Projet N°1 : Liens entre acteurs [1]

1. Introduction

Écrivez un programme qui détermine combien de "*degrés de séparation*" séparent deux acteurs.

```
$ 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
```

A rendre le : **30/09/2020**

2. Contexte

Selon le jeu "*Six Degrees of Kevin Bacon*", n'importe qui dans l'industrie cinématographique hollywoodienne peut être connecté à Kevin Bacon en six étapes, où chaque étape consiste à trouver un film dans lequel deux acteurs ont tous deux joué.

Dans ce problème, nous cherchons à trouver le chemin le plus court entre deux acteurs en choisissant une séquence de films qui les relie. Par exemple, le chemin le plus court entre *Jennifer Lawrence* et *Tom Hanks* est 2 : *Jennifer Lawrence* est connectée à *Kevin Bacon* en jouant tous les deux dans "**X-Men : First Class**", et *Kevin Bacon* est connecté à *Tom Hanks* en jouant tous les deux dans "**Apollo 13**".

Nous pouvons considérer cela comme un problème de recherche : nos états sont des personnes. Nos actions sont des films, qui nous font passer d'un acteur à un autre (il est vrai que dans un film plusieurs acteurs différents peuvent participer, mais ce n'est pas un problème). Notre état initial et notre état d'objectif sont définis par les deux personnes que nous essayons de relier. En utilisant la recherche par largeur (Bread-first search), nous pouvons trouver le chemin le plus court d'un acteur à l'autre.

3. Dossier degrees.zip

Le code de la distribution contient deux ensembles de fichiers de données CSV : un ensemble dans le grand répertoire (**large**) et un ensemble dans le petit répertoire (**small**). Chacun contient des fichiers avec les mêmes noms et la même structure, mais ***small*** est un ensemble de données beaucoup plus petit pour faciliter les tests et l'expérimentation.

Chaque ensemble de données est constitué de trois fichiers CSV. Un fichier CSV est juste une façon d'organiser les données dans un format texte : chaque ligne correspond à une entrée de données, avec des virgules dans la ligne séparant les valeurs de cette entrée.

Ouvrez le fichier ***small/people.csv***. Vous verrez que chaque personne a un identifiant unique, correspondant à son identifiant dans la base de données **IMDb**. Elle a également un nom et une année de naissance.

[1] https://cs50.harvard.edu/ai/2020

Ensuite, ouvrez ***small/movies.csv***. Vous verrez ici que chaque film a également un identifiant unique, en plus d'un titre et de l'année de sortie du film.

Maintenant, ouvrez ***small/stars.csv***. Ce fichier établit une relation entre les personnes dans ***people.csv*** et les films dans ***movies.csv***. Chaque ligne est une paire de valeurs *person_id* et *movie_id*. La première ligne (en ignorant l'en-tête), par exemple, indique que la personne ayant l'id **102** a joué dans le film ayant l'id **104257**. Si vous comparez cette ligne avec celles de ***people.csv*** et de ***movies.csv***, vous verrez que cette ligne indique que *Kevin Bacon* a joué dans le film "*A Few Good Men*".

Ensuite, jetez un coup d'œil dans le fichier ***degrees.py***. Dans son entête, plusieurs structures de données sont définies pour stocker les informations des fichiers CSV. Le nom ***dictionary*** est un moyen de relier une personne par son nom : il associe les noms à un ensemble d'identifiants correspondants (car il est possible que plusieurs acteurs aient le même nom). Le dictionnaire **people** associe l'identifiant de chaque personne à un autre dictionnaire avec des valeurs pour le nom de la personne, son année de naissance et l'ensemble des films dans lesquels elle a joué. Le dictionnaire **movies** associe l'identifiant de chaque film à un autre dictionnaire avec des valeurs pour le titre du film, l'année de sortie et l'ensemble des stars du film. La fonction ***load_data*** charge les données des fichiers CSV dans ces structures de données.

La fonction principale de ce programme charge d'abord les données en mémoire (le répertoire à partir duquel les données sont chargées peut être spécifié par un argument de ligne de commande). Ensuite, la fonction invite l'utilisateur à saisir deux noms. La fonction ***person_id_for_name*** récupère l'identifiant de n'importe quelle personne (et invite l'utilisateur à clarifier, dans le cas où plusieurs personnes ont le même nom). La fonction appelle ensuite la fonction ***shortest_path*** pour calculer le chemin le plus court entre les deux personnes, et imprime le chemin.

La fonction ***shortest_path*** n'est cependant pas implémentée. C'est là que vous intervenez !

4. Travail à faire

Terminer l'implémentation de la fonction ***shortest_path*** de sorte qu'elle renvoie le chemin le plus court de la personne ayant l'id source à la personne ayant l'id cible.

- En supposant qu'il existe un chemin de la source à la cible, votre fonction doit renvoyer une liste, où chaque élément de la liste est la paire suivante (movie_id, person_id) dans le chemin de la source à la cible. Chaque paire doit être un tuple de deux entiers.
    - Par exemple, si la valeur de retour de ***shortest_path*** était [(1, 2), (3, 4)], cela signifierait que l'acteur source a joué dans le film 1 avec la personne 2, la personne 2 a joué dans le film 3 avec la personne 4, et la personne 4 est la cible.
- S'il existe plusieurs chemins de longueur minimale entre la source et la cible, votre fonction peut renvoyer n'importe lequel d'entre eux.
- S'il n'y a pas de chemin possible entre deux acteurs, votre fonction doit renvoyer Non.
- Vous pouvez utiliser la fonction ***neighbors_for_person***, qui accepte l'identifiant d'une personne comme entrée, et renvoie un ensemble de paires (movie_id, person_id) pour toutes les personnes qui ont joué dans un film avec une personne donnée.

Vous ne devez rien modifier d'autre dans le fichier que la fonction ***shortest_path***, bien que vous puissiez écrire des fonctions supplémentaires et/ou importer d'autres modules de la bibliothèque standard Python.

5. Indications

a) Alors que la mise en œuvre de la recherche telle que vue dans le cours vérifie si le nœud est l'objectif lorsqu'il est sorti de la frontière, vous pouvez améliorer l'efficacité de votre recherche en vérifiant si un nœud est l'objectif lorsque des nœuds sont ajoutés à la frontière : si vous détectez un nœud d'objectif, il n'est pas nécessaire de l'ajouter à la frontière, vous pouvez simplement renvoyer la solution immédiatement.
b) Vous pouvez emprunter et adapter n'importe quel code des exemples du cours. Nous vous avons déjà fourni un fichier ***util.py*** qui contient les implémentations pour **Node**, **StackFrontier** et **QueueFrontier**, que vous pouvez utiliser (et modifier si vous le souhaitez).