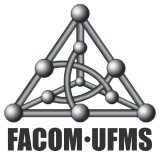

Estruturas de Dados

Prof. Bruno M. Nogueira
Faculdade de Computação - UFMS

Projeto Prático: Implementação e Comparação de Estruturas de Dados em um Sistema de Indexação de Textos

1 Descrição Geral

O objetivo deste projeto é construir um sistema que permita indexar textos e buscar palavras de forma eficiente. Os alunos deverão desenvolver um programa em Java que suporte todas as etapas do processo, desde a inserção de documentos até a busca por palavras indexadas.

O corpus utilizado será composto por 30 artigos do arXiv. Esses artigos serão convertidos para texto plano (`.txt`) e processados pelo programa desenvolvido.

2 Implementação das Estruturas de Dados

O sistema será baseado em duas principais estruturas de dados: Tabela Hash e Trie (Árvore Digital). Além disso, a compressão dos textos será feita utilizando o algoritmo de Huffman.

2.1 Tabela Hash com Duas Funções de Hash

A tabela hash será usada para armazenar os textos compactados, associando cada texto a uma chave gerada por uma função de hash. O sistema deverá implementar duas funções de hash distintas para comparação de desempenho.

Função de Hash por Divisão:

```java
public int hashDivisao(String texto, int M) {
    int soma = 0;
    for (char c : texto.toCharArray()) {
        soma += (int) c;
    }
    return soma % M;
}
```

Função de Hash DJB2:

```java
public int hashDJB2(String texto) {
    long hash = 5381;
    for (char c : texto.toCharArray()) {
        hash = ((hash << 5) + hash) + c; // hash * 33 + c
    }
    return (int) (hash % Integer.MAX_VALUE);
}
```

2.2 Fluxo do Sistema

O processo completo de inserção e recuperação de textos pode ser visto na Figura 1. Após inserir um documento, ele é compactado e armazenado na tabela hash. As palavras do documento são indexadas na Trie para facilitar buscas futuras. O programa permitirá que o usuário busque por uma palavra e obtenha os documentos onde ela aparece.

3 Fluxo do Sistema

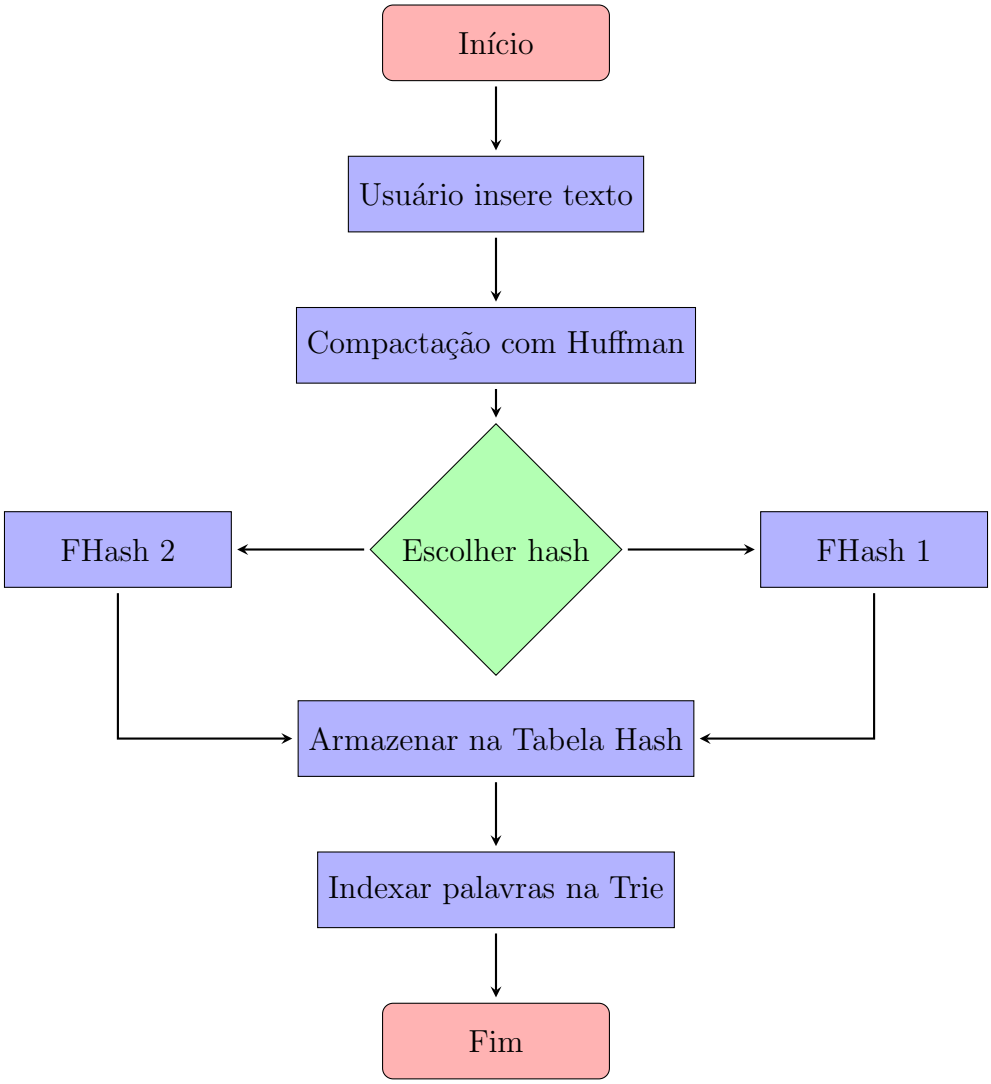

Figure 1: Fluxo do sistema de indexação e compressão de textos.

