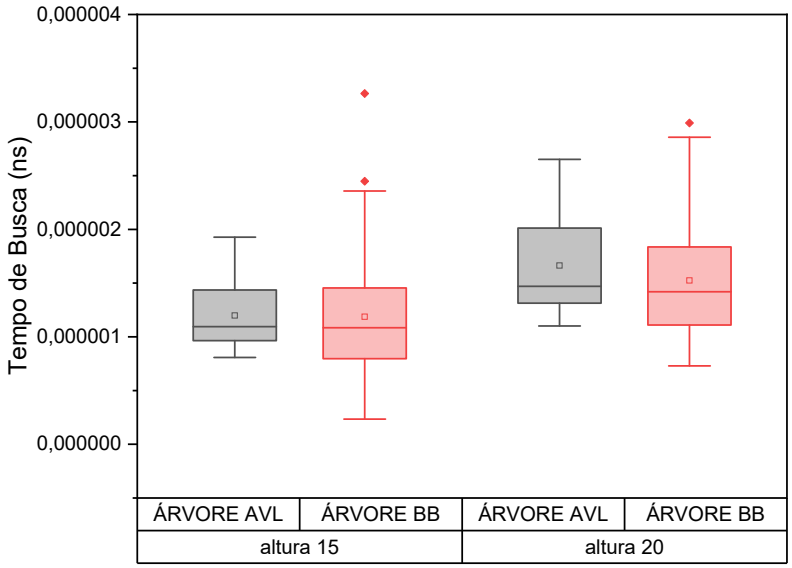

Figura 4. Variação do Tempo de Busca

A Tabela 2 mostra os dados parciais do experimento realizado, apresentando os tempos de busca e número de passos percorridos para encontrar determinado elemento inserido na árvore. Por motivos de exemplificação são apresentados apenas os dados relacionados as primeiras dez buscas realizadas em cada árvore e suas respectivas alturas. Os dados referentes ao números de passos percorridos corroboram com as informações apresentadas na Figura 4. É possível perceber que na árvore Binária de altura 15 o número de passos para encontrar um elemento vária de 5 a 13 passos, enquanto que na árvore AVL esse número vária entre 11 e 14 passos. Nas árvores de altura 20 esse padrão se repete, na árvore Binária o número de passos dados vária de 6 a 15 passos, enquanto que na árvore AVL vária de 16 a 19 passos. Tornando evidente a maior estabilidade e menor variância da árvore AVL graças ao balanceamento de suas ramificações.

Tabela 2. Tempos de Busca e Número de Passos nas Árvores

	Altura 15				Altura 20			
	Árvore BB		Árvore AVL		Árvore BB		Árvore AVL	
N buscados	passos	tempo (ns)	passos	tempo (ns)	passos	tempo (ns)	passos	tempo (ns)
# -1	6	0,00000206	11	0,000001793	6	0,000001223	16	0,000002566
# 375000	9	0,00000192	12	0,000001596	15	0,000001801	18	0,000001893
# 750000	12	0,000002357	12	0,000001326	12	0,000001458	18	0,000001791
# 1500000	7	0,00000132	12	0,000000995	9	0,000001187	17	0,00000163
# 3000000	9	0,000001527	14	0,00000132	12	0,000001523	17	0,00000193
# 4500000	13	0,000002448	14	0,000001339	15	0,000001722	19	0,000001691
# 5250000	11	0,000003264	13	0,000001059	14	0,000001558	18	0,000001574
# 5625000	7	0,000001894	15	0,000001315	10	0,00000115	16	0,000001461
# 5999999	5	0,000001455	14	0,000001143	9	0,000000962	17	0,000001323
# 6000000	5	0,000014897	14	0,000001062	9	0,000000969	17	0,000001265

4.2. Programa de Referência Cruzada

Um segundo experimento foi realizado para novamente analisar a quantidade de passos necessários para encontrar elementos dentro de uma árvore Binaria de Busca e uma árvore AVL. A medição foi realizada em um programa de referência cruzada, onde o conteúdo da árvore é formado por strings, inseridas a partir da leitura de um arquivo de texto.

O arquivo de entrada continha 152 strings para realizar a inserção nas árvores, sendo escolhidas aleatoriamente dez palavras para serem buscadas e observar o número de passos necessários para alcançar a palavra. A Tabela 3 apresenta os resultados obtidos, mostrando as palavras buscadas e o número de passos percorridos dentro das árvores. A Tabela torna evidente a diferença significativa entre a busca realizada nas duas estruturas. A Árvore Binária de Busca novamente apresenta indícios de uma maior variabilidade em relação ao número de passos realizados. Na Árvore Binária de Busca o número de passos vária entre 2 até 12 passo. Enquanto que na árvore AVL essa variabilidade diminuí significativamente, de 4 até 7 passos.

Tabela 3. Tempos de Busca e Número de Passos nas Árvores

	Árvore BB	**Árvore AVL**
Palavra Buscada	**Passos Percorridos**	**Passos Percorridos**
tubarão	2	4
venha	3	6
a	4	6
valor	5	7
normalmente	6	6
claro	6	6
mudança	6	7
das	8	7
convincentemente	10	5
correntes	12	7

5. Conclusão

Esse artigo técnico apresentou uma análise relativa as estruturas de dados baseadas em árvores Binária de Busca e AVL, com o objetivo de comparar a eficacia de ambas as estruturas nas situações apresentadas. Na primeira etapa foi observado o tempo de busca e o número de passos em árvores de valores inteiros. Um segundo experimento foi realizado visando medir exclusivamente o número passos para buscar elementos árvores de strings. Em ambos os casos os resultados relacionados a Árvore AVL se mostraram mais estáveis, apresentando um range de variação tempo e número de passos menor que na árvore Binaria de Busca. Com base nos dados apresentados, é possível afirmar que a árvore AVL é uma opção mais viável para tratar de grandes volumes de dados, dada a sua estabilidade. A árvore Binaria de Busca, por sua vez, apesar da sua instabilidade é uma opção recomendável para volumes de dados menores e que necessitem de um tempo de busca menor.

Referências

Celes, W., Cerqueira, R., and Rangel, J. L. (2004). Introdução a estruturas de dados. *Editora Campus*.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). Algoritmos: teoria e prática. *Editora Campus*, 2:296.