Universidade Federal de Minas Gerais
Programação e Desenvolvimento de Software II
Turmas TW e TM1 de 2023.1

Trabalho Prático: Máquina de Busca

Tempo esperado de desenvolvimento: 8 semanas
Data da entrega: 22/06 (definitiva - sem atraso)

---

1. Introdução

Documento é o termo genérico que designa objetos portadores de dados. Para que haja uma maior eficiência no processo de recuperação desses dados, eles precisam estar armazenados de uma forma que a informação contida neles possa ser recuperada. Uma *Máquina de Busca* pode ser vista como um programa que recupera informações em uma base de dados. A máquina de busca recebe como entrada uma expressão de busca (a.k.a. *query* ou *consulta*), processa a mesma, e dá como saída os documentos que, de acordo com a metodologia adotada, são os mais relevantes para a consulta fornecida.

A implementação de uma máquina de busca, consiste na implementação de três subsistemas: Coleta, Indexação, e Recuperação. A *coleta* consiste em reunir todos os documentos de interesse em uma única base de dados. A *indexação* consiste em processar estes documentos de forma que eles possam ser recuperados posteriormente. Já a *recuperação* consiste em processar consultas de forma a retornar os documentos mais relevantes para aquela consulta.

Para este trabalho, assume-se que a fase de coleta já foi realizada e que todos os documentos de interesse estão reunidos em uma pasta chamada ./`documentos/`. Sendo assim, neste trabalho, os alunos devem implementar apenas os subsistemas de Indexação e Recuperação.

2. Subsistema de indexação

Esta etapa consiste em construir e manter um *Índice Invertido*. Um índice invertido armazena os dados mapeando cada palavra às suas respectivas ocorrências nos documentos que compõem uma base de dados. Cada entrada de um índice invertido é uma determinada palavra e o valor associado a essa entrada é o conjunto de documentos em que ela ocorre. Para simplificar, assumiremos que a base de dados é imutável. Sendo assim, você precisará construir o índice invertido uma única vez. Por exemplo: considerando os documentos abaixo, o índice invertido correspondente seria o apresentado na Tabela 1.

Documento "d1.txt"
Quem casa quer casa. Porém ninguém casa. Ninguém quer casa também. Quer apartamento.

Documento "d2.txt"
Ninguém em casa. Todos saíram. Todos. Quer entrar? Quem? Quem?

Documento "d3.txt"
Quem está no apartamento? Ninguém, todos saíram.

Antes de ser inserida no índice invertido, cada palavra lida deve ser *normalizada*. A normalização consiste em simplificar a palavra de forma que diferentes formas da mesma palavra sejam padronizadas. Para simplificar, você pode assumir que os arquivos da entrada contém apenas caracteres ASCII[1], ou seja, não contém caracteres acentuados. Sendo assim, neste trabalho a normalização consiste apenas em (i) remover da palavra todos os caracteres que não sejam letras a-z e A-Z, e (ii) converter os caracteres restantes para letras minúsculas. Por exemplo, todas as palavras a seguir serão normalizadas para "**guardachuva**": "Guarda-Chuva", "guarda-chuva", "Guarda-Chuva!", "Guarda_Chuva," "guarda2chuva", e ".g.u.a.r.d.a-c.h.u.v.a.". **DICA:** Em C++, você pode implementar um índice invertido como este com um map<string, map<string, int>>. Encapsule o índice dentro de uma classe, e manipule ele apenas através dos métodos públicos da classe.

[1] https://pt.wikipedia.org/wiki/ASCII

Tabela 1: Índice invertido resultante.

Chave	Valor
apartamento	(d1.txt, 1), (d3.txt, 1)
casa	(d1.txt, 2), (d2.txt, 1)
em	(d2.txt, 1)
entrar	(d2.txt, 1)
esta	(d3.txt, 1)
ninguem	(d1.txt, 2), (d2.txt, 1), (d3.txt, 1)
todos	(d2.txt, 2) (d3.txt, 1)
no	(d3.txt, 1)
porem	(d1.txt, 1)
quem	(d1.txt, 1), (d2.txt, 2), (d3.txt, 1)
quer	(d1.txt, 2), (d2.txt, 1)
sairam	(d2.txt, 1), (d3.txt, 1)
tambem	(d1.txt, 1)

3. Subsistema de recuperação

O subsistema de recuperação é responsável por retornar os documentos que são mais relevantes para uma determinada consulta. Além disso, estes documentos são ordenados de forma que os mais relevantes são apresentados antes dos menos relevantes.

Para simplificar, as consultas serão consideradas como um conjunto de palavras. Ou seja, a ordem em que as palavras aparecem na consulta não é considerada. Cada consulta consiste em uma linha com várias palavras (lidas da entrada padrão). Após cada consulta, serão retornados somente os documentos que contém todas as palavras da consulta. Os nomes dos documentos retornados devem ser exibidos um por linha (na saída padrão). A ordem dos documentos que contém um número maior de *hits* devem ser exibidos primeiro que aqueles que têm um número menor *hits*. Em caso de empate, exibe-se primeiro o nome de arquivo que é lexicograficamente menor.

4. Sobre o desenvolvimento e a avaliação

GitHub (7 pts.): O projeto deve ser desenvolvido num projeto privado do GitHub. Os membros do projeto devem ser apenas os integrantes dos grupos e a(o) estagiária(o) docente que fará a correção. Todos os integrantes do grupo têm que participar da implementação. Isto será avaliado através dos commits que cada membro realizar.

Corretude (7 pts.): O trabalho será executado e testado em uma base de dados diferente daquela disponibilizada, onde será realizada uma série de consultas para validar se o sistema retornou a resposta certa.

Documentação (6 pts.): A documentação não deverá exceder o limite de 10 páginas, sendo submetida no formato PDF juntamente com o código fonte via GitHub. Cada grupo deve criar seu próprio repositório privado no site e enviar o link junto com a submissão do pdf contendo a documentação (via moodle). A documentação deverá contemplar pelo menos os 3 tópicos: Introdução, Implementação, e Conclusão. Na Introdução e Conclusão, o aluno deve descrever sua experiência com o trabalho, já a *Implementação* deve conter uma explicação clara e objetiva de como o problema foi resolvido, justificando os algoritmos e as classes utilizadas. Para auxiliar nessa atividade você pode utilizar, por exemplo, pseudocódigos e diagramas de classes. Não é necessário incluir trechos de código da sua implementação.

4. Considerações finais

Este trabalho pode ser feito em grupos de **duas ou três** pessoas. Grupos de uma ou quatro pessoas podem eventualmente ser aceitos quando devidamente justificados e aprovados explicitamente, por e-mail, pelo professor. Cada grupo deve submeter dois arquivos na atividade correspondente ao TP no Moodle. Um PDF contendo a documentação e um ZIP contendo o código do sistema. Este último deve conter um arquivo *Makefile* que compila todo o código e gera um único executável.

Dúvidas gerais sobre o enunciado do trabalho podem ser esclarecidas em sala de aula ou na monitoria, mas preferencialmente devem ser postadas no Moodle, para que todos os alunos fiquem cientes do esclarecimento.

O trabalho deve ser entregue impreterivelmente até o prazo indicado no Moodle. Não serão aceitas entregas após este prazo, exceto aquelas previstas pelas normas vigentes da UFMG e devidamente comprovadas. Sendo assim, o aluno que não entregar no prazo será avaliado em zero pontos. Recomendamos fortemente que o trabalho seja entregue alguns dias antes do prazo, para evitar eventuais imprevistos que certamente podem acontecer.

Desejamos a todos um ótimo trabalho.