Pontifícia Universidade Católica de Goiás
Departamento de Computação
CMP1054 - Estrutura de Dados II
Max Gontijo de Oliveira
2º Trabalho - Tabela Hash

1 Resumo

Deverá ser criado um motor de busca de documentos por palavras-chave ordenados por relevância, tais como os disponibilizados em sites de busca, como *Google* e *Yahoo*. **Deverá ser desenvolvido em grupos de dois ou três alunos**.

2 Detalhes

O motor de busca de documentos a partir de palavras-chave que será desenvolvido nesse trabalho deve manter um índice invertido. Dado um conjunto de documentos, um índice invertido é uma estrutura que tem uma entrada para cada palavra que exista em pelo menos um dos documentos da coleção. Cada entrada (cada palavra) dessa estrutura (tabela hash), está associada à uma lista de pares $\{id\_doc, qtde\}$, onde id_doc é o identificador do documento e $qtde$ é um contador que armazena a quantidade de ocorrências da palavra no documento em questão.

A partir desse índice invertido, uma consulta de documentos passa a ser simples consultas à tabela hash. A seção 2.2 vai descrever em detalhes como devem ser realizadas essas consultas.

2.1 Criando o índice invertido

Considere a existência de um arquivo com o nome **entrada.txt**. O conteúdo desse arquivo será uma lista de nomes de outros arquivos **.txt** (um por linha) que também estarão no mesmo diretório. Apenas a primeira linha é que não será um nome de arquivo: será um número indicando quantos arquivos tem nessa lista. Considere que não haverão caracteres de espaço no nome de nenhum desses arquivos. Segue exemplo de como seria o arquivo:

```
4
faroeste_caboclo-legiao_urbana.txt
a_ultima_pergunta-isaac_asimov.txt
o_guia_do_mochileiro_das_galaxias-douglas_adams.txt
cosmos-carl_sagan.txt
```

O programa deverá ser capaz de ler esse arquivo de entrada e, para cada linha (nome de outro arquivo que está no mesmo diretório), o programa deverá abrir e ler esse outro arquivo inteiramente para fazer uma análise de seu conteúdo, de modo todas as diferentes palavras de cada um desses arquivos irão compor o índice invertido.

A construção do índice deverá seguir a estrutura já mencionada e formalmente descrita a seguir:

- Deverá ser criada uma tabela hash onde a chave seja a palavra e o tipo de dado a ser guardado nessa tabela é uma lista de pares $\{id\_doc, qtde\}$.
  - Em cada par $\{id\_doc, qtde\}$, considere, INICIALMENTE, que id_doc seja o nome do arquivo (string) e $qtde$ seja o número de ocorrências da palavra em questão no arquivo identificado por id_doc.
- Sobre o texto dos arquivos:
  - Considere que haverão nos textos apenas letras, números, espaços, hifens, quebras de linha e sinais de pontuação.
  - Considere que não haverão nos textos caracteres com marcações especiais, tais como acentos, tremas, cedilhas, etc.
  - Considere que o texto não é sensível ao caso, ou seja, letras maiúsculas e letras minúsculas devem ser indistinguíveis. Assim, ocorrências no texto como Casa, casa, CASA ou cAsA, por exemplo, devem se referir à mesma palavra. **Dica:** converta todas as letras de uma palavra (chave) para minúsculo antes de armazenar no índice (tabela hash). NÃO É PARA ALTERAR OS ARQUIVOS.
  - Considere que uma palavra só poderá ser catalogada se a mesma tiver **pelo menos dois caracteres**.
  - Considere que uma palavra só poderá ser considerada palavra se a mesma se iniciar com uma letra.
  - Demais caracteres tais como espaços, quebras de linha e sinais de pontuação deverão ser igualmente ignorados como chaves, sendo úteis apenas para separar as palavras uma das outras.

2.2 Consultando o índice invertido

O seu motor de busca deverá conseguir efetuar três tipos de buscas:

- busca de documentos por uma única palavra;
- busca de documentos que contenham todas as palavras de um conjunto de duas ou mais palavras;
- busca de documentos que tenham pelo menos uma das palavras de um conjunto de duas ou mais palavras.

Efetuar consultas ao motor de busca é uma tarefa bem simples.Inicialmente, havia-se dito que a tabela hash armazenaria listas de pares $\{id\_doc, qtde\}$ sob palavras distintas como chave. Para conseguir ter a informação de quantidade de palavras distintas em cada arquivo, ao invés de id_doc ser uma string que tem o nome do arquivo, deverá ser o ponteiro para um objeto `Documento`, onde esse objeto tenha os atributos: nome do arquivo e quantidade de palavras distintas no arquivo. Cada vez que um arquivo for aberto para análise, deverá ser criado um objeto desse e toda vez que for criado um registro para a lista de pares $\{id\_doc, qtde\}$ (ou seja, encontrou uma palavra que ainda não havia aparecido no arquivo), o atributo "quantidade de palavras distintas"do objeto `Documento` referente ao arquivo sendo analisado deverá ser incrementado.

