Projeto 2

Técnicas de Busca e Ordenação

João Pedro Santos Silqueira
Maria Fernanda Andrade Rodrigues
Roberto Ramos Ferreira

Visão Geral do Projeto

Tarefas Implementadas

- **Tarefa 1:** Busca KMP
- **Tarefa 2:** Busca com Wildcard (*)
- **Tarefa 7:** Data Mining (emails, telefones, datas)
- **Tarefa 9:** Nuvem de Palavras

Decisão Arquitetura Principal

- Modularização em Classes Separadas:
  - Cada classe tem responsabilidade única e bem definida.
  - Se um módulo der problema, não afeta os outros.
  - Se quiser adicionar algo novo, não precisa mexer no que já funciona.

Tarefa 1 - Implementação KMP

Escolhas de Estruturas de Dados

- `vector<int> lps`: array de "failure function"
  - Acesso rápido a qualquer posição específica (O(1))
  - Crescimento dinâmico conforme necessário
- `vector<int> ocorrencias`: posições encontradas
  - Mesmas vantagens anteriores
  - Fácil iteração para exibição de resultados

Como funciona o algoritmo:

1. Preparação;
2. Busca;
3. Resultado;

Complexidade

- **Tempo:** O(n + m) vs O(n×m) da busca simples
- **Espaço:** O(m) para o array LPS

Tarefa 2 - Wildcard

Decisão: Algoritmo Simples

- **Abordagem direta:** comparação caractere a caractere.
  - Método mais simples no lugar do mais otimizado
  - Para textos de tamanho normal, diferença não é significativa
  - Código mais legível

Complexidade Assumida

- **Tempo:** O(n×m) - aceitável para o contexto
- **Benefício:** código 10x mais simples que versão otimizada

Tarefa 7 - Data Mining

Regex

- Linguagem de padrões: descreve formato de textos
- O que acontece:
  - **Regex:** Define o padrão a ser buscado
  - **Iterator**: Ele vai percorrer o texto procurando o padrão
  - **Armazena:** Guarda o que combina com o padrão
  - **Organização por tipo**: Emails numa lista, telefones em outra etc

Vantagens

- **Padrões complexos**: Cada tipo de dado (email, telefone, data) tem regras específicas
- **Flexibilidade**: É possível adicionar novos tipos facilmente
- **Manutenção**: Mudanças são só no padrão, não no código

Estrutura de Dados

- `map`: Cada tipo de dado (email, telefone) fica separado e organizado em ordem alfabética
- `vector<string>`: Para cada tipo, posso ter vários resultados
- **Exemplo prático**: Se encontro 3 emails e 2 telefones, fica tudo organizado por categoria

Tarefa 9 - Nuvem de Palavras

Etapa 1: Contagem de Palavras

- `map<string, int>`: palavra → frequência
- **Algoritmo:** percorrer caractere por caractere, identificar limites de palavras

Sistema de Stop Words

- `vector<string>` com palavras comuns ("de", "da", "do", "e"...)
- Para textos menores, busca linear é suficiente

Etapa 2: Problema da Ordenação

- **Map ordena por chave** (alfabética)
- **Preciso ordenar por valor** (frequência)
- **Solução:** converter para `vector<FrequenciaPalavra>`

Tarefa 9 - Nuvem de Palavras

Critério	Quick Sort	Merge Sort
Velocidade média	Mais rápido	Consistente
Pior caso	O(n²)	O(n log n)
Estabilidade	❌ Instável	✅ Estável
Previsibilidade	Varia	Sempre igual

Por que Estabilidade Importa?

- Palavras com mesma frequência mantêm ordem alfabética original
- **Resultado mais profissional e consistente**

OBRIGADO MEU CARO AMIGO TADEU E COMPANHIA :D