Thiago Sotoriva Lermen – 00313020
Leonardo Barros Bilhalva – 00315768
Professora: Renata Galante

Relatório

1. Introdução

Neste relatório consta as especificações das implementações de dois tipos de estruturas de dados a fim de realizar a leitura de informações a partir de um arquivo de texto. Além disso, foi realizada uma comparação entre os dois tipos de implementações, com o objetivo de aplicar os conceitos teóricos aprendidos em sala de aula em uma implementação e comparação prática das características das estruturas de dados utilizadas.

Assim, as implementações foram feitas com base em duas estruturas de dados distintas: Árvore Binária de Pesquisa (ABP) e Árvore Binária Com Balanceamento Por Altura (AVL). Através desses dados, que foram extraídos de dois arquivos de entrada para o programa (arquivo de texto e arquivo de operações), foram realizadas comparações entre as duas árvores a fim de evidenciar qual, dentre as duas implementações, possuiu o melhor desempenho para tipos de entradas diferentes.

Dessa forma, as comparações realizadas entre os dois tipos de estruturas de dados, tiveram enfoque no número de comparações durante toda a execução do programa, características como por exemplo, número de rotações, fator de balanceamento da árvore, altura e número de rotações, e temporizações realizadas durante a execução do programa, durante a inserção de nodos na árvore e busca de palavras e valores chaves na árvore.

2. Lógica de implementação

2.1. Árvore Binária Com Balanceamento Por Altura (AVL)

O primeiro método de implementação escolhido foi através de AVL devido, principalmente, pela redução da altura da árvore através das rotações feitas pelo critério de fator de balanceamento, que é alterado de acordo com a inserção e remoção de nodos. Assim como a lógica adotada na implementação da ABP, que veremos mais adiante, durante a leitura do arquivo texto, cada palavra lida é inserida na árvore, gerando, portanto, operações que serão analisadas. Para computar essas operações foram utilizadas variáveis que armazenam valores inteiros decimais, como por exemplo, o fator de balanceamento da árvore, a altura e a tempo decorrido durante a inserção.

```
typedef struct Informacao_Arvore{
    char palavra[MAX_PALAVRA];
    int quantidade;
}TipoInfo;

typedef struct TNodoA{
    TipoInfo info;
    int FB;
    struct TNodoA *esq;
    struct TNodoA *dir;
}pNodoA;
```

Figura 2.1.1. Estrutura de dados utilizada para representação de AVL.

Com isso, foi decido que a árvore teria seu critério de ordenamento baseado na ordem lexicográfica das palavras, porém com a inserção espelhada, ou seja, palavras que possuem ordem lexicográfica superior à da raiz estarão ao lado esquerdo da arvore binaria, enquanto as palavras que possuem ordem lexicográfica inferior à da raiz estarão na direita. Além disso, a estrutura de dados implementada de acordo com a figura 2.1.1, com campos voltados ao armazenamento de *strings* e valores numéricos decimais (frequência).

A cada iteração, durante a leitura do arquivo texto de origem, cada palavra retirada do texto é inserida na árvore seguindo o critério de ordenamento mencionado. Dessa forma, o número de comparações, rotações e temporizações nas realizações de operações de inserção de nodos em uma AVL são computados durante essas iterações. Além disso, durante esses laços de leitura do arquivo de operações, são realizadas buscas determinadas de acordo com os

valores do arquivo de operações, como por exemplo, busca de frequência de determinada palavra, ou palavras que possuem sua frequência entre dois valores dados.

Destarte, a cada iteração da leitura do arquivo de operações a função de impressão no arquivo de saída é realiza para cada valor de frequência dada, ou seja, seja $k1$ um inteiro que representa uma frequência mínima, $k2$, um inteiro que representa uma frequência máxima, para cada valor de m, tal que $k1 \leq m \leq k2$ é realizada uma busca que percorre toda a árvore com o objetivo de encontrar as palavras cujas as frequências têm valor m.

2.2. Árvore Binária De Pesquisa (ABP)

A implementação de ABP foi escolhida principalmente para podermos verificar a importância de um balanceamento de uma árvore, ou seja, teremos noção, dessa forma, da real importância do fator de balanceamento e um menor número de altura. Assim como fio utilizado na implementação da AVL, durante a leitura do arquivo texto, cada palavra lida é inserida na árvore, gerando, portanto, operações que serão analisadas. Para computar essas operações foram utilizadas variáveis que armazenam valores inteiros decimais, como por exemplo, o fator de balanceamento da árvore, a altura e a tempo decorrido durante a inserção. Além disso, o critério de ordenamento segue o mesmo, ou seja, ordenamento por ordem lexicográfica da mesma forma implementada na estrutura de dados AVL.

