UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

Josué Filipe Keglevich
Tatiana Pacheco de Almeida

MECANISMO DE SUGESTÃO DE PALAVRAS PARA CONSULTAS EM MOTORES DE BUSCA

Porto Alegre, 2015

1. INTRODUÇÃO

Este trabalho consiste na implementação de um mecanismo de sugestão de palavras para consulta em motores de busca. Essa atividade foi proposta, para fins didáticos, como trabalho final da disciplina de Estrutura de Dados, na qual diversas estruturas de dados e formas de implementá-las foram estudadas ao longo do semestre como, por exemplo, listas, filas, pilhas, árvores, grafos, entre outras. Cada estrutura tem sua característica e sua especificidade particular e sua utilização dependente do tipo de aplicação e do objetivo comportamental esperado do programa implementado.

O objetivo principal desta atividade é o emprego correto das estruturas de dados mais eficientes para a resolução do problema em questão, juntamente com a análise do tempo de execução do programa.

2. DESCRIÇÃO DA ATIVIDADE

Um mecanismo de sugestão de palavras consiste em tentar prever quais serão os próximos caracteres ou palavras que o usuário irá digitar.

Em motores de busca (ferramenta de busca) essas sugestões de palavras são realizadas com base em diversas métricas. A medida estatística de associação entre as palavras utilizada nesta atividade é: frequencia(palavra1, palavra2) / (sqrt(frequencia(palavra1) * frequencia(palavra2)).

O mecanismo foi implementado na linguagem de programação C e seu funcionamento consiste em duas fases. A primeira fase é a geração de estatística, a qual consiste basicamente em ler um arquivo no formato *"txt"*, passado por parâmetro para a função principal do programa, e gerar as estatísticas de acordo com as ocorrências das palavras no texto fornecido. A

segunda etapa é a fase de consulta, na qual o programa também irá ler um arquivo no formato *"txt"* com as palavras a serem consultadas. O programa gera como saída um arquivo, no formato *"txt"*, com as palavras consultadas, as palavras seguintes sugeridas em ordem decrescente de associação e o tempo gasto na busca.

3. FUNCIONAMENTO DA IMPLEMENTAÇÃO

Na implementação da fase de geração de estatística utilizamos, para a indexação do texto, a estrutura de dados Árvore AVL, na qual as palavras do texto foram inseridas em ordem alfabética, obtida por meio da utilização do código ASCII das letras.

Cada nodo da árvore possui a palavra e uma lista com as suas palavras adjacentes. Em nenhum momento é armazenado palavras repetidas, apenas os indices (variaveis que controlam a frequência de ocorrência) são incrementados.

Na fase de consulta, é verificado a ocorrência da palavra e de suas respectivas palavras adjacentes.

3.1 POR QUE ÁRVORE AVL?

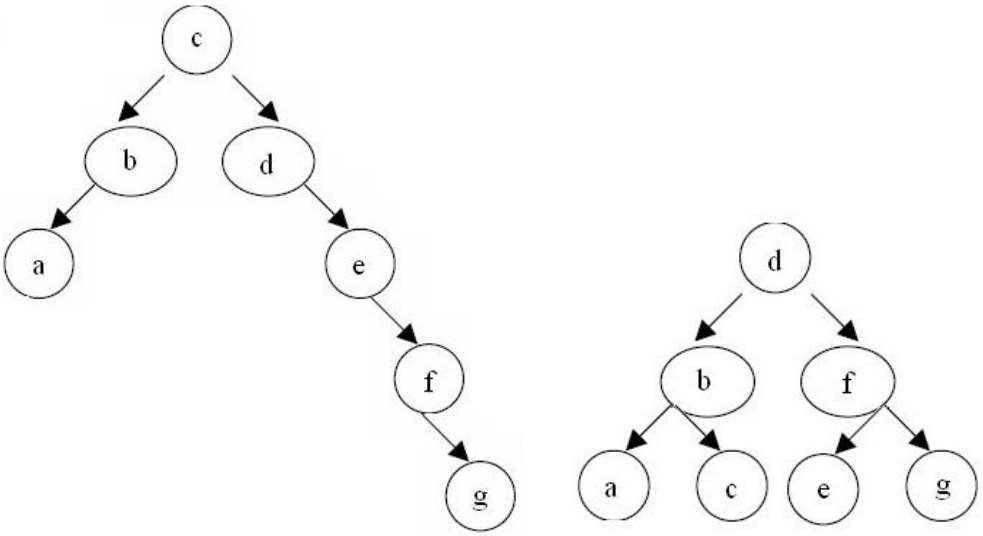
Árvore AVL - Adelson-Velskii e Landis - é uma árvore binária de pesquisa (ABP) construída de tal modo que a altura de sua subárvore direita difere da altura da subárvore esquerda de no máximo uma unidade, resumidamente, é uma árvore autobalanceada.

