



Sistema de Busca de Filmes e Cinemas

João Pedro Santos Silqueira
Maria Fernanda Andrade Rodrigues
Roberto Ramos Ferreira



Visão Geral do Projeto

Objetivo

- Sistema para busca de **filmes e cinemas**
- Implementação de filtros combinados usando operadores lógicos
- Foco em **performance** durante consultas

Funcionalidades Principais

- **Busca de Filmes:** por tipo, gênero, duração, ano
- **Busca de Cinemas:** por filmes exibidos, localização, preço
- **Combinação de filtros** com operadores "E"
- **Ordenação** dos resultados



Estruturas de Dados

Estrutura Movie

- Identificador
- Tipo do título
- Título principal
- Anos de lançamento
- Duração em minutos
- Lista de gêneros

Estrutura Cinema

- ID
- Nome do cinema
- Coordenadas geográficas
- Preço do ingresso
- IDs dos filmes em exibição



Estruturas de Dados

Vetor para armazenamento principal:

- Acesso sequencial eficiente para buscas
- Suporte nativo à ordenação

Unordered_map para índice:

- **$O(1)$** para busca de filmes por ID
- Essencial para busca em cinemas
- Trade-off: usa mais memória, mas ganha muito em velocidade



Menu

1. Buscar filmes

2. Buscar cinemas

3. Mostrar tipos de títulos

- Busca linear
- Set

4. Mostrar gêneros

- Busca linear
- Set

0. Sair



Algoritmos de Busca

Busca Linear com Filtros

- **Múltiplos critérios** simultâneos
- Dados não ordenados
- **Complexidade:** $O(n)$ - inevitável para este tipo de busca
- Otimização: **early return** em `movieMatchesCriteria`
- **Função** `movieMatchesCriteria`



Algoritmos de Ordenação

- **Complexidade:** $O(n \log n)$ - otimizada
- **Implementação:** Introsort (híbrido)
- **Customização:** Lambda functions para critérios específicos



Decisões de Projeto Importantes

1. Carregamento vs. Consulta

"É preferível um tempo considerável ao carregar, que ao consultar"

Nossa estratégia:

- Criar **índice hash** durante carregamento
- **Trade-off**: Mais tempo/memória no início, consultas rápidas

2. Tratamento de Dados Faltantes

- Valores padrão para campos vazios
- **Robustez** contra dados inconsistentes



Decisões de Projeto Importantes

3. Busca de Cinemas

- **Decisão importante:** Usar o índice hash para busca **$O(1)$** de filmes

4. Cálculo de Distância

- **Distância Euclidiana** simples



Análise de Performance

Medição de Tempo

Complexidades Implementadas

- **Carregamento:** $O(n)$ para filmes + $O(n)$ para índice
- **Busca de filmes:** $O(n)$
- **Busca de cinemas:** $O(c \times m)$ onde c = cinemas, m = filmes por cinema
- **Ordenação:** $O(n \log n)$



Demonstração Prática

Exemplo 1: Busca de Filmes

Cenário: Filmes de comédia lançados entre 2000-2010

Entrada:

- Gêneros: Comedy
- Ano mínimo: 2000
- Ano máximo: 2010

Processo:

1. Busca linear em todos os filmes
2. Aplica filtros sequencialmente
3. Ordena resultados por título



Demonstração Prática

Exemplo 2: Busca de Cinemas

Cenário: Cinemas com documentários, até R\$ 15,00, distância máx 1000

Entrada:

- Gêneros: Documentary
- Preço máximo: 15.00
- Centro: (20019, 510301)
- Distância máxima: 1000

Processo:

1. Para cada cinema: verifica distância e preço
2. Verifica se tem algum filtro além do de distância e preço que é feito separado
3. Usa hash map para busca rápida de filmes
4. Para cada filme do cinema: verifica se é documentário
5. Ordena por nome do cinema



Otimizações Implementadas

1. Hash Map para Índice

- **Antes:** $O(n)$ para encontrar filme por ID
- **Depois:** $O(1)$ para encontrar filme por ID
- **Impacto:** Crítico para busca de cinemas

2. Early Return

3. Short-Circuit Evaluation

- Testa filtros mais seletivos primeiro
- Evita computações desnecessárias



Tratamento de Casos Especiais

1. Filme Não Encontrado

"Caso o cinema referencie um código de filme que não existe você deve pegar o filme com o código maior mais próximo"

Nota: Implementamos verificação de nullptr, mas não a busca do "próximo maior"

2. Dados Inconsistentes

- Campos vazios ou "N" tratados como valores padrão
- Verificação de tamanho mínimo de campos



Pontos de Melhoria Identificados

Busca do "Filme Mais Próximo"

- **Atual:** Retorna nullptr se não encontra
- **Ideal:** Implementar busca do ID mais próximo
- **Solução:** Ordenar IDs e usar binary search



Comparação com Alternativas

Por que não Binary Search?

- **Múltiplos critérios** simultâneos
- Dados não ordenados por todos os campos
- **Custo de ordenação** seria maior que benefício

Por que não Hash Table para todos os filtros?

- **Flexibilidade:** Filtros dinâmicos pelo usuário
- **Simplicidade:** Linear search é mais direto





Por que STL sort?

- **Otimização:** Implementação altamente otimizada



Conclusões

Objetivos Alcançados

-  Sistema funcional de busca de filmes e cinemas
-  Combinação de múltiplos filtros
-  Performance adequada (carregamento vs. consulta)
-  Código limpo e bem estruturado

Conhecimentos Aplicados

- **Estruturas de dados:** Arrays, Hash Maps, Sets
- **Algoritmos de busca:** Linear search otimizada
- **Algoritmos de ordenação:** STL sort com comparadores
- **Otimização:** Trade-offs tempo vs. espaço