Dessa forma, a estrutura de dados da respectiva árvore, seguira a mesma forma da já implementada, ou seja, possuirá um campo para armazenar a palavra lida do texto e a frequência da mesma [. Figura 2.2.1].

```
typedef struct Informacao_Arvore{
    char palavra[MAX_PALAVRA];
    int quantidade;
}TipoInfo;

typedef struct TNodoA{
    TipoInfo info;
    int FB;
    struct TNodoA *esq;
    struct TNodoA *dir;
}pNodoA;
```

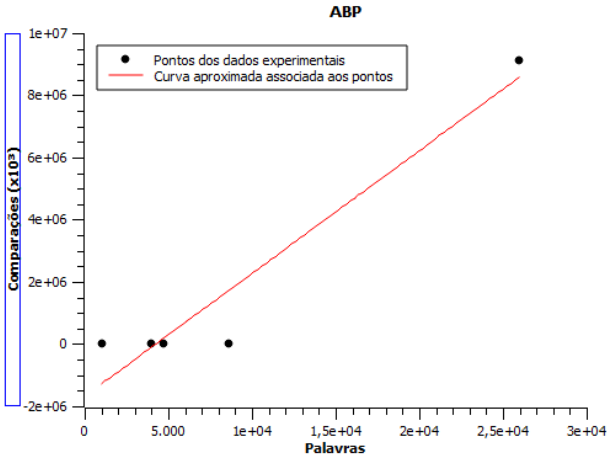
Figura 2.1.1. Estrutura de dados utilizada para representação de ABP.

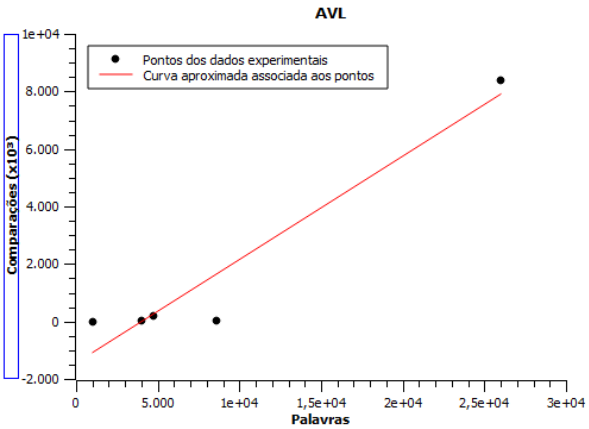
Assim como na implementação da AVL, a cada iteração, durante a leitura do arquivo texto de origem, cada palavra retirada do texto é inserida na árvore seguindo o critério de ordenamento mencionado. Dessa forma, o número de comparações e temporizações nas realizações de operações de inserção de nodos em uma ABP são computados durante essas iterações. Além disso, durante esses laços de leitura do arquivo de operações, são realizadas buscas determinadas de acordo com os valores do arquivo de operações, como por exemplo, busca de frequência de determinada palavra, ou palavras que possuem sua frequência entre dois valores dados. Da mesma forma que AVL, o método de leitura de palavras entre duas frequências dadas é a mesma, ou seja, percorre toda a arovre para cada valor de frequencia. Portanto o número de comparações nessa busca será a mesma para as duas árvores.

2. Comparações

Os arquivos de entrada, os quais continham informações a serem analisadas eram arquivos de texto no formato txt com formatação ANSI. Foram analisados 5 arquivos em inglês e português com uma variação de palavras compreendida entre 1029 e 25992 palavras distintas. Abaixo estão representados gráficos e uma tabela que contêm dados a respeito das comparações realizadas.

ARQUIVO	PALAVRAS POR ARQUIVO	COMPARAÇÕES ABP	COMPARAÇÕES AVL	TEMPORIZAÇÃO ABP (ms)	TEMPORIZAÇÃO AVL (ms)
1	1029	25570	23648	4	4
2	3995	213986	192818	21	21
3	4725	206345	186647	19	18
4	8626	1068999	1008890	104	102
5	25992	9126744	8367640	820	807





3. Conclusão

De acordo com os gráficos e com a tabela, apesar da sucinta diferença entre a inclinação das duas retas aproximadas aos pontos dos dados experimentais, percebe-se claramente que a implementação através de AVL se torna mais eficiente em relação à ABP durante a geração da árvore. Apesar do número de rotações de uma árvore AVL aumentar de acordo com o número de palavras, a AVL ainda assim demonstra mais eficiência, visto que a altura da árvore diminui em relação à ABP devido ao balanceamento por altura realizado na AVL. Assim, o fator de balanceamento se torna um fator extremamente significante no processo de otimização do número de comparações necessários para que a próxima palavra retirada do texto seja inserida, ou seja, quanto menor a altura da árvore, mais rápida é a busca por um nodo específico e assim, mais rápida é sua inserção de acordo com o crescimento da base de dados.

Todavia, com relação à temporização, não foi possível obter um dado concreto, devido, principalmente, pelo uso da CPU que influencia na variação da temporização. Não levando isso em consideração, percebe-se que há uma pequena diferença de tempo entre as duas estruturas de dados, devido, principalmente à pequena base de teste. Além disso, começa-se a observar que com o aumento da quantidade de palavras, o valor da temporização das duas implementações começa a se distanciar, tendo maior significância a AVL.

Portanto, ao analisarmos com cuidado os resultados das comparações entre as duas implementações, percebe-se que os valores retornados estão diretamente relacionados com a eficiência das implementações. Logo, teremos implementações, como a AVL que, de acordo com o crescimento da base de dados utilizada, começa a ter resultados mais significativos quando relacionamos essa implementação com a ABP, por exemplo.