	Universidade Federal de Viçosa **Campus Florestal** **Algoritmos e Estruturas de Dados II** **2019/1**

1º Trabalho Prático de CCF212 – 15 pts

APLICAÇÃO COM ÁRVORES DIGITAIS

Formação do grupo: O trabalho deverá ser desenvolvido em **trios**

Problema: Construção de índice invertido para máquinas de busca

Máquinas de busca, tais como Google, Bing, Yahoo, trabalham com a busca de palavras-chave em textos armazenados na Web. Para que os documentos contendo os termos sejam recuperados, os mesmos precisam ser devidamente indexados à priori. Nesse contexto, são utilizadas estruturas de dados que facilitem a recuperação das informações, como é o caso do uso de arquivos invertidos. Dada uma coleção de documentos, um índice invertido é uma estrutura contendo uma entrada para cada palavra (termo de busca) que aparece em, pelo menos, um dos documentos. Essa entrada associa a cada palavra do texto um ou mais pares do tipo <qtde, idDoc>, onde qtde corresponde ao número de vezes em que a palavra em questão apareceu em um determinado documento identificado por idDoc. O nome dado à estrutura indica que houve uma inversão da hierarquia da informação, ou seja, ao invés de uma lista de documentos contendo termos, é obtida uma lista de termos, referenciando documentos. Essa estrutura de índices é comumente implementada com base em *árvores* e tabelas *hash*, pois as mesmas não precisam ser reconstruídas a cada atualização.

Para exemplificar, considere os dois documentos mostrados abaixo[1]:

Texto 1 (arquivo1.txt) = "*Quem casa quer casa. Porem ninguem casa. Ninguem quer casa tambem. Quer apartamento.*"

Texto 2 (arquivo2.txt) = "*Ninguem em casa. Todos sairam. Todos. Quer entrar? Quem? Quem?*"

Supondo os identificadores 1 e 2 para os textos apresentados, arquivo1.txt e arquivo2.txt, respectivamente, o índice invertido para as palavras contidas nos textos pode ser visualizado na Tabela 1.

1Assuma que os textos não terão acentuação.

Tabela 1 – Índices invertidos para os textos apresentados

Palavra	**<qtde, idDoc>**	**<qtde, idDoc>**	**...**
apartamento	<1,1>		
casa	<4, 1>	<1, 2>	
em	<1, 2>		
entrar	<1, 2>		
ninguem	<2, 1>	<1, 2>	
porem	<1, 1>		
quem	<1, 1>	<2, 2>	
quer	<3, 1>	<1, 2>	
sairam	<1, 2>		
tambem	<1, 1>		
todos	<2, 2>		

Conforme pode ser observado na Tabela 1, para cada palavra, existe uma lista de pares de números <qtde, idDoc>, ordenada pelo campo idDoc. Essa lista poderia ser implementada como uma lista encadeada para cada palavra indexada.

Tarefas:

- **Receber como entrada *n* arquivos cujas palavras serão indexadas.**
- **Construir a árvore PATRICIA** para armazenar as palavras contidas nos textos, indexando-as com índices invertidos. Seu índice não deve diferenciar letras maiúsculas de minúsculas. Portanto, as palavras com letras em maiúsculas devem ser transformadas para minúsculas antes da inserção na estrutura.
- **Implementar uma função de busca por textos com base em termo (s) de busca,** utilizando o índice invertido para localizar os textos em que ele (s) aparece (m). Os textos devem ser retornados de forma ordenada pela sua relevância para a consulta, ou seja, textos que contém um maior número de

ocorrências do (s) termo (s) de busca devem aparecer primeiro. Para calcular a relevância de um texto para o (s) termo (s) de busca, você deve utilizar uma técnica de ponderação baseada no cálculo da frequência da ocorrência do (s) termo (s) nos documentos, conhecida como TF-IDF (*Term frequency – Inverse Document Frequency*). O componente IDF estima o quanto um termo ajuda a discriminar os documentos entre relevantes e não relevantes. Um termo que aparece em muitos documentos tem valor de IDF baixo, enquanto um termo que aparece em poucos documentos apresenta IDF alto, sendo um bom discriminador.

