TRABALHO DE AEDS II – ENG. DA COMPUTAÇÃO – CEFET-MG

COMPARATIVO ENTRE TEMPO DE EXECUÇÃO INSERÇÃO E PESQUISA PARA ÁRVORES BINÁRIA SIMPLES, AVL E VERMELHO-PRETA COM DIFERENTES VOLUMES DE ENTRADA E PESQUISAS

ALUNO: FREDERICO DANTÉS MACEDO NEVES
MATRICULA: 20203005142

PROFESSOR: MICHEL PIRES SILVA

DATA: 01/12/2021

LINK GITHUB: https://github.com/freddantes/EC-AEDS2-TREE

INTRODUÇÃO

As estruturas de dados são, para a ciência da computação, uma coleção que pode ser composta por valores, com suas interações, e/ou operações, que ocorrem entre estes valores e as estruturas. Ou seja, são estruturas que buscam auxiliar na criação, armazenamento e manipulação dos dados e com isso atingir o objetivo desejado com aquele agrupamento de dados e comandos.

Dentre as estruturas de dados, existe um tipo de estrutura denominado Árvore, que pode ser definido como um conjunto finito de elementos em que cada elemento é chamado nó e o primeiro elemento é chamado de raiz da árvore. Seus elementos são organizados de forma hierárquica, onde o elemento raiz fica no topo e os elementos subordinados a ele são conhecidos como nós filhos, sendo todos os outros que forem subordinados a esses, chamados de nós folha.

A Árvore é uma estrutura que utiliza de ponteiros para a representação dos nós filhos, caracterizando-a como uma estrutura dinâmica. Elá também não é considerada uma estrutura linear, já que os elementos que a compõem, não são armazenados de forma sequencial e nem mesmo estão todos encadeados. Cada elemento armazena um tipo de dado e ponteiros direcionados ao elemento à esquerda e à direita, permitindo que se utilize a recursão para a inserção dos valores na Árvore.

Existem vários tipos de Árvores, sendo que no presente documento serão apresentados três desses tipos existentes, sendo elas a Árvore Binária, Árvore AVL e Árvore Vermelho-Preta. No decorrer do trabalho as características e definição de cada uma delas, será abordada de forma a balizar o comparativo entre as performances de cada uma.

O PROBLEMA

O problema do presente trabalho, consiste na criação e manipulação de 3 bases de dados distintas, contendo cada uma delas 1.000, 10.000 e 1.000.000 de entradas respectivamente, de modo que serão gerados valores aleatórios nesses respectivos volumes. Gerados esses valores que servirão como base, a próxima etapa do problema consistirá em criar para cada base de dado um tipo de Árvore, dentre as três que serão estudadas no trabalho, ou seja, deveremos ter 3 bases de dados para cada uma dos 3 tipos de àrvores. Sendo assim teríamos as seguintes árvores:

- **Árvore Binária**
  - 1.000 entradas
  - 10.000 entradas
  - 1.000.000 entradas

- **Árvore AVL**
  - 1.000 entradas
  - 10.000 entradas
  - 1.000.000 entradas

- **Árvore Vermelho-Preta**
  - 1.000 entradas
  - 10.000 entradas
  - 1.000.000 entradas

Após a criação das Árvores, o passo seguinte será a geração de valores para a busca nessas estruturas, para tal, será utilizado o mesmo procedimento de criação dos valores da base de dados, os mesmo serão geradas de forma aleatória, porém os volumes dos arquivos de pesquisa, serão de 5.000, 10.000 e 100.000 itens a serem encontrados. Ou seja, para cada Árovre criada anteriormente, deveremos realizar 3 buscas, uma para cada arquivo de itens a serem encontrados. Portanto teríamos um esquema de estrutura e pesquisa conforme abaixo:

- **Árvore Binária**
  - 1.000 entradas
    - 5.000 itens
    - 10.000 itens
    - 100.000 itens
  - 10.000 entradas
    - 5.000 itens
    - 10.000 itens
    - 100.000 itens
  - 1.000.000 entradas
    - 5.000 itens
    - 10.000 itens
    - 100.000 itens

- **Árvore AVL**
  - 1.000 entradas
    - 5.000 itens

        - 10.000 itens
        - 100.000 itens
    - 10.000 entradas
        - 5.000 itens
        - 10.000 itens
        - 100.000 itens
    - 1.000.000 entradas
        - 5.000 itens
        - 10.000 itens
        - 100.000 itens

- **Árvore Vermelho-Preta**
    - 1.000 entradas
        - 5.000 itens
        - 10.000 itens
        - 100.000 itens
    - 10.000 entradas
        - 5.000 itens
        - 10.000 itens
        - 100.000 itens
    - 1.000.000 entradas
        - 5.000 itens
        - 10.000 itens
        - 100.000 itens

Os valores inseridos em cada uma das Árvores, não podem ser repetidos, por exemplo, na Árvore AVL de 10.000 entradas não pode existir mais de uma vez o valor 150.00, já no arquivo de pesquisa, é possível que se tenha valor repetido, portanto será possível que se tenha valores repetidos, não existentes ou válidos.

Como metodologia de avaliação do desempenho de cada uma das estruturas, será adotado o tempo de execução do processo, ou seja, será computado quanto tempo será gasto para a realização de todas as inserções, 9 sendo 3 para cada um dos 3 tipos de Árvore, e das pesquisas, que seriam 27 no total sendo 3 para cada uma das bases de dados inseridas.

DESENVOLVIMENTO

ÁRVORE BINÁRIA

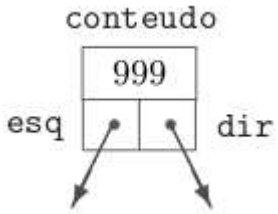
Definição

A estrutura Árvore Binária pode ser uma estrutura vazia, one não possui nenhum elemento, ou se caracteriza por ter um elemento distinto, chamado raiz, e dois ponteiros em direção a outras duas estruturas diferentes que seriam a subárvore esquerda e subárvore direita, sendo assim é possível observar que essa definição passa pela recursividade o que faz com que muitas operações utilizadas nessa estrutura sejam feitas com base na recursão. Isso é um dos motivos para que a Árvore Binária seja bastante utilizada na computação, sendo o principal uso para este tipo de estrutura, a árvore de busca.

