Algoritmos e Estruturas de Dados II

Trabalho Prático 4

Entrega: 23/11/09
Devolução: 10/12/09 (sem possibilidade de entrega com atraso)
Trabalho em dupla
Prof. Jussara Marques de Almeida

Problema 1: Construção de Índice Invertido para Máquinas de Busca

A base para o funcionamento de uma máquina de busca, tais como Google e Yahoo!, é a construção de um índice invertido para uma coleção de documentos. Dada a coleção de documentos, um índice invertido é uma estrutura contendo uma entrada para cada palavra (termo) que aparece em pelo menos um documento. Esta entrada associa à palavra pares do tipo <*count, doc_id*>, onde *doc_id* identifica um documento e *count* é o número de vezes em que a palavra em questão apareceu no documento *doc_id*. Documentos que não contém a palavra não precisam ser indicados.

Como exemplo, suponha uma coleção com apenas os dois documentos seguintes:

arquivo1.txt
Quem casa quer casa. Porem ninguem casa.
Ninguem quer casa tambem. Quer apartamento.

arquivo2.txt
Ninguem em casa. Todos sairam. Todos.
Quer entrar? *Quem*? *Quem*?

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

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

Note que, adjacente a cada palavra do índice está uma lista de pares de números representando ocorrências da palavra nos documentos. O primeiro número de cada par representa o número de vezes que a palavra ocorre em um documento e o segundo representa o identificador do documento em questão. Note que a lista está ordenada por identificadores de documentos.

Projete um sistema para produzir um índice invertido. 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. Portanto, ignore sinais de pontuação. Você pode assumir que os textos não terão acentuação.

b) Palavras com letras em maiúsculas devem ser primeiramente transformadas para minúsculas antes da inserção no índice. Desta forma, a mesma palavra apresentada com letras minúsculas ou maiúsculas não serão diferenciadas.

c) Apenas os primeiros C caracteres devem ser retidos nas chaves. Assim, duas palavras que não diferem nos primeiros C caracteres são consideradas idênticas. Assuma que C é um parâmetro de entrada do seu programa, e que ele pode assumir valores iguais ou superiores a 20.

d) Palavras constituídas por menos do que C caracteres devem ser preenchidas por um número apropriado de brancos.

e) Uma palavra pode ocorrer múltiplas vezes na mesma linha de um documento, ou mesmo em múltiplas linhas de um mesmo documento.

Organize o seu código da seguinte maneira. O sistema para construção de índice invertido deverá ser chamado como um procedimento (p.ex: CriaIndiceInvertido) pelo programa principal. Ou seja, além de operações de inicialização, o seu programa principal (versão 1) deve chamar a *operação* CriaIndiceInvertido, a partir da qual a construção é realizada.

Você fica responsável por criar coleções de documentos para testes. Submeta junto com o seu trabalho (no pacote zip) de coleção usada.

Soluções a Serem Apresentadas:

Para implementar a operação CriaIndiceInvertido, o seu programa deve utilizar as operações do Tipo Abstrato de Dados *Dicionário.* Você deve implementar QUATRO (4) versões do TAD *Dicionário*:

1. hash aberto (como dado em sala de aula)
2. hash aberto utilizando duas funções hash para resolução de conflito
3. hash com resolução de conflito via árvore binária sem balanceamento
4. árvore Patrícia

Para o TAD 2, você deverá utilizar duas funções hash, h1 e h2: h2 deverá ser utilizada para resolução dos conflitos de h1, enquanto que os conflitos gerados por h2 deverão ser resolvidos como no hash aberto tradicional. Note que h1 deve ser a mesma função utilizada no TAD 1 (hash aberto simples), para fins de comparação.

Sugestão: passe como parâmetro da operação CriaIndiceInvertido um indicador de qual TAD ela deve utilizar. Por exemplo, utilize os códigos 1 (HASHABERTOSIMPLES), 2 (HASHABERTOCOMPOSTO), 3 (HASHARVORE), 4 (PATRICIA), conforme a descrição acima.

O seu programa deve ser instrumentado para medir o desempenho, em termos do tempo de execução e do consumo de memória, de cada um dos TADs para diferentes cenários. Utilize contadores bem como chamadas do sistema para medições das estatísticas solicitadas. Execute diversos experimentos, variando a entrada e o tamanho *M* da tabela hash.

Problema 2 : Consultas sobre o Índice Invertido.