A partir do índice invertido, uma consulta de documentos contendo a palavra w resume-se em buscar na tabela hash, a lista de pares $\{id\_doc, qtde\}$ que está guardada sob essa chave (palavra w) e retornar uma lista contendo todos os documentos que existem nessa lista de pares.

Uma consulta de documentos contendo TODAS as palavras do conjunto de n palavras $W = \{w_1, w_2, \ldots, w_n\}$ resume-se em buscar na tabela hash, as n listas de pares $\{id\_doc, qtde\}$ referentes às n palavras da busca e retornar uma lista contendo todos os documentos que estão presentes em todas essas n listas, ignorando os documentos que não estão presentes em pelo menos uma delas. Obviamente não devem haver registros (nomes de arquivos) repetidos no retorno.

Uma consulta de documentos contendo ocorrências de PELO MENOS UMA das palavras do conjunto de n palavras $W = \{w_1, w_2, \ldots, w_n\}$ resume-se em buscar na tabela hash, as n listas de pares $\{id\_doc, qtde\}$ referentes às n palavras do conjunto W e retornar uma lista contendo todos os documentos que estão presentes nessas n listas. Obviamente não devem haver registros (nomes de arquivos) repetidos no retorno.

Contudo, ainda resta uma etapa na consulta que é ordenar os registros retornados, do mais relevante ao menos relevante registro retornado na busca.

No caso de uma consulta por uma única palavra, a ordenação pode ser simplesmente realizada em ordem decrescente pelo número de ocorrências da palavra w no arquivo.

Entretanto, quando a busca é feita com base em um conjunto de duas ou mais palavras, a ordem por relevância pode ser realizada de diversas maneiras.

Uma maneira bem simples consiste em, para cada arquivo que possui todas as palavras da busca, efetuar a soma das ocorrências de cada palavra da busca e ordenar de forma decrescente a lista de registros por essa soma.

O problema dessa abordagem é que ela não considera a relevância de cada palavra. Num conjunto de documentos existem palavras que distinguem com mais relevância um arquivo do outro do que outras palavras.

Por exemplo, uma palavra w_a que aparece em 20% dos arquivos é mais relevante para distinguir os arquivos do que uma palavra w_b que aparece em 95% dos arquivos. Assim, em uma busca por registros com TODAS ou PELO MENOS UMA das palavras do conjunto $W = \{w_a, w_b\}$ deveria considerar um peso maior para o número de ocorrências de w_a do que o número de ocorrências de w_b.

Outro exemplo seria o seguinte: suponha que dois arquivos a_1 e a_2 possuem a mesma quantidade de ocorrências da palavra w_b. Entretanto, suponha que o arquivo a_1 tenha 30 palavras distintas enquanto o arquivo a_2 tenha 100 palavras distintas. Dessa forma, a palavra w_b é mais relevante para o arquivo a_1 do que para o arquivo a_2.

2.2.1 Como deverá ser a implementação

A ordenação deverá se resumir a calcular a relevância r_i de cada registro i (arquivo) retornado da busca pelas n chaves e ordenar os registros de forma decrescente por essa relevância. Nesse trabalho, considere a seguinte forma realizar esse cálculo:

$$r_i = \frac{1}{q_i} \sum_{j=1}^{n} p_{ij}$$

, onde q_i é a quantidade de palavras distintas no arquivo i e p_{ij} é o peso da chave j na palavra i. Esse peso deverá ser calculado da seguinte maneira:

$$p_{ij} = f_{ij} \frac{log(N)}{d_j}$$

, onde f_{ij} é o número de ocorrências da palavra w_j no arquivo i, d_j é o número de documentos na coleção de arquivos que contém a chave w_j e N é o número de documentos de toda a coleção. Note que, se a chave w_j não aparece no documento i (caso em que a busca é realizada com mais de um termo e deverão ser retornados os arquivos que tiverem PELO MENOS UM dos termos de busca), então $f_{ij} = 0$.