Uma árvore nada mais é que um conjunto de nós interligados, cada nó pode ter até duas ligações, sendo uma com seu filho da esquerda e outra com seu filho da direita.

No momento da inserção, uma busca é realizada para encontrar o local correto onde a chave, neste caso a palavra, deve ser inserida. Após a inserção, a altura do nó pai e dos nós acima dele são atualizadas e para manter o balanceamento da árvore rotações simples ou duplas são realizadas, de acordo com a necessidade.

No momento da busca, a chave é comparada com a chave do nó que está sendo analisado e se a chave em questão for menor que a desse nó, a busca será realizada no seu filho da esquerda, caso a chave seja maior a busca será no filho da direita do nó analisado. Essa estrutura reduz significativamente o tempo de execução de uma pesquisa, visto que a árvore é percorrida parcialmente através de caminhamentos pré-definidos.

A figura abaixo ilustra claramente a diferença entre uma árvore binaria de pesquisa e uma árvore AVL na questão de balanceamento, característica que torna uma pequisa mais eficiente com a utilização de árvore AVL, pois a diferença de altura entre as suas subárvores é de no máximo 1.



Árvore ABP *Árvore AVL*

Também é visível nas figuras que uma busca na árvore AVL é muito mais eficiente que na outra árvore desbalanceada como, por exemplo, no caso de uma busca pelo nó “f”, na AVL é necessário apenas dois caminhamentos para encontrá-lo e, já na ABP, desbalanceada, é necessário utilizar 4 caminhamentos até encontrar a chave “f” e, assim, mais tempo computacional é utilizado.

No caso de uma árvore desbalanceada, pensando em uma escala muito maior, poderíamos ter uma sequência de inserções onde *palavra1 > palavra2 > palavra3 > ... > palavraN,* resultando assim em um nível muito alto e, posteriormente, uma busca pela *palavraN* gastaria um tempo computacional bem alto.

4. CONCLUSÃO

Devido aos dados serem inseridos de uma forma hierárquica e balanceada na árvore a busca fica otimizada, ao contrário do que aconteceria se utilizássemos, por exemplo, uma lista para armazenar as palavras, pois estaríamos a todo momento percorrendo a lista inteira, fato que geraria um grande aumento no tempo de execução da pesquisa.

Também, se utilizássemos algum outro tipo árvore, correríamos o risco de ter aumento no tempo de execução, visto que um caminhamento poderia ser muito mais comprido que outros, assim, com a utilização de árvore balanceada (AVL) a altura entre as subárvores é de no máximo uma unidade, eliminando assim a possibilidade de um caminhamento desproporcional a outros.

Em um primeiro momento, a atividade proposta foi desenvolvida utilizando apenas listas, porém a partir do momento que passamos a utilizar

árvore AVL para a tokenizaçao do texto obtivemos uma redução de tempo de execução significativa como, por exemplo, texto que levou mais de 400ms para a tokenização, com a utlização apenas de listas, levou 117ms com a utilização de árvore AVL, assim, concluimos que obtivemos sucesso nas escolhas das estruturas de dados escolhidas e também que o objetivo proposto foi alcançado, com base em análises de tempo de execução.