INTRODUÇÃO

Nesta atividade, trabalhamos para desenvolver um projeto utilizando o GitHub como plataforma de colaboração e versionamento. O objetivo foi criar um sistema de busca chamado "máquina de busca" usando um índice invertido e uma classe de recuperação.

Para isso, implementamos classes e métodos para carregar e processar os documentos, criar e manter o índice invertido, e realizar consultas de busca. Tivemos que lidar com desafios, como normalizar termos, remover caracteres especiais e calcular frequências nos documentos.

Além disso, o uso do GitHub facilitou nossa colaboração, permitindo compartilhar e integrar alterações de forma eficiente. Também nos ajudou a revisar o código e acompanhar o progresso do projeto.

IMPLEMENTAÇÃO

ESTRUTURA:

O projeto está organizado da seguinte forma:

```
meu_projeto
 |- Makefile
 |- src
 |   |- index.cpp
 |   |- search.cpp
 |   |- main.cpp
 |
 |- include
 |   |- index.h
 |   |- search.h
 |
 |- obj (gerado automaticamente)
 |
 |- bin (gerado automaticamente)
 |
 |- documentos
    |- ...txt
```

BIBLIOTECAS INCLUSAS E SUAS FUNCIONALIDADES:

iostream: Entrada e saída de dados.

vector: Contêiner dinâmico para armazenar elementos.

string: Manipulação e armazenamento de sequências de caracteres.

fstream: Operações de entrada e saída em arquivos.

sstream: Manipulação de strings como fluxos de entrada e saída.

unordered_map: Contêiner associativo de chave-valor implementado usando tabela hash.

unordered_set: Contêiner para armazenar elementos únicos, sem ordem garantida.

algorithm: Algoritmos genéricos para operações em contêineres.

dirent.h: Operações relacionadas a diretórios e arquivos.

cctype: Manipulação de caracteres, como testar tipos e conversão de maiúsculas/minúsculas.

EXPLICANDO O CÓDIGO E SUAS CLASSES

search.h:

Este arquivo contém a declaração da classe SearchEngine:

- class SearchEngine: Essa classe representa o mecanismo de busca. Ela possui um objeto InvertedIndex chamado index, que é responsável por armazenar o índice invertido e realizar as operações de busca.
- SearchEngine(const string& folderPath): O construtor da classe SearchEngine recebe um caminho de pasta como argumento. Ele instancia um objeto InvertedIndex chamado index, passando o caminho da pasta para o construtor do InvertedIndex. Isso permite carregar automaticamente os documentos da pasta especificada ao criar um objeto SearchEngine.
- void run(): Esta função é responsável por iniciar o mecanismo de busca e interagir com o usuário. Ela solicita uma consulta ao usuário, realiza a busca usando o objeto index e exibe os resultados.

search.cpp:

Este arquivo contém a implementação das funções da classe SearchEngine.

- void SearchEngine::run(): Essa função implementa a lógica para interagir com o usuário, solicitar consultas e exibir os resultados. Ela usa o objeto index para realizar a busca e chama as funções search e print_titles do objeto index.

main.cpp:

Este é o ponto de entrada principal do programa:

- int main(): A função main é o ponto de entrada do programa. Aqui, a pasta documentos é especificada como o caminho dos documentos para serem usados no mecanismo de busca.
- SearchEngine engine(folderPath): Um objeto SearchEngine chamado engine é criado, passando o caminho da pasta documentos para o construtor. Isso permite que o mecanismo de busca seja inicializado com a pasta correta.
- engine.run(): A função run do objeto engine é chamada para iniciar o mecanismo de busca e permitir que o usuário faça consultas.
- return 0: O programa retorna 0 para indicar que foi executado com sucesso.

main()**:** A função principal do programa. Ela cria uma instância do InvertedIndex, carrega os documentos do diretório especificado, e entra em um loop onde solicita ao usuário uma consulta de busca. A função realiza a busca no índice invertido e imprime os títulos dos documentos encontrados. O loop continua até que o usuário digite "exit" para sair do programa.

index.h:

Este arquivo contém a declaração da classe InvertedIndex e da classe Document:

- class Document: Essa classe representa um documento do mecanismo de busca. Ela possui três membros de dados: id, title e content, que representam o identificador, o título e o conteúdo do documento, respectivamente.
- class InvertedIndex: Essa classe representa o índice invertido. Aqui estão os principais elementos:
- Membros de dados: index, documents e next_id. O index é um mapeamento de termos para uma lista de documentos e suas frequências. O documents é um mapeamento de identificadores de documentos para objetos Document. O next_id é um inteiro que representa o próximo identificador disponível para um novo documento.
- Funções membros: normalize, load_documents, get_file_names, add_document, tokenize, search e print_titles. Essas funções são responsáveis por realizar operações relacionadas ao índice invertido, como normalizar termos, carregar documentos de uma pasta, obter nomes de arquivos, adicionar documentos ao índice, tokenizar uma string em palavras, realizar busca e imprimir os títulos dos documentos relevantes.

