UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS

ARTHUR CAMARGO
PABLO RODRIGUES

RELATÓRIO DE DESEMPENHO ENTRE ALGORITMOS DE ABP E ÁRVORES AVL PARA SOLUÇÃO DE PROBLEMA DE ANÁLISE DE SENTIMENTOS

Porto Alegre
2018

ARTHUR CAMARGO
PABLO RODRIGUES

RELATÓRIO DE DESEMPENHO ENTRE ALGORITMOS DE ABP E ÁRVORES AVL PARA SOLUÇÃO DE PROBLEMA DE ANÁLISE DE SENTIMENTOS

O projeto de comparação de eficiência de algoritmos baseados em ABP e AVL para solução do problema de análise de sentimentos, foi solicitado pela professora Viviane Moreira, com o objetivo de avaliação da disciplina de Estrutura de Dados, do curso de Ciência da Computação do Instituto de Informática da UFRGS.

Porto Alegre
2018

1. INTRODUÇÃO

Este relatório tem como finalidade apresentar os fatores observados na implementação de um algoritmo com base em estruturas de dados ABP e AVL, cujo objetivo é medir o desempenho de ambas estruturas na resolução de um problema de análise de sentimentos.

Aqui será mostrado de maneira detalhada os processos do algoritmo, tais como, a implementação da inserção das palavras léxicas em uma ABP, e o processo na resolução do problema com implementação da inserção das palavras léxicas inseridas em uma AVL. Também, será possível analisar informações de execução destes casos a fim de compará-los para se chegar a uma conclusão.

2. DESCRIÇÃO TEÓRICA DAS ESTRUTURAS DO PROJETO

2.1 Árvores Binárias de Pesquisa

Em ciência da computação, uma Árvore Binária de Pesquisa é uma árvore binária, ou seja, todo nodo tem no máximo 2 filhos, onde, cada nodo é associado a uma chave que determina a sua posição na árvore.
Tendo como característica que:

I. dado um nodo, todos os nodos da subárvore esquerda possuem uma chave menor do que a do nodo e, todos os nodos da subárvore direita possuem uma chave maior do que a do nodo.

2.2 Árvores AVL

Em ciência da computação, uma árvore AVL é uma árvore binária de pesquisa balanceada, ou uma árvore completa, cujo o objetivo é minimizar o número de comparações efetuadas ao percorrer a árvore. Para se manter este objetivo, árvores AVLs tem como característica, o processo de modificação da árvore caso a inserção de um determinado nodo quebre a propriedade de uma árvore AVL, que é:

I. A altura da subárvore direita de um nodo difere da aultura da subárvore esquerda deste nodo de no máximo 1.

O valor obtido da diferença da altura da subarvore esquerda de um nodo com a altura da subarvore direita deste mesmo nodo é chamado de **fator de balanceamento** do nodo. Todos os nodos de uma árvore AVL obedecem um fator que varia entre -1 e 1.
Se um nodo conter um fator de balanceamento menor do que -1 ou maior do que 1, a árvore perde a propriedade de uma AVL. Logo é necessário realizar uma rotação para balancear a árvore novamente.

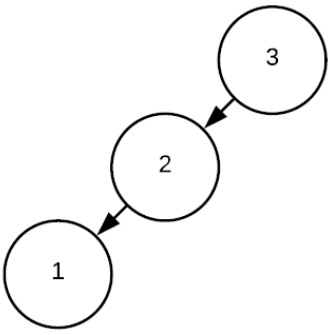

Figura 2.1: ABP desbalanceada

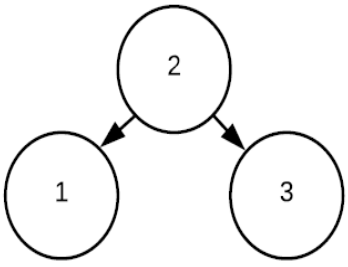

Figura 2.2: árvore AVL

3 . IMPLEMENTAÇÃO DO ALGORTIMO

Por consetimento da dupla, a estrutura de dados escolhida entre AVLs, Rubro-Negras e Árvores Splay, foi a estrutura AVL, já que se mostra adequada para percorrer uma Árvore Binária de Pesquisa quando se necessita de muitos acessos à árvore e pelas outras estruturas serem mais complexas de se implementarem. Embora, em casos de inserção em AVLs, o processo em algumas situações não seja o mais eficiente.

