Desenho e Análise de Algoritmos (DAA)
2° Semestre 2023/2024

Minitrabalho 2 – "Seis graus de separação"

Introdução

"Seis graus de separação" é uma teoria popular que sugere que quaisquer dois indivíduos no mundo podem ser conectados através de uma cadeia de conhecidos num máximo de seis passos.

O objetivo deste trabalho é implementar em Python o jogo "O Oráculo de Bacon", muito popular no final dos anos 90 e que ainda está disponível online. O jogo consiste em nomear um ator, X, como o "centro do universo" de Hollywood (por exemplo, o ator Kevin Bacon) e, de seguida, tentar encontrar a distância mínima entre o "centro do universo" e todos os restantes atores/atrizes em Hollywood.

A Fig.1 abaixo ilustra um exemplo do jogo tomando o ator Kevin Bacon como o "centro do universo". Neste caso, o "número de Bacon" é definido da seguinte forma: Kevin Bacon tem "número de Bacon" de 0; as pessoas que atuaram num filme com Kevin Bacon têm "número de Bacon" de 1; as pessoas que participaram num filme com alguém que tem um "número de Bacon" de 1 têm um Bacon Number de 2; e assim por diante. Finalmente, os atores que não tem uma cadeia de relação com Kevin Bacon têm "número de Bacon" infinito.

Dado um ator "centro do universo", X, e qualquer ator/atriz de Hollywood, o objetivo do jogo será determinar o seu "número de X" e a correspondente sequência de filmes e atores que conduzem ao "centro do universo", X, como ilustrado na Fig.1.

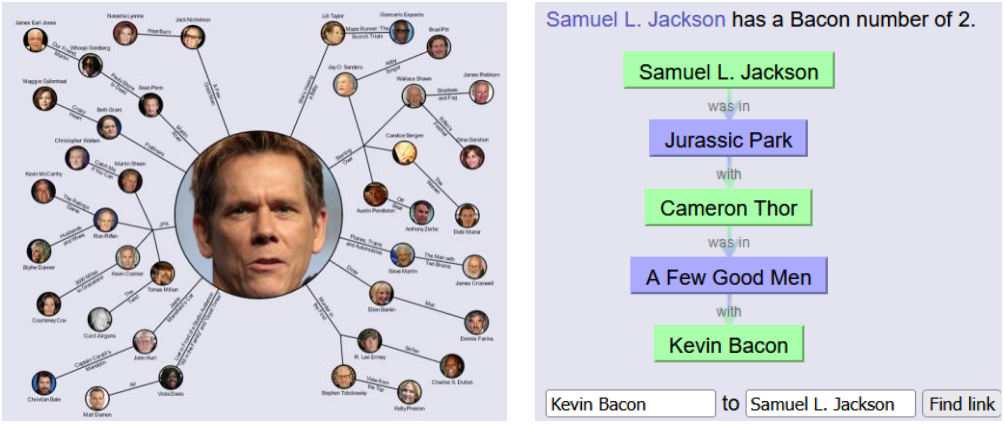

Fig. 1 Exemplos do jogo "The Oracle of Bacon".

Requisitos do trabalho

O grupo deve criar o código pedido e responder às questões seguintes num *Jupiter notebook* e esse ficheiro deve ser submetido no Moodle conforme as indicações apresentadas na secção "Submissão".

Nesta secção, apresenta-se um encadeamento de itens que permitem um desenvolvimento sustentado do trabalho. Portanto, a estrutura do relatório **deve** seguir a mesma estrutura apresentada abaixo:

1) Representação dos dados

A tarefa inicial consiste em ler e carregar os dados numa estrutura de dados adequada. Isto implica, essencialmente, em concetualizar a relação ator-filme como um grafo.

Para a sua implementação, deverá usar como base a classe `Graph`/`Digraph` fornecida nos exercícios da Semana 7, ou deve implementar a sua classe `Graph` de raiz. **Não é permitida** a utilização de outros módulos Python para a representação de grafos.

Com base no conjunto de operações que devem ser suportadas pela sua aplicação (apresentadas na Secção 2), descreva nesta secção os detalhes da estrutura de dados escolhida e o tipo de grafo resultante da sua interpretação do problema. Note que não existe apenas uma "única" representação possível, mas algumas

representações podem ser mais/menos eficientes e mais fáceis/difíceis de programar. Você, como desenvolvedor, deve analisar qual será a melhor solução para o problema, mantê-la consistente e fazê-la funcionar.

1.a) [2 valores] Descreva o modelo de grafo que utilizou para concetualizar a relação ator-filme do problema proposto. Liste quais foram os critérios que levaram à sua decisão e as possíveis vantagens e desvantagens da representação usada.

