Amigos de Graduação Indo ao Cinema

Trabalho 2

SCC0223 Estruturas de Dados I

Cody Stefano Barham Setti – 4856322
Luís Henrique de Queiroz Veras – 14592414
Raphael Mendes Batista – 15497660
Vinícius de Sá Ferreira – 15491650

08 de Dezembro de 2024

1 Descrição do Trabalho Feito

1.1 Visão Geral

Para desenvolver o sistema de cadastro de alunos de graduação e seus interesses cinematográficos, uma certa estrutura de dados foi fundamental: a árvore de Adelson-Velskii e Landis, ou, mais brevemente, a AVL.

Duas instâncias de AVL foram utilizadas, uma árvore para armazenar os alunos cadastrados e outra para armazenar todos os filmes mencionados pelos alunos. A chave de ordenação utilizada na AVL de alunos foi o número USP. Os filmes, por sua vez, foram ordenados de forma alfabética. Portanto, o próprio nome de cada filme serve como sua respectiva chave.

Por sua vez, os filmes favoritos de cada aluno individual foram armazenados em forma de lista ordenada (com cabeçalho). Entretanto, é importante frisar que os nós dessas listas foram reaproveitados da AVL de filmes – para economizar memória – logo, as listas, na realidade, são uma sequência estratégica de ponteiros.

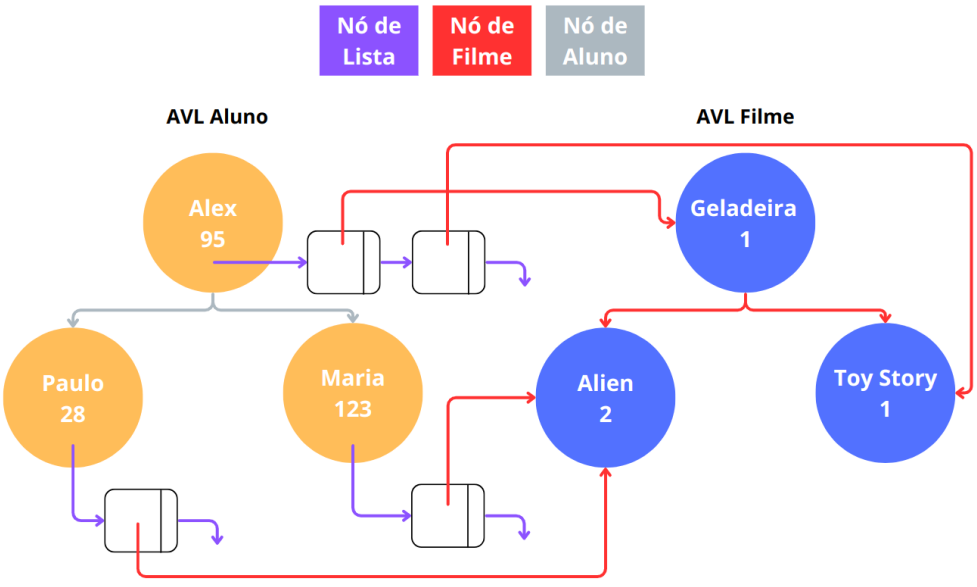


Figura 1: Visão esquemática do armazenamento dos dados do trabalho

Finalmente, vale salientar que a estrutura de lista ordenada foi reaproveitada para a funcionalidade de impressão dos filmes mais queridos.

1.2 A Biblioteca 'AVL'

Tratemos agora da implementação dessas estruturas basilares na linguagem de programação C, contida nos arquivos `<avl.h>` e `<avl.c>`.

1.2.1 Implementação da AVL

Primeiramente, ao se tratar da AVL, seus nós foram instanciados no tipo de dado `no`. Tal dado faz uso de um `enum`, para dá-lo a flexibilidade de armazenar as informações de um aluno ou de um filme. A *flag* utilizada para indicar cada caso foi denominada de `type`, onde `type=0` indica o armazenamento do tipo `aluno` e `type=1` do tipo `filme`.

As variáveis armazenadas dentro desses dois tipos são

`aluno`:

- `*nome`: um ponteiro do tipo `char` para referenciar o primeiro caractere da *string* que contém o nome do aluno.
- `nroUSP`: uma variável do tipo `int` para armazenar o número USP do respectivo aluno.
- `*filmes_fav`: um ponteiro do tipo `NoLista` (o qual será explicado posteriormente), para referenciar o cabeçalho da lista de filmes favoritos do aluno.

`filme`:

- `*nome`: um ponteiro do tipo `char` para referenciar o primeiro caractere da *string* que contém o nome do filme.
- `frequencia`: uma variável do tipo `int` que registra o número de alunos que adicionaram à sua lista de favoritos o respectivo `filme`.

O conjunto de `type` com `aluno`, ou então, com `filme`, é amalgamado no tipo `elem`. Por fim, além de seu `elem`, todo `no` possui também as variáveis

- `FB`: uma variável do tipo `int`, utilizada para armazenar o fator de balanceamento do respectivo `no` em relação à AVL à qual pertence.
- `altura`: uma variável do tipo `int`, utilizada para armazenar a altura do respectivo `no` em relação à AVL à qual pertence.
- `*esq`: um ponteiro a uma variável do tipo `no`, que cumpre o papel de filho esquerdo na AVL.
- `*dir`: um ponteiro a uma variável do tipo `no`, que cumpre o papel de filho direito na AVL.

Para as AVL's em si, criou-se o tipo de dado `AVL`, que possui, simplesmente, um ponteiro `*raiz`, que referencia o `no` ocupando a raiz atual da árvore.

1.2.2 Implementação da Lista Ordenada

A instanciação dos nós das listas ordenadas ao invés de utilizar o tipo `no`, aproveitou um tipo distinto, o `NoLista`, que contém duas variáveis:

- `*filme`: um ponteiro para variáveis do tipo `no`, a fim de referenciar itens da AVL de filmes.
- `*prox`: um ponteiro para variáveis do tipo `NoLista`, para referenciar o próximo nó da lista ordenada.