3.1 Inicialização

Após a passagem de paramêtros para a main, o usuário se depara com um menu semelhante ao da figura 3.1 para escolher entre encerrar o programa ou executar um dos algoritmos. Ou da ABP ou da árvore AVL.

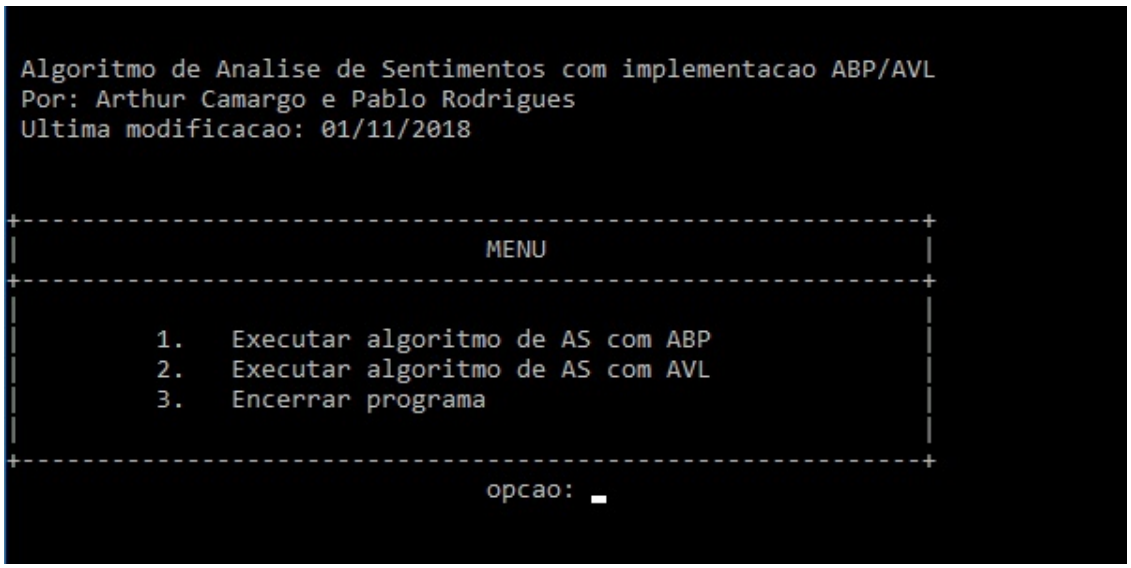

Figura 3.1: exibição do menu de opções do programa

3.2 Tratamento de implementação baseado em ABP

Quando o usuário opta por executar o algoritmo de ABP, todos os tratamentos são feitos necessariamente sobre funções responsáveis por lidar apenas com ABPs, com exceção da função de consulta. Supondo que o arquivo seja validado com sucesso, a operação de remoção de uma palavra junto com o seu respectivo peso é realizada enquanto a remoção de uma palavra não resultar na flag “EOF”, ou seja, fim de arquivo. A cada iteração, uma palavra junto com seu peso é removida do texto léxico para enfim ser passada para a função que insere esta palavra na ABP. Quando a função recebe esta palavra, junto com a árvore a ser inserida a palavra, é realizado um processo recursivo no qual a árvore é percorrida de forma metódica utilizando a chave(ordem lexicográfica), respeitando as propriedades de uma ABP. Achado a posição de inserção, o nodo é inserido no seu respectivo lugar.

