Trabalho G1 – Índice invertido

Um índice invertido é uma lista de palavras com a indicação dos locais no texto onde cada palavra chave ocorre. A maioria dos livros apresentam no seu final o índice remissivo como uma forma de facilitar a busca por informações.
Como exemplo, suponha um arquivo contendo um texto constituído como abaixo:

Arquivo 1: Good programming is not learned from
Arquivo 2: generalities, but by seeing how significant
Arquivo 3: programs can be made clean, easy to Linha
Arquivo 5: human-engineered, efficient, and reliable,
Arquivo 6: by the application of common sense and
Arquivo 7: by the use of good programming practices.

Supondo os identificadores 1 e 2 para arquivo1.txt e arquivo2.txt, respectivamente, o índice invertido seria:

and 4,5, 6
be 3
by 2,6,7
…

to 3,4

Note que a lista de palavras chave está em ordem alfabética. Adjacente a cada palavra está uma lista de números de linhas, um para cada vez que a palavra ocorre no texto.

Projete um sistema para produzir um índice invertido utilizando as estruturas de dados vistas em aula. Para facilitar, este sistema deve receber como entrada o nome de um arquivo entrada.txt com formato: N arquivo1.txt arquivo2.txt arquivo3.txt ..... arquivoN.txt

A primeira linha deste arquivo contém um número N que representa o número de documentos da coleção. Cada linha a seguir contém o nome do arquivo que contém um dos documentos da coleção. No exemplo acima, há N documentos que estão armazenados nos arquivos arquivo1.txt, arquivo2.txt, ..., arquivoN.txt. Você pode assumir que estes arquivos, caso existam (você precisa testar), estarão no diretório corrente de execução.

O sistema deverá processar cada um dos arquivos, lendo palavra após palavra e construindo o índice invertido. Ele deverá também associar a cada documento um doc_id único e associar, em memória, este identificador com o nome do documento. Para extrair as palavras de um texto, você pode utilizar o procedimento ExtraiPalavras mostrado na página 205 do livro ``Projeto de Algoritmos”, Nívio Ziviani.

Cabe ressaltar que:
a) Uma palavra é considerada como uma sequência de letras e dígitos, começando com uma letra;
b) Apenas os primeiros cl caracteres devem ser retidos nas chaves. Assim, duas palavras que não diferem nos primeiros cl caracteres são consideradas idênticas;
c) Palavras constituídas por menos do que c1 caracteres devem ser preenchidas por um número apropriado de brancos.
d) Uma palavra pode ocorrer múltiplas vezes na mesma linha de um documento, ou mesmo em múltiplas linhas de um mesmo documento.

Detalhamento do trabalho:

A) O programa deve ler a entrada de um arquivo .txt ou .doc.
B) Deverá ser permitido a inserção de novas palavras após o carregamento do arquivo;
C) Deverá ser utilizado um algoritmo de árvore para implementar o problema;
D) Será considerado o tempo como um dos critérios para a avaliação. Ou seja, se a árvore estiver com alto grau de desbalanceamento, provavelmente as consultas terão um custo maior;
E) Não poderá ser utilizado estruturas prontas, presentes em alguma biblioteca.
F) Será realizada uma aula para apresentação do andamento do trabalho (data a definir)
G) Cada dupla fica responsável pela geração do arquivo de teste. O arquivo deve apresentar a estrutura descrita no exemplo.

Avaliação:

1. *Compilação: 10%*
2. *Execução correta: 40%*
    - *Serão feitos vários testes. Cada teste com um nível de dificuldade maior, onde o arquivo de saída do programa será comparado com um "gabarito". O aluno receberá nota máxima se ambos forem idênticos.*
3. *Estilo de programação: 10%*
    - *Código bem identado, comentado (sem excesso), bem estruturado, utilizando as operações do TAD corretamente, nomes de variáveis significativos, modularização, etc.*
4. *TAD bem definido no código: 10%*
5. *Documentação: 20%*
6. *Apresentação do andamento com a estrutura do código pronto e a lógica para resolver o problema: 10%*

Atrasos:

Será descontado $2^d * 5$ do trabalho (onde d representa o número de dias). Por exemplo, atrasou 3 dias terá um desconto de 40% da nota total.