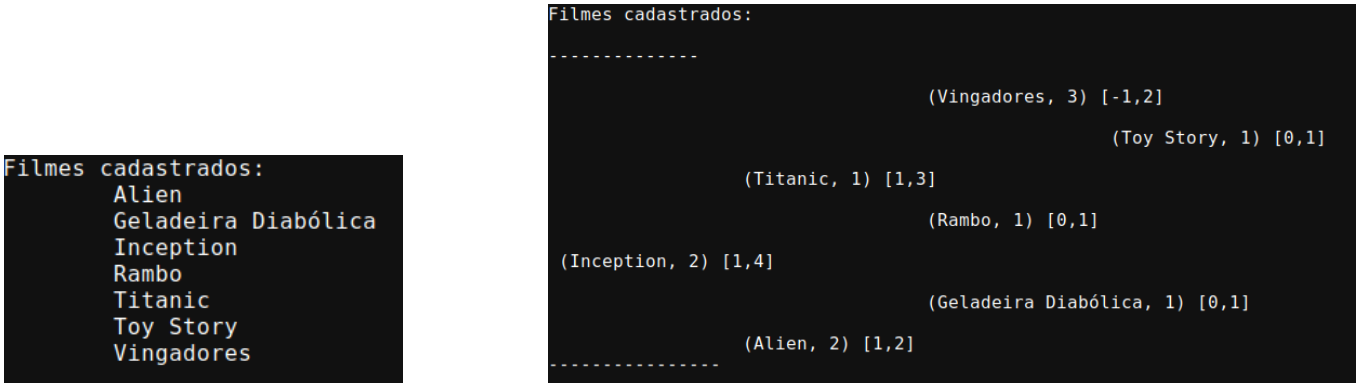
O tipo `Lista` foi definido para instanciar-se a lista, propriamente dita. De variáveis, ele contém simplesmente um ponteiro `*ini`, que aponta ao seu primeiro `NoLista`, o cabeçalho.

1.2.3 As Funções da Biblioteca 'AVL'

Por fim, os arquivos `<avl.h>` e `<avl.c>` possuem algumas funções.

Primeiro, há algumas correspondentes ao funcionamento elementar de uma AVL: `Criar()` e `FinalizarAVL()`, `inserir()` e `remover()`, que também rebalanceiam a árvore, e `buscar()`, que aproveita a ordenação da árvore para fazer uma busca eficiente (tempo logarítmico) por um elemento na árvore.

Ademais, para a conveniência do usuário, a biblioteca já inclui duas formas de imprimir os dados da AVL, encapsulados nas funções `imprimirAVL()` e `imprimirGrafo()`.



(a) `imprimirAVL()` (b) `imprimirGrafo()`

Figura 2: Os dois modos de imprimir as AVL's

Por fim, de interesse, há as funções `imprimir_dadosAVL()` e `salvar_sistema()`, que serão explicadas posteriormente, já que se referem diretamente a funcionalidades pedidas no projeto.

1.3 A Biblioteca 'Aluno'

Os arquivos `<aluno.h>` e `<aluno.c>` encapsulam as funcionalidades específicas à AVL de alunos[1].

Nenhum tipo de dado novo é definido, apenas funções. Algumas, elementares, para a inserção e remoção de filmes da lista de favoritos de cada aluno. E outras mais elaboradas. Os dois algoritmos de sugestão de parceiros de cinema, `sugerir_similar()` e `sugerir_diferente()`; e uma função para suporte de edição de arquivos, `carregar_arquivo()`.

1.4 A Biblioteca 'Filme'

A biblioteca 'Amigos de Graduação Indo ao Cinema' é completa com os arquivos `<filme.h>` e `<filme.c>`, cujo conteúdo resume-se à função `mais_queridos()`, que lista o catálogo de filmes dos alunos em ordem de popularidade, mais suas funções auxiliares (escondidas do usuário).

1.5 Funcionalidades da Biblioteca

Conforme as especificações do trabalho, a biblioteca 'Amigos de Graduação Indo ao Cinema' deve ser capaz de:

1. Cadastrar alunos em seu sistema;
2. Remover de seu sistema o cadastro de um aluno;
3. Dado um aluno, adicionar um filme à sua lista de prediletos;
4. Dado um aluno, remover um filme de sua lista de prediletos;
5. Listar todos os alunos cadastrados;

[1]Para seu funcionamento, é necessário a invocação da Biblioteca 'AVL'.

6. Buscar um aluno específico no sistema;
7. Listar os filmes cadastrados em ordem alfabética;
8. Buscar um filme específico no sistema;
9. Dado um aluno, indicá-lo um colega com gosto cinematográfico similar ao seu;
10. Dado um aluno, indicá-lo um colega com gosto cinematográfico diferente ao seu;
11. Salvar em um arquivo texto todas as informações armazenadas no sistema;
12. Fornecer os dados técnicos das ABB's utilizadas: o número de nós de cada árvore, suas alturas totais e o maior fator de balanceamento dentre os nós das árvores;
13. Listar os filmes em ordem de popularidade;
14. Cadastrar alunos e seus filmes favoritos a partir de arquivo;
15. Finalizar todos os dados;
16. Limpar a tela;
17. Modo de visualização.

Na função `main()`, um menu é apresentado ao usuário, mostrando as 17 opções (mais a opção '0', para sair do sistema). Ele então digita qual quer utilizar e, internamente, um `switch` faz o controle de quais funções da biblioteca chamar. Vejamos como cada funcionalidade foi implementada:

Funcionalidade 1

O cadastro de alunos no sistema é feito pela função `inserir()`, que, essencialmente, é um algoritmo de inserção de nós em uma AVL, ou seja, primeiro insere o aluno como uma folha, respeitando a estrutura de uma ABB, e, então, caso necessário, faz rotações para ajustar o balanceamento da árvore, a fim de que seja AVL.

Utilizou-se a altura de cada nó para a checagem de balanceamento, assim há homogeneidade entre o funcionamento dos algoritmos de adição e remoção de alunos.

