**Relatório de Entraga do Trabalho Final**

INF01203 - Estrutura de Dados

Mateus Davi Simon

Ricardo de Araújo Coelho

Em nosso trabalho optamos por realizar as comparações entre uma implementação utilizando ABP não balanceada e uma usando árvores AVL. Nossa hipótese é de que a árvore AVL irá ter um performance melhor, pois nesta aplicação a quantidade de inclusões na árvore se limita ao número de carácteres que serão codificados em morse, tendo um limite aproximado de 100 inclusões, já o número de consultas é virtualmente ilimitado, dependendo do número de carácteres no texto a ser convertido.

Para criar a aplicação utilizamos as implementações das árvores AVL e ABP já disponíveis no Moodle da disciplina com as modificações necessárias. O programa recebe o nome dos arquivos da tabela morse, do arquivo a ser convertido e do arquivo destino por argumentos da linha de comando, como especificado. Caso seja passado um quarto argumento adicional com a string ‘avl’ o programa usará árvore AVL, caso contrário usará ABP não balanceada.

Além da contagem do número de comparações feitas nas consultas das árvores também adicionamos a contagem de tempo que o programa utiliza para realizar as operações de carregamento da tabela de conversão (onde ocorrem as inclusões) e a conversão do texto (consultas).

Para realização dos testes, utilizamos a tabela de conversão como foi fornecida no Moodle e um versão desordenada dela, e dois arquivos de texto um pequeno (78Kb) (disponível no Moodle) e um grande(58Kb). A ideia de utilizar uma tabela desordenada foi para mostrar o efeito negativo que a inclusão de valores ordenados pode ter na ABP não balanceada.

**Resultados:**

Conforme mostra a Tabela 1, foram geradas 8 combinações diferentes de testes. As árvores de pesquisa testadas são diferenciadas quanto ao balanceamento na inserção, sendo: ABP quando a inserção não leva em conta nenhum tipo de balanceamento, e AVL quando é utilizado o algoritmo AVL na inserção. Além disso, o nome possui uma letra, P ou G, que indica se o teste foi realizado com o arquivo

Pequeno, ou o arquivo Grande de conversão respectivamente. E por último, o nome da árvore pode conter mais um indicador “alt”, que significa que o arquivo da tabela do código morse utilizado é uma versão alternativa do arquivo original, tendo a ordem de inserção trocada a fim de que não ocorresse uma inserção ordenada para montar a árvore.

| Árvore de Pesquisa | | nº de comps | t inserção (us) | t consulta (ms) | t total (ms) |
|---|---|---|---|---|---|
| ABP | P | 727,610 | 85.6 | 56.23 | 56.32 |
| ABP | G | 547,049,574 | 73.7 | 16,287.62 | 16,287.69 |
| ABP alt | P | 374,288 | 93.4 | 56.30 | 56.39 |
| ABP alt | G | 284,811,858 | 62.3 | 15,496.18 | 15,496.24 |
| AVL | P | 296,178 | 106.9 | 52.99 | 53.10 |
| AVL | G | 224,960,202 | 98.7 | 14,964.43 | 14,964.53 |
| AVL alt | P | 325,161 | 128.5 | 54.84 | 54.97 |
| AVL alt | G | 250,167,186 | 76.8 | 15,111.60 | 15,111.68 |

*Tabela 1: Resultado das medidas de cada um dos testes realizados*

As Figuras 1 e 2 mostram gráficos comparativos do número de comparações necessárias feitas na consulta da árvore binária de pesquisa para todos os caracteres do arquivo a ser convertido para morse. Tanto para o arquivo pequeno quanto para o grande, o número de comparações feitas usando a ABP ordenada foi significativamente maior. Isto se deve ao fato de que como a inserção foi feita de maneira ordenada, a árvore ficou com a altura máxima, comportando-se assim, como uma lista ordenada. Em termos de medida de comparações, esse é o pior caso possível, tendo a AVL mostrado o melhor desempenho (uma redução de quase 60% nos dois casos).

