Relatório técnico de Estruturas de Dados 2 sobre árvores binárias de busca

Elievelto Edimar da silva
elievelton@gmail.com
Estruturas de Dados 2 - Juliana Oliveira de Carvalho

Resumo

As árvores são estruturas que servem para armazenar um grande volume de dados, permitindo realizar busca, inserção e remoção de itens de forma muito eficiente, porém, há algumas diferenças entre alguns tipos de árvores, e que podem possuir vantagens ou desvantagens, por isso, ao utilizar uma determinada estrutura devemos levar em consideração para que ela foi projetada e qual problema ela se adequa mais a resolver. Este trabalho tem como objetivo analisar as diferenças no desempenho e eficiência na realização de operações de buscas e inserções executadas em diferentes árvores, sendo elas: árvore binária de busca e árvore AVL.

Palavras-chave: Algoritmos, Estruturas de Dados, Árvores AVL, Árvore Binária de Busca.

Introdução

O trabalho proposto consiste na codificação de alguns tipos de estruturas de dados, a fim de analisar o tempo de execução de operações de busca de alguns elementos nessas estruturas. Após isso, será feito uma comparação entre os resultados obtidos em nanosegundos analisando como as estruturas se comportam, se houve melhoria de desempenho, e qual árvore se mostra mais eficiente.

Para realizar os testes, foram escolhidos 2 tipos de árvores, árvores binárias de busca, e árvores AVL. Após a codificação das funções para manipular as estruturas, foram criadas 30 árvores binárias e AVL de diferentes alturas, e foi realizada a busca de 9 valores distintos em cada árvore criada. Com isso, podemos analisar a diferença de desempenho entre elas levando em consideração o número de passos até achar um elemento das árvores. O tempo será computado ao inserir e ao buscar, no inserir será computado a inserção na criação de cada árvore com 1000 elementos, da mesmo forma a busca será feita dentro de um nó especifíco com 1000 elementos. Também será analisado uma segunda problemática que consiste em organizar índices analíticos de um livro, que teremos que organizar conforme foi proposto na questão e que será também contabilizado o tempo de inserção e de busca para fins de comparativo.

Todos os códigos foram feitos na linguagem C utilizando o editor de texto da Microsoft Visual Studio Code em um computador com sistema operacional **Linux 64 bits, 16 GB de memória RAM, SSD 1T e processador i7**. As árvores foram construídas utilizando structs, alocação dinâmica de memória, TADs e métodos presentes da linguagem, e as operações para manipulação das estruturas foram feitas utilizandorecursividade, visto que, árvores são estruturas originalmente recursivas.

Seções Específicas

Existem diversos tipos de estruturas de dados lineares como: vetores, listas encadeadas, pilhas, filas. Porém, essas estruturas não são eficientes para a realização de buscas ou inserções de novos elementos, pois, os dados estão organizados de modo que o algoritmo precise realizar muitas comparações entre os elementos da estrutura. E uma forma de realizar buscas de forma eficiente seria com a busca binária. Porém, para realizar este modelo de busca em estruturas lineares, os dados precisam estar organizados de forma sequencial.

As árvores são estruturas não lineares, ou seja, cada elemento da árvore poderá ter mais de um sucessor. No caso das árvores binárias de busca, cada nó da árvore pode possuir um ou dois sucessores (Nós filhos). Essa árvore é muito eficiente, pois os dados estão organizados seguindo uma lógica recursiva onde, todos os valores menores que a raiz ficarão na esquerda e todos os valores maiores que a raiz ficarão na direita, permitindo que a busca binária seja feita na sua forma original mesmo que os dados sejam inseridos de forma não sequencial.

Com isso, as árvores são capazes de armazenar uma grande quantidade de dados e realizar buscas de forma muito eficiente. Porém, existe um cuidado na hora de trabalhar com árvores binárias, pois como os dados serão inseridos de forma aleatória, pode ocorrer da árvore se transformar em uma espécie de lista encadeada, tendo uma sub-árvore bem maior que as outras. E para resolver isso, foi criada a árvore AVL, onde, cada vez que for inserido um novo nó, a árvore será reorganizada para mantê-la sempre balanceada. E apesar da árvore AVL ser mais lenta na inserção de valores, ela permite uma organização mais otimizada dos dados, sendo capaz de armazenar um maior número de elementos que na árvore binária.

