INTRODUÇÃO

Nesta atividade, trabalhamos para desenvolver um projeto utilizando o GitHub como plataforma de colaboração e versionamento. O objetivo era criar um sistema de busca chamado "máquina de busca" usando um índice invertido. Durante o projeto, melhorei minhas habilidades de programação e trabalhei com outros membros do grupo.

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

/*

*** Bibliotecas inclusas.**

***/**

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <sstream>
#include <unordered_map>
#include <unordered_set>
#include <algorithm>
#include <cassert>
#include <dirent.h>
#include <cctype>
#include <locale>
using namespace std
```

~~index.h~~

```
#ifndef INDEX_H
#define INDEX_H
class Document {
public:
   int id;
   string title;
```

```
    string content;
    Document() = default;
    Document(int id, const string& title, const string& content) : id(id), title(title),
content(content) {}
};
class InvertedIndex {
public:
    InvertedIndex() : next_id(1) {}
    InvertedIndex(const string& folderPath) {
        load_documents(folderPath);
    }
/**
     * Pesquisa por documentos que contenham a consulta especificada.
     * @param query A consulta de pesquisa.
     * @return Um vetor de pares representando os resultados da pesquisa. Cada par contém
o ID do documento e a frequência
     *         dos termos da consulta nesse documento.
     */
    vector<pair<int, int>> search(const string& query) const;
    /**
     * Imprime os títulos dos documentos nos resultados da pesquisa.
     * @param results Os resultados da pesquisa.
     * @param query A consulta de pesquisa.
     */
    void print_titles(const vector<pair<int, int>>& results, const string& query) const;
private:
    unordered_map<string, unordered_map<int, int>> index;
    unordered_map<int, Document> documents;
    int next_id;
    /**
     * Remove os acentos do termo fornecido.
     * @param term O termo para remover os acentos.
     * @return O termo sem acentos.
```

```
    */
    string removeAccents(const string& term) const;
    /**
     * Normaliza o termo fornecido, convertendo-o para minúsculas e removendo os acentos.
     * @param term O termo para normalizar.
     * @return O termo normalizado.
     */
    string normalize(const string& term) const;
    /**
     * Carrega os documentos da pasta especificada.
     * @param folderPath O caminho para a pasta contendo os documentos.
     */
    void load_documents(const string& folderPath);
    /**
     * Obtém os nomes dos arquivos na pasta especificada.
     * @param folderPath O caminho para a pasta.
     * @return Um vetor de nomes de arquivos.
     */
    vector<string> get_file_names(const string& folderPath);
    /**
     * Adiciona um documento ao índice.
     * @param doc O documento a ser adicionado.
     */
    void add_document(const Document& doc);
    /**
     * Tokeniza a string fornecida em um vetor de termos individuais.
     * @param str A string a ser tokenizada.
     * @return Um vetor de termos.
     */
    vector<string> tokenize(const string& str) const;
};
#endif  // INDEX_H
```

~~search.h~~

```
#include "index.h"
class SearchEngine {
public:
    InvertedIndex index;
    /**
     * Construtor da classe SearchEngine.
     * @param folderPath O caminho para a pasta contendo os documentos.
     */
    SearchEngine(const string& folderPath) : index(folderPath) {}
    /**
     * Executa o mecanismo de busca interativo.
     */
    void run();
};
#endif  // SEARCH_H
```

~~main.cpp~~

```
/**
 * @file main.cpp
 * @brief Programa principal para executar o mecanismo de busca de documentos.
 */
#include "search.h"
int main() {
    // Caminho para a pasta contendo os documentos
    string folderPath = "./documentos";
    // Cria uma instância do mecanismo de busca
    SearchEngine engine(folderPath);
    // Executa o mecanismo de busca
    engine.run();
    return 0;
}
// Comando para compilar e executar o programa:
// g++ main.cpp index.cpp search.cpp -o meu_programa && ./meu_programa
```

EXPLICANDO O CÓDIGO E SUAS CLASSES

Document (classe): Essa classe representa um documento e possui três membros de dados: id (identificador do documento), title (título do documento) e content (conteúdo do documento).

InvertedIndex (classe): Essa classe implementa um índice invertido para pesquisa de documentos. Ela possui os seguintes membros de dados:

Index: Um unordered_map que mapeia termos normalizados para um segundo unordered_map que mapeia IDs de documentos para a frequência de ocorrência desse termo no documento.

documents: Um unordered_map que mapeia IDs de documentos para os próprios documentos.

next_id: Um inteiro que representa o próximo ID disponível para atribuir a um documento.

normalize(const string& term): Essa função recebe um termo e retorna sua versão normalizada. A normalização envolve a remoção de caracteres especiais, conversão para letras minúsculas e remoção de números.

remove_accents(const string& term): Essa função recebe um termo e transforma o caractere acentuado na sua versão padrão, seguindo uma tabela de conversão.

load_documents(const string& folderPath): Essa função carrega os documentos de um diretório especificado pelo folderPath. Para cada arquivo no diretório, ele lê o título e o conteúdo do arquivo e adiciona o documento ao índice invertido por meio da função add_document.

get_file_names(const string& folderPath): Essa função retorna os nomes dos arquivos presentes em um diretório especificado pelo folderPath.

add_document(const Document& doc): Essa função adiciona um documento ao índice invertido. Ela adiciona o documento ao mapeamento documents e atualiza o índice invertido index com as palavras e suas frequências de ocorrência no documento.

tokenize(const string& str): Essa função recebe uma string e a divide em tokens (palavras) com base nos espaços como delimitadores. Ela retorna um vetor contendo os tokens resultantes.

search(const string& query): Essa função realiza uma busca no índice invertido com base em uma consulta (query). Ela percorre os termos da consulta, encontra os documentos relevantes e calcula uma pontuação para cada um com base nas frequências dos termos nos documentos. Os resultados são retornados em um vetor de pares (ID do documento, pontuação), ordenados pela pontuação.

print_titles(const vector<pair<int, int>>& results, const string& query): Essa função imprime os títulos dos documentos encontrados na busca. Se não houver resultados, uma mensagem informando que nenhum documento foi encontrado é exibida.

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.

CONCLUSÃO

Participar deste projeto no GitHub foi uma experiência positiva de trabalho em equipe e colaboração. Aprendi a importância da organização e comunicação para o sucesso do projeto. Melhorei minhas habilidades de programação, especialmente na implementação de algoritmos e manipulação de dados.

Durante a implementação do sistema de busca, percebi 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 me proporcionou aprendizado valioso em programação. Através desse projeto, pude aplicar meus conhecimentos, adquirir novas habilidades e trabalhar de forma eficaz em equipe utilizando ferramentas de colaboração como o GitHub.

João Henrique Voss Teixeira - 2022056080