Vale notar também que a execução do programa utilizando a tabela morse alternativa fez com que o número de comparações caísse drasticamente em relação a ABP, pois a inserção não ordenada foi uma forma de explicitamente diminuir a altura da árvore. Observando-se a “AVL alt” vemos que o número de comparações foi levemente maior do que o número de comparações da “AVL”, o que de fato faz sentido, pois como a AVL já se encarrega de deixar a árvore balanceada na inserção, isso faz com que não se tenha sensibilidade na altura da árvore com relação à ordem em que os elementos são inseridos.

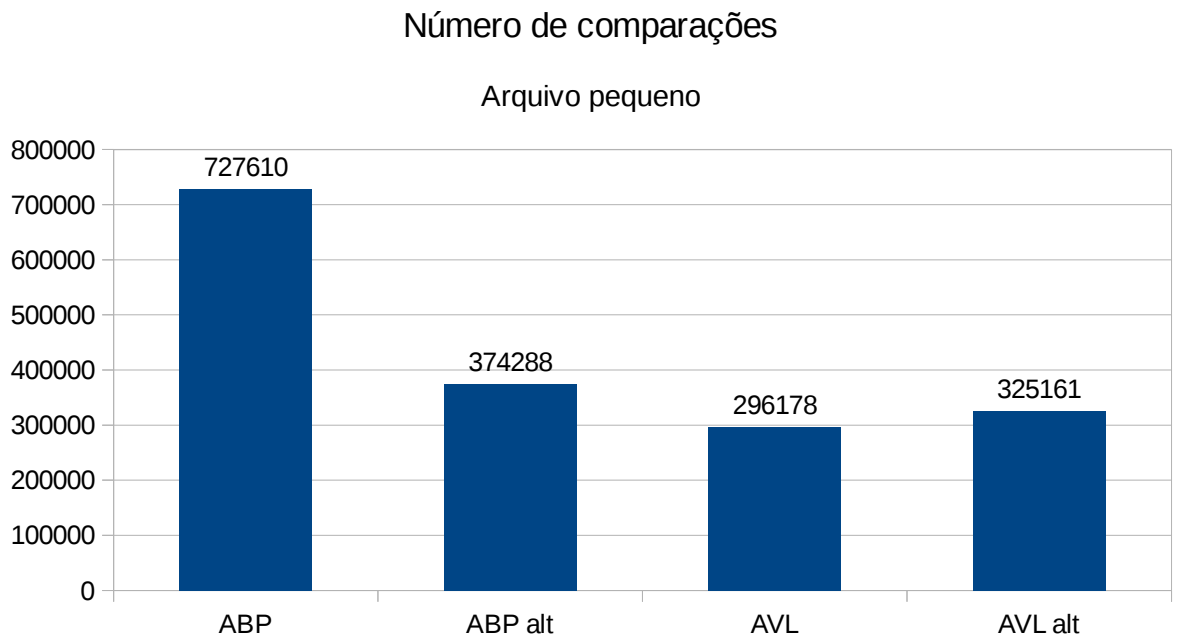


*Figura 1: Número de comparações totais na operação de consulta à arvore de pesquisa para arquivo de teste pequeno.*

