Parte superior do formulário

**Trabalho Final (2.0)**

O IMDB disponibiliza periodicamente arquivos com informações sobre a indústria cinematográfica estruturados em formato TSV (*tab separated values*). Dentre esses arquivos estão o ***name.basics.tsv.gz*** e o ***title.basics.tsv.gz*** (que podem ser encontrados em [datasets.imdbws.com](https://datasets.imdbws.com/)). Segue um trecho do primeiro:

*nconst primaryName birthYear deathYear primaryProfession knownForTitles*

*nm0000001 Fred Astaire 1899 1987 soundtrack,actor,miscellaneous tt0072308,tt0045537,tt0050419,tt0053137*

*nm0000002 Lauren Bacall 1924 2014 actress,soundtrack tt0037382,tt0117057,tt0038355,tt0075213*

Embora aparentem ser espaços, os valores de cada linha estão separados pelo caracter tab (\t). A primeira linha é o cabeçalho. A partir da segunda linha, o valor da primeira coluna (*nconst*) contem o ID de um artista e o valor da última coluna (*knownForTitles*) contem uma sequência separada por vírgulas dos IDs dos filmes de destaque deste artista. O segundo arquivo está organizado de maneira análoga, contendo os IDs, nomes e outras informações a respeito dos filmes.

O objetivo deste trabalho é **implementar em linguagem C (utilizando apenas os cabeçalhos padrão)** um programa que identifique quais filmes possuem artistas em comum, utilizando para tanto as estruturas de dados vistas no curso até então. Mais especificamente, **o programa deve gerar um grafo (representado como uma lista de adjacências) tal que os vértices sejam correspondam aos filmes e as arestas correspondam a filmes que tiveram artistas em comum no elenco**. Segue um delineamento do processo:

(1) Implemente um *array* dinâmico para armazenar os artistas.

(2) Implemente uma lista de adjacências para armazenar os filmes. (A lista de adjacência é formada por um *array* dinâmico para os vértices e uma lista encadeada para as arestas de cada vértice.)

(3) Crie uma *struct* para os artistas. O nome do artista deve ser alocado dinamicamente e seus filmes de destaque devem ser guardados em uma lista encadeada. Exemplo:

*struct actor {*

*int id;*

*char \* name;*

*struct node \* movies;*

*};*

(4) Crie uma *struct* para os filmes. Como eles serão os vértices de uma lista de adjacências, caso prefira você já pode adaptar essa *struct* para ter informações sobre as adjacências. Exemplo:

*struct movie {*

*int id;*

*char \* title;*

*struct node \* neighbors;*

*};*

(5) Efetue a leitura do arquivo de artistas. (O arquivo é extenso. Visando facilitar seus testes, coloque um contador no laço de leitura para interromper a leitura quando achar necessário.) Extraia de cada linha o ID (observe que este pode ser convertido para *int* caso sejam removidas as duas letras iniciais), o nome do artista e sua lista de filmes. Guarde os dados de cada linha em sua devida *struct* e insira as *structs* no *array*.

(6) De modo similar, efetue a leitura do arquivo de filmes, salvando os IDs (como *int*) e os títulos dos filmes nas devidas *structs* e insira as mesmas na lista de adjacências. As linhas que não correspondam a filmes (isto é, que não possuam o valor *movie* na segunda coluna) devem ser ignoradas. Inicialmente o grafo não deve ter arestas.

(7) Percorra o *array* de artistas, verifique os seus filmes de destaque e forme no grafo uma clique entre esses filmes. A clique é formada inserindo-se arestas (caso ainda não existam) entre todos os vértices em questão (caso existam).

(8) Observe que encontrar os filmes por ID na lista de adjacências custa tempo linear, de modo que a inserção das arestas poderá ser quadrática. Para acelerar esse processo é possível usar uma árvore de busca binária balanceada como um mapa entre os IDs dos filmes e seus índices no *array* que está na base da lista de adjacências. Desse modo torna-se possível recuperar o índice de um filme em tempo logarítmico e encontra-lo na lista em tempo constante. (Para facilitar a implementação você pode utilizar a AVL disponibilizada pelo professor. Observe que esta etapa não é essencial para que o algoritmo funcione, porém, sem ela o tempo de execução será bastante alto.)

(9) Visite o grafo e imprima o mesmo na linguagem *DOT* ([graphviz.org/doc/info/lang.html](https://graphviz.org/doc/info/lang.html)) em um arquivo chamado *input.dot*. Nessa impressão os nomes dos filmes devem ser usados como identificadores, tal como no exemplo a seguir:

*graph { concentrate=true*

*"Matrix" -- "Matrix Reloaded";*

*"Matrix" -- "John Wick";*

*"Titanic" -- "Inception";*

*"Inception" -- "Dark Knight Rises"*

*}*

O arquivo pode ser transformado em imagem através do seguinte comando em um terminal Linux:

*dot -Tsvg input.dot > output.svg*

**O trabalho deve ser feito em equipes de até quatro pessoas e submetido nesta página até a data especificada ao lado (apenas um membro da equipe precisa submeter e apenas o código-fonte deve ser submetido, sem os arquivos adicionais). As defesas ocorrerão durante a aula desse mesmo dia.** Não serão aceitas entregas que ultrapassem o prazo. Cada grupo terá até 10 minutos para apresentar seu trabalho ao professor. **A atribuição da nota será individual e apenas aos integrantes presentes.** Durante a defesa o professor poderá solicitar explicações e modificações no código para qualquer um dos integrantes. Os códigos devem compilar e rodar normalmente nos computadores da universidade, mas a defesa pode ser feita no computador de um dos integrantes da equipe. Instruções adicionais poderão ser dadas ao longo das próximos aulas e/ou acrescentadas nesta especificação.

Parte inferior do formulário