(Serão explicadas detalhadamente em seguida em index.cpp)

index.cpp

1. **Classe Document:**

A classe Document representa um documento a ser indexado no índice invertido. Cada documento possui um ID, título e conteúdo.

Membros e funções:

- **id**: Variável de membro para armazenar o ID do documento.
- **title**: Variável de membro para armazenar o título do documento.
- **content**: Variável de membro para armazenar o conteúdo do documento.
- **Construtor**: Um construtor é utilizado para inicializar os membros do documento com os valores fornecidos.

2. **Classe InvertedIndex:**

A classe InvertedIndex representa um índice invertido, uma estrutura de dados usada para indexar documentos em um sistema de busca textual. Ele permite pesquisar palavras-chave e recuperar os documentos relevantes que as contêm.

Membros e funções:

- **normalize(const string& term)**: Essa função é responsável por normalizar um termo, convertendo-o para letras minúsculas, removendo caracteres não alfanuméricos e números, mas mantendo os espaços. Ela é usada para processar os termos antes de adicioná-los ao índice invertido ou pesquisar por eles.
- **load_documents(const string& folderPath):** Essa função é usada para carregar documentos de um diretório. Ela recebe o caminho do diretório como entrada, obtém os nomes dos arquivos contidos nesse diretório e lê o conteúdo de cada arquivo. Em seguida, ela adiciona os documentos ao índice invertido usando a função add_document.
- **get_file_names(const string& folderPath):** Essa função é usada para obter os nomes dos arquivos contidos em um diretório. Ela recebe o caminho do diretório como entrada e retorna um vetor com os nomes dos arquivos encontrados.
- **add_document(const Document& doc):** Essa função é usada para adicionar um documento ao índice invertido. Ela recebe um objeto Document como entrada, armazena o documento em uma estrutura de dados interna e atualiza o índice invertido com os termos presentes no documento.
- **tokenize(const string& str)**: Essa função é responsável por tokenizar uma string em palavras separadas. Ela recebe uma string como entrada e retorna um vetor de strings contendo os tokens individuais.
- **search(const string& query):** Essa função realiza uma busca no índice invertido com base em uma consulta. Ela recebe uma string de consulta como entrada, normaliza a consulta, divide-a em termos e encontra os documentos relevantes que contêm todos os termos. Ela retorna uma lista de pares (ID do documento, frequência de correspondência) ordenada por relevância.
- **print_titles(const vector<pair<int, int>>& results, const string& query)**: Essa função imprime os títulos dos documentos encontrados em uma busca. Ela recebe os resultados da busca (pares de ID do documento e frequência de correspondência) e a consulta original como entrada e exibe os títulos dos documentos correspondentes.

RESUMINDO

O search.h e o search.cpp implementam a classe SearchEngine responsável pela interação com o usuário e a execução do mecanismo de busca. O main.cpp é o ponto de entrada do programa, onde um objeto SearchEngine é criado e a função run é chamada. O index.h contém as classes InvertedIndex e Document, que representam o índice invertido e os documentos, respectivamente, enquanto index.cpp é responsável pela implementação dessas classes.

CONCLUSÃO

Participar deste projeto no GitHub foi uma experiência positiva de trabalho em equipe e colaboração. Aprendemos a importância da organização e comunicação para o sucesso do projeto e melhoramos nossas habilidades de programação ao decorrer do projeto.

Durante a implementação do sistema de busca, percebemos como é crucial usar algoritmos eficientes e estruturas de dados adequadas para obter um desempenho satisfatório. A escolha correta dos algoritmos de normalização de termos, remoção de caracteres especiais e cálculo de frequência de termos foi fundamental para obter resultados precisos nas consultas.

O uso do GitHub como plataforma de versionamento e colaboração facilitou a integração do trabalho de cada membro do grupo. Foi mais fácil revisar o código, compartilhar ideias e resolver conflitos. A ferramenta também permitiu acompanhar o progresso do projeto e manter um histórico completo das alterações realizadas.

Resumindo, essa experiência não apenas resultou no desenvolvimento de um sistema funcional, mas também proporcionou um aprendizado valioso em programação. Através desse projeto, foi possível aplicar nossos conhecimentos, adquirir novas habilidades e trabalhar de forma eficaz em equipe.

INTEGRANTES

João Henrique Voss Teixeira – 2022056080

Bernardo Prosdocimi Lamounier Soares – 2021032250