2.2.2 Consideração sobre a implementação

Inicialmente, havia-se dito que a tabela hash armazenaria listas de pares $\{id\_doc, qtde\}$ sob palavras distintas como chave. Para conseguir ter a informação de quantidade de palavras distintas em cada arquivo, ao invés de id_doc ser uma string que tem o nome do arquivo, sugere-se que esse valor deverá ser o ponteiro para um objeto `Documento`, onde esse objeto tenha os atributos: nome do arquivo e quantidade de palavras distintas no arquivo. Cada vez que um arquivo for aberto para análise, deverá ser criado um objeto desse e toda vez que for criado um registro para a lista de pares $\{id\_doc, qtde\}$ (ou seja, encontrou uma palavra que ainda não havia aparecido no arquivo), o atributo "quantidade de palavras distintas"do objeto `Documento` referente ao arquivo sendo analisado deverá ser incrementado.

2.3 A estrutura de dados tabela hash

O uso da estrutura de tabela hash é obrigatório nesse trabalho no ponto em que foi mencionada. Existem bibliotecas para o C++ que possuem há a implementação de tabela hash pronta. Evidentemente, neste trabalho é essencial e indispensável que a tabela hash a ser utilizada deve ser implementada pelo grupo.

A tabela hash que deve ser implementada deve utilizar endereçamento aberto. Isso significa que não haverão listas ligadas e o tratamento de colisão deverá ser tratado com sondagem de hash duplo.

As duas funções hash desenvolvidas para o cálculo do hash duplo deverão ser criadas pelo grupo seguindo as boas práticas de criação de funções hash vistas em sala de aula ou outras que o aluno encontrar em bibliografias consolidadas.

O tamanho do vetor da tabela hash deverá ser criado com base em um argumento passado pelo construtor. A tabela hash poderá ter também um construtor padrão (sem argumentos) que determine um tamanho fixo para o vetor.

2.4 Programa principal

O programa principal deverá funcionar da seguinte forma:

1. Programa abre o arquivo **entrada.txt** e constrói o índice invertido.
2. Programa disponibiliza um menu para que o usuário escolha o tipo de busca que deseja fazer:
   - trazer registros com ocorrência de TODOS os termos de busca (palavras chave)
   - trazer registros com ocorrência de PELO MENOS UM dos termos de busca (palavras chave)
3. Usuário escolhe o tipo de busca e informa os termos de busca (palavras chave) separados por espaço (tal como em sites de busca como *Google* e *Yahoo*)
4. Programa efetua a busca conforme solicitação do usuário e apresenta os nomes dos documentos (arquivos) encontrados na busca, ordenados pela relevância, do mais relevante ao menos relevante, um em cada linha.
5. Programa volta ao passo 2, possibilitando que o usuário faça uma nova busca (não limpar a busca anterior até que o usuário faça uma nova).

3 Entrega

O trabalho deverá ser entregue em duas etapas. A primeira é o envio dos arquivos de código fonte para o endereço `max.gontijo.oliveira@gmail.com` com o seguinte assunto:

`[CMP1099] Trabalho 2 - Tabela Hash`

A segunda é a apresentação do trabalho em sala de aula diretamente para o professor.

3.1 Código fonte

Os arquivos de código fonte deverão ser compactados em um único arquivo (ZIP, RAR, 7Z, etc.) e enviados até o dia **05/10/2015** às **18:45**.

3.2 Apresentação

Todos os alunos deverão apresentar o trabalho ao professor em sala de aula. As datas de apresentações serão definidas em sala de aula, de modo que o início se dará em **05/10/2015**.

A apresentação será realizada individualmente e irá considerar a capacidade do aluno de explicar todo o código desenvolvido ou partes específicas determinadas pelo professor. Além disso, poderão ser solicitadas alterações no dia da apresentação.

Não será admitido desconhecimento de partes importantes do trabalho por parte de um dos membros do grupo.

4 Avaliação

A avaliação levará em consideração os seguintes aspectos:

- Entrega no prazo
- Execução do programa
  - Programa compila e executa
  - Atendimento aos requisitos (execução e funcionamento correto)
- Entendimento do problema
- Conhecimento sobre a estrutura de tabela hash
- Conhecimento sobre o código implementado
- Participação efetiva no desenvolvimento do código
- Iniciativa para responder perguntas e/ou explicar partes do código sem que o professor tenha que se dirigir ao aluno, não deixando tudo para apenas um membro do grupo explicar sozinho
- Atendimento aos requisitos bônus

A nota de cada aluno i será calculada de acordo com a seguinte fórmula:

$$\sqrt[n+1]{nota_i \times \left(\prod_{j=1}^{n} nota_j \right)}$$

, onde n é o número de alunos do grupo (2 ou 3), $nota_i$ é a nota individual do aluno i e $nota_j$ é a nota individual de cada aluno j do grupo do aluno i (incluindo o próprio aluno i). Essa nota individual já inclui todos os aspectos de avaliação já citados.