PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS - LOURDES

DIEGO HENRIQUE XAVIER DOS SANTOS, MARCOS VINÍCIUS NUNES REIS,
RAFAEL GEORGETTI GROSSI E VITOR DANIEL SILVA MELO

# TRABALHO PRÁTICO - FASE V

ALGORITMOS E ESTRUTURAS DE DADOS III

BELO HORIZONTE - MG

2025

DIEGO HENRIQUE XAVIER DOS SANTOS, MARCOS VINÍCIUS NUNES REIS,
RAFAEL GEORGETTI GROSSI E VITOR DANIEL SILVA MELO

# TRABALHO PRÁTICO - FASE V

ALGORITMOS E ESTRUTURAS DE DADOS III

Trabalho apresentado ao curso superior de Ciência da Computação da PUC-MG Lourdes para o cumprimento das exigências da disciplina Algoritmos e Estruturas de Dados III

Orientador: Walisson Ferreira de Carvalho

BELO HORIZONTE - MG
2025

# SUMÁRIO

**FORMULÁRIO..... 4**
1. Qual campo textual foi escolhido para aplicar os algoritmos de casamento de padrões? Por quê?.....4
2. Explique o funcionamento do KMP implementado.....4
3. Explique o funcionamento do Boyer–Moore implementado.....4
4. Descreva como integrou os algoritmos ao sistema.....4
5. Quais dificuldades encontrou na implementação dos dois algoritmos?..... 4
**REPOSITÓRIO GITHUB..... 5**

# FORMULÁRIO

**1. Qual campo textual foi escolhido para aplicar os algoritmos de casamento de padrões? Por quê?**

O campo textual selecionado para a aplicação dos algoritmos de casamento de padrões foi o campo username. A escolha desse campo deve-se ao fato de que ele é uma informação textual curta, porém altamente relevante para buscas no sistema. Usuários frequentemente procuram registros específicos utilizando nomes de usuário, e esse campo apresenta variações que se beneficiam de um mecanismo eficiente de busca por padrões.

**2. Explique o funcionamento do KMP implementado.**

No pré-processamento, o algoritmo calcula o vetor LPS (Longest Prefix Suffix), que indica, para cada posição do padrão, o tamanho do maior prefixo que também é sufixo. Esse vetor permite que, durante a busca, o algoritmo não precise retornar ao início do padrão após um caractere incompatível, evitando comparações redundantes. Durante a busca, o algoritmo percorre o texto e o padrão simultaneamente. Se os caracteres forem iguais, ambos os índices avançam. Caso ocorra um mismatch após algumas correspondências, o algoritmo utiliza o valor do vetor LPS para reposicionar o índice do padrão, evitando retroceder no texto. Quando o índice do padrão alcança seu tamanho total, significa que uma ocorrência completa foi encontrada, sendo registrado o índice correspondente no texto.

**3. Explique o funcionamento do Boyer–Moore implementado.**

A heurística Bad Character cria, por meio de um HashMap, uma tabela que associa cada caractere do padrão à última posição em que ele aparece. Quando ocorre um mismatch, o padrão é deslocado para alinhar essa ocorrência com a posição correspondente no padrão — ou para fora do texto caso o caractere não exista no padrão. A heurística Good Suffix calcula deslocamentos baseados nos sufixos do padrão. Quando parte final do padrão coincide com o texto, mas ocorre um mismatch anterior, o algoritmo desloca o padrão usando o maior sufixo já compatível, permitindo pular várias posições sem perder possíveis ocorrências. Na fase de busca, o padrão é comparado da direita para a esquerda, começando pelo final. Quando ocorre um mismatch, o algoritmo calcula dois possíveis deslocamentos: o da heurística de bad character e o da heurística de good suffix, aplicando o maior entre eles. Isso permite grandes saltos, especialmente quando o padrão é longo ou quando há pouca repetição de caracteres no texto.

**4. Descreva como integrou os algoritmos ao sistema.**

Os algoritmos foram encapsulados em uma classe utilitária (PatternMatching) contendo métodos estáticos. Isso permite que qualquer parte do sistema invoque a busca por meio de chamadas simples, como searchKMP ou searchBoyerMoore, passando o texto e o padrão como parâmetros.A integração foi feita de forma modular, sem alterar as estruturas principais do sistema. Os métodos retornam uma lista com todos os índices onde o padrão foi encontrado, facilitando sua utilização em componentes como telas de busca, filtros ou módulos de análise. Assim, o sistema pode escolher dinamicamente qual algoritmo utilizar, dependendo da necessidade de desempenho ou características da consulta.

**5. Quais dificuldades encontrou na implementação dos dois algoritmos?**

Para suportar textos com caracteres especiais, acentos e Unicode, foi necessário adaptar a heurística de Bad Character para utilizar HashMap<Character, Integer>, já que tabelas fixas baseadas em ASCII seriam insuficientes.

## REPOSITÓRIO GITHUB

- https://github.com/vmelooo/tp-aeds3