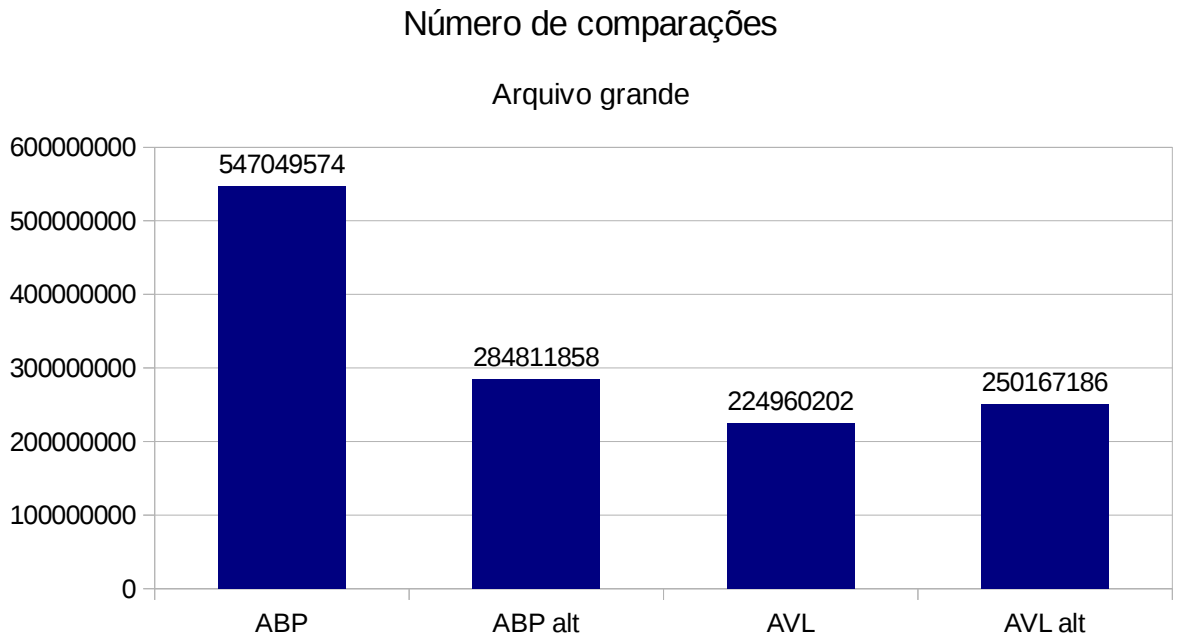


*Figura 2: Número de comparações totais na operação de consulta à arvore de pesquisa para arquivo de teste grande.*

A Figura 3 mostra o tempo de inserção da tabela do código morse na árvore de pesquisa. O objetivo aqui é analisar o custo de se utilizar o algoritmo de balanceamento da AVL. Cabe ressaltar que como o número de inserções é relativamente pequeno, o tempo total de inserção é bem curto (na ordem de dezenas de microssegundos), o que faz com que não se consiga se medir com a precisão desejada os valores. Um exemplo disso é que a inserção tanto para o arquivo de teste pequeno quanto para o grande deveriam ser iguais, pois a parte que muda no programa é apenas a parte da consulta, porém na aferição das medidas esses valores deram diferentes. Entretanto, é possível observar que na média, os valores de inserção da ABP sem balanceamento foram menores (basta observar que em geral os tempos para a ABP estão abaixo da linha vermelha pontilhada, que indica o tempo médio de inserção, enquanto que os valores para a AVL estão no geral acima da linha). Isso vai de acordo com o esperado, pois a AVL perde um tempo a mais no processamento da execução do balanceamento.

