Ranqueamento de Documentos

Michel Pires, Centro Federal de Educação Tecnológica de Minas Gerais 9 de outubro de 2024

Observações:

1. O trabalho deve ser realizado individualmente, e qualquer indício de cópia será desconsiderado na avaliação.
2. O programa deve ser compatível com sistema operacional Linux, utilizando *gcc/g++* versão 11 ou superior. Cabe ao aluno validar possíveis erros de compilação e execução. A compilação deve seguir os padrões já estabelecidos na disciplina.
3. Linguagens aceitas: *C* / *C++*.

Neste trabalho, o aluno será desafiado a implementar um sistema de ranqueamento de documentos com base no algoritmo *TF/IDF* (do inglês, *Term Frequency-Inverse Document Frequency*). O foco é exercitar e revisar conceitos fundamentais abordados na disciplina de *Algoritmos e Estrutura de Dados I*, como análise assintótica, listas, pilhas, filas, métodos de ordenação e tabelas hash. Ao mesmo tempo, o trabalho prepara os alunos para o estudo de estruturas mais avançadas, como árvores e grafos, que serão introduzidas ao longo do curso.

1 Descrição do Problema

Neste trabalho, cada aluno receberá como entrada:

1. Um conjunto de arquivos de texto contendo documentos aleatórios.
2. Cada aluno deve preparar uma lista de frases de pesquisa, onde cada frase deve ser comparada aos documentos fornecidos para determinar a relevância de cada arquivo em relação à frase.

O objetivo é ranquear os documentos de acordo com a relevância de cada um para uma frase de pesquisa específica, utilizando a métrica *TF/IDF* para calcular tal relevância.

2 Estrutura do Trabalho

- **Leitura dos Documentos**: Cada documento será lido e processado, utilizando uma *lista* ou uma *fila* para armazenar os termos presentes no documento. Os termos de cada documento devem ser normalizados (remover pontuação, converter para minúsculas, remover *stop words*, etc.).

- **Cálculo do TF/IDF**: *TF* (do inglês, *Term Frequency*): Frequência de cada termo em um documento. *IDF* (do inglês, *Inverse Document Frequency*): Mede a importância de um termo considerando todos os documentos do conjunto. O cálculo do $\frac{TF}{IDF}$ pode ser facilitado por uma *tabela hash*, onde cada termo será mapeado para a frequência com que aparece em cada documento.
- **Ranqueamento de Documentos**: Para cada frase de pesquisa, calcular a relevância de cada documento, somando os valores de *TF/IDF* dos termos que aparecem na frase. Ordenar os documentos de acordo com os valores de relevância utilizando um método de ordenação, como o *QuickSort* ou *MergeSort*, garantindo um custo computacional eficiente.

2.1 Desafios Computacionais

A solução implementada utilizando as estruturas já conhecidas (*listas*, *pilhas*, *filas*, *tabelas hash* e *métodos de ordenação*) é suficiente para resolver o problema. Contudo, essas estruturas apresentam maior custo computacional, especialmente no que diz respeito ao acesso, busca e ordenação de dados. Por exemplo, o uso de *listas* para armazenar termos implica em operações de busca e inserção com custo $\mathcal{O}(n)$. A ordenação dos documentos com métodos tradicionais, como o *QuickSort*, possui complexidade $\mathcal{O}(n \log n)$, que é adequada, mas pode ser aprimorada com o uso de árvores.

2.2 Preparação para Estruturas Avançadas

Embora a solução proposta com as estruturas já estudadas seja funcional e suficiente para atender a todos os requisitos do trabalho, o aluno será incentivado a refletir sobre como o uso de *estruturas em árvore*, como *Árvores Binárias de Busca* (*BST*) ou *Árvores AVL*, poderia melhorar a eficiência. Por exemplo, ao invés de usar uma lista para armazenar os termos dos documentos, uma árvore balanceada poderia reduzir o tempo de busca de $\mathcal{O}(n)$ para $\mathcal{O}(\log n)$. Além disso, a solução atual prepara o aluno para compreender a aplicação de *grafos*, ao lidar com a relação entre documentos e termos de pesquisa como um grafo, onde os vértices representam os documentos e os termos conectam os documentos com base na relevância.

3 Entrega e Avaliação

O trabalho será avaliado de acordo com os seguintes critérios:

- **Correção**: A implementação gera os resultados corretos, ordenando os documentos conforme a relevância de cada frase de pesquisa.
- **Eficiência**: Embora a solução utilize estruturas básicas, a análise do custo computacional de cada etapa deve ser discutida.

- **Organização**: O código deve ser bem estruturado, com o uso adequado de funções e modularidade.
- **Análise Crítica**: O aluno deve apresentar uma breve discussão sobre como o uso de estruturas mais avançadas poderia melhorar o desempenho da solução.

3.1 Documentação

A documentação deve conter as seguintes seções:

- Uma discussão sobre as *estruturas de dados* escolhidas, comparando-as com alternativas possíveis e justificando as escolhas feitas.
- Uma descrição detalhada das operações implementadas e suas complexidades.
- Instruções de compilação e execução, com exemplos de entrada e saída.
- Discussão sobre o desempenho do sistema, utilizando métricas como tempo de execução e uso de memória.

4 Entrega e Pontuação

Data de Entrega: 31/10/2024 **Valor**: 10 pontos