3.3 Tratamento de implementação baseado em árvore AVL

Já quando o usuário opta por executar o algoritmo com tratamento de uma AVL, de maneira equivalente as palavras são retiradas uma a uma junto com seu peso do arquivo

de modo que, a cada remoção a palavra removida é passada para a função de inserção na árvore AVL. Diferente de uma ABP, a árvore AVL possuí características determinantes que tem o objetivo de balancear a árvore para que a busca por um determinado nodo seja feita de modo mais eficiente. Quando uma palavra é repassada para a função de inserção, através de comparações lexicograficas a árvore AVL é percorrida de maneira recursiva. Quando enfim se chega ao nodo correto, a palavra é inserida e os fatores de balanceamento são verificados. Caso um dos fatores de balanceamento se mostre inválido, considerando as propriedades de uma árvore AVL, é determinado pelo algoritmo, dependendo da posição onde o nodo foi inserido como filho, esquerda ou direita, se a rotação necessária será uma rotação ou a direita ou a esquerda. Dentro de cada caso essas funções analisam se deve ser realizada uma rotação simples ou uma rotação dupla para se balancear a árvore novamente.

3.4 Processo de atribuição de sentimento a uma frase

Dito de maneira simples, após a inserção de todas as palavras contidas no arquivo léxico em uma das duas estruturas, abre-se um arquivo texto, que contém frases. A cada palavra removida do texto, é feito uma busca pela palavra em uma das estruturas com o auxílio da função de *Consulta_Palavras()* para assim somar seu peso ao “sentimento” final da frase. Quando encontrado um fim de frase, a soma total dos sentimentos é inserido antes desta em um arquivo de saída. Uma nova contagem começa para a próxima frase até que assim se encerre o arquivo.

4. ANÁLISE DE DESEMPENHO

Para demonstrar o desempenho de ambas as árvores foram feitos testes controlados a fim de exibir o número de comparações nos casos de inserção e o número de consultas necessárias em cada estrutura para se encontrar as palavras passadas as funções.

Para uma análise inicial a fim de testar os desempenhos das estruturas, utilizamos um arquivo texto que continha 5 palavras ordenadas de maneira lexicográfica para testarmos de modo simplificado, a alteração no número de comparações para uma árvore binária de pesquisa desbalanceada, e o que ocorreria com a inserção das mesmas palavras e mantendo a ordem de inserção em uma árvore AVL .

As palavras utilizadas para teste foram: *apple*, *ball*, *house*, *kangaroo* e *zebra.*

O processo de inserção em uma ABP não é nada mais do que a procura e inserção de acordo com a chave. Como se trata de um teste controlado, eventualmente como já explicado, a ABP resultante é uma árvore linearmente ordenada, como mostra a figura 4.1. Similar a uma lista.

O ponto notável é o que acontece quando essas palavras são inseridas em uma árvore AVL e como o algoritmo trabalha com isso.
(1) quando inserimos a primeira palavra, *apple*, estamos inserindo em uma árvore vazia, logo o fator de balanceamento deste nodo é 0. **(2)** quando inserimos a palavra *ball*, temos uma uma árvore com um nodo *apple*, então precisamos saber em que posição inserir a palavra *ball* e atribuí-la ao ao lugar correto. Inserida ao lado direito da palavra *apple*, seu fator se torna 0, e o fator da palavra *apple* se torna -1. O interesse de nosso teste se encontra na inserção de palavras que gerem rotações, como a palavra *house*. **(3)** quando

inserimos a palavra *house* na árvore AVL, após as comparações durante o percorrer da árvore, a palavra *house* é inserida à direita da palavra *ball*, pois *house* é maior do que *apple* e *ball*, lexicograficamente. Porém, se relembrarmos a etapa seguinte a atribuição de um nodo em uma árvore AVL, veremos que agora o fator de *ball* se torna -1 e o fator de *apple* se torna -2. Logo, é preciso que a árvore gere uma rotação para balancear a árvore novamente, como mostra a figura 4.2. Neste caso, uma rotação à esquerda. Após o ajuste dos fatores, a palavra *kangaroo* é inserida. **(4)** A inserção da palavra *kangaroo* não afeta o balanceamento da árvore AVL. Contudo, a inserção da palavra *zebra* sim. Igualmente ao processo ocorrido com a palavra *house*, é necessário uma rotação para balancear a árvore AVL novamente. Veja figura 4.6.