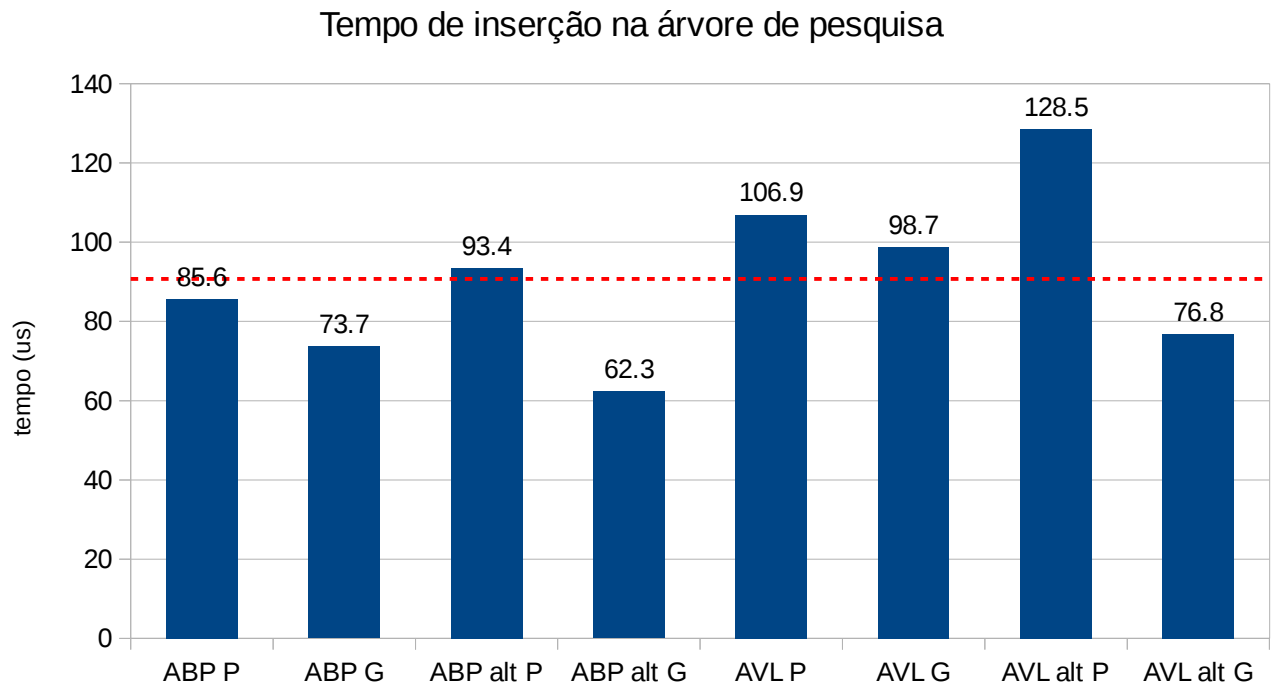


*Figura 3: Tempo de inserção na árvore de pesquisa. O traço vermelho indica o valor médio*

A próxima análise é em relação ao tempo de consulta. A Figura 4 mostra o tempo de consulta para o arquivo de teste pequeno. O desempenho da arvore AVL se mostra consideravelmente melhor, o que é de se esperar, tento em vista que o número de comparações em relação a ABP sem balanceamento é menos da metade. Contudo, como o arquivo é pequeno, o número de consultas não conseguiu refletir em diferença observável no tempo de consulta da ABP ordenada em relação a ABP com tabela morse alternativa, inclusive essa última teve um tempo de resposta pior do que o a ABP ordenada. Isso pode ser explicado pela imprecisão das funções de aferição dos tempos e também a carga do sistema operacional no momento do teste.

Na Figura 5 tem-se a relação dos tempos de consulta para o arquivo de teste grande. Agora, tento em vista que o número de consultas é bem elevado, pode-se notar uma diferença um pouco mais considerável nos tempos de execução para cada uma das árvores (nota-se agora que a escala de tempo está em segundos, e na tabela anterior era dado em milissegundos). As execuções com AVL foram as mais rápidas, sendo mais de 1 segundo mais rápidas que a ABP ordenada, e quase meio segundo mais rápidas que a “ABP alt”. A explicação se dá pelo tempo a mais que a ABP desbalanceada leva para processar as comparações a mais. Da Tabela 1 vemos que da ABP ordenada para a AVL, temos mais de 322 milhões comparações a mais. Assim, ao contrário do teste feito com o arquivo pequeno, para um número de consultas muito grande, começamos a observar uma relação de proporção mais bem definida de número de comparações por tempo de execução (ABP alternativa sendo quase 0.8 segundos mais rápida que a ABP ordenada mostra isso).

