Universidade Federal do Espírito Santo
Departamento de Informática
Professora: Claudine Badue
Disciplina: Estruturas de Dados II
Ano/Período: 2017/2
Curso: Ciência da Computação
Atividade: Exercício de Programação 3

Índices Invertidos

Definição

Um índice invertido (ou arquivo invertido) é um mecanismo orientado a palavra para indexar uma coleção de documentos a fim de acelerar a tarefa de busca [Baeza-Yates and Ribeiro-Neto, 2011]. A estrutura do índice invertido é composta de dois elementos: o *vocabulário* e as *ocorrências*. O vocabulário é o conjunto de todas as palavras diferentes no texto. Para cada palavra no vocabulário, o índice armazena cada documento que contém a palavra e o número de vezes que a palavra ocorre no documento. Por esta razão ele é chamado de *índice invertido*, porque podemos reconstruir o texto usando o índice. Esse é o principal índice usado atualmente e é também o mais antigo.

O modo mais simples de representar os documentos que contém cada palavra do vocabulário é uma matriz cujas células armazenam o número de vezes que uma palavra ocorre em um documento – a matriz termo-documento. Isso é ilustrado na Figura 2(a) para a coleção de documentos da Figura 1. Essa representação simples é muito rápida, dado que apenas um acesso à matriz é requerido para determinar se um documento contém uma dada palavra.

To do is to be. To be is to do.	To be or not to be. I am what I am.	I think therefore I am. Do be do be do.	Do do do, da da da. Let it be, let it be.
d_1	d_2	d_3	d_4

Figura 1: Pequena coleção de exemplo composta de quatro documentos [Extraído da Figura 3.6 de Baeza-Yates e Ribeiro Neto (2011)].

Matriz termo-documento					**Ocorrências como Listas Invertidas**
	d_1	d_2	d_3	d_4	
to:	4	2	-	-	to: [1,4],[2,2]
do:	2	-	3	3	do: [1,2],[3,3],[4,3]
is:	2	-	-	-	is: [1,2]
be:	2	2	2	2	be: [1,2],[2,2],[3,2],[4,2]
or:	-	1	-	-	or: [2,1]
not:	-	1	-	-	not: [2,1]
i:	-	2	2	-	i: [2,2],[3,2]
am:	-	2	1	-	am: [2,2],[3,1]
what:	-	1	-	-	what: [2,1]
think:	-	-	1	-	think: [3,1]
therefore:	-	-	1	-	therefore: [3,1]
da:	-	-	-	3	da: [4,3]
let:	-	-	-	2	let: [4,2]
it:	-	-	-	2	it: [4,2]
(a)					(b)

Figura 2: (a) Índice invertido básico e a matriz termo-documento para a coleção ilustrada na Figura 1. Cada entrada da matriz indica a frequência do termo no documento. Note que o vocabulário armazena o número de documentos nos quais um termo aparece e que entradas com 0 na matriz são substituídas por "–" para facilitar a leitura. (b) A matriz termo-documento é substituída por listas

invertidas, cujas entradas são compostas do documento contendo o termo e a frequência do termo correspondente [Extraído da Figura 9.1 de Baeza-Yates e Ribeiro Neto (2011)].

O principal problema dessa solução simples é que ela requer muito espaço (proporcional ao número de documentos multiplicado pelo tamanho do vocabulário). Como essa é uma matriz esparsa (a maioria dos documentos contém um pequeno subconjunto do vocabulário), a solução é associar uma lista de documentos com cada palavra. O conjunto de todas essas listas é chamado de "ocorrências". A Figura 2(b) mostra as listas invertidas associadas com o mesmo exemplo. O espaço usado é proporcional às ocorrências das palavras nos documentos e é bem menor do que o tamanho do texto.

A pesquisa em um índice invertido tem geralmente três passos [Ziviani, 2011]:
1. *Pesquisa no vocabulário*: As palavras presentes na consulta são isoladas e pesquisadas no vocabulário.
2. *Recuperação das ocorrências*: As listas de ocorrências de todas as palavras encontradas no vocabulário são recuperadas.
3. *Manipulação das ocorrências*: As listas de ocorrências são processadas para encontrar os documentos mais prováveis de serem relevantes de acordo com algum modelo de ordenação.

Logo, pesquisar em um índice invertido sempre começa pelo vocabulário, sendo pois interessante mantê-lo em um arquivo separado. Na maioria das vezes, esse arquivo cabe na memória principal.

A pesquisa de palavras pode ser realizada usando qualquer estrutura de dados que torne a busca eficiente, como *hashing*, árvore *trie* ou árvore B. As duas primeiras têm custo $O(m)$, em que m é o número de caracteres da consulta. Entretanto, armazenar as palavras na ordem lexicográfica é barato em termos de espaço e muito competitivo em desempenho, já que a pesquisa binária pode ser empregada com custo $O(\log n)$, sendo n o número de palavras da consulta.

Implementação

Projete um sistema para produzir um índice invertido usando listas invertidas. Para cada palavra extraída de cada documento, no vocabulário, o sistema deverá inserir a palavra e, na lista invertida da palavra, o identificador do documento e a frequência da palavra no documento. Para extrair as palavras de um documento, use o procedimento disponível em http://www2.dcc.ufmg.br/livros/algoritmos/cap5/codigo/c/5.43-extraipalavra.c e o alfabeto disponível em alfabeto.txt abaixo.

Utilize um método eficiente para verificar se uma palavra lida do texto pertence ao índice. Para resolver esse problema, implemente o índice como uma tabela *hash* com endereçamento aberto e *hashing* linear para resolver colisões.

Observe que existe uma desvantagem na utilização do *hashing*: após atualizado todo o índice, é necessário imprimir suas palavras em ordem alfabética, sendo necessário ordenar a tabela *hash* que contém o índice.

Escreva o índice em um arquivo de texto denominado arquivosaida.txt.

Utilize o exemplo anterior para testar seu programa. Utilize também o conjunto de documentos disponível em documentos.zip abaixo.

Instruções de Desenvolvimento

Este trabalho deverá ser desenvolvido em grupos de dois alunos.

O programa deste trabalho deverá ser implementado usando a linguagem C.

O programa deverá também ser modularizado. Crie arquivos de extensão .c e .h para cada módulo do programa. Crie arquivos exclusivos para definir tipos abstratos de dados.

O programa deverá também ser compilado e executado com o aplicativo make. O programa deverá ser compilado com o comando make all e executado com o comando make run.

Instruções de Entrega

Submeta, até às 23:55 horas da data limite de entrega, o arquivo makefile, os arquivos do código (*.c e *.h), os arquivos de entrada (alfabeto.txt e documentos.zip) e o arquivo de saída (arquivosaida.txt).

Maiores Detalhes

Maiores detalhes serão discutidos em sala de aula. Considerações feitas em sala terão valor superior ao daquelas contidas nesta especificação.

Referência

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: the Concepts and Technology Behind Search.* 2nd edition. Addison Wesley, 2011.

N. Ziviani. *Projeto de Algoritmos: com Implementações em PASCAL e C*. 3a edição revista e ampliada. Cengage Learning, 2011.