Programming

Discipline:

Web programming

Programming language:

Python

Paper instructions:

Prerequisites

Before starting this programming assignment, participants should be able to:

Write object-oriented code in Python

Implement a graph ADT and common graph algorithms

Write Markdown and code cells in Jupyter Notebook

Create plots with matplotlib

Overview and Requirements

For this programming assignment, we are going to tackle the important social network problem of finding an actor's "Bacon number" Links to an external site.: starting with an actor, see if they have been in a movie with someone who has been in a movie with someone who has been in a movie ... who has been in a movie with Kevin Bacon Links to an external site.. They're usually at most 6 steps away.

There are plenty of other 6-degrees-of-separation phenomena in social networks. In a geekier version, the center of the universe is Paul Erdos Links to an external site., a prolific author and coauthor, and people are characterized by their Erdos numbers Links to an external site.. The highest known finite Erdos number is 13. Remarkably, there are a number of people who have both small Erdos numbers and small Bacon numbers (number = steps away):

Dan Kleitman Links to an external site.has total Erdos-Bacon number of 3 (Erdos 1, Bacon 2), but the Bacon number is due to a role as an extra.

Danica McKellar Links to an external site.has an Erdos-Bacon number of 6, and is both a professional actress (The Wonder Years and West Wing) and wrote a published math paper as well as supplemental math texts designed for teenage girls (Math Doesn't Suck, Kiss My Math, and Hot X: Algebra Exposed).

Note: for this assignment, code one Jupyter Notebook that tells the story of your data science endeavor. All auxiliary code (e.g. graph classes and functions) are to be in separate .py files and are imported into the notebook.

Program Details

In this problem you will write a program to play the Kevin Bacon game. The vertices in this network graph are actors and the edge relationship is "appeared together in a movie". The goal is to find the shortest path between two actors. Traditionally the goal is to find the shortest path to Kevin Bacon. The following output from the sample solution shows how the game is played:

To quit the program, type return in answer to a question.

Enter the name of an actor: Diane Keaton

Diane Keaton's number is 2

Diane Keaton appeared in Hanging Up (2000) with Meg Ryan

Meg Ryan appeared in In the Cut (2003) with Kevin Bacon

Enter the name of an actor: Buster Keaton

Buster Keaton's number is 5

Buster Keaton appeared in Limelight (1952) with Claire Bloom

Claire Bloom appeared in Haunting, The (1963) with Julie Harris

Julie Harris appeared in Requiem for a Heavyweight (1962) with Mickey Rooney

Mickey Rooney appeared in Erik the Viking (1989) with Tim Robbins

Tim Robbins appeared in Mystic River (2003) with Kevin Bacon

Enter the name of an actor:

So based on the data set we supply for this problem, Diane Keaton's Bacon Number is two, and Buster Keaton's Bacon Number is five.

Shortest Path Computation

The easiest way to play the Kevin Bacon game is to do what is called breadth-first search (BFS) in the movie data graph. This builds a tree of shortest paths from every actor who can reach Kevin Bacon back to Kevin Bacon. Or more generally, given a root BFS builds a shortest-path tree from every vertex that can reach the root back to the root. It is a tree where every vertex points to its parent, and the parent is the next vertex in a shortest path to the root.

Note: In class, we implemented a BFS to print out vertices in the graph. This BFS does not build a shortest-path tree. Later, we will implement Dijkstra's algorithm for a weighted graph. Dijkstra's algorithm does build a shortest-path tree. For this assignment, our graph structure is unweighted, so we can use either a BFS generated shortest-path tree or a modified version of Dijkstra's algorithm to generate the tree.

To implement BFS we use a queue. We also need a graph, which is to be represented using a Graph class (see the lesson notes for how to do this). The result of our BFS is the shortest-path tree described above.

