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 2 3 e árvore Rubro Negra.

Palavras-chave: Algoritmos, Estruturas de Dados, Árvores 2 3, Árvore Binária de Busca Árvore Rubro Negra.

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 2 3, e árvores Rubri Negra. Após a codificação das funções para manipular as estruturas, foram inseridos 100 elementos nas árvores 2 3 e Rubro Negra, e foi realizada a busca de 30 valores distintos na 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 no momento de realizar as buscas. O problema consiste em simular uma loja de vendas de calçados, que será usado para inserir, remover e editar calçados.

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 2 3, onde, cada vez que for inserido um novo nó, a árvore será reorganizada para mantê-la sempre balanceada. Uma das suas principais vantagens é deixar a árvore perfeitamente balanceada, entretanto uma parte negativa que observados é que a remoção de um elemento pode ser extremamente demorado , devido a árvore precisar sempore se organizar a cada remoção.

Não podemos deixar de falar da árvore Rubro Negra uma solução que é um pouco mais lenta quando possui uma grande quantidade de valores no nó que a árvore 2 3 , pois ela usa de um artifício semelhante a ávore AVL, usando de suas rotações para efetuar o balanceamento, além de usar cores , Preto e Vemelho em vez da profundidade do nó, como acontece na AVL mostrando-se um pouco mais lenta para inserção para grande quantidade de elementos, mas nas questões de buscas ela se mantém com valores bem semelhantes ao árvore 2 3.

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 pasta nomeada por Por (Segunda Prova), em que a 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 que possibilita a criação de Árvores 2 3 preenchidas com 100 objetos calçados. O programa permite realizar busca por ID, faz a medição do tempo de cada busca, então são buscados 30 calcaçados e feito o medição do tempo dessas buscas que serão apresentando mais adiante.

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 30 valores, sendo

eles gerados de forma aleatório.

Na imagem 1 podemos observar o tempo gasto para realizar a busca dos 30 elementos, essa contabilidade foi iniciada quando buscou o primeiro elemento e encerrada no trigéssimo elemento buscado, contabilizando assim o tempo gasto total para essa busca.

