ECE241 PROJECT 2: Thinking in Graph

Due: Nov 27, 2018, 11PM on Moodle

Introduction:

We live in a world where a lot of things are connected, especially with the development of the Internet. Six degrees of separation is the idea that everything is six or fewer steps away from each other so that a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps. In order to understand the world better, we need to think in a graph way, everything is connected. In this project, we are going to model our Million Song Dataset in a new way, Graph.

In this assignment, you will still read in a subset of the Million Song Dataset from an Excel file. Beside the fields such as title, artist, song duration, track ID in the previous assignment, there is a new attribute of the collaborative artists, which have contributed to produce the songs. You can find interest connections when you analyze this information, such as a new collaborative artist to recommend.

Task Overview

In this project, you will perform the following specific tasks.

1. Manage your data in Python classes and objects. You should store songs in the **Song** object, manage songs in **SongLibrary** and build an artist graph in **ArtistConnections** class.
    a. For the **Song** class and the **SongLibrary** class, you can use the same one from project 1. (You can just include the basic structure, or you can add your own functions such as sorting or searching at your own convenience.)
    b. For the **ArtistConnections** class, you should build a graph based on the connections of the artists. You should have a **Vertex** class that store the basic information of an artist, which generally contains artist name, the songs he/she writes (an array) and the collaborative artists (an array) he/she has written songs together. (The information on the last column of the csv file is the collaborative artist list, they are separated using ';'. Take the first song for example:

```
0,Qing Yi Shi,Leon Lai,203.38893,5237536
,Nick Ingman;Tree;DJ Spinn and DJ Rashad;Ray Pillow;Soul Embraced;Green
Day;Erja Lyytinen
```

For artist “Leon Lai”, there are 7 artists who collaborated with him on the song “Qing Yi Shi”. So, when you make a new vertex of “Leon Lai”, you need to add those artists into his *coArtist* dictionary (key as the name, value as 1 since there is one song that they work together). If there is another song that have “Leon Lai” and “Nick Ingman” together, you should update the value in their *coArtist* dictionary as 2. **NOTE** that, you only need to connect “Leon Lai” and the 7 artists. There is no need to connect those 7 artists pairwise.

2. Build the artist graph from the database of songs using the *load_graph* method in **ArtistConnections** class. The method takes the input data file name and reads the artist information.
3. Implement a *search_artist* function to search for the basic information of an artist based on the artist name. Return a tuple (the number of songs he/she wrote, the collaborative artist list). For example, if “Leon Lai” wrote 10 songs and has collaborative artists [A, B, C], you can just use

```
return 10, ["A", "B", "C"]
```

4. In the class **ArtistConnections, i**mplement a *find_new_friends* function that returns a list of two-hop neighbors of a given artist. The two-hop neighbor means the artist that does not write songs with you but have written songs with your collaborative artists.
5. In the class **ArtistConnections, i**mplement a *recommend_new_collaborator* function to find the (one) most potential artist you will work with next. We quantify the potential here as the number of songs that the artist from two-hop neighbors (list from the previous task) have written with your collaborative artists, the more the better. You need to return the name of that artist and the total number of songs he has written with your collaborative artists. (If there are several artists with the same total number, just return one of them.)
6. In the class **ArtistConnections,** implement a *shortest_path* function to compute the shortest path (number of hops away) from a given artist to all the other artists. You should return a dictionary for all the vertices, where the key is the artist name and value is the path length.

Hints and suggestions

Successfully completing the project and achieving a good grade requires completing the project as described above and clearly commenting the code. As always, it makes sense to start the project early. Unless you are an amazing programmer, you probably won’t be able to finish in one day. Build your project code step by step. For example, verify that you have successfully read in the database before attempting a BFS or DFS search. Then, make sure the search works before writing and testing code for *find_new_friends* or *shortest_path*, etc.

What to submit:

For Task 1-6, you should submit your code to Gradescope for auto-grading. Remember to comment your code properly.

Reminder: The course honesty policy requires you to write all code yourself, except for the code that we give to you. Your submitted code will be compared with all other submitted code for the course to identify similarities. Note that our checking program is not confused by changed variable or method names.

Grading:

- Code works on Gradescope (90%)
- Program structure and comments (10%)