O projeto está dividido em 2 etapas, o desenvolvimento dos programas para solucionar os problemas cobrados em cada questão e a escrita de um relatório onde deve conter os resultados e conhecimentos obtidos, fazer a comparação entre os resultados obtidos e apresentar casos de usos onde cada uma dessas estruturas se adequa. Para organizar o código desenvolvido para cada questão, foi utilizado uma estrutura de pastas nomeadas por Por (Prova1, Prova2), onde cada pasta contém outras duas pastas chamadas de (Q1,Q2) e nelas estará o código desenvolvido para cada questão. Já o relatório está organizado em 6(Cinco) seções, sendo elas Resumo, Introdução, Seções Específicas, Resultados da Execução do Programa, Conclusão e Apêndice.

Resultados da execução do programa

1 Questão 1

Para a 1ª questão foi desenvolvido um programa utilizando a linguagem C quepossibilita a criação de Árvores BB preenchidas com 1000 números gerados de forma aleatória. O programa permite realizar busca por valores, faz a medição do tempo de cada inserção e busca e salva em um arquivo .txt, permite que você imprima o nível da folha de maiorprofundidade e o nível da folha de menor profundidade e exibe quantas das 30 árvores teve a profundidade menor e maior foram igual 1, 2, 3, 4 e assim por diante.

Para o funcionalidade de busca foi utilizado o algoritmo de percorrimento em árvore pós-ordem, onde ele acessa primeiro o lado esquerdo do nó, depois o direito e só então acessa o nó. Para analisarmos essa estrutura e seus tempos gastos, separamos uma lista com 9 valores, sendo eles: **56, 90, 568, 5902, 10023, 467, 50000, 80458, 1234**. Geramos 30 árvores e realizamos a busca desses valores em cada uma das 30 árvores.

Na tabela 1 podemos observar o tempo gasto para gerar cada uma das 30 árvores. Jána tabela 2, observamos os tempos gastos para realizar a busca de cada valor da sequência em cada árvore. A

tabela 3 demonstra a diferença entre a maior profundidade e menor profundidade e em quantas árvores elas apareceram.

Tabela 1*(Tempo de criação)*

Árvore	**Tempo(s)**	**Árvore**	**Tempo(s)**	**Árvore**	**Tempo(s)**
Arv 1	0.000171	Arv 11	0.000128	Arv 21	0.000120
Arv 2	0.000154	Arv 12	0.000131	Arv 22	0.000140
Arv 3	0.000173	Arv 13	0.000125	Arv 23	0.000141
Arv 4	0.000167	Arv 14	0.000131	Arv 24	0.000136
Arv 5	0.000115	Arv 15	0.000116	Arv 125	0.000150
Arv 6	0.000121	Arv 16	0.000133	Arv 26	0.000159
Arv 7	0.000118	Arv 17	0.000117	Arv 27	0.000140
Arv 8	0.000134	Arv 18	0.000119	Arv 28	0.000152
Arv 9	0.000130	Arv 19	0.000115	Arv 29	0.000119
Arv 10	0.000136	Arv 20	0.000118	Arv 30	0.000142

Tabela 2*(Tempo das buscas)*

Árvore	**Tempo(s)**	**Árvore**	**Tempo(s)**	**Árvore**	**Tempo(s)**
Arv 1	0.000002	Arv 11	0.000003	Arv 21	0.000002
Arv 2	0.000001	Arv 12	0.000002	Arv 22	0.000001
Arv 3	0.000002	Arv 13	0.000001	Arv 23	0.000002
Arv 4	0.000002	Arv 14	0.000002	Arv 24	0.000004
Arv 5	0.000002	Arv 15	0.000002	Arv 25	0.000003
Arv 6	0.000002	Arv 16	0.000002	Arv 26	0.000002
rv 7	0.000001	Arv 17	0.000003	Arv 27	0.000002
Arv 8	0.000002	Arv 18	0.000001	Arv 28	0.000001
Arv 9	0.000002	Arv 19	0.000002	Arv 29	0.000001
Arv 10	0.000001	Arv 20	0.000003	Arv 30	0.000002

Tabela 3*(Contagem da diferença entre maio profundidade e menor profundidade)*

Diferença	**Quantidade**
13	3 vezes
14	5 vezes
15	3 vezes
16	6 vezes
17	5 vezes
18	3 vezes
19	1 vezes
20	4 vezes

Questão 2

A solução para a questão 2 é bem parecida com a desenvolvida para o primeiroproblema, com diferença que para essa a estrutura de árvore utilizada foi a AVL. A árvore AVL é um tipo de árvore binária que se balanceia, ou seja ela se ajusta parase manter balanceada mesmo fazendo inserções ou deletando valores.