Funcionalidade 2

Para cumprir tal funcionalidade primeiro é chamada a função `limpar_filmes_fav()`, que remove da AVL de filmes aqueles cuja frequência torna-se 0, com a remoção do aluno em questão, e atualiza as frequências (subtrai 1) daqueles de sua lista ainda mencionados por outros alunos.

Em seguida, a função `remover()` é chamada, a qual remove da AVL de alunos o aluno em questão, certificando-se de rebalancear a árvore após.

Funcionalidade 3

A função que, dado um aluno, adiciona um filme à sua lista de prediletos é a `inserir_filme_fav()` da biblioteca 'Aluno'.

Ela opera de forma recursiva. Primeiramente, se o cabeçalho da lista de filmes favoritos não apresenta filmes na sequência, temos um caso base: o novo filme é inserido logo após o cabeçalho e retorna-se à função o valor `1`, um `int` que corresponde ao sucesso na inserção do filme na lista.

Se esse não for o caso, vai-se percorrendo os nós da lista recursivamente até encontrar-se um cujo título de filme venha após, em ordem alfabética, o do novo que quer-se inserir na lista, indicado por um retorno de `-1` (outro caso base).

Com esse valor, a função sabe, então, que deve inserir o filme anteriormente a tal nó, para preservar-se a ordem alfabética e as chamadas anteriores de função todas retornam o valor `1`.

Caso todos os filmes da lista antecedam, em ordem alfabética, o filme novo que deseja-se inserir, a última vez que a função será chamada será para um nó `NULL`. A função também tem retorno `-1` neste caso e, portanto, ao voltar a penúltima chamada de função, insere o filme ao final da lista.

Caso o usuário tente inserir um filme que já está na lista, na hora de compará-lo a si mesmo, não adentrará nenhum dos condicionais que checam se seu título vem estritamente antes ou após o

do nó, retornando o valor `0` (mais um caso base). As chamadas anteriores da função então também assumem tal valor e nada na lista é alterada.

Por fim, caso haja erro de `malloc`, a função retorna `0`, pois nesse caso também não é possível inserir o filme na lista, já que falta espaço na memória para gravá-lo nela.

Funcionalidade 4

Para a remoção de um filme predileto, está às ordens a função `remover_filme_fav()`, cujo funcionamento é similar ao da função `inserir_filme_fav()`, e a qual também pertence à biblioteca 'Aluno'.

A função é primeiro chamada para o cabeçalho da lista e, logo após, antes de cessar tal chamada, faz-se outra para o próximo nó. Se ele for nulo, a lista é inalterada e o chamado da função finalizado. Caso contrário, analisa-se se o filme de tal nó é o desejado para remoção. Se sim, o encadeamento da lista é rearranjado sem ele e ele é finalizado. Se não, prossegue-se ao próximo nó da lista.

Ademais, na remoção, a frequência de aparição do filme no catálogo é diminuída. Se chegar a 0, no trecho de código da função `main()` dedicado à funcionalidade 4, invoca-se a remoção do filme da AVL de filmes.

Funcionalidade 5

A função que encarrega-se de listar todos os alunos cadastrados é a `imprimirAVL()`, cujo funcionamento resume-se a chamar a função `imprimir_recursivo()` para sua raiz.

Tal função, por meio de uma recursão, percorre a árvore *em ordem*: imprime a sub-árvore esquerda da raiz, depois a raiz e por fim sua sub-árvore direita. Consequentemente, os alunos são impressos em ordem crescente de número USP.

Alternativamente, há a função `imprimirGrafo()`, que invoca a função `imprimir_recursivo2()`, para imprimir os dados em forma de árvore ao invés de lista. A alternância entre os dois modos de impressão é realizado na funcionalidade 17.

Funcionalidade 6

A busca de um aluno específico no sistema é feita pela função `buscar()`, a qual aproveita a ordenação da AVL: se a chave do elemento procurado for menor do que a do nó atual sendo analisado, procede-se ao seu filho esquerdo; caso contrário, o direito. Ou seja, `buscar()` é uma função recursiva.

Há algumas condições de parada: se a chave de um nó for a mesma daquela do aluno procurado, retorna-se o endereço de tal nó. Se chegar-se em um nó `NULL`, o retorno da função é `NULL`, para indicar que o aluno procurado não encontra-se na AVL.

Funcionalidade 7

A listagem em ordem alfabética dos filmes cadastrados é feita novamente pela função `imprimirAVL()`.

Já que a árvore é de busca e o chaveamento da AVL de filmes é conforme seus nomes, a impressão *em ordem* – precisamente o modo adotado por `imprimirAVL()` – garante que os filmes sejam listados alfabeticamente.

Funcionalidade 8

Como no caso da funcionalidade 6, a busca é feita por meio da função `buscar()`, da biblioteca 'AVL'.

Funcionalidade 9

Primeiramente, é necessário explicar a métrica elegida para determinar o aluno de gosto mais similar ao colega em questão: o número de filmes em comum entre os dois. O último aluno encontrado na

árvore (percorrida em ordem) de maior número de filmes em comum (excluindo o próprio colega), é elegido como aluno de gosto mais similar.

Tal funcionalidade é encapsulada na função `sugerir_similar()`. Por meio de uma recursão, ela percorre a AVL de alunos em ordem e, para cada nó, chama a subfunção `filmes_fav_comum()`. Se `filmes_fav_comum()` retornar um número maior ou igual do que o atual máximo, este é atualizado.

Por sua vez, quanto ao funcionamento interno de `filmes_fav_comum()`: aproveita-se de um `while()` aliado ao fato das duas listas de filmes favoritos estarem em ordem alfabética para contar o número de filmes em comum: um índice `i` é usado para percorrer a lista do colega, e outro `j` para a lista do nó atual.

- Se o título do filme referente ao índice `i` (filme na lista do colega) for menor do que o do indicado por `j` (filme na lista do nó atual), faz-se um incremento em `i`.
- Se for maior, faz-se um incremento em `j`.
- Se forem iguais, faz-se um incremento em ambos `i` e `j`, e no número de filmes em comum.

