Write a program that determines how many “degrees of separation” apart two actors are.

Example:

$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

**Background**

To Find the degrees of separation between two actors:

* Find the shortest sequence of movies that connects two actors
* Count the number of Movies

E.g. To Find the degrees of separation between JL and TH:

* JL and KB both star in X-men
* KB and TH both star in Apollo 12
* Hence there are 2 degrees of separation

For This Search Problem

* States – The Actors
* Actions – The Movies
* Note that one movie can an agent from one actor to multiple actor

**Using Breadth First Search,** We can find the shortest path from one actor to another

**CSV Files – Can be ignored**

There are Two CSV Directories

* Large – For Actual Implementation
* Small – For Testing and Experimentation

In each directory there are 3 CSV Files

* people.csv – contains the id, name and birth year of an actor
* movies.csv – contains the id, title and year of release of a movie
* stars.csv – joins a person\_id to a movie­\_id. It establishes a relationship between the people in peoples.csv and the movies in movies.csv

**Code Files**

Data Structures

* names = {}
  + # A dictionary that maps a name to a set of corresponding person\_id (it is possible that multiple actors have the same name)
* people = {}
  + # A dictionary that maps a person\_id to a dictionary of the person’s: name, birth, movies (a set of movie\_ids)
* movies = {}
  + # A dictionary that maps movie\_ids to a dictionary of the movie’s: title, year, stars (a set of person\_ids)

**Functions**

* load\_data – loads data from the CSV files into names, people and movies
* person\_id\_for\_name - retrieves the id for any person (and handles prompting the user to clarify, in the event that multiple people have the same name)
* shortest\_path – not implemented
* main
  + Calls load\_data loads data (from a csv directory specified in command line) into memory (i.e. the data structures)
  + Asks the user to type in two names
  + calls person\_id\_for\_name to create a source id and target id
  + calls the shortest\_path function to compute the shortest path between the two people,
  + Prints out the path.

**Problem Specification**

* Complete shortest\_path such that it returns the shortest path from the person with id source to person with id target
* If there is no path, the function should return none
* If there is a path, the function should return a list
  + where each item is the next (movie\_id, person\_id) pair in the path form the source to the target.
  + Each pair should be a tuple of two strings
  + E.g. the return value [(1,2),(3,4)] highlights the source stared in movie 1 with person 2 and person 2 stared in movie 3 with person 4. Person 4 is the target
* If there are multiple paths of minimum length from the source to the target, the function can return any of them

**Tips and Rules**

* You can use util.py which contains the lecture implementation of Node, StackFrontier and QueueFrontier, which can be modified.
* You can call neighbors\_for\_person
  + Accepts the person id as an input
  + Returns a set of (movie\_id, person\_id) pairs for all people who starred in the movie with a given person.
* You cannot modify anything else inf the file other than the shortest path function
* You can write additional function or import other standard python libraries