Algoritmos e Estruturas de Dados II

Prof. Laura Assis

Trabalho 1 - Dicionário - AVL

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Objetivo:

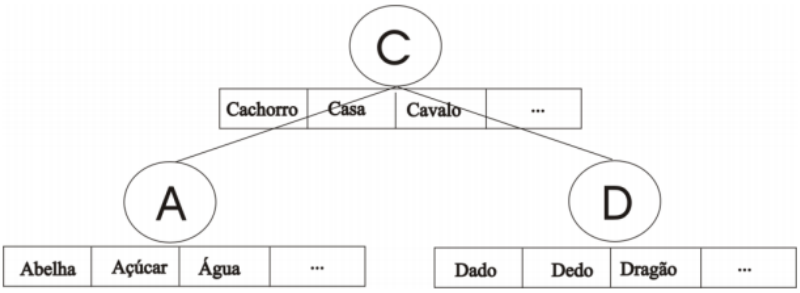
Este trabalho prático consiste na implementação de um **Dicionário** utilizando Árvore AVL.

Como visto, as árvores binárias de busca são estruturas de grande importância quando queremos indexar dados em memória com possibilidade de acesso rápido. Porém, a estrutura de árvore binária de busca não garante que teremos um tempo de acesso médio mínimo, já que não há garantia, para todos os nós, que as subárvores direitas e esquerdas estejam sempre balanceadas (ou seja, possuam aproximadamente a mesma altura). Pensando nisso, algumas árvores binárias de busca balanceadas foram desenvolvidas, uma delas é a árvore AVL.

O objetivo deste trabalho é implementar uma árvore AVL para representar um dicionário. Cada nó da árvore terá:

- Um campo referente a uma letra do alfabeto, sobre a qual serão realizadas as buscas, inserções e remoções de palavras.
- Um campo para armazenar as palavras em uma lista. Desse modo, cada nó também deverá ter uma lista encadeada, a qual deve permitir inserir, remover, buscar e imprimir as palavras que iniciem com a letra correspondente a esse nó da árvore.

A figura a seguir ilustra este dicionário



Operações:

O programa deve ter as seguintes funcionalidades implementadas para uma AVL e lista encadeada:

1. **Busca:** Permite verificar se uma palavra se encontra no dicionário. Recebe uma palavra, busca o nó com sua primeira letra, se o nós existir então busca a palavra na lista daquele nós. Retorno sucesso ou insucesso.
2. **Inserção:** permite inserir uma palavra ainda inexistente no dicionário. Se o nó já existir basta inserir no lugar certo na fila, caso contrário deve criar o nó e a fila, e então inserir a palavra.
3. **Remoção:** permite remover uma palavra, se a palavra existir no dicionário deve remover da lista. Se a mesma for a última a ser removida, o nó da árvore também deve ser removido usando o **antecessor**.
4. **Impressão de um nó:** Permite imprimir todas as palavras daquele nó. Informar se o nó não existir.
5. **Impressão da árvore:** Imprimir cada nó da árvore AVL (em pré-ordem). Para cada nó imprimir a lista de palavras.

Quando a aplicação for executada realizará a leitura de um arquivo contendo uma lista de palavras com as quais irá criar o dicionário. A lista será encerrada quando obter o caracter “0”. Logo em seguida receberá a lista de operações a serem realizadas e o dado necessário para realizar cada operação requisitada.

Será fornecido arquivos de exemplo com uma entrada e o formato de saída. A implementação fornecida pelo aluno deve seguir o exemplo de saída fornecido. O programa deve fornecer saída exatamente igual a pedida pois dada uma determinada entrada 1.in será comparado a saída de seu programa com a saída esperada em 1.out. O arquivo "palavrasordem.txt" pode ser usado para testar o dicionário criado. Arquivos menores de testes devem ser criados e utilizados pelo aluno para avaliar e debugar a aplicação.

Entrega:

O que deve ser entregue?

1. Um arquivo com o **código** do sistema em linguagem C (pelo run codes).
2. Um arquivo em pdf com o **relatório** descrevendo o que foi feito, estruturas usadas, particularidades de suas ideias implementadas, as operações. O relatório deve ter, pelo menos, as seções: resumo, introdução, desenvolvimento, testes, conclusão, bibliografia. Está disponibilizado no Teams um template em Latex para produção do relatório. A entrega do relatório deve ser feita pela tarefa do Teams e deve ter de 6 a 8 páginas.

   Na parte de resultados do relatório deve apresentar um gráfico mostrando a curva do tempo a medida que a quantidade de nós da AVL cresce (relação número de nós $\times$ tempo).
3. Uma apresentação do trabalho de aproximadamente 15 será realizada em laboratório.

Importante!

Use o carregamento dos dados através de arquivo para ganhar tempo e conseguir inserir um volume maior de dados rapidamente e realizar testes. Note que não se faz necessário realizar a manipulação direta do arquivo apenas preparar o arquivo com os dados para carregar em seu programa e redirecionar a entrada para o arquivo ao testar seu código.

Dica! Comece a fazer o trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar!