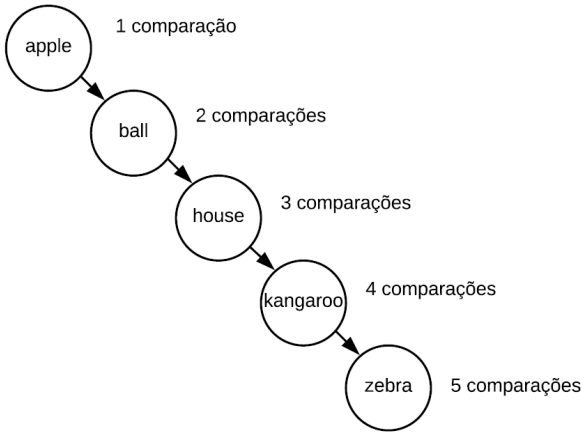

Figura 4.1: inserção das cinco palavras em ABP

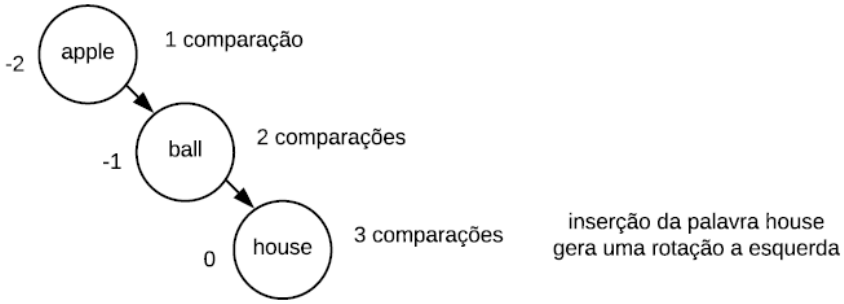

Figura 4.2: terceira inserção em AVL

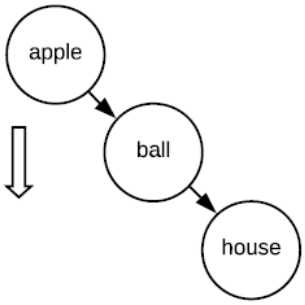

Figura 4.3: rotação a esquerda em AVL

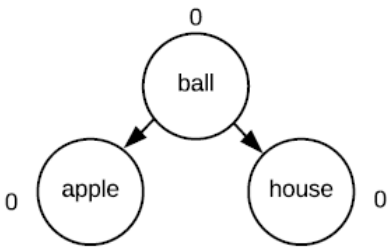

Figura 4.4: AVL após rotação

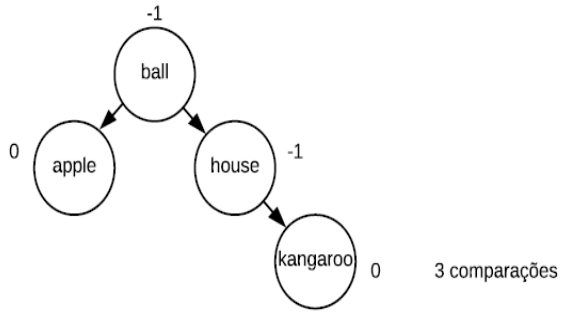

Figura 4.5: inserção da palavra "kangaroo" em AVL

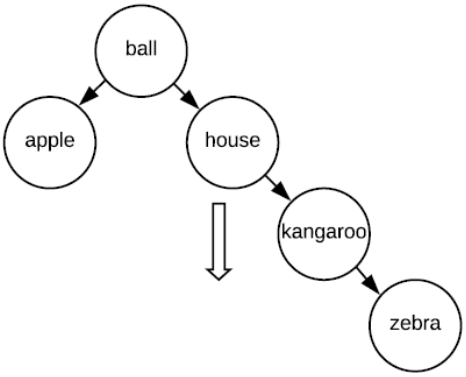

Figura 4.7: rotação a esquerda em AVL

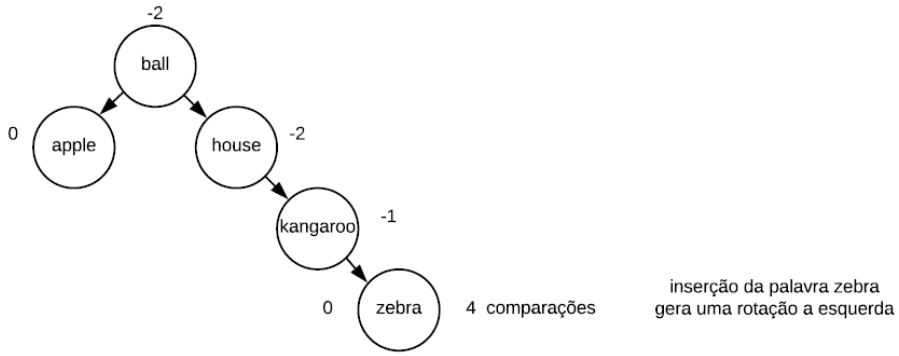

Figura 4.6: inserção da palavra "zebra" em AVL

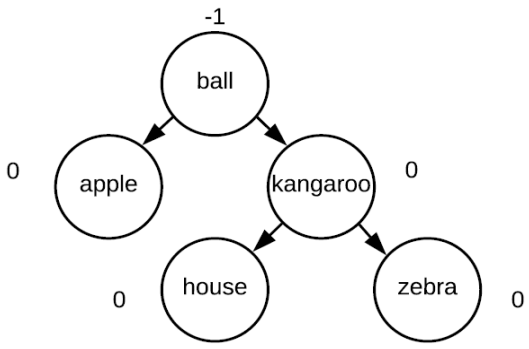

Figura 4.8: árvore AVL resultante

Os resultados e a importância destes dois processos algorítmicos distintos se dá quando resolvemos comparar o número de comparações necessárias para inserir um elemento em ambas as árvores e o número de consultas para se achar um elemento.

1 + 2 + 3 + 4 + 5 = **15**

Enquanto na árvore AVL seria:

1 + 2 + 3 + 3 + 4 = **13**

Também, se tivermos um texto apenas com a palavra *zebra,* podemos notar que o número de comparações no pior caso, pois *zebra* é a última “palavra”, em uma árvore ABP é de 10 consultas, conforme a função *Consulta_Palavra(),* enquanto em uma árvore AVL é apenas 6.

É notável que com apenas 5 palavras o número de comparações e consultas já se difere. Assim podemos notar a eficiência de uma árvore AVL em relação a uma ABP quando se trata de uma quantidade considerável de elementos e quando temos que lidar com problemas de consulta e de desbalanceamento. É um bom exercício imaginar este processo com 2000 palavras. O número de palavras no arquivo *lexicon_2k_shuffled.txt.*

Abaixo segue os resultados das execuções dos dois algoritmos, utilizando os arquivos de teste:

- lexicon_2k_shuffled.txt
- movieReviews1000.txt

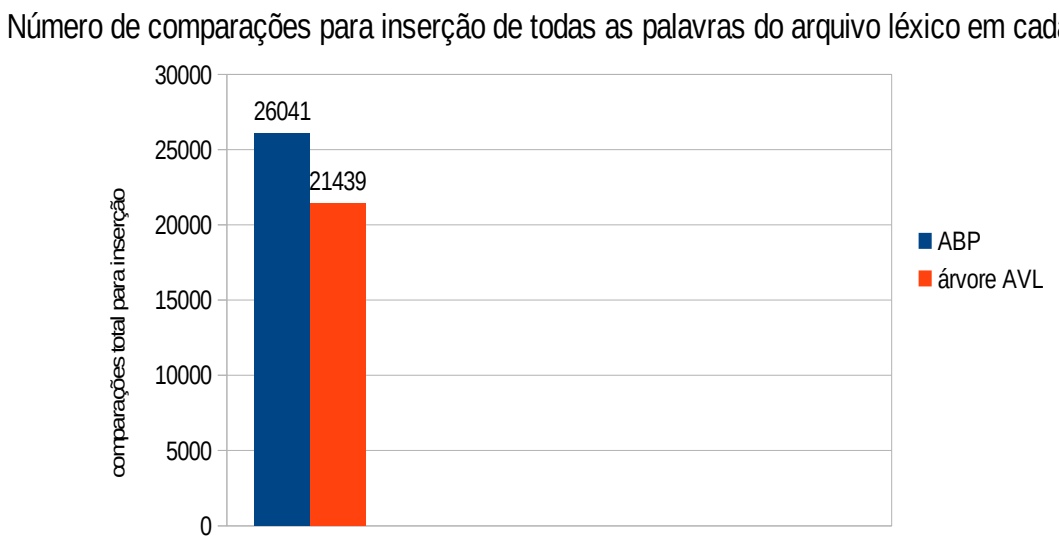

Figura 4.9: número de comparações necessárias em cada estrutura

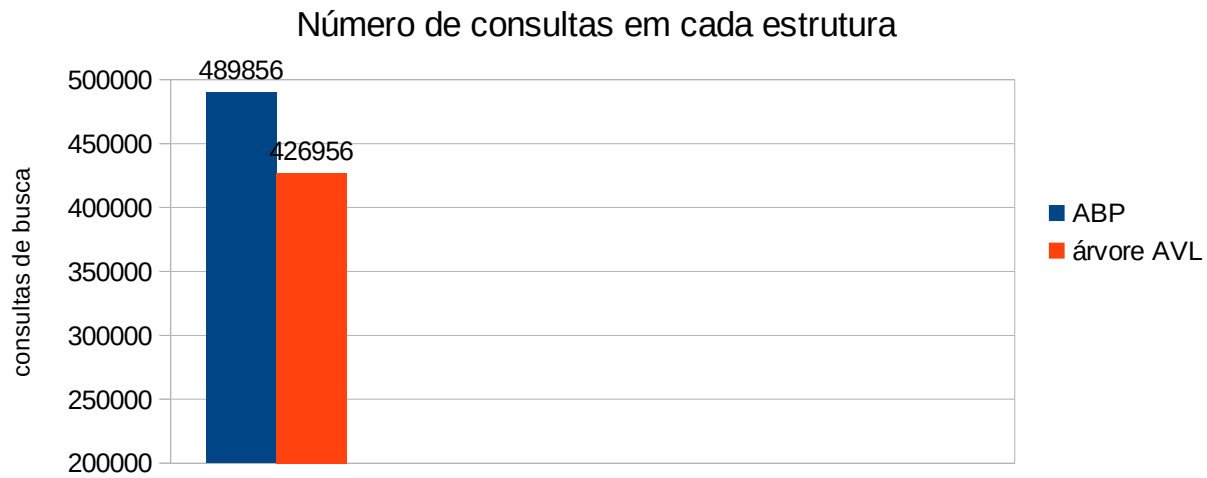

Figura 4.10: número de consultas necessárias em cada estrutura

Resultados finais utilizando os arquivos de testes:

- lexicon_2k_sorted.txt
- movieReviews5000.txt

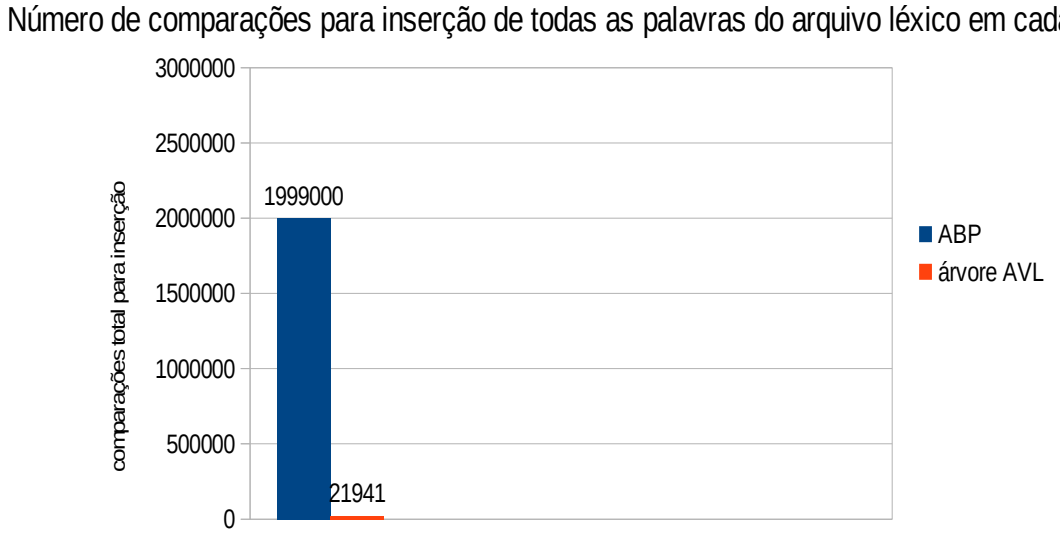

Figura 4.11: número de comparações necessárias em cada estrutura

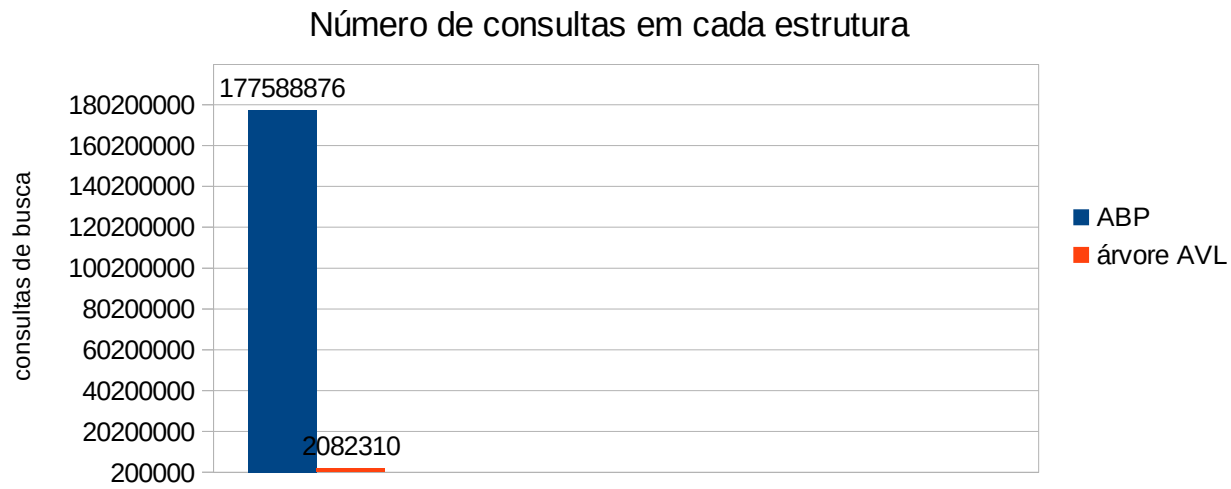

Figura 4.12: número de consultas necessárias em cada estrutura

5. CONCLUSÃO

Em suma, foi observado que a implementação do algoritmo de uma ABP se mostrou menos eficiente que a implementação de uma árvore AVL, pois, muitas vezes em casos de comparação, o desbalanceamento possível em ABP acaba por gerar um desempenho inferior em questão de acessibilidade aos seus nodos. Assim, AVLs acabam por ter um desempenho melhor por conta de sua caracteristica de balanceamento, o que facilita buscas e inserções quando lidamos com árvores com grande quantidade de elementos.