Nesta parte do projeto, você utilizará o índice invertido construído utilizando o TAD 2 (HASHABERTOCOMPOSTO) para realizar consultas de um ou mais termos. O objetivo é, dado uma consulta, retornar uma lista de documentos, ordenados (em ordem decrescente) pela relevância para a consulta. Existem vários métodos de estimar a relevância de documentos de uma coleção para uma consulta. Vocês utilizarão um simples, baseado na frequência dos termos da consulta em cada documento (TF – *term frequency*) da coleção bem como na frequência inversa dos documentos (IDF – *inverse document frequency*). O componente IDF estima o quão bem um termo ajuda a

discriminar os documentos entre relevantes e não relevantes: um termo que aparece em muitos documentos (valor de IDF baixo) não é um bom discriminador, enquanto que um termo que aparece em poucos documentos (IDF alto) é um bom discriminador dos mesmos.

Em outras palavras, dada uma consulta com *q* termos, t_1, t_2, ..., t_q, 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 n_i é o número de termos distintos do documento i e w^i_j é o peso do termo t_j no documento i, que é calculado como:

$$w^i_j = f^i_j \frac{\log(N)}{d_j}$$

onde f^i_j é o número de ocorrências do termo t_j no documento i , d_j é o número de documentos na coleção que contém o termo t_j e N é o número de documentos na coleção. Se o termo t_j não aparece no documento i, $f^i_j = 0$.

No exemplo dado acima, uma consulta "*quer todos*":

- dois termos (*q = 2*): *quer* e *todos*
- há dois documentos: N = 2
  - o documento 1 tem 7 termos : $n_1 = 7$
  - o documento 2 tem 8 termos : $n_2 = 8$
- 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 $d_j = 2$.
  - $w^1_1 = 3*\log(2)/2 = 1.5$
  - $w^2_1 = 1*\log(2)/2 = 0.5$
- 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 $d_j = 1$
  - $w^1_2 = 0*\log(2)/2 = 0$
  - $w^2_2 = 2*\log(2)/2 = 1$
- Logo, as relevâncias dos documentos pra esta consulta são:
  - r(1) = 1/7 *(1.5 + 0) =
  - r(2) = 1/8 *(0.5 + 1) =
  - você deve retornar a lista de documentos
    arquivo1.txt
    arquivo2.txt

Crie um novo programa que estenda o sistema criado para, além de realizar a operação CriaIndiceInvertido, inclua também as seguintes operações, a serem executadas sobre o índice invertido construído:

a) Realizar consultas com um ou mais termos: deve retornar uma lista com os nomes dos documentos, ordenada pela relevância dos mesmos para a consulta dada. Esta lista não deve conter documentos cuja relevância (como calculado acima) esteja abaixo de um limiar L (parâmetro do seu sistema).
b) Imprimir o índice invertido: imprime as palavras em ordem alfabética, uma por linha. À frente de cada palavra, você deve imprimir a lista das ocorrências, ordenada pelo índice do documento. Para cada elemento da lista, você deve imprimir o nome do documento (e não o identificador) e o número de ocorrências da palavra no documento.

Construa um novo programa principal. Diferentemente do programa anterior, este deve ser *interativo*, isto é, deve permitir ao usuário, primeiramente, solicitar a construção do índice remissivo, fornecendo conjunto de palavras e texto, para em seguida permitir a execução de uma ou mais das operações acima.

Organize o seu código de forma que:

- as implementações relativas a cada TAD estejam em arquivos diferentes, um para cada TAD.
- as operações sobre os índices remissivos inclusive a operação de criação estejam em um arquivo separado.
- o programa principal (main em C) relativo ao Problema 1 esteja em um arquivo separado. Chame-o, por exemplo, de mainCriacao.
- o programa principal (main em C) relativo ao Problema 2 esteja em um outro arquivo. Chame-o, por exemplo, de mainSistema

Crie um arquivo **Makefile** e utilize o utilitário **make** para compilação e geração de código executável automaticamente. Crie uma opção runall no seu Makefile que dispara todas as compilações necessárias, gerando ao final os dois executáveis, mainCriacao e mainSistema, para um dado TAD.

As regras para realização e entrega do trabalho estão disponíveis no *site* da disciplina. **Além das informações listadas nessa página**, a documentação deste trabalho deverá conter também:

- uma descrição das estruturas de dados
- uma descrição da instrumentação realizada
- um estudo comparativo do desempenho relativo dos vários TADs nos diferentes cenários avaliados, considerando as duas estatísticas: tempo de execução e consumo de memória. Para tanto, apresente gráficos e tabelas com as estatísticas coletadas para os diferentes valores de M avaliados. **Discuta** os resultados na luz da teoria apresentada em sala de aula. Em particular: *quais são os compromissos de desempenho observados?*Ou, em outras palavras, *quais algoritmos apresentaram melhores resultados para cada cenário?*
- uma descrição da sintaxe e semântica das operações implementadas
- estudo da complexidade das operações de cada TAD.
- estudo da complexidade das operações sobre o índice invertido, considerando, separadamente, cada TAD.

Submeta o seu Makefile juntamente com os demais arquivos, no pacote zip.