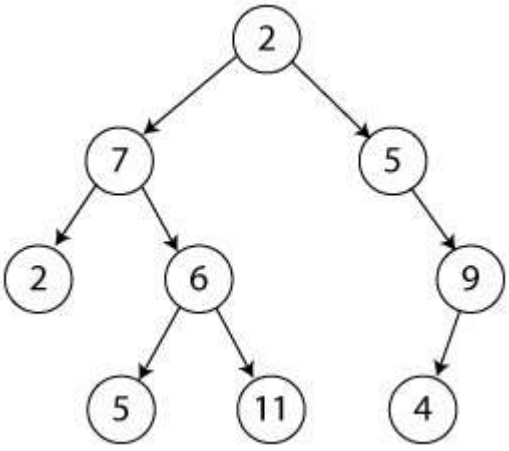
Ela é um conjunto de registros, nós, que satisfaz certas condições, que não são dadas explicitamente, mas que ficarão claras implicitamente pelo contexto. Cada um desses nós possui um endereço, e também cada nó é formado por um valor inteiro, considerado o conteúdo, e dois ponteiros sendo que cada um deles aponta para o endereço do nó filho, um para o filho esquerdo e um para o filho direito.

A profundidade de um nó é a distância deste nó até a raiz, sendo que um conjunto de nós que esteja com a mesma profundidade é conhecido como nível, sendo que a maior profundidade de um nó e a altura da árvore.

```
typedef struct reg {
      int  conteudo; // conteúdo
      noh *esq;
      noh *dir;
   } noh; // nó
```



O campo útil, ou seja, onde estará o dado propriamente dito, neste caso um valor inteiro, neste exemplo está denominado como conteudo, ele é que trará a informação a que se busca, os outros dois campos servirão apenas para dar estrutura à Árvore, ou seja, manter a lógica de organização entre os nós filhos/pais.



Para precorrer a estrutura da árvore, como por exemplo para realizar uma pesquisa, existem três principais ordens para realizar, sendo elas a pré-ordem, em-ordem e pós-ordem. Em pré-ordem a varredura é realizada de forma que o primeiro nó, é o nó raiz, seguido pela subárvore esquerda em pré-ordem e, finalmente, a subárvore direita, também, em pré-ordem. No em-ordem a estratégia adotada é começar pela subárvore esquerda, em-ordem, passar então para o nó raiz e finalizar pela subárvore direita, em-ordem. Já na estratégia pós-ordem primeiro lê-se os nós da subárvore esquerda em pós-ordem, depois passa-se à subárvore direita, também em pós-ordem e finaliza no nó raiz.

Implantação

Para realizar a implantação da árvore binária, utilizaremos os métodos contidos nos códigos expostos em sala, de forma a inserir e estruturar a árvore binária.

- CreateTree() - Método para criação da Árvore Binária Simples, retornando NULL, significa que cria uma árvore vazia.
- InsertTree() - Método para inserção do valor na árvore, insere a estrutura árvore, com o registro e inserindo os ponteiros de filhos.
- Pesquisa() - Método para percorrer a árvore pesquisando o valor estipulado.
- IsInTree() - Verifica se o valor se encontra na árvore, semelhante à pesquisa.
- Antecessor() - Método que fornece ao ponteiro o endereço do antecessor para realizar a remoção.
- RemoveTree() - Método para a remover o registro determinado.
- Preordem() - Método para a percorrer a árvore em pre-ordem.
- Central() - Método para a percorrer a árvore em em-ordem.
- Posordem() - Método para a percorrer a árvore em pos-ordem.

ÁRVORE AVL

Definição

A Árvore AVL, nada mais é do que uma Árvore Binária balanceada, ou seja, é uma Árvore Binária porém nesse caso as alturas das duas subárvores de todo os nós nunca se difere de 1. Nela o balanceamento do nó é definido como a altura de sua subárvore esquerda menos a altura da sua subárvore direita. Sendo assim, cada nó em uma Árvore AVL tem o balanceamento de 1, -1 ou 0, e, em caso de este balanceamento do nó estar diferente de um desses três valores, isso quer dizer que a Árvore não está balanceada.

Pelo fato de a Árvore AVL ser balanceada, em caso de a probabilidade de pesquisar um dado for a mesma para todos os dados, ou seja, não existe predileção entre os dados a serem pesquisados, então a estrutura AVL determinará uma busca mais eficiente, pois o percurso estará em uma distribuição mais uniforme.

Porém ao realizar metodos de inserção e de remoção, pode-se incorrer no problema de desbalancear a árvore, o que ocorreria, por exemplo, caso fosse inserido um nó filho esquerdo em um nó com balanceamento de 1, ou o inverso, um nó filho direito em um nó com balanceamento -1. Caso este fato ocorra, é necessário que se realize o rebalanceamento, ou seja, que o percurso em-ordem da árvore transforamda seja o mesmo da árvore original e que a mesma fique balanceada.

Para isto, é realizada uma manobra na Árvore AVL que é a rotação, que consiste em um movimento que pode ser realizado para a esquerda ou para a direita, a depender do desbalanceamento que necessitará ser corrigido. Essa rotação deve ser feita respeitando-se que o percurso em-ordem original seja mantido e que a mesma fique balanceada. Existem casos de desbalanceamento em que será necessário mais do que apenas uma rotação para que esteja balanceada novamente.

Para o rebalanceamento é necessário calcular qual o Fator de Balanceamento (FB), a fim de verificar qual a toração será realizada e a quantidade de rotação. Esse fator é descoberto através da subtração do valor de altura da subárvore direita do nó, pela altura da subárvore esquerda do nó a se calcular o Fator de Balanceamento. Caso o FB seja negativo, quer dizer que é necessário realizar a rotação para a direita, e em caso do mesmo ser positivo, a rotação será no sentido inverso, a esquerda.