O `while` é finalizado quando ambos os índices tornam-se nulos, o que significa que todos filmes foram comparados.

Funcionalidade 10

A métrica adotada neste caso é análoga ao do caso anterior, entretanto, elege-se o último aluno com o menor – ao invés de maior – número de filmes em comum com o colega.

Tal funcionalidade é encapsulada na função `sugerir_diferente()`, cuja única diferença em relação à função `sugerir_similar()` é que o aluno a se sugerir é atualizado caso `filmes_fav_comum()` encontrar outro com um número menor ou igual – ao invés de maior ou igual – de filmes em comum.

Funcionalidade 11

Para salvar as informações do sistema em um arquivo `.txt`, tem-se a função `salvar_sistema()`. Ela cria ou sobrescreve o arquivo `saida_sistema.txt`, pertencente ao diretório `./src/`, conforme o seguinte padrão desenvolvido: percorrendo a AVL de alunos *em ordem*, para cada nó seu, a função cria uma linha correspondente no arquivo e insere nela, na ordem apresentada, as seguintes informações,:

1. Nome do aluno;
2. Número USP do aluno;
3. Nome dos filmes favoritos.

Utiliza-se o ponto e vírgula como separados dessas informações.

Funcionalidade 12

Os três dados técnicos sobre as ABB's utilizadas são fornecidos conjuntamente pela função `imprimir_dadosAVL()`, que invoca três sub-rotinas:

1. `numero_nos()`: Uma função recursiva que percorre a árvore em pré-ordem, dando um incremento unitário ao seu contador de nós[2] a cada recursão. Seu passo base é quando depara-se com um nó `NULL`. Neste caso, retorna 0.
2. `altura()`: Já que a altura de cada nó deve ser monitorada para o balanceamento da AVL, a função `altura()` aproveita-se desse fato e simplesmente recupera a altura de sua raiz.

[2]O próprio retorno da função.

3. `maior_FB()`: O fato de optarmos pelo uso de AVL traz um de seus maiores facilitadores aqui. Pela teoria de AVL, sabe-se que, se houver uma diferença de altura entre suas sub-árvores, ela terá módulo igual a 1, pois diferenças maiores não são admitidas. Se não houver diferenças de altura, isso implica que o FB de todos seus nós será 0.

   Consequentemente, `maior_FB()` simplesmente percorre a AVL em *pré-ordem* e, se encontrar um nó com $|\text{FB}| = 1$, cessa a busca, retornando 1. Caso percorra a árvore inteira sem adentrar o condicional, retorna 0.

Funcionalidade 13

A listagem dos filmes em ordem de popularidade é encapsulada na função `mais_queridos()`. A função começa criando tal lista. Caso haja espaço na memória para tanto, prossegue para inserção dos filmes nela, feita pela função auxiliar `inserirListaFreq()`.

Por sua vez, `inserirListaFreq()` percorre a AVL de filmes *em ordem* e, para a ordenação, faz uso da função `inserir_recursivo()`.

Finalmente, `inserir_recursivo()`, dada a lista de filmes populares em formação e um nó da AVL de filmes, checa a popularidade do nó em relação aos elementos da lista, inserindo-o após os de popularidade estritamente maior (os detalhes são análogos ao funcionamento da função `inserir_filme_fav()`, pois, em ambos os casos, está-se reordenando uma lista após a inserção de um elemento novo).

Vale salientar que os filmes de mesma popularidade estarão listados em ordem alfabética, pois a AVL de filmes foi percorrida *em ordem*, como também sempre insere-se filmes de frequência 1 ao fim da lista, tanto para preservar esse 'sub-ordenamento', como para fins de otimização.

Funcionalidade 14

O cadastro de alunos e seus respectivos filmes favoritos no sistema a partir de um arquivo `.txt` é feito por meio da função `carregar_arquivo()`.

Arquivos para leitura, igualmente aos para salvar o sistema, são mantidos no diretório `./src/`. Entretanto, apenas um foi desenvolvido: `entrada_sistema.txt`. Sua formatação segue a mesma lógica do arquivo `salvar_sistema.txt`: cada linha é dedicada a um aluno, cujos dados são separados pelo ponto e vírgula e, presume-se que sua ordem seja

1. Nome do aluno;
2. Número USP;
3. Nome dos filmes favoritos.

Não é necessário fornecer informações completas. Entretanto, alguns casos podem acusar erros, como, por exemplo, fornecer o nome de um aluno sem número USP.

São feitas checagens: se o aluno já encontra-se no sistema (acusa um erro nesse caso), e se filmes favoritos foram repetidos, para que as listas dos alunos não contenham redundância, nem haja erro na popularidade do catálogo.

Funcionalidade 15

A finalização de todos os dados é feita chamando-se a função `FinalizarAVL()` duas vezes. Uma para a árvore de alunos, e outra para a de filmes.

Tal função percorre as árvores de forma *pós-ordem*. Caso contrário, a referência a sub-árvores seria perdida ao deletar-se a raiz primeiro.

Para o caso da AVL de alunos, uma função auxiliar `finalizarLista()` é invocada antes de deletar-se o respectivo nó, a fim de finalizar também a lista de filmes favoritos do aluno correspondente. Isso é feito percorrendo-se a lista de filmes favoritos até seu fim e, então, voltando e deletando seus nós no caminho, o que inclui o cabeçalho da lista.

Para a AVL de filmes, a função `FinalizarAVL()` imediatamente finalizar os nós, já que não há listas atrelados a eles.

Por fim, salienta-se que depois dessas chamadas de funções, resta uma raiz para cada AVL. Entretanto, no arquivo `main.c`, em que a funcionalidade 15 está implementada, as AVL's são definitivamente finalizadas.

Funcionalidade 16

Simplesmente, a tela do terminal é limpada por meio do comando nativo `system("clear")` para a implementação em LINUX, ou então o comando `system("cls")` para a em Windows, visando a qualidade de vida do usuário.