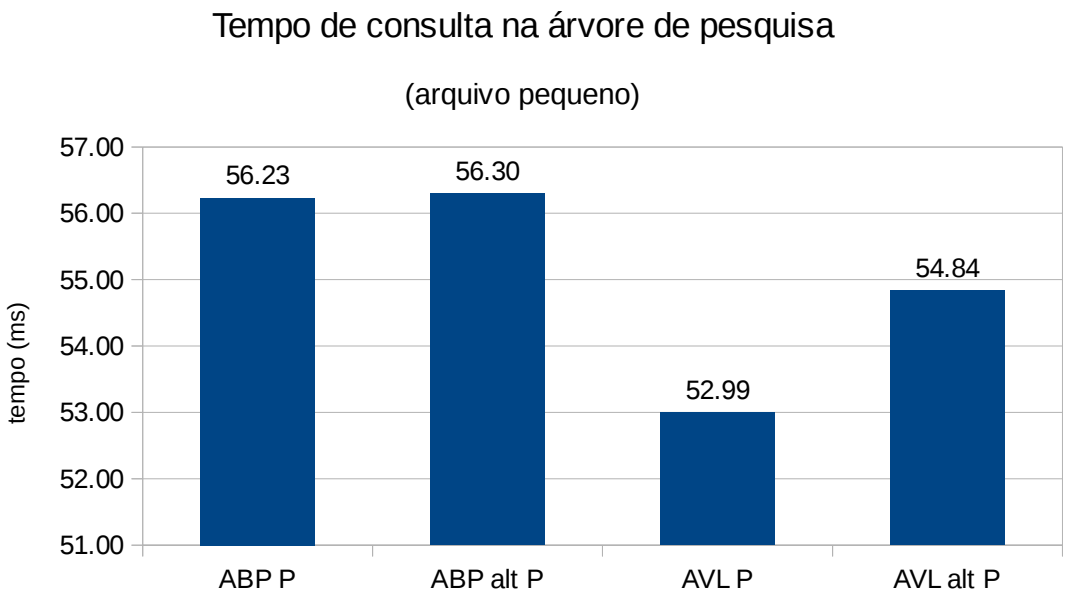


*Figura 4: Tempo de consulta na árvore de pesquisa para o arquivo de teste pequeno*

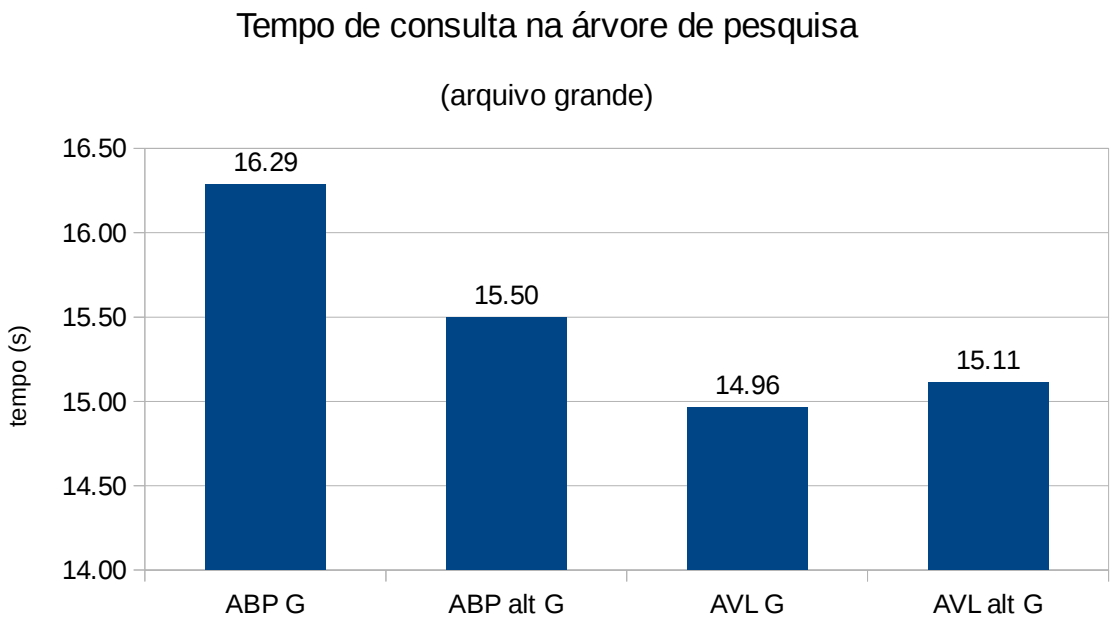


*Figura 5: Tempo de consulta na árvore de pesquisa para o arquivo de teste grande*

Por fim, temos a análise do tempo total de execução do programa de conversão para morse de cada uma das árvores utilizadas. As Figuras 6 e 7 mostram os respectivos gráficos para o teste com o arquivo pequeno e arquivo grande. Em ambos os gráficos, o tempo total se mostra muito semelhante ao tempo de consulta. Isso se dá pois o tempo de consulta é muito maior que o tempo de inserção nessa aplicação. Para arquivo o teste com arquivo pequeno, o tempo de consulta é na ordem dos milissegundos enquanto o de inserção é microssegundos. Já considerando-se o arquivo de teste grande, essa diferença se torna ainda maior, sendo o tempo de consulta medido na ordem dos segundos.