Existem dois casos que ocorrem para o balaceamento, no primeiro o nó raiz tem FB de 2 ou -2, tendo um filho, na direção da inserção, com FB de 1 ou -1, sendo o mesmo sinal do nó raiz, sendo assim a rotação necessária para a correção seria uma rotação simples, seja para a esquerda ou para a direita a depender da necessidade de balanço. No outro caso possível, seria um nó raiz com FB de 2 ou -2, tendo um filho, na direção da inserção, com FB de 1 ou -1, também, porém o que se altera aqui é que o sinal do FB do nó raiz seja inverso ao FB do nó filho, sendo assim será necessário realizar uma rotação dupla, primeiro rotacionando o nó com FB de 1 ou -1 na direção necessária e após isso, realiza-se nova rotação no nó com fator 2 ou -2 porém em direção contrária à rotação anterior.

Implantação

Alguns métodos a serem utilizados na Árvore AVL, obviamente, serão parecidos com os da Árvore Binária Simples por conta da AVL se tratar de uma árvore binária mas com mais propriedades, logo o básico será compartilhado entre elas, por este motivo na implantação só será exposto os demais métodos.

- InsertTree() - Neste caso a única alteração será que ao o bloco terá uma informação a mais, será o peso.

- Rebalance() - Função apra verificar necessidade de balancear.

- GetWeight() - Função para pegar o peso do nó.

- RotacaoSimplesDireita() - Rotação realizada, à direita, para fazer o balanceamento da estrutura.

- RotacaoSimplesEsquerda() - Rotação realizada, à esquerda, para fazer o balanceamento da estrutura.

- RotacaoDuplaDireita() - Rotação realizada, à direita, e depois à esquerda para fazer o balanceamento da estrutura.

- rotacaoDuplaEsquerda() - Rotação realizada, à esquerda, e depois à direita para fazer o balanceamento da estrutura.

ÁRVORE VERMELHO-PRETA

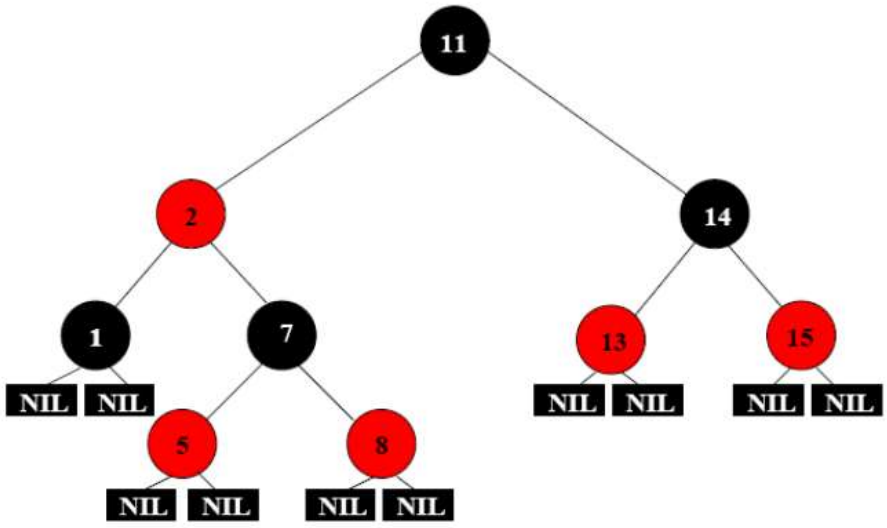
Definição

A Árvore Vermelho-Preta, assim como a Árvore AVL, é também uma Árvore Binária balanceada, mas com algumas particularidades, como também ocorre com a Árvore AVL, o que faz com que ambas sejam estruturas diferenciadas de uma Árvore Binária Simples. Na Árvore Vermelho-Preta os nós folhas (nil), não são, de todo, relevante e não contém dados, não precisando que elas sejam mantidas em memória, bastando um ponteiro apontando para o nulo identificando-as.

Além das regras tradicionais das Árvores Binárias, na Árvore Vermelho-Preta cada nó tem um atributo de cor, que obviamente será vermelho ou preto, e também possuem como requisitos adicionais que:

- Um nó da árvore é vermelho ou preto.
- A raiz é preta.
- Todo nó folha (nil) é preto.
- Se um nó é vermelho então ambos filhos são pretos.
- Para todo nó, todos os caminhos do nó até as folhas descendentes contém o mesmo número de nós pretos.

Para não precisar representar todas as folhas, é possível, para fins de economia de memória, e consequentemente custo, fazer todos os ponteiros de nós folha (nil) apontare para uma mesma folha, chamada de nó sentinela.



Com as regras acima descritas, fica garantida uma propriedade primordial das Árvores Vermelho-Preta, a de que o caminho mais longo da raiz a qualquer folha não seja mais do que duas vezes o caminho mais curto da raiz qualquer outra folha naquela árvore. Dessa forma, é possível considerar a árvore com um balanceamento aproximado, afinal para certas operações consideramos o

tempo de pior caso proporcional à altura da árvore e com o limite do caminho mais longo, garante que a árvore seja eficiente em situações de pior caso.