The pseudocode describing BFS is:

insert root into an empty queue Q and into a new directed graph T

until Q is empty

dequeue Q to get next vertex V\_des to process

for each edge E that is incident to V\_des in G

let V\_src be the other end of the edge

if V\_src is not in T

add V\_src to T and add an edge with the same label as E from V\_src to V\_des in T

enqueue V\_src in Q

return T

When you are done, T holds a shortest-path or BFS tree. To find the Bacon number of an actor, look the actor up in T. If there is no vertex for that actor in T, then the actor is not connected to the root. If the actor is there, follow edges of T back to the root, printing movies (edge labels) and actors (vertices) along the way.

Dataset

Download bacon.zip Links to an external site.(thanks to Brad Miller at Luther College) and construct a graph from the datasets contained within the zip file. The three main files, actors.txt, movies.txt, and movie-actors.txt are large: 9,235 actors, 7,067 movies, and 21,370 movie-actor pairs, resulting in 32,337 edges.

Note: while you are developing your program use smaller versions actorsTest.txt, moviesTest.txt, movie-actorsTest.txt, whose data represent the graph:

vertices:

"Kevin Bacon", "actor1", "actor2", "actor3", "actor4", "actor5", "actor6"]

edges:

("Kevin Bacon", "actor1", "movie1")

("Kevin Bacon", "actor2", "movie1")

("actor1", "actor2", "movie1")

("actor1", "actor3", "movie2")

("actor3", "actor2", "movie3")

("actor3", "actor4", "movie4")

("actor5", "actor6", "movie5")

The files are all formatted the same way. Each line has two quantities separated by a "|". In the actors file the quantities are actorID and actorName. In the movies file they are movieID and movieName. In the movies-actors file they are movieID and actorID, indicating that the actor associated with actorID appeared in the movie associated with movieID.

Use the file contents to build a graph whose vertices are labeled with actor names (not IDs). Create an edge between two actors if they appeared in the same movie, and label that edge with the name of that movie. You should assume that no movie appears twice in the movies file and that no actor appears twice in the actors file. It is OK for there to be multiple edges between a pair of actors if they appeared together in multiple movies. You may find it useful to create maps (e.g. dictionaries) for mapping IDs to actor names and IDs to movie names. You can also use a map to figure out which actors appeared in each movie, and can use that information to add the appropriate edges to the graph. This may take a little thought, but try it by hand on the small data set given above.

Note: When opening these files, you may need to specify the encoding so Python doesn't crash: open(r"actors.txt", "r", encoding="latin-1")

Bacon Game

Implement the Bacon game. Perform BFS on the graph with "Kevin Bacon" as root and hold onto the BFS tree returned. Then ask for a series of actors. For each one, print out the path between that actor and the source, or say that none exists. This will require following a path from the chosen actor in the BFS tree back to the root (see above for an example of how this might be formatted). If the user gives a name that is not in the original graph (not the tree) say so and prompt again.

Test your program on the movieTest.txt, actorTest.txt, and movie-actorTest.txt files. Make sure to demonstrate that your program works for boundary conditions. When you are sure that your program works on the test data, change it to use the movie.txt, actor.txt, and movie-actor.txt files. Demonstrate that your program works for these as well.

Path Visualization

Using a python library for visualization (such as graphviz or networkx), visualize the path from the user-specified actor to Kevin Bacon. The vertices in the graph should be labeled with the actor's name and the edges should be labeled with the movie name. For example, a path from Owen Wilson to Kevin Bacon:

Grading Guidelines

This assignment is worth 100 points + 5 points bonus. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

5 pts for correctly reading in the data files

5 pts for correctly cross-indexing IDs to names

15 pts for correctly using Graph class and object to store the network

10 pts for correctly looping over queue to visit nodes in BFS

10 pts for correctly maintaining the shortest path tree in BFS

10 pts for correct text-based path trace from actor back to Kevin Bacon

15 pts for correctly visualizing the path trace from user-specified actor back to Kevin Bacon

10 pts for implementing an interactive interface (read names, print results)

10 pts for correctly computing at least two interesting statistics about the graph network

5 for correctly handling boundary cases

5 pts for adherence to proper programming style and comments established for the class