UB Social

CS50AI with Python

1.Degrees

Mateus Schwede

Problemática

- Conforme jogo "Six Degrees" de Kevin Bacon, qualquer pessoa na indústria cinematográfica de Hollywood pode ser conectada a Kevin Bacon em 6 etapas, onde cada etapa consiste em encontrar um filme em que 2 atores estrelaram;
- Neste problema, estamos interessados em encontrar caminho mais curto entre quaisquer 2 atores, escolhendo sequência de filmes que os conecte. Por exemplo, caminho mais curto entre Jennifer Lawrence e Tom Hanks é 2: Jennifer Lawrence está conectada a Kevin Bacon por ambos estrelarem "X-Men: Primeira Classe", e Kevin Bacon está conectado a Tom Hanks por ambos estrelarem "Apollo 13";
- Podemos enquadrar isso como problema de busca: nossos estados são pessoas. Ações são filmes, que nos levam de um ator a outro (é verdade que um filme pode nos levar a vários atores diferentes, mas isso é bom para este problema). Estado inicial e estado objetivo são definidos pelas 2 pessoas que estamos tentando conectar. Ao usar busca em largura, podemos encontrar caminho mais curto de um ator a outro.

Instruções

- Degrees (graus de separação), escreva programa que determine quantos "graus de separação" dois atores estão separados;
- Baixe código de https://cdn.cs50.net/ai/2023/x/projects/0/degrees.zip e descompacte-o.

```
$ python degrees.py large
Loading data...
Data loaded.
Name: Emma Watson
Name: Jennifer Lawrence
3 degrees of separation.
1: Emma Watson and Brendan Gleeson starred in Harry Potter and the Order of the Phoenix
2: Brendan Gleeson and Michael Fassbender starred in Trespass Against Us
3: Michael Fassbender and Jennifer Lawrence starred in X-Men: First Class
```

Planejamento

- O código de distribuição contém 2 conjuntos de arquivos de dados CSV: um conjunto no diretório grande e outro no diretório pequeno. Cada um contém arquivos com mesmos nomes e mesma estrutura, mas small é conjunto de dados muito menor para testes e experimentação. Cada conjunto de dados consiste em 3 arquivos CSV. Abra small/people.csv. Cada pessoa tem id exclusivo, correspondendo ao seu id no banco de dados do IMDb. Eles também têm nome e ano de nascimento;

- Em seguida, abra small/movies.csv. Cada filme também tem id exclusivo, além de título e ano de lançamento do filme. Agora, abra small/stars.csv. Este arquivo estabelece relação entre pessoas (people.csv) e filmes (movies.csv). Cada linha é um par de valor 'person_id' e valor 'movie_id'. A 1ª linha (ignorando cabeçalho), por exemplo, afirma que pessoa id 102 estrelou filme id 104257. Verificando isso em people.csv e movies.csv, perceberá que esta linha está dizendo que Kevin Bacon estrelou filme "A Few Good Men".

Planejamento

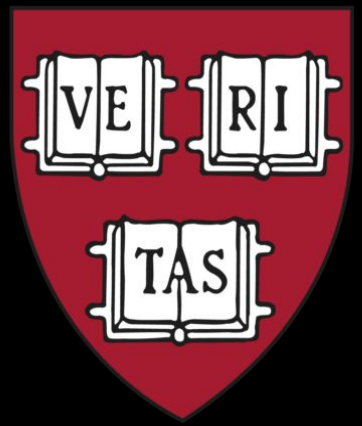
- Após, confira degrees.py. No topo, várias estruturas de dados são definidas para armazenar informações dos arquivos CSV. Dicionário de nomes é uma maneira de procurar pessoa pelo nome, mapeando nomes para conjunto de IDs correspondentes (porque é possível que vários atores tenham mesmo nome). Dicionário de pessoas mapeia ID de cada pessoa para outro dicionário com valores para nome da pessoa, ano de nascimento e conjunto de todos filmes que ela estrelou. Dicionário de filmes mapeia ID de cada filme para outro dicionário com valores para título do filme, ano de lançamento e conjunto de todas estrelas do filme. Função 'load_data' carrega dados dos arquivos CSV para tais estruturas de dados;
- Função principal neste programa, 1º carrega dados na memória (diretório onde dados são carregados pode ser especificado por argumento de linha de comando). Então, função solicita que usuário digite 2 nomes. Função 'person_id_for_name' recupera id de qualquer pessoa (e lida com solicitação do usuário para esclarecer, no caso de várias pessoas terem mesmo nome). Então, chama função 'shortest_path' para calcular caminho mais curto entre 2 pessoas e o imprime;
- Função 'shortest_path', está sem implementação, o qual será o desafio criá-la.

Especificações

- Conclua a função 'shortest_path', de modo que ela retorne caminho mais curto da pessoa com id source (origem) para pessoa com id target (alvo);
- Supondo que haja caminho da origem para alvo, função deve retornar lista, onde cada item é próximo par (movie_id, person_id) no caminho da origem para alvo. Cada par deve ser tupla de 2 strings. Exemplo, valor de retorno de shortest_path é [(1,2), (3,4)], significaria que origem estrelou filme 1 com pessoa 2, pessoa 2 estrelou filme 3 com pessoa 4, e pessoa 4 é alvo;
- Se houver vários caminhos de comprimento mínimo da origem ao destino, função poderá retornar qualquer um deles;
- Se não houver caminho possível entre 2 atores, função deverá retornar None;
- Pode chamar função 'neighbors_for_person', que aceita id de pessoa como entrada e retorna conjunto de pares (movie_id, person_id) para todas pessoas que estrelaram filme com determinada pessoa;
- Não deve modificar nada no arquivo além da função 'shortest_path', embora possa escrever funções adicionais e/ou importar outros módulos da biblioteca padrão do Python.

Submissão

- Visual Studio Code online: https://cs50.dev
- Testar precisão da lógica do algoritmo: check50 ai50/projects/2024/x/degrees
- Testar estilização do código: style50 degrees.py
- Para submissão:
  - Em https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b, entre com GitHub e autorize CS50;
  - Instale pacote Git, Python 3 (e pip), instalando pacotes: pip3 install style50 check50 submit50
  - Submeta o projeto: submit50 ai50/projects/2024/x/degrees
- Verificar avaliação: https://cs50.me/cs50ai.



UB Social

Mateus Schwede
HBS ID 202400167108 - DCE ID @00963203