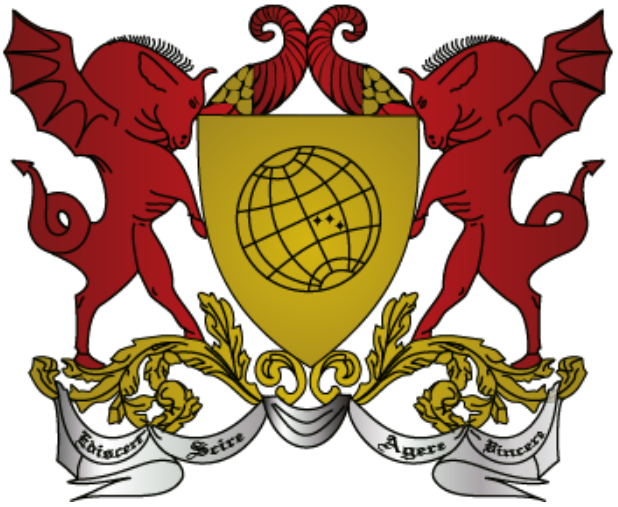

Universidade Federal de Viçosa - Campus Florestal
Ciência da Computação - 3º Período

Trabalho Prático de AEDS II – 02

Gerenciador de Árvore B em C + GTK

Bruno Marra - 3029
Gustavo Viegas - 3026
Heitor Passeado - 3055

Florestal

Maio de 2018

1. Introdução

Após o estudo da árvore digital B, foi proposta uma implementação de tal estrutura para gerenciamento e armazenamento dos dados na mesma. O intuito era analisarmos a complexidade e a quantidade de registros é possível armazenar dentro de uma estrutura desse tipo sem aumentar significantemente a altura na árvore, e consequentemente, há um ganho no quesito busca (inclusive em um trabalho de busca em memória secundária, onde cada nível inferior é um novo acesso a disco).

Utilizamos como base o código do livro-texto, e a partir das funções da árvore implementadas, partimos para o desenvolvimento de uma interface de comunicação com o sistema usando a biblioteca GTK (GIMP Tool Kit), onde criamos uma interface bastante amigável para o usuário final. Após isso, implementamos as comparações e fizemos análises de desempenho a partir de uma árvore com 100.000 registros numéricos ordenados aleatoriamente.

2. Desenvolvimento

Para criação da árvore B, usamos como base a implementação proposta pelo Prof. Dr. Nivio Ziviani, na qual ele disponibiliza em seu próprio site[1]. Como neste trabalho, não havia a necessidade de adaptação na estrutura em si, tivemos apenas que entender o código fornecido pelo professor e modificar a sintaxe original, para fazer mais sentido lexicalmente que a proposta pelo mesmo. Logo, a partir do ponto onde entendemos a estrutura da árvore, partimos para a criação do TAD "file.h", responsável pela leitura e interpretação do arquivo, bem como a alimentação da árvore a partir dos dados armazenados no arquivo.

Para que fosse possível estruturar uma comunicação amigável com o usuário final, utilizamos a biblioteca gráfica GTK (GIMP Tool Kit), que consiste numa biblioteca para criar interfaces gráficas de usuário. É licenciado pela licença LGPL, e portanto você pode desenvolver com o GTK software aberto, livre ou mesmo comercial e não-livre, sem ter de gastar nada com licenças ou royalties. Mas, o GTK só nos dá suporte a escrita da interface, para o desenvolvimento do layout e do design da tela,

[1] Referência de Código Árvore B:
http://www2.dcc.ufmg.br/livros/algoritmos/cap6/codigo/c/6.3a6.9-arvore-b.c

utilizamos o Glade, que consiste em uma ferramenta RAD para permitir o desenvolvimento rápido e fácil de interfaces de usuário para o kit de ferramentas GTK + e o ambiente de área de trabalho GNOME. Após tudo isso, nossa tarefa foi linkar nossas funções a tela com as interfaces e enfim, podermos testar efetivamente o desempenho da estrutura.

Figura 1 - GTK + Glade

Para o arquivo de 100.000 registros, e uma árvore de ordem M=2, notamos que a altura da árvore não passou de 8, validando o fato de que a estrutura é extremamente bem adaptada para otimizar e percorrer com eficiência os níveis da árvore, o que também facilita ao buscar alguma chave, pelo fato da árvore ser binária. A Tabela 1 mostra um comparativo de alguns dados importantes, com mais detalhes, que foram analisados pelo grupo após a inserção do arquivo com vários registros.

Dados/Valor de M	8	64	512
Altura	7	6	4
Número de acesso a disco	7	6	4
Operações na inserção (22)	2603	1912	1432
Operações na pesquisa (22)	815	689	321
Operações na remoção (22)	1910	1805	1978

Tabela 1 - Dados sobre a estrutura após 100.000 registros

A partir dos resultados obtidos, podemos observar que a medida que aumentamos o M para um número grande de registros, podemos ver um ganho significativo no número de

operações, principalmente na pesquisa, onde vemos um ganho gigantesco. Um detalhe interessante, foi observar que não necessariamente o aumento do M resulta num melhor desempenho na remoção, que por exemplo obteve um resultado um pouco pior que para M = 8. Ou seja, executamos um grande número de operações quando fazemos uma operação de remoção ou de inserção, mas notamos um número bem baixo de operações para encontrar um registro, dando mais ênfase a eficiência das árvores na pesquisa.

3. Conclusão

Após o estudo e realização do trabalho, ficou evidente que árvores binárias, apesar de difícil a visualização e implementação são estruturas eficientes para pesquisa. Seu tempo de pesquisa é O(log(n)). Como podemos observar na tabela, a altura final da árvore é algo que impacta diretamente no tempo de pesquisa, otimizando o número de acesso a disco e automaticamente aumentando a eficiência da estrutura para essa finalidade. Logo, o uso de árvores de pesquisa é extremamente eficiente quando o objetivo maior de uma aplicação é a busca contínua de dados, em uma massa de dados muito alta, podemos notar que as árvores nos proporcionam resultados muito bons nesse quesito.

Um comparativo interessante nesse aspecto, seria a altura de uma árvore binária em relação a altura de uma árvore B. A diferença entre as duas é exorbitante, o que acarretaria em um número muito maior de descidas na estrutura, diminuindo a eficiência, ainda mais quando o intuito maior de uma árvore b é a pesquisa em memória secundária. Logo, fica evidente a eficiência e eficácia no uso dessa estrutura.