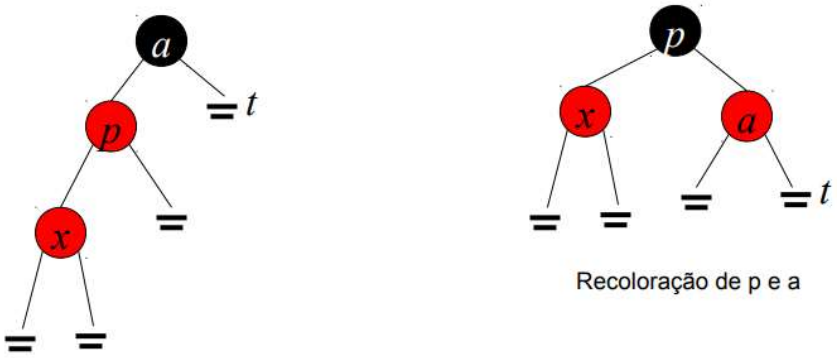
Isso pode ser confirmado se observar que não existem dois nós vermelhos em sequência, pois ambos filhos de um nó vermelho são pretos. Sendo assim o caminho mais curto possível passa por todos os nós na cor preta, enquanto o pior caminho possível é a alternância de nós pretos e vermelhos. Ao considerar a propriedade de que todos os caminhos máximos têm o mesmo número de nós pretos, conclui-se que nenhum caminho é mais do que duas vezes mais longo que qualquer outro caminho, denotando sua eficiência para o pior caso.

Sempre que ocorre alguma operação, como inserção e/ou remoção, assim como na Árvore AVL, é feito uma conferência nas propriedades que garantem à estrutura um estado de balanceamento, e, em caso de alguma das propriedades não ser satisfeita, são realizadas rotações de forma que garantam a correção necessária à estrutura para que ela permaneça balanceada. Esse processo se parece bastante com o processo de balanceamento da outra árvore, porém nesse caso deverá ser alterada também a cor do nó de forma a se manter a propriedade da árvore.

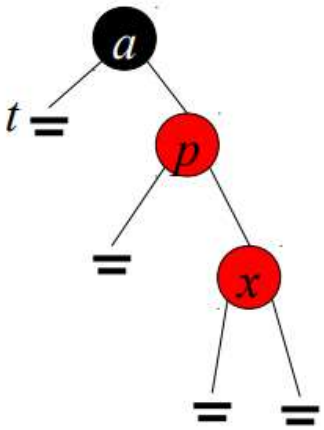
Para manter a propriedade da mesma quantidade de nós pretos nos caminhos, iremos considerar toda inserção como um nó vermelho, de forma a ajustar a cor do mesmo quando necessário. Caso a inserção seja feita em uma árvore vazia bastará alterar a cor do nó inserido para preto de forma a garantir a propriedade de que a raiz é preta.

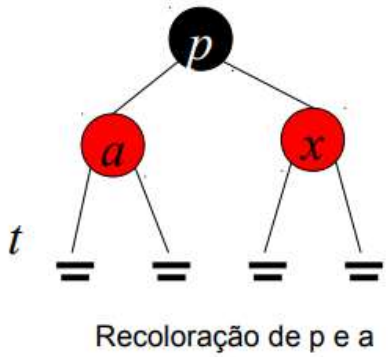
Caso ao inserir o nó x, o tio de x for vermelho, será necessário alterar a cor do avô, tio e pai, caso o avô já seja vermelho é necessário balanceamento, se o avô for a raiz, deverá ser mantido como preto. No caso de o tio do elemento inserido ser preto, será necessário realizar a rotação para que as propriedades sejam respeitadas e neste caso seriam 4 variações de rotação para balancear a estrutura, sendo elas:

- Rotação à direita



- Rotação à esquerda





Recoloração de p e a

- Rotação dupla Esquerda: uma sequencia da Rotação à direita seguida de uma Rotação à esquerda.
- Rotação dupla Direita: uma sequencia da Rotação à esquerda seguida de uma Rotação à direita.

Implantação

- rotacaoEsquerda() - Rotação à esquerda para balancear mas neste caso existe, em caso de necessidade, da alteração da cor do nó.

- rotacaoDireita() - Rotação à direita para balancear mas neste caso existe, em caso de necessidade, da alteração da cor do nó.

- InsertVP() - Inserção do nó, sempre na cor vermelha, e com cor alterada de acordo com as propriedades.

- PesquisaVP() - Pesquisa conforme as outras, mas por conta da particularidade da estrutura, utiliza da variavel cor também.

PERGUNTAS

3.1) - Qual das estruturas é mais rápida para pesquisa e porquê?

Em consulta à bibliografia utilizada durante a elaboração do presente trabalho, foi informado de que para a pesquisa, o mais interessante seria a utilização da AVL, em caso de utilização maior de pesquisa na estrutura, fato este explicado pelo balanceamento da AVL ser maior do que o da Árvore Vermelho-Preta. Porém devido a avaliação dos números obtidos pelo programa, chega-se em uma outra conclusão, pelo menos no caso exemplificado.

Sendo assim, através da análise dos dados obtidos através da medição do tempo de execução dos procedimentos de inserção e pesquisa para cada um dos tipos de árvore e também em cada um dos volume da base de dados e itens a serem pesquisados, chega-se à conclusão que a árvore mais rápida é a Árvore Vermelho-Preta, considerando que ela é balanceada o que a torna mais eficiente que a binária simples, e além disso, por ela possuir um pior caso mais eficiente que o pior caso da Árvore AVL, o que faz com que as repetições de pesquisa sejam menores que as demais. As propriedades adicionais dessa estrutura, Árvore Vermelho-Preta, garantiram que uma pesquisa seja mais eficiente visto que diminuem o maior caminho, o que faz com que em quanto maior o volume de pesquisa, mais eficiente ele se torna.

Ainda considerando a pesquisa, podemos considerar que para uma base de dados menor (1.000), a variação entre a velocidade de execução da Árvore Vermelho-Preta para as outras duas varia de 12% a 19% para mais se considerarmos os volumes de pesquisa variados, tendo uma média de variação de 13.6% (AVL) e 17.5% (Simples).

Considerando a base de dados média (10.000) a variação oscila em uma amplitude maior, saindo de 6.4% (AVL pesquisa 10.000) até 29.1% (Simples pesquisa 10.000), na média a variação fica entre 12.9% (AVL) e 24.8% (Simples).

