CS 5010: Programming Design Paradigms

Fall 2017

HW 6 – Data Collections, Graphs, Social Networks and Recommendation Systems

Assigned: Tuesday, October 31, 2017, Due: Monday, November 13, 2017 by 6pm

College of Computer and Information Science
Northeastern University – Seattle

Team Work

With this assignment, we are switching to a team work. Both (all) team members are expected to equally contribute to the assignments, but it is up to each team to divide the work amongst team members. Please keep in mind that **all team members will get the same grade for the assignment**.

Assignment Goals

The purpose of this assignment is to give you an additional practice with **graphs**, their representations, and operations. You will explore:

- Connected and complete graphs,
- Subgraphs and cliques,
- Use of graphs in social networks, and
- Recommendation systems on social networks.

Background

In a decade or less, social networks have become a significant part of our lives, and today we use them for a variety of things - for example, as a professional point of contact, as a way to connect with friends and family, as a way to share our opinion about movies, restaurants, and places to visit, but also as a way to share our photos, code and other contributions.

For many end users, social networks are attractive because they make it easier to connect with anyone anywhere in the world, and share thoughts, ideas or contributions. And if, by any chance, those posts, tweets, updates, grams, snaps and gits go viral - then even better!

For many social network owners and researchers, social networks are attractive because they give an unprecedented, (almost) real time, access to thoughts, sentiment, and preferences (as well as other information) - data that traditionally either did not exist, was very hard to access, or was considered private and/or sensitive, and was therefore closely guarded.

But as many other things, social networks are sensitive to trends, and are only useful if they are "popular". Some simple ways to measure "popularity" of a social network are through the **number of active monthly users**, and the **total number of users**. As long as a network is growing, in the number of members and their activity, it can be considered popular.

One important contributing factor to the growth of every social network is its ability to recommend people and things that might be of interest to current users. For example, every time you use a social network such as Facebook, Twitter or LinkedIn, it will recommend other people that you may want to befriend, follow or connect with. Additionally, it may even send you a daily list of recommendation via an email.

Your Task

Your task in this assignment is to build a simple recommendation system for the Twitter social network. You will create a program **RecommendationSystem** that takes as input three required parameters: **file1.csv**,

file2.csv and **file3.csv**, and up to three additional parameters, **processingFlag**, **numberOfUsersToProcess**, and **numberOfRecommendations**. Additional information about these arguments is provided in the next section.

Based on the data stored in files `file1.csv` and `file2.csv`, and the processing options, defined as additional input parameters, your program then recommends a certain number of new people that an individual node may want to start following. In making such **"Who-to-Follow"** recommendations, your program should rely on four ordered recommendation criteria, **1. Newbies mimic a friendliest friend**, **Friend of a friend is a friend**, **Always follow the influencer**, and **When in doubt, branch out**. Additional information about each of the recommendation criteria are provided below.

Finally, your program should store the computed recommendations in file `file3.csv`, and provide meta-information on the standard output (console) as described below.

Implementation Details

One of the goals of this assignment is to expose you to graph theory, different representations of graphs, and different operations on them. We will do so by applying graph theory to social networks, and representing a Twitter social network as a **directed graph**, in which every user represents a node in a graph, and there exist some edge $\{x, y\}$ is user x follows user y on Twitter.

As usual, different approaches may exist, but we are looking for graph theoretic approaches that are not NP-hard, or NP-complete. Should it be the case that some of your approaches are NP-hard/NP-complete, you will want to explain in your write up why is the proposed approach the only possible approach, and if there exists some suitable heuristics.

Processing Input Arguments: Your program, `RecommendationSystem`, is expected to always take three input arguments:

1. **file1.csv** - a String, denoting the name of a CSV file containing information about the users (nodes) in the considered Twitter network.
2. **file2.csv** - a String, denoting the name of a CSV file containing information about the edges, where an edge $\{x, y\}$ denotes that user x follows user y on Twitter.
3. **file3.csv** - a String, denoting the name of the CSV file used to store the output data.

The order of required arguments should be as defined above, and if one of the arguments is missing, the program should terminate with an appropriate message. In addition to the required parameters, your program should take up to three input parameters, defined as follows:

1. **processingFlag** - a character from the set {s, e, r}, where `s` represents the flag to process users from the beginning, `e` a flag to process users from the end, and `r` a flag to process users in some random order.
2. **numberOfUsersToProcess** - an integer in the set [1, TOTAL NUMBER OF USERS], defining the number of users for whom the program should generate recommendations. The default value for this parameter is 100, and a constant TOTAL NUMBER OF USERS is a function of the dataset being processed.
3. **numberOfRecommendations** - an integer in the set [1, 100], defining the number of recommendations made for a single user. The default value is 15.

The Twitter Data Set: In this assignment, we will use a modified version of the original Twitter data set, the **Arizona State University Twitter Data Set**, that was made available in 2009, by R. Zafarani and H. Liu, the members of the Arizona State University, School of Computing, Informatics and Decision System Engineering. The original dataset can be accessed here, and it consists of 11316811 nodes (users), and 85331846 edges (friends/followers relationships).
The original data in the data set is organized in two files, and in this assignment, we will follow the same convention:

1. **nodes.csv** - representing the file of all the users. This file works as a dictionary of all the users in this data set, and in the original data set, it contains the node IDs of all the users in the data set. In our modified version, in addition to node IDs, it also contains:

  - **The date of profile creation** in the format MM/DD/YY
  - **Gender**, represented as one of the characters `M`, `F` or `O`, denoting respectively **male, female** and **other**.
  - **Age**, represented as an integer in the range [0, 100].
  - **City**, represented as a String.
2. **edges.csv** - representing friendship/followership network among the users, where a follower/friend is represented as a directed edge.

