Lista de Exercícios sobre Árvores

1- Quantos antecedentes tem um nó no nível n de uma árvore binária? Prove!

2 - Escreva um programa que implementa funções para: a) contar o número de nós em uma árvore binária; b) contar o número de folhas; c) contar o número de filhos à direita; d) contar a altura da árvore.

3- Escreva um programa de referência cruzada que construa uma árvore binária de pesquisa a partir de um arquivo de texto, incluindo todas as palavras, e registre os números das linhas em que essas palavras foram usadas. Esses números de linhas devem ser armazenados em listas ligadas associadas aos nós da árvore. Depois do arquivo de entrada ter sido processado, imprima em ordem alfabética todas as palavras do arquivo de texto, junto com a lista de números de linhas correspondentes nas quais as palavras ocorrem.

4- Questão de Pesquisa: A estrutura heap é um tipo particular de árvore binária, em que (i) o valor de um nó não é menor do que os valores armazenados em seus filhos, (ii) a árvore é perfeitamente balanceada e as folhas do último nível estão todas nas posições mais à esquerda. Qual é, no melhor caso, o número de comparações e de trocas necessários para criar um heap usando (a) o método de Williams (Algorithm 232: Heapsort) e (b) o método de Floyd (Algorithm 245: Treesort 3)?

5 - Árvore AVL: Questão única:

Faça um programa que leia um arquivo texto (em *.txt*) e imprima, em ordem alfabética, as palavras e a suas frequências no texto, além de indicadores de performance dos algoritmos implementados.

A leitura do arquivo deverá desprezar espaços em branco e sinais de pontuação, que serão considerados separadores de palavras. Além disso, a leitura deverá converter todas as letras maiúsculas em minúsculas.

A pesquisa e inserção das palavras do texto deverão ser implementadas com as seguintes estruturas:

1. Pesquisa Binária (utilizando um vetor dinâmico para armazenar as palavras).
2. Árvore Binária de Pesquisa sem balanceamento.
3. Árvore Binária de Pesquisa com balanceamento (Árvore AVL).

Coloque contadores no seu programa para determinar o número de comparações de chaves e atribuições dos registros necessários para montar a tabela de frequências em cada uma das estruturas acima. Conte apenas o número de comparações para montar a estrutura (operações de inserir e pesquisar). Você não deve considerar as operações na fase de impressão ordenada.

Calcule também o tempo que cada estrutura leva para montar a tabela. Analise, por meio dos dados coletados, a eficiência de cada estrutura.

A entrada de dados será um arquivo de texto contendo um texto qualquer. O arquivo texto deverá usar a codificação UTF-8. Seu algoritmo receberá dois parâmetros: (i) nome do arquivo texto que deve ser lido e (ii) estrutura a ser utilizada para impressão das frequências.

Exemplos de execução:

```
./tp entrada.txt pesquisa_binaria
```

```
./tp entrada.txt arvore_binaria
```

```
./tp entrada.txt arvore_avl
```

Exemplo de arquivo texto:

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur.
```

Deverá ser impresso na tela um resumo completo da execução para todas as três estruturas implementadas. O tempo gasto por cada estrutura. Em seguida, as frequências das palavras devem ser impressas. Para esta impressão, deverá ser utilizada a estrutura indicada pelo parâmetro passado na linha de comando. Segue um exemplo para indicar o formato da saída:

```
arvore avl:
    53 comparações
    0.99 segundos

arvore binaria:
    103 comparações
    1.19 segundos

pesquisa binaria:
    103 comparações
    1.89 segundos

frequencia das palavras:
    in: 10
    ipsum: 5
    lorem: 3
    ut: 20
```