Passando para a base de dados maior (1.000.000) a variação teve uma amplitude maior, ainda, isso devido ao fato de a menor variação ser o rendimento menor de apenas 4.8% (AVL pesquisa 10.000) frente à maior diferença de 33.3% (Simples pesquisa 5.000), ou seja, um terço mais rápido para a Árvore Vermelho-Preta nessa situação. Na média para este cenário de base de dados a variação seria de 9.4% (AVL) e 27.9% (Simples).

3.2) - Há diferença de tempo das inserções, o que afeta cada estrutura em termos de mecanismo de manipulação / balanceamento?

O tempo de inserção dos dados nas árvores sofre variação conforme o tipo da árvore devido ao fato de cada método de inserção utilizar-se de uma metodologia diferente, como por exemplo, o fato de que na Árvore Binária simples, a inserção ser mais rápida devido ao fato de não existir nenhuma etapa de validação da estrutura, como ocorre nas estruturas balanceadas, o que faz com que o tempo de inserção seja maior ao existir a conferência.

Outro ponto que chama a atenção é o fato do tempo de inserção da AVL, isso devido ao comportamento do procedimento chamado, pois nesse caso o procedimento a todo momento de inserção faz a consulta da estrutura e realiza o balanceamento que for necessário, através das rotações simples e duplas, demandando um tempo maior nesta etapa, mas que provavelmente será compensado no momento da pesquisa, visto que a estrutura estando organizada, o tempo de pesquisa será reduzido.

Com relação à Árvore Vermelho-Preta verifica-se a capacidade da mesma na operação de inserção, afinal se considerarmos que ela possui um melhor pior caso em relação à AVL, isso faz com que a utilização da mesma se torne melhor para a inserção, garantindo uma maior rapidez nesse ponto em relação à AVL e uma vantagem na hora da pesquisa em relação à Árvore Binária Simples, que apesar de possuir uma inserção melhor, compensa negativamente durante a operação de busca, o que torna mais viável para a utilização no que diz respeito à inserção, da Árvore Vermelho-Preta.

3.3) - Considerando as três bases de entrada, para quais você indica cada uma das estruturas e porquê?

Tendo em vista que para definir qual estrutura deve ser usada em cada uma das bases de entrada, devemos também pensar nas demais operações que seriam executadas nessas bases, como pesquisa, a fim de avaliar qual seria mais adequada levando em consideração a sua capacidade e seu custo.

Iniciando pela menor base de dados (1.000), podemos considerar que neste caso, valeria a pena utilizar-se de uma estrutura mais barata como a Árvore Binária Simples, isso devido ao fato de que a inserção dela é a mais rápida dentre todas e a variação de performance dela em relação à demais com essa base de dados é pequena, em torno de 12%, mas que poderia ser compensado na performance do processo de inserção.

Quando passamos para a base de entrada média (10.000) é importante atentar-se para a operação de pesquisa, pois com uma base maior, esta etapa do processo demandará mais tempo, sendo primordial ter uma estrutura mais preparada para tal, no caso uma árvore balanceada. Se compararmos as performances, tanto de inserção quanto de pesquisa, da Árvore Vermelho-Preta, veremos que a mesma se torna a melhor em relação à AVL. Porém existe um outro fator que deve ser observado que é a questão da implantação dela ser mais complexa em relação à AVL, devido ao fato das propriedades adicionais, logo deve-se avaliar a situação a necessidade compensa a maior complexidade, mas fato é que quando a base de dados aumenta, é primordial trabalhar com a estrutura balanceada.

Conforme já dito anteriormente, quanto maior a base de dados, mais necessário é utilizar uma estrutura balanceada de forma a facilitar o trabalho de pesquisa. Sendo assim, ao passar para a base de entrada maior (1.000.000) fica indispensável utilizar-se de uma árvore que seja benéfica tanto na inserção quanto na pesquisa de dados, e como vimos a Árvore Vermelho-Preta leva vantagem na pesquisa, mas, além disso, possui performance bastante superior na inserção devido a verificação de balanceamento dela demandar menos que a AVL, quase 100% mais de velocidade neste caso.

DADOS ANÁLISE PERFORMANCE

Tabela Medidas Inserção

Foi elaborada a tabela com os tempos medidos para a inserção de cada um dos volumes de base de dados, em cada um dos três tipos de árvores estudadas. De posse desses tempos, foi feito um cálculo da Velocidade de Inserção, que basicamente seria o número de inserções (volume da base de dados) dividido pelo tempo medido em segundos, com isso obtemos a quantidade de inserções por segundo para usar como comparativo de performance.

ARVORE	TEMPO INSERÇÃO (seg)	INSERÇÃO / seg
VOLUME BASE DE ENTRADA: 1.000		
Binária Simples	0.001177	849,582
AVL	0.001930	518,162
Vermelho-Preta	0.001239	807,103
VOLUME BASE DE ENTRADA: 10.000		
Binária Simples	0.013046	766,504
AVL	0.026267	380,709
Vermelho-Preta	0.013180	758,748
VOLUME BASE DE ENTRADA: 1.000.000		
Binária Simples	2.222470	449,950
AVL	4.207637	237,663
Vermelho-Preta	2.335276	428,215

Variação Velocidade de Inserção (Inserção / seg)

De posse dessa variável, velocidade de inserção, passamos ao cálculo da variação de um tipo de árvore para a outra, levando sempre em consideração a mesma base de entrada para realizar o comparativo, ou seja, compara-se Simples, AVL e Vermelho-Preta com 1.000 entradas, depois as três novamente com 10.000 e finaliza comparando as três para a base de dados de 1.000.000. Para fazer a variação, consideramos como base o melhor valor, ou seja, a maio velocidade de inserção, que no caso seria a estrutura que adicionou mais dados por segundo, neste caso a Simples.