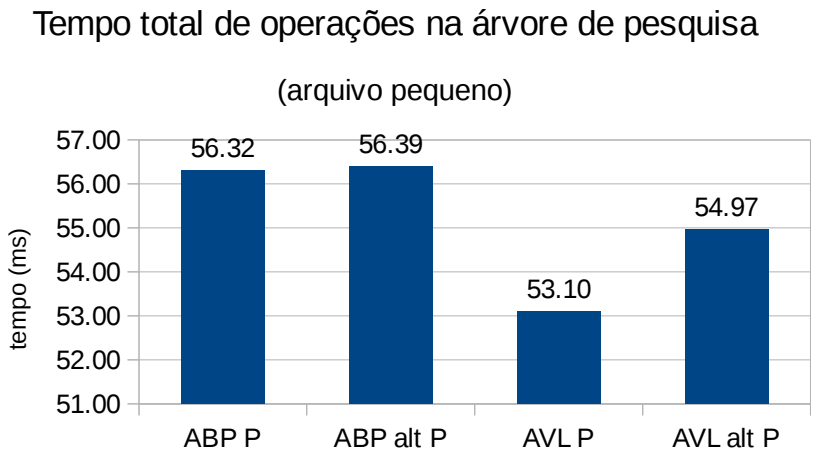


*Figura 6: Tempo total de execução das operações na árvore de pesquisa para arquivo pequeno.*

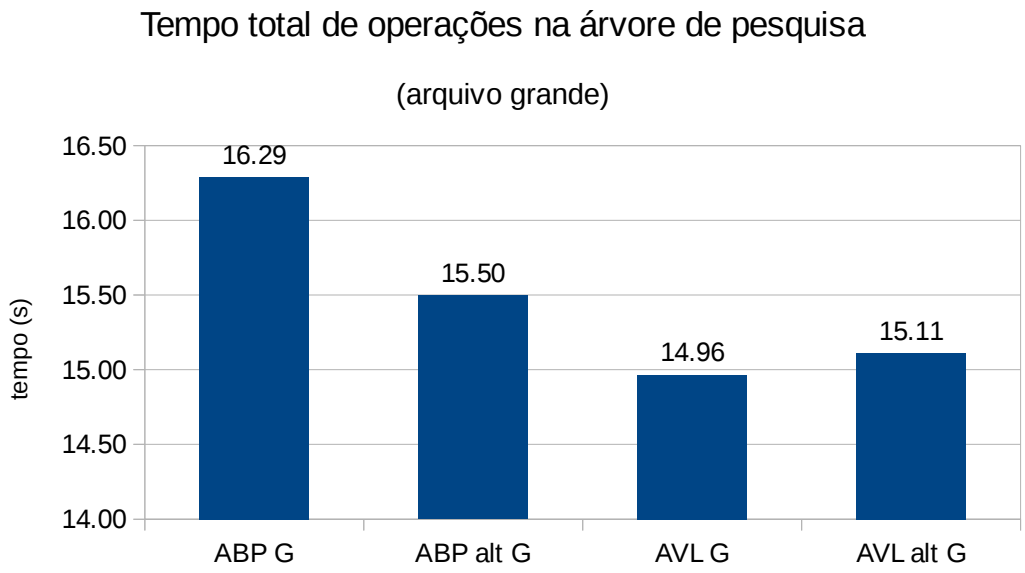


*Figura 7: Tempo total de execução das operações na árvore de pesquisa para arquivo grande.*

**Conclusão:**

Para o presente trabalho, a escolha pela AVL se mostrou uma opção vantajosa para se estruturar a árvore de pesquisa. A hipótese de que a AVL seria uma escolha correta para melhorar o desempenho do programa se confirmou. Isso pode ser atribuído ao fato de que a aplicação é caracterizada por ter um número de consultas muito mais expressivo do que o número de inserções. O fato da árvore ter relativamente poucas inserções, traz a ideia de que e a operação de consulta não deve ser o gargalo na execução do programa (o palpite é que as funções de leitura e escrita nos arquivos sejam mais demoradas), por isso que uma execução com o dobro do número de comparações reflete numa diferença não tão acentuada no tempo de execução por exemplo. Talvez em uma aplicação onde o tamanho da ABP fosse consideravelmente maior, poderíamos observar um aumento no tempo de consulta, o que representaria uma fatia maior no tempo de processamento do programa.