Funcionalidade 17

Desenvolveu-se um modo de visualização alternativa para os dados das AVL's. No lugar de listar os alunos ou filmes, são mostrados conforme seu armazenamento nas árvores.

Logo, a funcionalidade 17 permite ao usuário alternar entre o modo de visualização em lista e o em árvore, este segundo internamente denominado de `impressaoGrafo()`.

2 Estruturas de Dados Utilizadas e Justificativas

As estruturas de dados utilizadas foram AVL e lista dinâmica encadeada ordenada com cabeçalho.

A estrutura de AVL foi utilizada por dois motivos. Primeiro, ela é excelente para receber e armazenar ordenadamente dados fornecidos de forma aleatória. Segundo, o fato dela preservar um balanceamento sempre, garante que o custo temporal de percorrê-la sempre seja bem próximo de $O(\log(n))$, onde n é o número de nós na árvore.

Por sua vez, já que cada aluno deve ter uma respectiva lista de filmes favoritos, ordenados de forma alfabética, utilizou-se uma lista com cabeçalho, pois o cabeçalho facilita a implementação de algoritmos de ordenação.

3 Apresentação do Sistema

3.1 Compilador Utilizado

```
nait@nait:~/Documents/USP/2SEM/EstrutDados_I/13sem/Trabalho02/Linux$ gcc --version
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

nait@nait:~/Documents/USP/2SEM/EstrutDados_I/13sem/Trabalho02/Linux$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.5 LTS
Release:        22.04
Codename:       jammy
```