Os testes realizados para esse problema foram os mesmos realizados para o problema 1. A sequência escolhida para realizar as buscas foi a mesma usada na questão anterior. Ao compararmos os valores contidos na tabela 1 com os valores da tabela 4, podemos observar que, como citado antes, a Árvore BB acaba fazendo inserções mais rápidas o que a torna uma estrutura ideal para problemas que necessitam de muitas inserções e poucasbuscas.

Já ao compararmos a tabela a tabela 2 com os valores contidos na tabela 5, podemos observar a eficiência que AVL tem em suas buscas. Ao observarmos os tempos obtidos pelo arvore BB é possível observar uma discrepância considerável entre os valores, já olhando os tempos gastos nas buscas realizadas na AVL vemos que a diferença entre os valores é bem pequena. Isso acontece porque os nós que estão no mesmo nível da árvore tem uma quantidade de filhos bem parecida, assim não importa para que lado ela vá realizar a busca, a quantidade de passos percorridos será próximo, todos os nó que estão no mesmo nível possuem uma quantidade de filhos quase igual.

Outro comparativo importante de ser feito é entre as tabela 3 e 6, nelas podemos observar que as diferenças entre a maior e menor profundidade da árvore BB é bem maior que os apresentado pela árvore avl, isso acontece porconta de que a AVL todos os nós filhos estão no mesmo nível, devido a inserção ser feita já de forma a balancear os nós.

Tabela 4*(Tempo de criação)*

Árvore	**Tempo(s)**	**Árvore**	**Tempo(s)**	**Árvore**	**Tempo(s)**
Arv 1	0.014338	Arv 11	0.016176	Arv 21	0.017820
Arv 2	0.012543	Arv 12	0.016043	Arv 22	0.016803
Arv 3	0.015792	Arv 13	0.016116	Arv 23	0.016755
Arv 4	0.017016	Arv 14	0.015616	Arv 24	0.017598
Arv 5	0.014749	Arv 15	0.015943	Arv 125	0.015805
Arv 6	0.015963	Arv 16	0.016142	Arv 26	0.016915
Arv 7	0.016868	Arv 17	0.016034	Arv 27	0.017008
Arv 8	0.017318	Arv 18	0.015940	Arv 28	0.016090
Arv 9	0.019539	Arv 19	0.016694	Arv 29	0.017699
Arv 10	0.014071	Arv 20	0.016433	Arv 30	0.018111

Tabela 5*(Tempo das buscas)*

Árvore	**Tempo(s)**	**Árvore**	**Tempo(s)**	**Árvore**	**Tempo(s)**
Arv 1	0.000002	Arv 11	0.000001	Arv 21	0.000001
Arv 2	0.000001	Arv 12	0.000001	Arv 22	0.000001
Arv 3	0.000001	Arv 13	0.000002	Arv 23	0.000002
Arv 4	0.000001	Arv 14	0.000001	Arv 24	0.000001
Arv 5	0.000001	Arv 15	0.000002	Arv 125	0.000001
Arv 6	0.000001	Arv 16	0.000001	Arv 26	0.000001
Arv 7	0.000001	Arv 17	0.000001	Arv 27	0.000001
Arv 8	0.000001	Arv 18	0.000001	Arv 28	0.000001
Arv 9	0.000002	Arv 19	0.000001	Arv 29	0.000002
Arv 10	0.000001	Arv 20	0.000001	Arv 30	0.000001

Tabela 6*(Contagem da diferença entre maio profundidade e menor profundidade)*

Diferença	**Quantidade**
3	3 vezes
4	27 vezes

Questão 3

O programa desenvolvido para solucionar o problema proposto é um programa que deve simular um índice analítico de um livro. Para a construção dessa solução, foi usado uma árvore BB, onde, cada elemento da árvore é composta por um Termo, uma nova Árvore BB com Sub Termos deste termo e as Páginas que contêm esse termo. Além disso, a sub-árvore de subtermos, contém em cada elemento, o sub termo e as páginas em que esse subtermo aparece. O programa desenvolvido disponibiliza um menu para que o usuário possa inserir ostermos e subtermos, assim como, as páginas que contêm esses termos e subtermos.

