Project 3 Small World - AKA 6 degrees of Kevin Bacon

In this project you will investigate the degree of separation of Holywood actors, also known as the Kevin Bacon game. As you may know, Kevin Bacon is a prolific actor who has appeared in many movies. We assign Kevin Bacon himself a Kevin-Bacon-number of 0. Any actor (except Kevin Bacon himself) who has starred in a movie with Kevin Bacon has a Kevin-Bacon-number of 1. Any remaining actor who has been in the same cast as an actor whose Kevin-Bacon-number is 1 has a Kevin-Bacon-number of 2, and so on.

For example, Meryl Streep has a Kevin-Bacon-number of 1 because she appeared in The River with Kevin Bacon. Nicole Kidman has a Kevin-Bacon-number of 2 because she did not play with Kevin Bacon in any movie, but she was in Cold Mountain with Donald Sutherland, and Sutherland appeared in Animal House with Kevin Bacon.

Check out the Wiki page on the six degrees of Kevin Bacon. And check out an online version of this game, The Oracle of Bacon. The goal of this project is, given the name of an actor, to find his/her Kevin-Bacon-number and the shortest alternating sequence of actor-movie pairs that lead to Kevin Bacon. And, to find the average distance of all actors to KB.

When it comes to Holywood, the idea is that even if every actor has a relatively small number of co-actors, there is a relatively short chain of movies/actors separating two actors from each other. If the theory of the six-degrees of separation is true for Holywood, it would imply that most actors will have a KB-number of 6 or less. That is, the average KB-number is < 6. By computing the average KB number of all actors, you will, therefore, test this theory.
Here are various lists of movies and the actors that you'll be using:

- BaconCast_06.txt: movies released in 2006 [movies=8780, actors=84236]
- BaconCast_00_06.txt: movies released since 2000 [movies=52195, actors=348497]
- BaconCastFull.txt: movies [movies=285462, actors=933874]
- ActionCast.txt: action movies [movies=14938, actors=139861]
- PopularCast.txt: popular movies [movies=4527, actors=122406]

Each line gives the name of a movie followed by the cast. Since names have spaces and commas in them, the / character is used as a delimiter.

```
'Breaker' Morant (1980)/Fitz-Gerald, Lewis/Steele, Rob (I)/Wilson, Frank (II)/Tingwell, Charles 'Bud'/Cassell, Alan (I)/Rodger, Ron/Knez, Bruno/Woodward, Edward/Cisse, Halifa/Quin, Don/Kiefel, Russell/Meagher, Ray/Procanin, Michael/Bernard, Hank/Gray, Ian (I)/Brown, Bryan (I)/Ball, Ray (I)/Mullinar, Rod/Donovan, Terence (I)/Ball, Vincent (I)/Pfitzner, John/Currer, Norman/Thompson, Jack (I)/Nicholls, Jon/Haywood, Chris (I)/Smith, Chris (I)/Mann, Trevor (I)/Henderson, Dick (II)/Lovett, Alan/Bell, Wayne (I)/Waters, John (III)/Osborn, Peter/Peterson, Ron/Cornish, Bridget/Horseman, Sylvia/Seidel, Nellie/West, Barbara/Radford, Elspeth/Reed, Maria/Erskine, Ria/Dick, Judy/Walton, Laurie (I)
'burbs, The (1989)/Gage, Kevin/Hahn, Archie/Feldman, Corey/Gordon, Gale/Drier, Moosie/Theodore, Brother/Katt, Nicky/Miller, Dick (I)/Hanks, Tom/Dern, Bruce/Turner, Arnold F./Howard, Rance/Ducommun, Rick/Danziger, Cory/Ajaye, Franklyn/Scott, Carey/Kramer, Jeffrey (I)/Olsen, Dana (I)/Gains, Courtney/Picardo, Robert/Hays, Gary/Davis, Sonny Carl/Gibson, Henry (I)/Jayne, Billy/Stevenson, Bill (I)/Katz, Phyllis/Vorgan, Gigi/Darbo, Patrika/Schaal, Wendy/French, Leigh/Fisher, Carrie/Benner, Brenda/Newman, Tracy (I)/Stewart, Lynne Marie/Haase, Heather (I)
...
```

Reading and representing the data:

Your first task will be to read and load the data in memory into an appropriate data structure. Essentially, you'll need to store:

- for each movie, all the actors in that movie
- for each actor, all the movies the actor played in

Actor-movie relationships as a graph: We can think of the actor-movie relationship as forming a *graph*: the vertices in the graph are all actors and movies, and the edges are the relationships between them. That is, there is an edge between every movie and every actor who played in that movie.

Computing Kevin-Bacon numbers.

More generally, computing Actor-A-numbers: Given two vertices in a graph, a path is a sequence of edges connecting them. There may be more than one path in a graph connecting two vertices. A *shortest path* is a path with minimum length among all paths between two vertices; here the *length* of a path is the number of edges on the path.

Let's phrase the KB problem in terms of paths in the graph: To find the Kevin-Bacon-number of an actor X, we need to find the shortest path connecting X to Kevin Bacon. Generally speaking, for an arbitrary actor A, we need to find the shortest path connecting X to A.

Your goal is to write a method that takes two actor names A and B, finds the A-number of B (that is, a shortest path from A to B) and displays nicely the movie-actor chain to A. Shortest paths are not necessarily unique; that is, there may be several paths of the same minimum length connecting A to X. In this case, we just want to compute one of them (does not matter which one).
It turns out that you can compute shortest paths in a graph using a strategy that you have seen while searching: breadth-first search (BFS). Start from the vertex representing the source (actor A); add all its neighbors to a queue. These are all the actors with an A-number of 1. Then add to the queue all neighbors of these neighbors, and so on. It is not hard to see (and we'll argue this in class) that using breadth-first search from A you find the shortest paths from A to all other vertices (that are connected to A).

Some things to think of:

1. How to represent a node in the queue while doing BFS. Well, it is a String representing a vertex in a MovieGraph. But you also need to keep track of the actual path of a queue node to the start vertex. At the end, you want to trace back the path to A. Hint: think of how we stored the path out of a maze.
2. How do you handle duplicate nodes in the queue: that is, you may want to enqueue a vertex that is already in the queue.
3. How do you keep track of the cost of a node in the queue to the start node? At the end, you need to print this distance, which is actually the A-number. Hint: you'll need to store the distance of each node.

Note that there is nothing special about Kevin Bacon, and that the same approach can be computed to compute shortest paths between any two actors in Holywood. It would be nice if you make your methods general enough, not customized for Kevin Bacon.

Efficiency: One thing to think about is efficiency. Some of the graphs are very large. Note that, to compute a path from A to B, you need to run BFS from A until reaching B. So, one way to compute the average path length from A for all actors is to run this process for each actor B. This is extremely inneficient, and you will not be able to use it on anything but the smallest graph. You want to think about running BFS from A until the end (until reaching all nodes that can be reached), and compute in this way all the paths from A in the same time.

What to turn in:

submit on canvas a python filed named *Project3_Bacon_yourname.py*

Final comments:

You need to understand that there is not one **"right"** way to do it. There are easier ways, and there are harder ways. There are more efficient ways, and less efficient ways. There are ways that will be easy to program, and there are ways that will take a lot of effort to make work. **YOU** are the creator of your world. Understand what it is that your world needs to do, decide how to model your world, keep it consistent, and make it work. Have fun!