ARVORE	INSERÇÃO / seg	VAR. INSERÇÃO / seg
VOLUME BASE DE ENTRADA: 1.000		
Binária Simples	849,582	-
AVL	518,162	39.01%
Vermelho-Preta	807,103	5.00%
VOLUME BASE DE ENTRADA: 10.000		
Binária Simples	766,504	-
AVL	380,709	50.33%
Vermelho-Preta	758,748	1.01%
VOLUME BASE DE ENTRADA: 1.000.000		
Binária Simples	449,950	-
AVL	237,663	47.18%
Vermelho-Preta	428,215	4.83%

Tabela Medidas Pesquisas

Conforme feito para as inserções, com as informações de tempo gasto em cada grupo de pesquisa realizado, sendo que no grupo considera-se o tipo da árvore, o volume de pesquisa e o volume da base de dados, logo, possuimos algumas variantes e novamente iremos criar uma variável que servirá como parâmetro de performance. No caso iremos realizar a mesma lógica, pegando o volume de pesquisa e dividindo pelo tempo em segundos de forma a encontrarmos a velocidade de pesquisa que nos fornecerá a quantidade de pesquisas realizadas por segundo.

ARVORE	VOLUME PESQUISA	TEMPO PESQUISA (seg)	PESQUISAS / seg
VOLUME BASE DE ENTRADA: 1.000			
Binária Simples	5,000	0.005621	889,577
AVL	5,000	0.005429	921,065
Vermelho-Preta	5,000	0.004731	1,056,792
Binária Simples	10,000	0.011104	900,597
AVL	10,000	0.011008	908,414
Vermelho-Preta	10,000	0.009635	1,037,904
Binária Simples	100,000	0.106032	943,110
AVL	100,000	0.100174	998,261
Vermelho-Preta	100,000	0.089443	1,118,028
VOLUME BASE DE ENTRADA: 10.000			
Binária Simples	5,000	0.006607	756,819
AVL	5,000	0.006116	817,494
Vermelho-Preta	5,000	0.005607	891,742
Binária Simples	10,000	0.014747	678,093
AVL	10,000	0.012152	822,934
Vermelho-Preta	10,000	0.011421	875,592
Binária Simples	100,000	0.126431	790,948
AVL	100,000	0.121923	820,191
Vermelho-Preta	100,000	0.098890	1,011,224
VOLUME BASE DE ENTRADA: 1.000.000			
Binária Simples	5,000	0.015311	326,560
AVL	5,000	0.012889	387,935
Vermelho-Preta	5,000	0.011482	435,474
Binária Simples	10,000	0.029321	341,050
AVL	10,000	0.023100	432,900
Vermelho-Preta	10,000	0.022047	453,580
Binária Simples	100,000	0.227414	439,726
AVL	100,000	0.211007	473,918
Vermelho-Preta	100,000	0.189526	527,632

Variação Velocidade de Pesquisa (Pesquisa / seg)

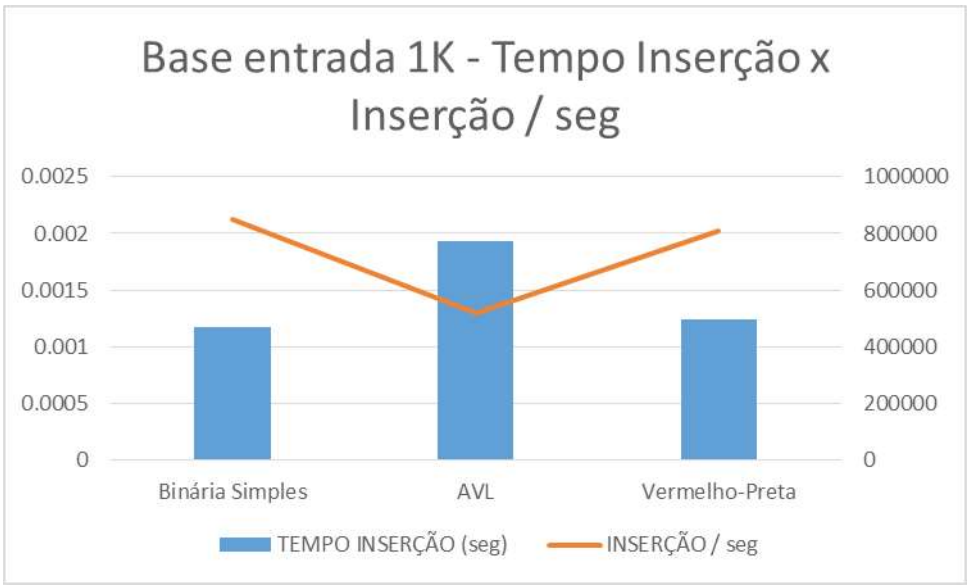
Por se tratar de 3 casos diferentes de base de dados, e depois para cada um desses casos, termos 3 volumes de pesquisas e 3 estruturas diferentes, iremos comparar primeiramente a variação dessa velocidade de pesquisa, caso a caso, levando sempre como base a maior velocidade, e após será feito uma pela média de cada base de entrada.