Dada uma consulta com q termos, t1, t2, ..., tq, a relevância de um documento i, r(i), é computada como:

$r(i) = \frac{1}{n_i} \sum_{j=1}^{q} w^i_j$	Onde *ni* = número de termos distintos do documento i $w_{j,i}$= peso do termo tj no documento i
$w^i_j = f^i_j \frac{\log(N)}{d_j}$	$f_{j,i}$ = número de ocorrências do termo tj no documento i dj = número de documentos na coleção que contém o termo tj N = número de documentos na coleção. Se o termo tj não aparece no documento i, $f_{j,i}$ = 0

Para exemplificar, considere uma consulta com os termos "*quer*" e "*todos*"[2]:

- dois termos (q = 2)
- dois documentos: N = 2
  - o documento 1 tem 7 termos (n1 = 7)
  - o documento 2 tem 8 termos (n2 = 8)
- o número de ocorrências do primeiro termo (quer), no documento 1 é 3 e no documento 2 é 1, logo $f_{1,1}$ = 3 e $f_{2,1}$ = 1. Além disto dj = 2.
  - $w_{1,1}$ = 3*log(2)/2 = 1.5
  - $w_{1,2}$ = 1*log(2)/2 = 0.5

2 Exemplo retirado do material da Profa. Jussara Marques de Almeida (UFMG)

• o número de ocorrências do segundo termo (todos), no documento 1 é 0 e no documento 2 é 2, logo $f_{1,2} = 0$, $f_{2,2} = 2$ e dj = 1

- $w_{2,1} = 0*\log(2)/1 = 0$
- $w_{2,2} = 2*\log(2)/1 = 2$

• Logo, as relevâncias dos documentos pra esta consulta são:

- r(1) = 1/7 *(1.5 + 0) = 0.21
- r(2) = 1/8 *(0.5 + 2) = 0.31

A partir das relevâncias calculadas para cada documento, o método retornará os documentos ordenados da seguinte forma:

Texto 2 (arquivo2.txt)

Texto 1 (arquivo1.txt)

- **Implementar o recurso de autopreenchimento para auxiliar o usuário na inserção dos termos da busca:** utilize uma árvore TRIE do tipo TST para isso, ou seja, as palavras deverão ser armazenadas também na árvore TRIE.

- **Implementar um menu com as seguintes opções:** a) construir o índice invertido, a partir dos textos de entrada, usando o TAD árvore PATRICIA; b) imprimir o índice invertido, contendo as palavras em ordem alfabética, uma por linha, com suas respectivas listas de ocorrências e; c) buscar palavra (s) no (s) texto (s), a partir do índice construído.

Entrega:

➔ O trabalho deverá ser entregue via PVANET, por um dos integrantes do trio, através de um **único** arquivo compactado, contendo:
  - código-fonte do programa em C;
  - arquivos utilizados como entrada (textos para testes e termos de busca);
  - arquivo "leiame.txt" com explicações de uso do programa;
  - o arquivo compactado deverá conter o nome e a matrícula dos integrantes do trio.

➔ Data de entrega: **22/04/19**

➔ **Data de apresentação/entrevista: 23 e 25/04/19**

	**Universidade Federal de Viçosa** **Campus Florestal** **Algoritmos e Estruturas de Dados II** **2019/1**

Comentários Gerais:

- O trio deverá tomar como base, os códigos discutidos em aula, retirados do livro texto da disciplina (Ziviani, 2010). Outras fontes poderão ser consultadas;
- O código-fonte DEVERÁ ser devidamente comentado;
- As implementações relativas a cada TAD devem estar em arquivos separados;
- As operações referentes à leitura e à carga dos dados devem estar em um arquivo separado;
- As operações referentes à montagem do índice invertido devem estar em um arquivo separado;
- Atenção quanto ao uso e inicializações de variáveis no programa principal, que podem comprometer o funcionamento do seu código (Encapsular funções sempre que possível);
- Os arquivos fornecidos como exemplo devem ser utilizados apenas para fins de verificação e validação do algoritmo. No entanto, pelo menos 2 outros textos, com um maior número de palavras devem ser utilizados (mínimo de 500 palavras) e entregues junto ao código;
- O trio deverá ser identificado no cabeçalho de TODOS os arquivos do código-fonte;
- Apesar de o trabalho ser em trios, a nota poderá ser individualmente atribuída, a critério do professor (entrevistas individuais poderão ser realizadas);
- Trabalhos compartilhados (copiados) ou obtidos da internet terão nota zero para todos os envolvidos. Atenção ao conceito F!;
- Trabalhos entregues em atraso não serão considerados (Nota = 0).
- Durante o desenvolvimento do trabalho, caberá ao trio propor e construir uma implementação para o problema apresentado. Não cabe à professora e ao monitor a análise de erros em código-fonte, nem tampouco o fornecimento de detalhes técnicos da solução a ser construída.