Nota: São listadas abaixo algumas perguntas que devem estar claras na sua resposta:

- *O que são os vértices e as arestas no seu modelo de grafo. Qual foi o critério para esta escolha? Por exemplo, a sua escolha facilita a implementação de alguma operação específica? Ou faz com que as operações fiquem mais eficientes (em relação ao tempo e ao espaço em memória)?*
- *A sua representação do problema resulta em que tipo de grafo (não orientado, orientado, pesado, com multiarestas, acíclico, cíclico, bipartido, etc)?*
- *Que tipo de modificações teve de realizar na classe* `Graph` *fornecida (teve de inserir novos atributos/métodos e porquê?); ou como implementou a sua classe* `Graph`*?*

1.b) [1,5 valores] Implemente uma função auxiliar que irá construir um grafo (do tipo `Graph` implementado) a partir dos dados de um ficheiro `.txt` fornecido. A função deve receber como argumento o nome do ficheiro e devolver um objeto do tipo `Graph`.

Nota 1: Cada linha do ficheiro fornecido apresenta o nome de um filme seguido de uma lista dos atores que participaram no filme. Para separar os nomes que aparecem numa linha, o caracter `'/'` é utilizado como delimitador.

```
Harry Potter and the Chamber of Secrets (2002)/Davis, Warwick (I)/Knight, Tom (I)/…
Harry Potter and the Goblet of Fire (2005)/Gambon, Michael/Claydon, Steve (II)/…
Harry Potter and the Prisoner of Azkaban (2004)/Phelps, James (I)/Murray, Devon/…
…
```

São anexadas duas bases de dados ao enunciado do projeto:

- `small_dataset.txt`: Contém a lista de filmes classificados no IMBD em 2011: 4.527 filmes e 122.406 atores (~3,5MB). Deve ser utilizada para realizar testes enquanto implementa a sua API.
- `large_dataset.txt`: Contém todos os filmes do IMBD em 2011: 285.460 filmes e 933.864 atores (~60MB). Deve ser usada para gerar os resultados dos testes definidos na Secção 3. Note que, para conseguir testar com esta base de dados, terá de ter disponível um espaço de memória RAM suficiente, uma vez que o grafo irá ser todo carregado para a memória.

Nota 2: Se achar mais fácil ter esta função implementada na classe `HollywoodOracle` (Secção 2), é livre de o fazer.

2) API HollywoodOracle

A sua aplicação terá de suportar as operações listadas na API `HollywoodOracle` apresentada abaixo.

2.a) [1,5 valores] Descreva a estrutura de dados que utilizou para implementar a sua API. Apresente uma análise informal do espaço em memória total utilizado pela sua aplicação. Mais especificamente, indique quanto espaço de memória os atributos da sua classe `HollywoodOracle` ocupam em relação ao número de filmes e atores/atrizes que existem na base de dados analisada.

Nota: Em relação ao espaço em memória, deve também apresentar uma estimativa do espaço real ocupado pela classe. Para isso, pode utilizar a função `getsizeof()` do módulo Python `sys`, e verificar o tamanho dos atributos do grafo construído e também de qualquer outro atributo que componha a sua implementação da classe `HollywoodOracle`.

2.b) [12 valores] Implemente cada uma das operações listadas na API `HollywoodOracle` abaixo. Para cada operação, deve apresentar uma análise informal da sua complexidade em relação ao tempo e ao espaço extra de memória utilizado (em relação ao número de filmes/atores).

Nota 1: Pode haver vários caminhos de distância mínima entre o "centro do universo" e um ator qualquer. Neste trabalho, será suficiente apresentar um destes caminhos.

Nota 2: Comece por testar cada um dos métodos com um ficheiro que tenha apenas 5-10 filmes antes de realizar testes na base de dados `small_dataset.txt`, para ter a certeza de que a sua resolução está correta. No relatório, apresente os resultados dos testes que foram realizados para cada método (exceto `all_movies()` e `all_actors()`) usando a base de dados `small_dataset.txt`.

`class` **`HollywoodOracle`**		Cotação (valores)
`__init__(filename)`	[construtor] Inicia os atributos definidos para a classe. Recebe o nome do ficheiro (filename) a partir do qual será criada a sua estrutura de grafo. Configura, por defeito, o "centro do universo" como o ator "*Bacon, Kevin*" e calcula o "número de Bacon" para cada ator existente na base de dados.	5,0
`all_movies()`	Devolve um iterável com todos os filmes na base de dados.	0,25
`all_actors()`	Devolve um iterável com todos os atores na base de dados.	0,25
`movies_from(a)`	Devolve um iterável que lista os filmes em que o ator dado (a) já atuou.	0,5
`cast_of(m)`	Devolve um iterável com os atores que atuaram num dado filme (m).	0,5
`set_center_of_universe(x)`	Configura o ator dado (x) como novo "centro do universo" e calcula novamente o "número de X" para cada ator existente na base de dados.	0, 25
`number_of_X(a)`	Devolve o "numero de X" do ator dado (a), considerando o "centro do universo" atual, X.	0,25
`path_to_X(a)`	Devolve a sequência de filmes e atores que conduzem ao ator "centro do universo", X (semelhante ao exemplo dado na Fig. 1). **Nota:** Caso devolva apenas a sequência de atores, a cotação sofrerá um desconto de 0,75 valor.	1,5
`max_number_of_X()`	Devolve: • O "número de X" máximo para os atores existentes na base de dados, excluindo todos os atores que não estão relacionados com o "centro do universo" atual (e cujo "número de X" é infinito). • O número de atores que não estão relacionados com o "centro do universo" atual.	1,0
`count_number_of_X(n)`	Devolve o número de atores/atrizes com "numero de X" igual ao valor inteiro dado (n).	1,0
`average_number_of_X()`	Devolve o valor médio do "número de X" para os atores existentes na base de dados, excluindo todos os atores que não estão relacionados com o "centro do universo" atual (e cujo "número de X" é infinito).	1,5