3.2 Rodagem do Programa

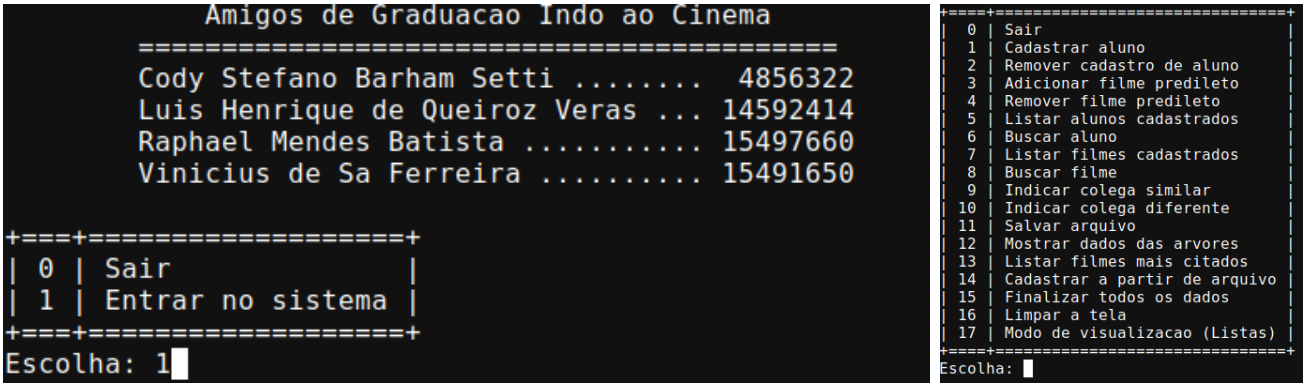


Figura 3: Menus

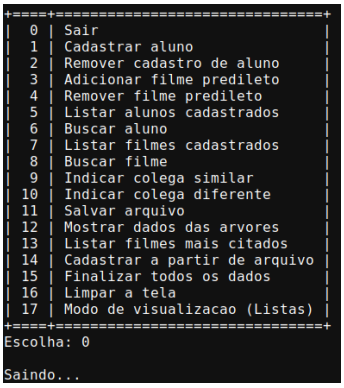


Figura 4: Sair

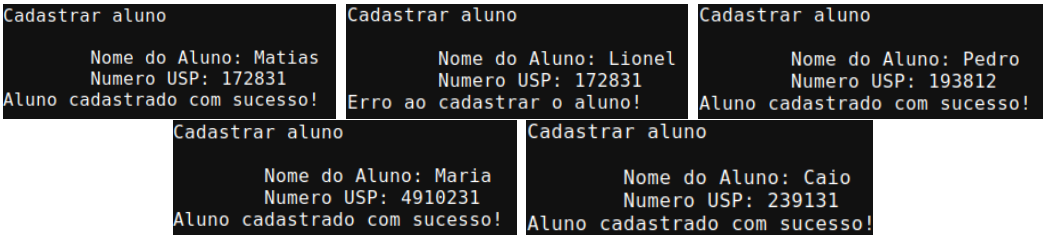


Figura 5: Cadastro de alunos

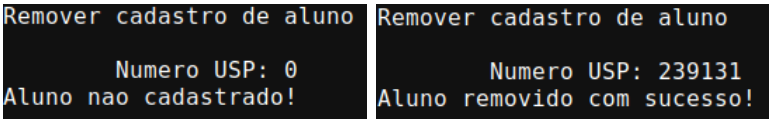


Figura 6: Remove cadastro de aluno

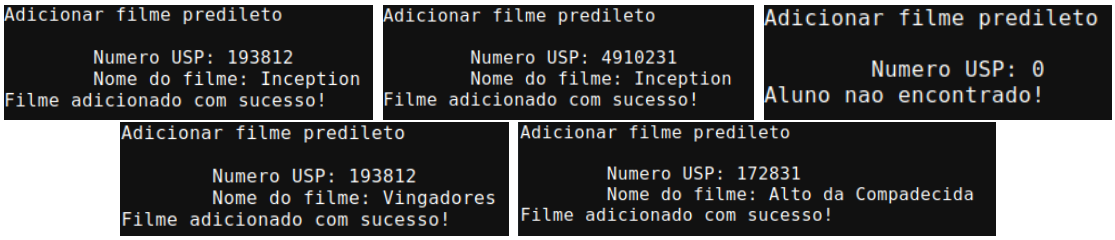


Figura 7: Adiciona os filmes prediletos dos alunos

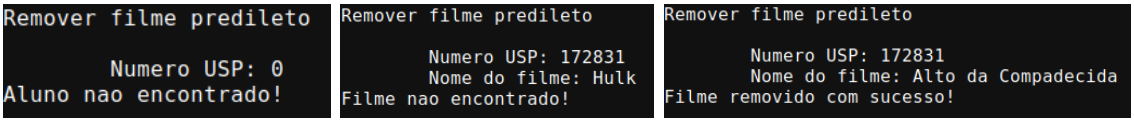


Figura 8: Remove os filmes prediletos dos alunos

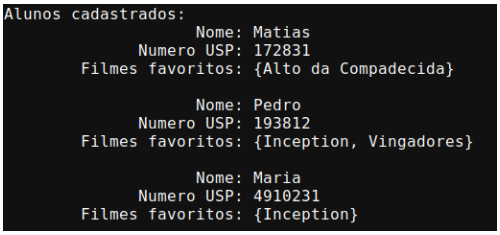


Figura 9.1: Antes da remoção do filme predileto de Matias

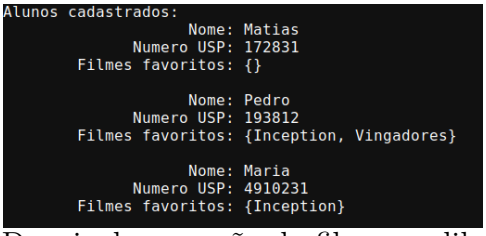


Figura 9.2: Depois da remoção do filme predileto de Matias

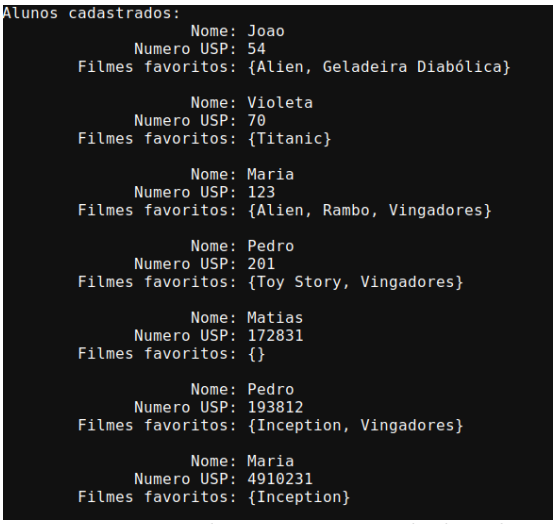


Figura 9.3: Depois de carregar os dados de arquivo

Figura 9: Lista alunos cadastrados, no modo de visualização `Listas`

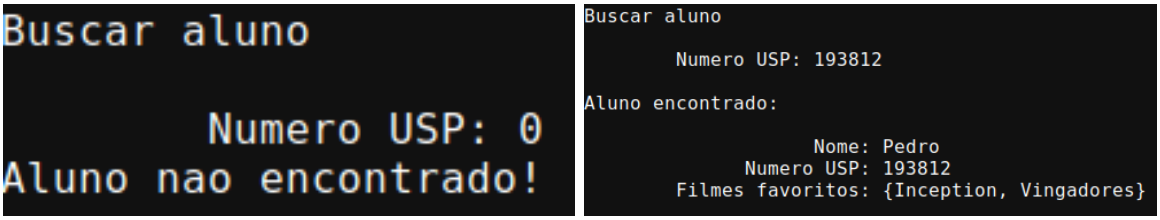


Figura 10: Busca alunos

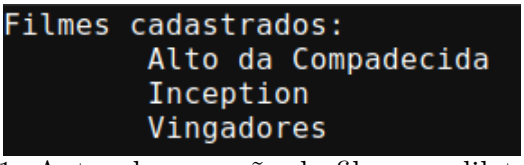


Figura 11.1: Antes da remoção do filme predileto de Matias

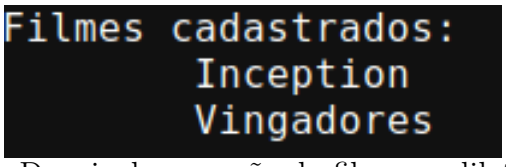


Figura 11.2: Depois da remoção do filme predileto de Matias

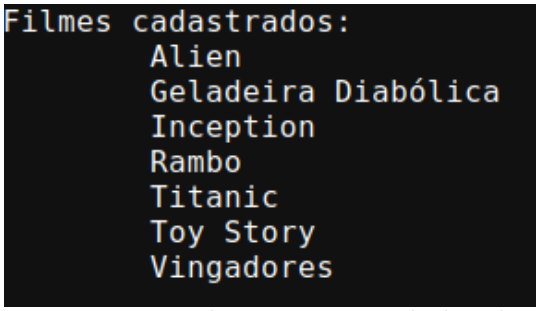


Figura 11.3: Depois de carregar os dados de arquivo

Figura 11: Lista alunos cadastrados, no modo de visualização `Listas`

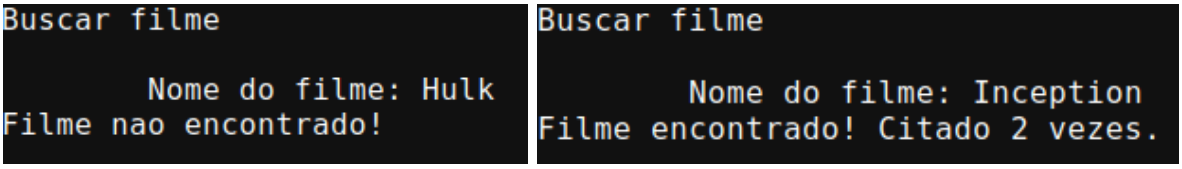


Figura 12: Busca filmes

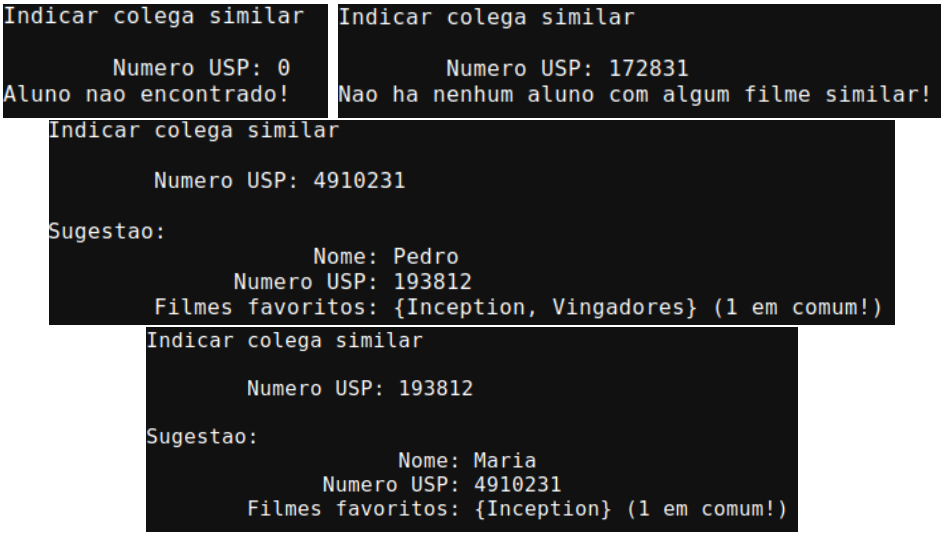


Figura 13: Indica um colega que tenha filmes similares

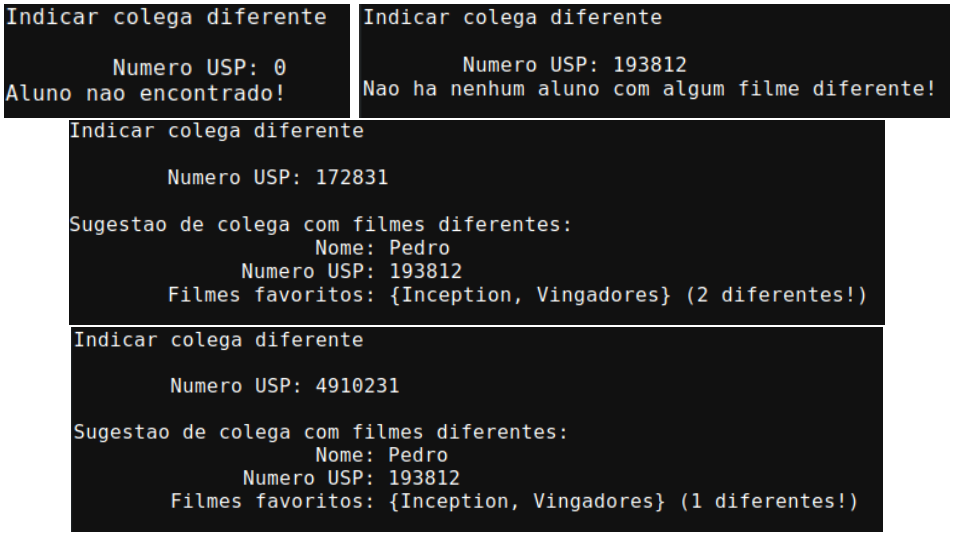


Figura 14: Indica um colega que tenha filmes diferentes

