Curso de Graduação em Ciência da Computação
Disciplina: Estrutura de Dados I
Relatório
Professor: Mário Luiz R. Oliveira

Relatório

Elias Eduardo Silva Rodrigues **Nº: 0015920**
Maria Eduarda da Silveira **Nº: 0035483**

1 Introdução

O trabalho proposto era o desenvolvimento de um programa para a criação de índices remissivos. Um índice remissivo é um arquivo que serve como referência de um texto. De forma mais clara, um índice remissivo é uma lista em ordem alfabética que lista as palavras chaves de um texto e as linhas referentes onde ela aparece. Para a geração de tal arquivo, é necessário realizar várias buscas, o que pode causar uma demora no processamento do software caso não seja usada uma estrutura de dados adequada.

Assim sendo, o objetivo deste trabalho é a comparação entre várias estruturas de dados, logo foi desenvolvido as estruturas árvore binária de busca, árvore AVL, lista encadeada e tabela hash.

2 Implementação

Nesta seção ser apresentado como as estruturas foram implementadas e demostrações das estruturas de dados.

2.1 Decisões de projeto

Para o desenvolvimento do trabalho foi usado a linguagem de programação C. O trabalho foi codificado usando o editor de texto Sublime Text 3, foi usado o compilador GCC versão 7.4.0, sistema operacional GNU/Linux Ubuntu 18.04.1 e o controle de versão usando o Git.

Foi criado uma TAD de árvore binária de busca(*ABB.c*, *ABB.h*), onde os nós a esquerda da raiz principal possuem valores menores e os nós a direita valores maiores. Uma árvore AVL(*AVL.c*, *AVL.h*), onde a árvore se mantém balanceada usando os métodos de rotacionamento para direita e esquerda. Uma lista encadeada(*lista.c*, *lista.h*), cada célula da lista é uma palavra e que possui um vetor contendo os números das linhas onde ela aparece. E uma tabela hash com encadeamento externo e open hash(*HASH.c*, *HASH.h*). Além desses arquivos foi criado o arquivo principal(*main.c*), sendo realizado nesse arquivo a manipulação de strings, leitura e gravação de arquivos de saída, nesse arquivo também é chamado todas as estruturas para a realização do trabalho.

Na tabela hash foram usados três métodos diferentes para o cálculo da posição da palavra. O primeiro método usado foi a função hash sugerida na referência Ziviani (ZIVIANI et al., 2004), onde é realizado o mod do tamanho da palavra com o tamanho da tabela hash. O segundo método usado foi pensando em causar colisões,

então foi um mod do tamanho da palavra por 2. E a última função hash usada foi o método de Horner.

2.2 Compilação e Execução

Para facilitar na hora de compilar e executar foi criado o arquivo sh(*compila.sh*). Basta executar este arquivo que o programa principal(*main.c*), será compilado e executado.

Na compilação do arquivo *main.c*, é definida uma função *rodaEstrutura()*, que recebe como parâmetro o número da estrutura a ser executada e o arquivo de saída que receberá o resultado. A cada momento é passado um número de uma estrutura até que todas sejam executadas e salvos os respectivos resultados no arquivo de saída.

Após o término da execução será mostrado o tempo gasto na geração do índice remissivo de cada estrutura.

3 Testes Realizados

Para a realização dos testes foi disponibilizado pelo professor um arquivo de texto(carta-pero-vaz-caminha-filtrado.txt), que possuí um texto relativamente grande para a geração do índice remissivo. As palavras chaves a serem encontradas ficariam a critério dos alunos, então foi criado um arquivo de texto(chave.txt), que contém as palavras a serem encontradas no arquivo do professor. Os resultados encontrados podem ser visualizados nas tabelas seguintes.

Execução	**ABB**	**AVL**	**Lista**
1	0.189262	0.283853	0.219517
2	0.236476	0.302474	0.307309
3	0.233507	0.294574	0.254826
4	0.224877	0.254088	0.253812
5	0.223383	0.33446	0.257312
Média	**0,221501**	**0,2938898**	**0,2585552**
Desvio Padrão	**0,01885926219**	**0,02917626768**	**0,03138018681**

Tabela 1: Tabela com os valores de tempo das execuções do sistema.

Execução	Hash Função 1	Hash Função 2	Hash Função 3
1	0.197645	0.252028	0.335113
2	0.185361	0.273938	0.298728
3	0.247559	0.251767	0.261284
4	0.191727	0.30384	0.278413
5	0.218898	0.247089	0.26358
Média	**0,208238**	**0,2657324**	**0,2874236**
Desvio Padrão	**0,02533751438**	**0,0237167682**	**0,03055754895**

Tabela 2: Tabela com os valores de tempo das execuções do sistema.

A **Tabela 1** indica o tempo em mili-segundos gasto para a geração do índice remissivo das estruturas de árvore binária e AVL e lista encadeada, assim como a média e o desvio padrão. A **Tabela 2** indica o tempo em mili-segundos gastos para a geração do índice remissivo para o estrutura da tabela hash com suas três diferentes funções de cálculo de posições, além da média e desvio padrão.

4 Análise dos Resultados

O seguinte questionamento foi feito pelo professor: "Qual das estruturas de dados testadas é indicada para implementar de forma eficiente um índice remissivo?".

Após a execução do programa e a análise feita nos resultados obtidos pode-se concluir que a estrutura que obteve o melhor resultado para a geração do índice remissivo foi a estrutura de tabela hash com a função de hash 1, ou seja, a função de hash do Ziviani. A estrutura que obteve o pior resultado na bateria de testes foi a estrutura da árvore AVL. Isso se deve ao fato de que a AVL precisa de um tempo para o seu balanceamento que sobrepõe o tempo de outras estruturas. Porém, como dito, a tabela hash se manteve como a melhor estrutura, mas a mesma possui uma dificuldade na implementação devido as colisões que podem ocorrer.

5 Conclusão

Pode-se concluir que a melhor estrutura para a geração do índice remissivo é a tabela hash. Também que durante o desenvolvimento do trabalho foi reforçado a implementação de todas as estruturas aprendidas em sala de aula, algoritmos de balanceamento e o mais principal, a experiência obtida para o escolha das estruturas em cada caso.

Referências

ZIVIANI, N. et al. *Projeto de algoritmos: com implementações em Pascal e C*. [S.l.]: Thomson Luton, 2004. v. 2.

Listagem dos Códigos-Fontes

ABB.c
ABB.h
AVL.c
AVL.h
HASH.c
HASH.h
lista.c
lista.h
main.c
compila.sh

Arquivos usados para teste: *chave.txt teste.txt*
Arquivo adicional: *README.md*