Imagem 1*(Tempo de busca)*

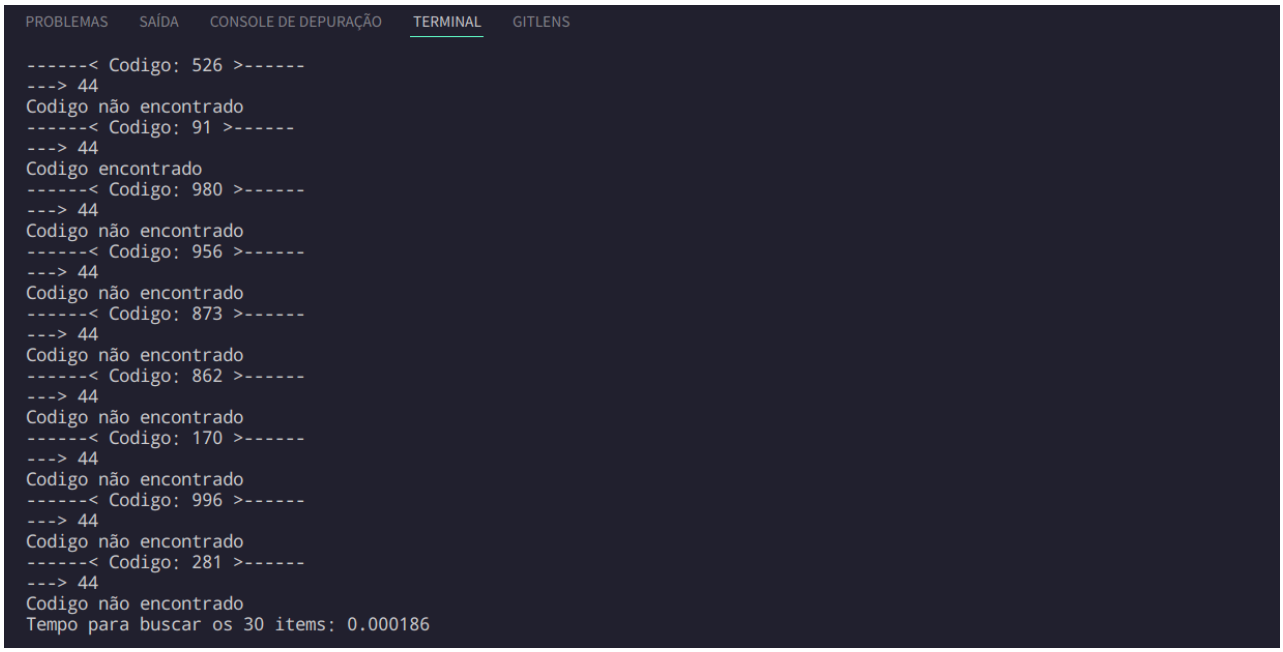
Questão 2

A solução para a questão 2 é bem parecida com a desenvolvida para o primeiro problema, com diferença que para essa a estrutura de árvore utilizada foi a Rubro Negra. A árvore Árvore Rubro Negra é um tipo de árvore binária que se balanceia, ou seja ela se ajusta para se 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 feito de forma aleatório, buscando também elementos que não existissem na árvore. Ao compararmos os valores contidos na imagem 1 com os valores da imagem 2, podemos observar que, como citado antes, a Árvore 2 3 acaba tendo buscas mais custosas, mas isso não é de regra, pois fizemos várias buscas e observamos que os valores ficaram de **0.000175 à 0.000258** ficando bem semelhante as buscas Rubro Negra.

Já ao observar a imagem 2 que está diretamente relacionada as buscas na árvore rubro negra, foram busca mais rápidas, mas também não é de regra e tudo está relacionado diretamente aos elementos buscados e se eles existem um não na árvore, seus valores ficaram bem semelhantes as da árvores 2 3 tendo valores de **0.000168 à 0.000186**.

Imagem 2 *(Tempo de busca)*



Conclusão

Com os resultados obtidos na execução das buscas realizadas em cada tipo de árvore, podemos concluir que a árvore do tipo Rubro Negra, apesar de sua inserção ser mais lenta em relação a árvore 2 3. Ela se comportou de maneira mais otimizada, mesmo possuindo a mesma quantidade de elementos. Mas com os valores bem próximos da árvore 2 3 , isso se deve por as duas árvores serem balanceada, mas devido a árvore 2 3 possuir 2 informações e a possibilidade de percorrer na esquerda, centro e direita, acaba deixando as buscas um pouco mais lentas em relação a árvore rubro negra.

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 Rubro Negra. E na escolha de uma estrutura de dados eficiente, a árvore Rubro Negra pode ser a melhor opção, mas tudo vai depender da quantidade de dados a serem armazenadas, pois uma árvore 2 3 terá uma profuncidade bem menor e isso em uma grande quantidade de dados pode fazer uma enorme diferença.

Usar árvore 2 3 ou Rubro Negra vai depender do seu problema a resolver, se você precisar inserir milhões de elementos é mais interessante usar a árvore 2 3, pois usar a Rubro Negra se torna mais custoso na hora de inserir, isso relacionado diretamente a profundidade da raiz. A árvore Rubro Negra possui uma altura bem maior que a árvore 2 3 e se torna mais eficiente ao efetuar uma busca. Já a árvore 2 3 se comportou de modo mais interessante na hora de inserir, a medida que vai se criando novos nós a inserção poderá ser aplicada sem precisar fazer nenhuma manipulação dos valores, entretato, ela na hora de remover elementos, observamos que é bem mais lenta, pois as remoções precisam fazer diversas movimentaçãoes e rotações de elementos e depois ainda que se precupar com os elementos do centro, coisa que não acontece na árvore Rubro Negra.

Também observamos que os códigos da árvore 2 3 são de mais fácil compreensão que a da árvore Rubro Negra, pois o codificador precisa ficar a todo momento observando os elementos e realizando rotação, simples e duplas, assim como ocorre na árvore AVL, além de se precupar com as cores dos nós, não somento do pai, mas também do imão, tio e avô.

Apêndice

Os códigos utilizados para resolver as questões serão enviados em uma pasta compactada no formato .zip, pois, considerando a quantidade de arquivos e a quantidade de linhas de cada arquivo, seria inviável adicioná-lo a este documento.