Figura 15: Salva os dados no arquivo `src/saida_sistema.txt`

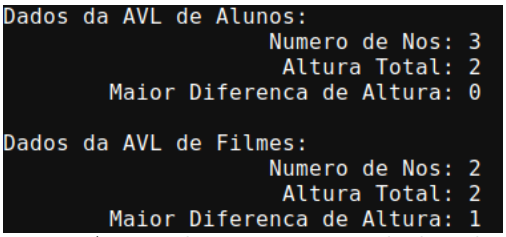


Figura 16.1: Antes de carregar os dados de arquivo

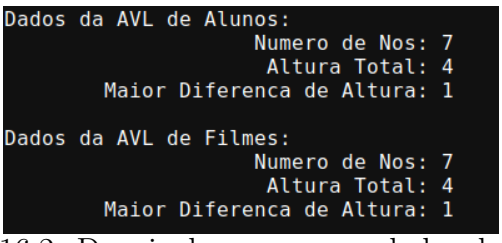


Figura 16.2: Depois de carregar os dados de arquivo

Figura 16: Mostra os dados da AVL de alunos e da AVL de filmes

```
Filmes mais citados:

        Citado 2 vezes: Inception
        Citado 1 vezes: Vingadores
```

Figura 17.1: Antes de carregar os dados de arquivo

```
Filmes mais citados:

        Citado 3 vezes: Vingadores
        Citado 2 vezes: Alien
        Citado 2 vezes: Inception
        Citado 1 vezes: Geladeira Diabólica
        Citado 1 vezes: Rambo
        Citado 1 vezes: Titanic
        Citado 1 vezes: Toy Story
```

Figura 17.2: Depois de carregar os dados de arquivo

Figura 17: Mostra os filmes mais citados em ordem decrescente

```
Insira os dados em 'src/entrada_sistema.txt' da seguinte forma:
        NOME_ALUNO_1;NROUSP_1;FILME_FAV_1;FILME_FAV_2;FILME_FAV_3;...;FILME_FAV_N
        NOME_ALUNO_2;NROUSP_3;FILME_FAV_1;FILME_FAV_2;FILME_FAV_3;...;FILME_FAV_N
        ...

        Pressione qualquer tecla para continuar...

Dados carregados com sucesso!
```

```
1 Maria;123;Alien;Rambo;Vingadores
2 Joao;54;Alien;Geladeira Diabólica
3 Violeta;70;Titanic
4 Pedro;201;Toy Story;Vingadores
5
```

Figura 18: Lê os dados do arquivo `src/entrada_sistema.txt`

```
Dados apagados com sucesso!
```

```
Nao ha alunos cadastrados.
```

```
Nao ha filmes cadastrados.
```

```
Nao ha alunos cadastrados.

Nao ha filmes cadastrados.
```

```
Filmes mais citados:

Nao ha filmes cadastrados.
```