3) Testes

3.1) [1 valor] Usando a base de dados `large_dataset.txt`, implemente um script/função que utiliza a API `HollywoodOracle` e apresenta um histograma do número de atores com o mesmo "número de Bacon" (para todos os "números de Bacon" diferentes de infinito). Considere como o "centro do universo" o ator "*Bacon, Kevin*".

> **Nota 1:** Para além do gráfico, imprima os resultados encontrados, ou seja, imprima o número de atores com cada "número Bacon" possível (numa linha diferente).
>
> **Nota 2:** Imprima também "número de Bacon" médio encontrado (ou seja, o resultado da operação `average_number_of_X()`).

3.2) [1 valor] Usando a base de dados `large_dataset.txt`, implemente um script/função que utiliza a API `HollywoodOracle` e apresenta um gráfico do "número de X" médio considerando como o "centro do universo" cada um dos 20 atores mais populares de Hollywood nos anos 2000, segundo o IMBD. Ou seja, a função `average_number_of_X` deve ser executada para cada um dos 20 atores de modo a obter o seu "número médio de X". A lista de atores mais populares encontra-se no ficheiro `top20imbd.csv` anexado.

> **Nota 1:** Para além do gráfico, imprima os resultados, ou seja, imprima os valores do "número de X" médio de cada um dos 20 atores (numa linha diferente).
>
> **Nota 2:** Se, por qualquer razão, não for possível executar este teste na base de dados `large_dataset.txt`, deve justificar a razão do problema e, em alternativa, executar o teste utilizando o ficheiro `small_dataset.txt`. Só serão aceites justificações sustentadas por factos (por exemplo, capturas de ecrã do erro e/ou tempo de execução).

3.3) [1 valor] Se a teoria dos seis graus de separação for verdadeira para Hollywood, isso implicaria que a maioria dos atores terá um "número de X" de 6 ou menos, o que significa que o "número de X" médio seria menor do 6. Para testar esta teoria, implemente um script/função que utiliza a API `HollywoodOracle` com a base de dados `small_dataset.txt`, e apresenta um gráfico do "número de X" médio considerando como o "centro do universo" 1000 atores selecionados de forma aleatória da base de dados.

> **Nota:** Este teste pode demorar bastante tempo a ser concluído. Por esta razão, sugerimos que escreva os resultados num ficheiro para que seja possível recuperá-los caso a execução seja interrompida.

4) Questões Éticas

Tente resolver os problemas apenas com os integrantes do seu grupo antes de colaborar. Escreva as suas respostas por suas próprias palavras. Nunca deve partilhar o ficheiro fonte com as suas soluções com integrantes de outros grupos.

4.a) Se colaborou com alguém fora do seu grupo, indique aqui os respetivos nomes.

4.b) Deve citar todas as fontes que utilizou fora do material da UC.

Qualquer indício de plágio implica, automaticamente, na reprovação na avaliação periódica.

Consoante a gravidade dos indícios de plágio, estes serão apropriadamente investigados e, caso se confirme que um estudante cometeu plágio, este estará automaticamente reprovado à UC.

Submissão

O prazo para entrega deste trabalho é o **final do dia 12 de maio de 2023**. A entrega deve ser feita através do Moodle.

Deverá cumprir escrupulosamente as seguintes regras para a entrega:

- Deve entregar o relatório num *Jupiter notebook* com a mesma estrutura apresentada na secção "Requisitos de projeto". O ficheiro **.ipynb** deve conter o histórico da sua execução, incluindo os outputs gerados, de forma a compor o relatório.
- Deve garantir que é possível executar o relatório sem erros antes de submetê-lo.
- Deve submeter o ficheiro **.ipynb** do relatório ou um ficheiro **.zip** que inclua o relatório e quaisquer outros ficheiros necessários para a correta execução do Jupyter notebook. Não serão aceites outros formatos.

Nota: Os trabalhos que não cumprirem as regras acima só serão tratados e corrigidos após os integrantes do grupo estarem aprovados nas outras componentes de avaliação periódica (e, caso não fiquem aprovados, os trabalhos nem sequer serão vistos).