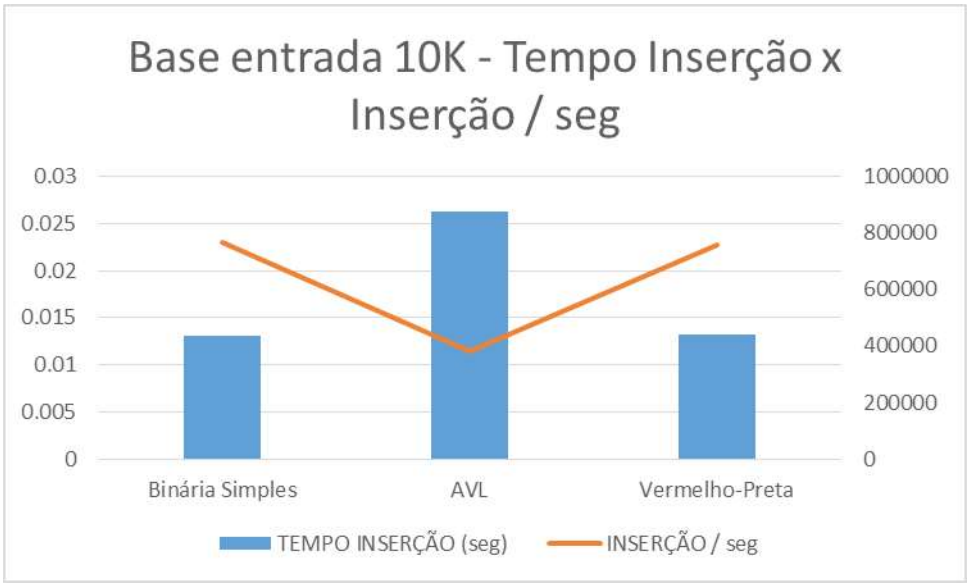
ARVORE	VOLUME PESQUISA	PESQUISAS / seg	VAR. PESQUISA / seg
VOLUME BASE DE ENTRADA: 1.000			
Binária Simples	5,000	889,577	18.80%
AVL	5,000	921,065	14.74%
Vermelho-Preta	5,000	1,056,792	-
Binária Simples	10,000	900,597	15.25%
AVL	10,000	908,414	14.25%
Vermelho-Preta	10,000	1,037,904	-
Binária Simples	100,000	943,110	18.55%
AVL	100,000	998,261	12.00%
Vermelho-Preta	100,000	1,118,028	-
VOLUME BASE DE ENTRADA: 10.000			
Binária Simples	5,000	756,819	17.83%
AVL	5,000	817,494	9.08%
Vermelho-Preta	5,000	891,742	-
Binária Simples	10,000	678,093	29.13%
AVL	10,000	822,934	6.40%
Vermelho-Preta	10,000	875,592	-
Binária Simples	100,000	790,948	27.85%
AVL	100,000	820,191	23.29%
Vermelho-Preta	100,000	1,011,224	-
VOLUME BASE DE ENTRADA: 1.000.000			
Binária Simples	5,000	326,560	33.35%
AVL	5,000	387,935	12.25%
Vermelho-Preta	5,000	435,474	-
Binária Simples	10,000	341,050	33.00%
AVL	10,000	432,900	4.78%
Vermelho-Preta	10,000	453,580	-
Binária Simples	100,000	439,726	19.99%
AVL	100,000	473,918	11.33%
Vermelho-Preta	100,000	527,632	-

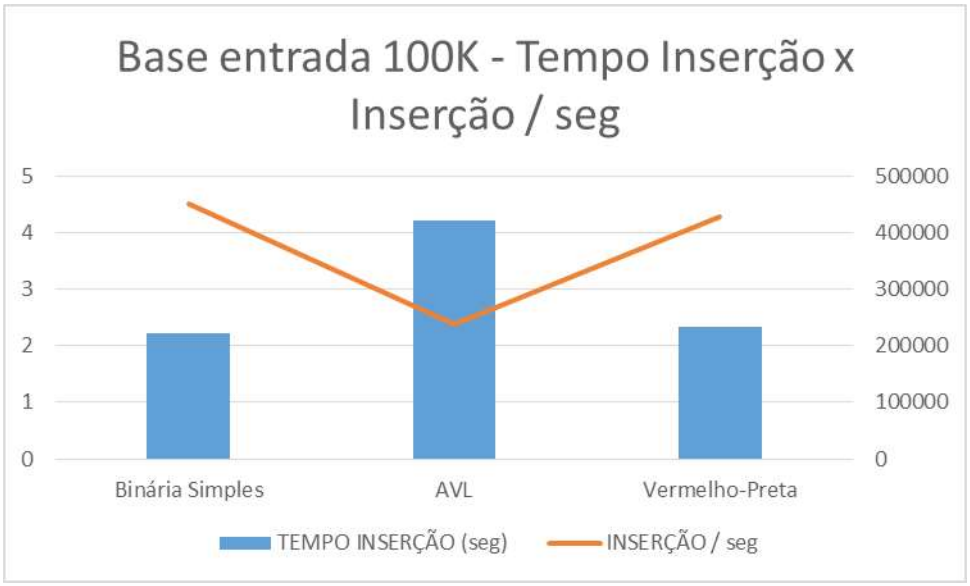
Agora se considerarmos as médias de velocidade de pesquisa para cada volume de base de dados chegariamos na variação dessa velocidade conforme abaixo:

ARVORE	MÉD. PESQUISAS / seg	VAR. PESQUISA / seg
VOLUME BASE DE ENTRADA: 1.000		
Binária Simples	911,095	17.54%
AVL	942,580	13.61%
Vermelho-Preta	1,070,908	-
VOLUME BASE DE ENTRADA: 10.000		
Binária Simples	741,953	24.83%
AVL	820,206	12.92%
Vermelho-Preta	926,186	-
VOLUME BASE DE ENTRADA: 1.000.000		
Binária Simples	369,112	27.94%
AVL	431,585	9.42%
Vermelho-Preta	472,228	-