For convenience, we have provided two data sets:

- **nodes_small.csv, edges_small.csv**, consisting of 100 users, and their corresponding followership network, and
- **nodes_10000.csv, edges_10000.csv**, consisting of 10000 users, and their corresponding followership network.

You may want to use a smaller network during the development and debugging phase, but your program should finish in a reasonable amount of time for the larger network, as well as for the original Twitter data set.

Recommendation System: Social networks recommendation systems are an extremely active area of research and development, and many sophisticated method to make a personalized recommendation exist, and/or are being developed. In this assignment, our goal is to implement one very simple recommendation system that consists of four **ordered** criteria:

1. **Newbies mimic a friendliest friend**
2. **Friend of a friend is a friend**
3. **Always follow the influencer**
4. **When in doubt, branch out**

Let's examine each of these criteria in detail.

Newbies Mimic a Friendliest Friend: If a node has only been a member of a network for a month or less, then such a node is likely still actively building their network. One quick way to grow a network is to mimic a behavior of the most active friend, and start following the same people that node is following.

So, under this criterion, if some user A has been a member of Twitter for a month or less, your recommendation system should find that user's friend (a node that the user follows) with the largest number of friends (nodes that that nodes follows), say some node B. It should recommend as friends of A all of the friends of B, that are not already friends of A.

If node A does not have any friends yet (is not following any nodes yet), or the system can not make a sufficient number of recommendations, then the recommendation system should move on to other criteria.

Friend of a Friend is a Friend: A very popular way to build some social network is to connect with friends of the existing friends, as there is a higher likelihood that we already know those people, and have interests in common with them.

So, under this criterion, when making recommendations for some user A, your system should consider all of the friends of A (the nodes that A is already following), and it should find their friends, say some nodes in set $\mathcal{B}$. It should then recommend as friends of A those nodes from $\mathcal{B}$ that are not already friends of A.

If node A does not have any friends, or the system cannot make sufficient number of recommendations, your program should proceed to the next criterion. If, on the other hand, there are too many possible recommendations, then your program should release the required number of them in the ascending order, starting from the candidate node with the smallest node ID.

Always Follow the Influencer: Another popular way to grow a network is to follow the so-called **influencers**, the users (nodes) that have managed to attract a large number of friends (followers) through their social network activity.

So, under this criterion, when making recommendations for some user A, your system should take a global view of the network, and recommend as friends those nodes that have more than 25 followers for the small data set (`nodes_small.csv, edges_smallcsv`), and more than 250 followers for the regular data set (`nodes_10000.csv, edges_10000.csv`).

If the system cannot make sufficient number of recommendations, your program should proceed to the next criterion. On the other hand, if there are more potential candidates then the required number of recommendations, your program should again release the required number of them in the ascending order, starting from the candidate node with the smallest node ID.

When in Doubt, Branch Out: Last criterion for our recommendation system does not really care about security, privacy and personalization. Under this criterion, your system is expected to grow a social network of some user A by randomly proposing some nodes from the network as potential friends, until the required number of recommendations for a user is reached.

Generating the Output File: Your program should store the generated recommendations into the third provided CSV file, `file3.csv` (if the file does not exists, it should be created). The CSV file should contain header:

1. **Node ID** - representing the ID of a currently processed node
2. **Recommended nodes** - representing a **space-separated** list of node IDs to follow

For every processed Twitter user, the information indicated in the header should be provided as a row in the file. Additionally, in the attempt to anticipate "up and coming influencers", your program should **print out on the console** top ten most frequently recommended node IDs during the program execution.

Bonus Part - 2 points

Using the same Twitter data set, propose and implement one or more of your own "Who-to-Follow" recommendation criteria. In doing so, please use the writeup to explain the idea behind every recommendation criterion that you propose. As always, you are allowed to reuse all of the objects and interfaces that you wrote for the original approach.
(*Note: Once again, the possible bonus points do not reflect the time and effort that may be required to implement the bonus problem. Please focus on the the required part of the assignment first, and attempt the bonus part only after your have fulfilled all of the components of the required part.)*

What To Submit?

When submitting your assignment, please use the CCIS GitHub repo that you have listed as your team's repo. You should continue using the same Maven archetype that we used in previous assignments. For this assignment, you will want to submit the following:

1. Class `RecommendationSystem`. That class will be assumed to be the starting point of your program, and will be called from the command line with input arguments.
2. All classes that you have developed for this assignment.
3. All abstract classes and interfaces that you extended and implemented in this assignment.
4. All classes that you have developed to test your code.
5. A UML diagram, corresponding to the design of your program.
6. A brief write-up, which summarizes the main relations between your classes, and how does your program handle errors and/or exceptions. Additionally, in this assignment, you will want to comment on how are you encoding and accessing the information about the social network, and the pros and cons of your approach. You will also want to comment on the time complexity of your approach, especially if your approach is (or likely is) NP-complete.
7. If you attempted the bonus question, please include class `RecommendationSystemBonus`, all of the classes that it uses, and the new UML that corresponds to this program. Please also update your write-up, to explain your recommendation criterion, and comment on this pros and cons.