Relatório do Trabalho 2 de Estrutura de Dados

João Pedro Pereira Balieiro - Nº USP 12676615

10/12/2024

1 Descrição do Trabalho

Este trabalho implementa um sistema para gerenciar cadastros de alunos e suas preferências por filmes, utilizando conceitos de Estruturas de Dados, listas encadeadas e árvores binárias de busca balanceadas (Árvore SBB). O objetivo principal é permitir operações diversas sobre os dados dos alunos, incluindo busca, recomendação e análise dos registros.

2 Estruturas de Dados Utilizadas

2.1 Árvore Binária de Busca Balanceada (SBB)

A estrutura principal para armazenar os alunos é uma árvore binária de busca balanceada, onde o número USP de cada aluno é utilizado como chave. Essa estrutura foi escolhida por oferecer eficiência em operações de busca.

2.2 Lista de Filmes (Encadeada)

Cada nó da árvore contém uma lista encadeada de filmes, representando as preferências do aluno. A lista é ordenada alfabeticamente para simplificar as operações de busca e exibição.

2.3 Top 3 Alunos com Mais Filmes (Estrutura Auxiliar)

Para a funcionalidade extra, foi criada uma estrutura auxiliar que armazena os três alunos com mais filmes cadastrados.

3 Funcionalidades Implementadas

3.1 Cadastro de Alunos

Permite inserir um aluno no sistema informando:

- Número USP.
- Nome do aluno.
- Lista de filmes favoritos. A entrada é dinâmica, permitindo que o usuário insira quantos filmes desejar. A lista é ordenada alfabeticamente para padronização.

Após o cadastro, o aluno e sua lista de filmes são inseridos na Árvore SBB, utilizando o número USP como chave.

3.2 Listagem de Alunos

Utilizando um percurso em ordem na Árvore SBB, exibe a lista completa de alunos cadastrados, apresentando:

- Número USP.

- Nome do aluno.
- Lista de filmes cadastrados para cada aluno.

3.3 Busca de Alunos

A busca de alunos é realizada utilizando a estrutura da Árvore Binária de Busca Balanceada (SBB). A operação segue o seguinte algoritmo:

1. Recebe como entrada o número USP do aluno a ser buscado, que é a chave principal de cada nó da árvore.
2. Inicia a busca a partir do nó raiz da árvore.
3. Em cada nó, compara a chave buscada com a chave armazenada no nó atual:
    - Se a chave buscada for menor que a chave do nó atual, a busca é continuada na subárvore esquerda.
    - Se a chave buscada for maior, a busca prossegue na subárvore direita.
    - Se as chaves forem iguais, o aluno é encontrado, e seus dados são retornados.
4. Caso um nó nulo (folha) seja alcançado antes de encontrar a chave, o sistema indica que o aluno não está cadastrado.

Complexidade: Como a árvore é balanceada, a busca possui complexidade $O(\log n)$, onde n é o número de nós na árvore.

3.4 Listagem de Filmes Diferentes

Coleta os filmes cadastrados para todos os alunos, eliminando duplicatas e exibindo a lista final em ordem alfabética. Esta funcionalidade utiliza uma lista encadeada auxiliar para armazenar os filmes únicos.

3.5 Busca de Filmes

A busca de filmes ocorre em duas etapas:

1. Primeiro, todos os filmes cadastrados no sistema são coletados em uma lista auxiliar, que armazena apenas títulos únicos.
2. Para verificar se um filme específico está no sistema:
    - A lista de filmes únicos é percorrida e comparada com o título informado.
    - Se o filme for encontrado, uma mensagem de confirmação é exibida.
    - Caso contrário, o sistema informa que o filme não foi citado por nenhum aluno.

Complexidade: A busca por filmes na lista única possui complexidade $O(n)$, onde n é o número de filmes diferentes cadastrados.

3.6 Recomendação de Colegas para Cinema

A busca de um colega com interesses similares é realizada percorrendo toda a árvore para comparar a lista de filmes do aluno consultado com as listas de outros alunos:

1. A similaridade é calculada pelo número de filmes em comum entre dois alunos.
2. O sistema armazena o colega com a maior similaridade encontrada durante o percurso.
3. Caso nenhum aluno tenha filmes em comum, o sistema não faz uma recomendação.

3.7 Indicação de Colega com Perfil Diferente

A busca do colega com perfil mais distinto segue uma lógica semelhante à busca por similaridade, mas priorizando o menor número de filmes em comum:

1. Durante o percurso na árvore, a similaridade entre o aluno consultado e outros alunos é calculada.
2. O sistema armazena o colega com a menor similaridade encontrada.
3. Se todos os colegas tiverem uma similaridade alta, o sistema indica que não há sugestões com perfis muito diferentes.

3.8 Exportação de Dados

Exporta todas as informações cadastradas no sistema para um arquivo texto. O arquivo gerado apresenta:

- Dados de cada aluno (número USP, nome e filmes cadastrados).
- Filmes ordenados em ordem alfabética.

3.9 Dados Técnicos da Árvore

Exibe informações técnicas sobre a Árvore SBB:

- Número total de nós.
- Altura total da árvore.
- Maior diferença entre as alturas das subárvores de qualquer nó.

3.10 Remoção de Alunos

Remove um aluno da Árvore SBB com base em seu número USP. Após a remoção, a árvore é reestruturada dinamicamente, mantendo sua estrutura balanceada.

3.11 Funcionalidade Extra: Top 3 Alunos com Mais Filmes

Identifica os três alunos com o maior número de filmes cadastrados. Utiliza uma estrutura auxiliar para armazenar e ordenar os dados dos alunos, exibindo os resultados em ordem decrescente.

4 Como Compilar e Rodar o Programa

5.1. Pré-requisitos

- Compilador C (recomendado: GCC versão 9.3 ou superior).
- Sistema Operacional: Linux ou Windows.

4.1 Compilação

Utilize o `CMakeLists.txt` fornecido para configurar e compilar o projeto:

```
cmake .
make
```

4.2 Execução

Após a compilação, o executável gerado será `trabalho_ed__`. Execute-o da seguinte forma:

```
./trabalho_ed__
```

4.3 Telas de Exemplo

Menu principal:

```
Menu do Sistema:
1. Criar cadastro de aluno
2. Listar todos os alunos
3. Buscar aluno no sistema
...
```

Cadastro de aluno:

```
Informe o número USP: 123456
Informe o nome: João Silva
Digite os nomes dos filmes:
...
```