3.1 Programa Esperado

Espera-se que a entrega contenha um arquivo que permita o teste manual das implementações. O programa deve permitir:

1. Inserir um conjunto de documentos: O usuário fornece o caminho para uma pasta, contendo os documentos a serem indexados.
2. Escolher a função de hashing usada.

3. Buscar por uma palavra: O sistema exibe os documentos que contêm a palavra.

Exemplo de Entradas e saídas:

```
> Inserir documentos: /home/bruno/documentos
Documentos inseridos com sucesso!

> Qual a função de hashing (divisao/djb2): divisao
Documentos indexados com sucesso!

> Buscar palavra: aprendizado
A palavra "aprendizado" foi encontrada nos seguintes documentos:
- artigo1.txt
- artigo5.txt

> Buscar palavra: computação
A palavra "computação" foi encontrada nos seguintes documentos:
- artigo3.txt
- artigo7.txt
- artigo9.txt
```

4 Medição de Desempenho

Considerando toda a base de documentos, os seguintes aspectos deverão ser medidos:

- Tempo de indexação: Tempo necessário para inserir e indexar os documentos.
- Tempo de busca: Tempo necessário para encontrar uma palavra na Trie.
- Consumo de memória: Memória utilizada pelas estruturas.

Para os experimentos de busca, o grupo deverá definir pelo menos 100 expressões, com pelo menos 90% delas contidas nos documentos, para buscar. Os testes devem ser realizados automaticamente, com um arquivo próprio, diferente do teste manual.

5 Relatório Final

O relatório deve seguir o modelo da SBC e incluir:

- Introdução e objetivos do projeto.
- Descrição das estruturas de dados e corpus utilizado.
- Metodologia de compressão e indexação.
- Resultados e análise comparativa.
- Conclusões e sugestões de melhorias.

6 Entrega e Avaliação

- Prazo de Entrega: 24/11.
- Itens para Entrega: Relatório, código-fonte, dados utilizados na avaliação, expressões de busca utilizados no teste e vídeo de apresentação.
- Critérios de Avaliação: Correção das implementações, clareza na explicação e análise comparativa.

7 Dicas sobre a base de dados a ser utilizada

Recomenda-se que os alunos priorizem a escolha de arquivos em formato de coluna única, pois PDFs com múltiplas colunas podem apresentar desafios durante a extração de texto, resultando em trechos fora de ordem ou sobrepostos. Isso garantirá uma conversão mais precisa e facilitará o processo de limpeza e indexação, evitando a necessidade de manipulação manual intensa ou uso de ferramentas específicas para preservar o layout.

7.1 Conversão de PDFs para Texto Plano

Os textos escolhidos para o corpus podem estar em formato PDF, sendo necessário convertê-los para texto plano (`.txt`) para facilitar a indexação e manipulação pelos algoritmos desenvolvidos. A conversão precisa preservar a estrutura essencial do texto para garantir que os dados estejam corretos para busca e compressão.

7.1.1 Ferramentas Recomendadas

As seguintes ferramentas e bibliotecas são sugeridas para realizar a conversão:

- **Python:** Uso da biblioteca `PyMuPDF` (ou `fitz`), que oferece uma API eficiente para extrair textos de PDFs.
- **Tika:** Ferramenta baseada em Java que suporta diversos formatos de arquivo, incluindo PDF.
- **PDFMiner:** Biblioteca Python voltada para extração de texto e análise de layout.

7.1.2 Exemplo de Conversão em Python

O exemplo abaixo mostra como utilizar a biblioteca `PyMuPDF` para extrair texto de um PDF:

```
import fitz  # PyMuPDF

# Abrindo o arquivo PDF
pdf_document = fitz.open("documento.pdf")

# Extraindo texto de cada página
with open("documento.txt", "w", encoding="utf-8") as txt_file:
    for page_num in range(pdf_document.page_count):
```

```
        page = pdf_document.load_page(page_num)
        text = page.get_text()
        txt_file.write(text + "\n")

pdf_document.close()
```

Recomenda-se que cada grupo faça uma verificação manual da qualidade do texto extraído, especialmente em documentos PDF com formatos complexos (e.g., fórmulas matemáticas, tabelas). Isso garante que os dados estejam prontos para serem usados nas etapas de indexação e busca.

7.1.3 Erros Comuns e Como Corrigi-los

- **Texto fora de ordem:** Pode ocorrer em PDFs com colunas. Use ferramentas como Tika, que preservam melhor o layout.
- **Caracteres estranhos:** Pode ser necessário corrigir a codificação do texto para `UTF-8`.
- **Perda de conteúdo:** Verifique se todas as páginas foram lidas corretamente e se não houve falhas na extração.