Figura 19: Finaliza todos os dados das duas AVL's

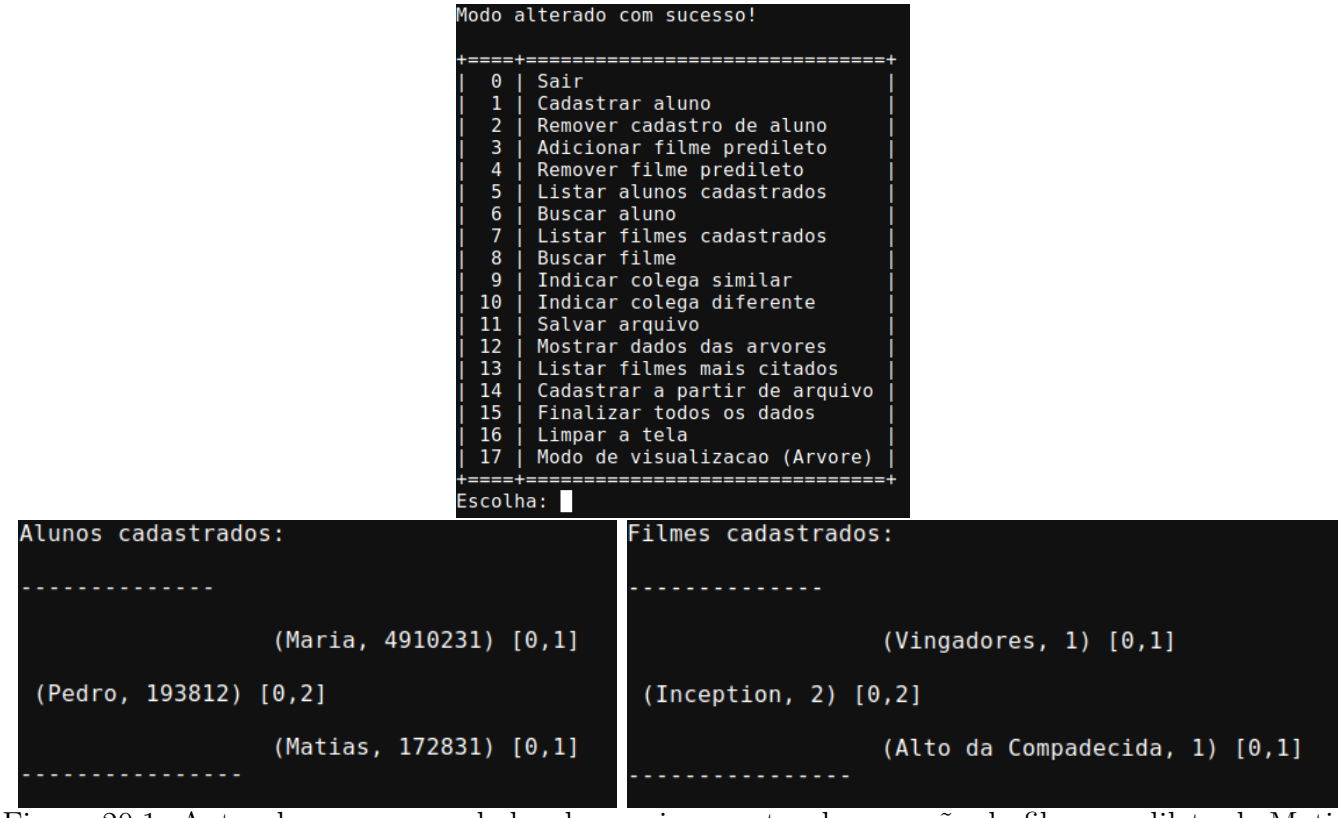


Figura 20.1: Antes de carregar os dados de arquivo e antes da remoção do filme predileto de Matias

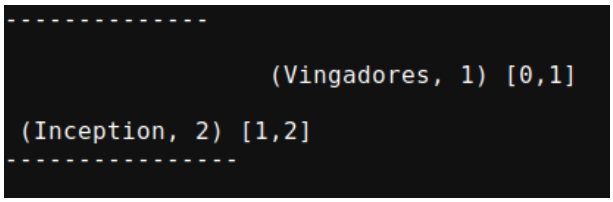


Figura 20.2: Antes de carregar os dados de arquivo e depois da remoção do filme predileto de Matias

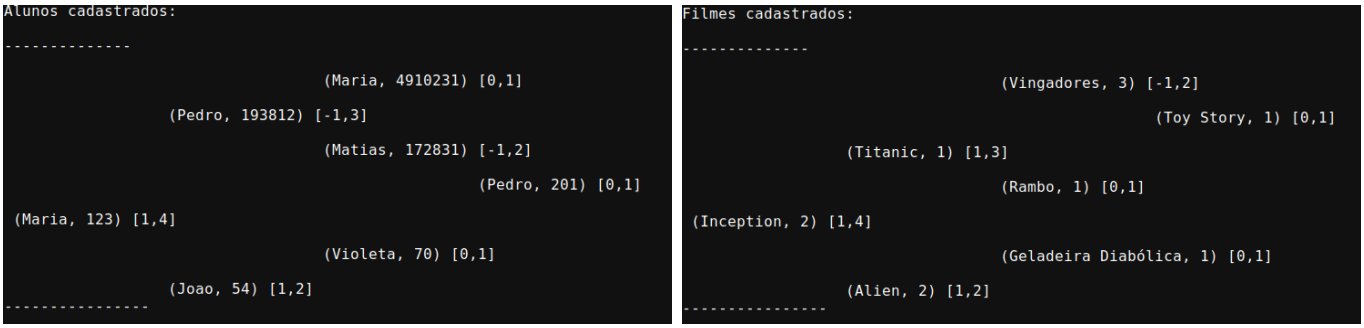


Figura 20.3: Depois de carregar os dados de arquivo

Figura 20: Alterna o modo de visualização de `Listas` para `Arvore`