Como teste para esse problema foi gerado uma árvore a partir de um arquivo .txt(teste.txt), que está incluso na pasta comprimida, contendo alguns termos, subtermos e suas respectivas páginas e em seguida, através do menu, foi realizada a busca por alguns desses termos(Listados na tabela). Em cada busca realizada foi anotado o tempo gasto para buscar cada termo.

Na tabela 7 podemos observar os termos buscados e o tempo gasto em cada busca.

Tabela 7*(Tempo gasto na inserção e busca de cada termo na Árvore BB)*

Palavras	**Tempo para inserir(s)**	**Tempo para buscar(s)**	**Existe na Árvore**
Casa(termo)	0.000003	0.000001	Sim
Telhado(sub)	0.000004	0.000002	Sim
Parede(sub)	0.000002	0.000002	Sim
Porta(sub)	0.000006	0.000001	Sim
Janela(sub)	0.000002	0.000002	Sim
Carro(termo)	0.000002	0.000001	Sim
Direção(sub)	0.000002	0.000002	Sim
Pneu(sub)	0.000003	0.000002	Sim
Motor(sub)	0.000002	0.000002	Sim

Banco(sub)	0.0000002	0.000002	Sim

Questão 4

Um desenvolvido para resolver esse problema proposto é bem semelhante à solução anterior, com a diferença de uma solução de árvore utilizada para esse problema com a Árvore AVL. Foram utilizados os mesmos animais citados anteriormente e, além desses, foram utilizados os animais de balanceamento para balancear a árvore sempre que necessário. A tabela 8 mostra os resultados ao realizar as inserções e buscas de cada palavra na árvore AVL.

Fazendo um comparativo entre esta tabela e a tabela 7, podemos observar que em quase todas as buscas a AVL foi mais eficaz do que uma árvore binária, realizando buscas mais otimizadas. Isso ocorreu por conta de como uma árvore AVL está organizada, mesmo como ambas as árvores organizadas por sua ordem alfabética, a binária de seus lados possui mais elementos do que o outro lado, sendo que a forma esse equilíbrio ocorre, depende bastante da ordem em que os valores são inseridos.

Já a AVL, além de estar organizado por ordem alfabética, ela está balanceada, o que significa que tanto o seu lado direito quanto o seu lado esquerdo possuem uma quantidade de filhos bem próximos ou até mesmo iguais.

Tabela 8 *(Tempo gasto na inserção e busca de cada termo na Árvore AVL)*

Palavras	**Tempo para inserir(s)**	**Tempo para buscar(s)**	**Existe na Árvore**
Casa(termo)	0.0000003	0.000001	Sim
Telhado(sub)	0.0000005	0.000002	Sim
Parede(sub)	0.0000006	0.000002	Sim
Porta(sub)	0.0000011	0.000001	Sim
Janela(sub)	0.0000005	0.000002	Sim
Carro(termo)	0.0000003	0.000001	Sim
Direção(sub)	0.0000003	0.000002	Sim
Pneu(sub)	0.0000003	0.000002	Sim
Motor(sub)	0.0000003	0.000002	Sim
Banco(sub)	0.0000005	0.000001	Sim

Conclusão

Com os resultados obtidos na execução das buscas realizadas em cada tipo de árvore, podemos concluir que a árvore do tipo AVL, apesar de sua inserção ser mais lenta em relação a binária. Ela se comportou de maneira mais otimizada, mesmo possuindo bem mais elementos. E o tempo de busca da AVL é mais rápido, visto que possui uma altura menor que a árvore binária, mesmo com a quantidade de elementos igual.

Já a árvore binária, pode variar conforme os dados são inseridos. Caso sejam inseridos de forma ordenada, o tempo de busca da será muito alto. Fazendo com que os resultados sejam mais variados e uma performance inferior a AVL. E na escolha de uma estrutura de dados eficiente, a árvore AVL pode ser a melhor opção.

Usar árvore AVL ou BB vai depender do seu problema a resolver, se você precisar inserir milhões de elementos é mais interessante usar a árvore binária, pois usar a AVL se torna mais custoso na hora de inserir, podendo ser até 100 vezes mais lento em relação à árvore binária. Em compensação a árvore AVL possui uma altura menor e se torna mais eficiente ao efetuar uma busca. Já a árvore BB se comportou de modo mais interessante na hora de inserir, entretato, ela poderá ter uma profundidade maior e isso nas buscas pode não ser tão interessante. Portanto a pergunta que se deve fazer é :Vou inserir mais ou vou buscar mais? A resposta será fundamental para escolher qual das soluções deverá ser usada no problema.