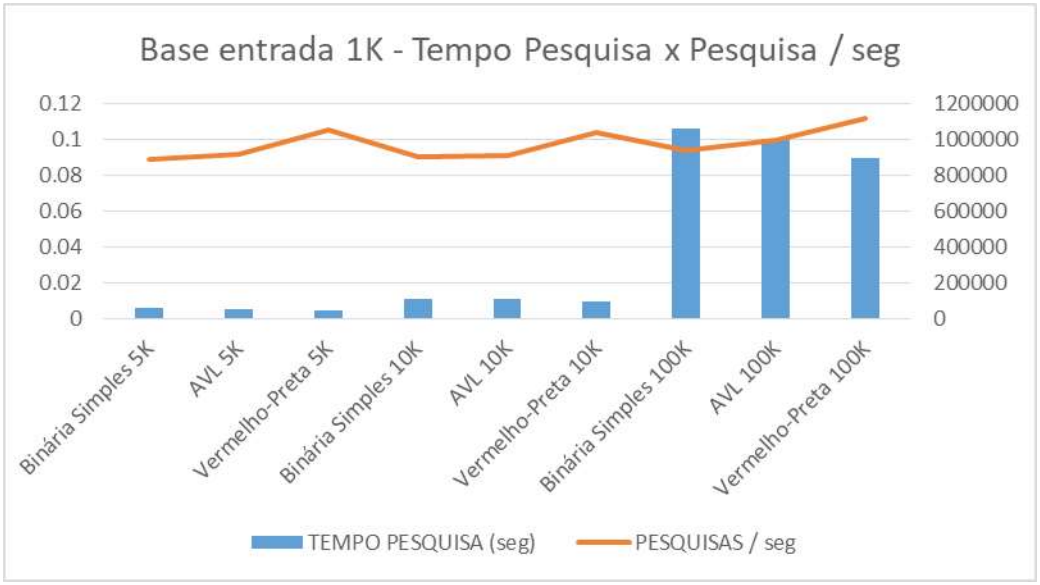
Gráficos Inserção



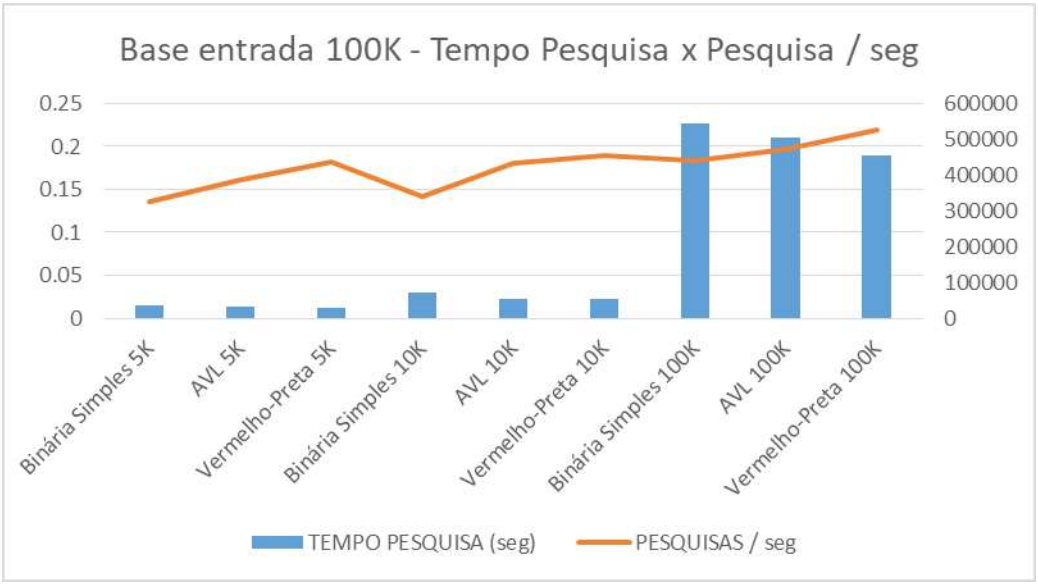




Gráficos Pesquisa




Base entrada 10K - Tempo Pesquisa x Pesquisa / seg
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1200000
1000000
800000
600000
400000
200000
0
Binária Simples 5K
AVL 5K
Vermelho-Preta 5K
Binária Simples 10K
AVL 10K
Vermelho-Preta 10K
Binária Simples 100K
AVL 100K
Vermelho-Preta 100K
TEMPO PESQUISA (seg)
PESQUISAS / seg


Base entrada 100K - Tempo Pesquisa x Pesquisa / seg
0.25
0.2
0.15
0.1
0.05
0
600000
500000
400000
300000
200000
100000
0
Binária Simples 5K
AVL 5K
Vermelho-Preta 5K
Binária Simples 10K
AVL 10K
Vermelho-Preta 10K
Binária Simples 100K
AVL 100K
Vermelho-Preta 100K
TEMPO PESQUISA (seg)
PESQUISAS / seg

CONCLUSÃO

Conclui-se após o desenvolvimento do trabalho e a análise dos dados obtidos, que todas as estruturas de árvore estudadas neste presente projeto, apresentam vantagens e desvantagens, sendo necessário o conhecimento das propriedades e particularidades de cada uma delas, a fim de decidir qual será a estrutura utilizada.

Isto fica claro no momento em que uma estrutura é melhor do que a outra, mas apenas para bases de dados maiores, pois em bases pequenas elas se equivalem, além disso o custo dessa que é melhor em bases grandes é mais elevado, portanto deve-se avaliar se este custo elevado é válido em função da necessidade da estrutura. Foi exemplificado no estudo ao mostrar que por mais que a Árvore Vermelho-Preta seja mais eficiente que a Árvore Binária Simples, não faz sentido utilizar em base pequena visto que seu custo de implementação e computacional é maior.

Além disso, é preciso avaliar o todo da estrutura, ou seja, como ela se comporta perante todas as operações às quais estará sujeita, como por exemplo, saber que embora uma estrutura seja vantajosa no momento de uma inserção, pode ser menos eficiente do que outra em um momento de pesquisa, e vice-versa. Portanto fica comprovada a necessidade de conhecer as estruturas de forma a escolher a mais adequada à situação.

REFERÊNCIAS BIBLIOGRÁFICAS

http://linguagemc.com.br/arquivos-em-c-categoria-usando-arquivos/

https://www.ic.unicamp.br/~francesquini/mc202/files/aula16-18.pdf

http://www.facom.ufu.br/~madriana/PF/TP7-Arvores.pdf

http://www.ic.uff.br/~fabio/Aula-arvores-1.pdf

http://www.facom.ufu.br/~backes/gsi011/Aula12-ArvoreRB.pdf

https://www.ti-enxame.com/pt/data-structures/diferenca-entre-arvores-vermelho-pretas-e-arvores-avl/1071894541/

https://sites.google.com/site/proffdesiqueiraed/aulas/aula-10---arvores