Assignment 6 Writeup

The goal of this assignment is to implement a recommendation system that recommends friends for users in a social network. The users (nodes) information is provided in a csv file (nodes_small.csv or nodes_10000.csv), and the connections (edges) information is provided in another csv file (edges_small.csv or edges_10000.csv).

The RecommendationSystem class reads in the command-line arguments and constructs a CmdHandler to handle the arguments. The CmdHandler validates the format of arguments and returns the information using getters. If the arguments are invalid, an InvalidInputException will be thrown with an error message; otherwise, the CmdHandler object will be passed to the constructor of RecommendationSystem to starts the recommendation and outputs results at the end.

Here is an example of input arguments:

```
nodes_small.csv edges_small.csv output.csv
--processing-flag e --number-of-users-to-process 50 --number-of-recommendations 10
```

where the arguments on the second line are optional and can be skipped.

Here are the steps that a recommendation system takes to complete the recommendation:

1. Builds a network based on the info in nodes file and edges file (see below for details)
2. Picks the requested number of users from the network following a processing flag (picks users from the lowest ID or highest ID or picks randomly)
3. For each user selected, recommends friends for that user until the number of recommendations reaches the requested number. There are four criteria of friend recommendation (see assignment 6). The recommendation will start from the first criterion and move on to the next criterion if not enough friends have been recommended.
4. Outputs the results to the desired output file
5. Prints to console the top ten most frequently recommended user IDs

To build a network based on the csv files, a few steps are taken:

(a) Converts the csv file to a list of strings, each string represents a line in the file
(b) Uses a parser (CsvParser) to parses each line to a list of string, now the list represents a line
(c) If the original csv file is a nodes file, uses a UserGenerator to generate a user (node) given the list of strings from CsvParser. If the original csv file is a edges file, then uses a EdgeGenerator to generate an edge given the list of strings from the CsvParser.
(d) Adds users and edges to the Network

Inside the Network class, we use a map of Map<Integer, IUser> to store the users in this network, where the integer is for the user ID. We use another map (Map<Integer, Set<Integer>>) to store the connections between users, where the key is the user ID, and the value is the set of friend IDs of that user. The Network class provides a getter to return all users in this network, which will be used for RecommendationSystem to select users at the beginning. The Network also provides a getter to get friend IDs of a certain user, which will be used during the recommendation process.

When the recommendation system selects users to process, if the processing flag is ‘s’ or ‘e’, that means the program should select users from the lowest IDs or highest IDs, thus sorting of IDs is required, which will take O(NlgN) time. If the processing flag is ‘r’ (random), then the time complexity of selecting users is O(N) in the worst case.

We have two new criteria in RecommendationSystemBonus. The first one is to choose users in the same city and similar age (between some range, such as 3 years). The second one is to recommend old users, which means we can sort all users based on their created date and recommend to the user. These two criteria are not based on the user's friends, so they will not need to load all data from connectionsMap, instead, just go through all users in the network.