Lab 4 – A More Complex Problem

(a.k.a. Putting Everything Together)

Release: 10 Feb 2020 (Mon, Week 6)
Due: 16 Feb 2020, 11pm (Sun, Week 7)

Some Words:

This lab is interesting: the "Quality Score" of your solution is significant for this lab. You will need to be familiar with lists, loops and conditionals in order to complete this lab. There will be many "correct" solutions to this exercise but some correct solutions will be of higher quality than the others. Feel free to compare solutions with your classmates.

Instructions:

- There is 1 question in this exercise to be completed individually.
- For this exercise, your team ID is your name (i.e. you are the only member in your team).
- You need to submit code for this exercise at the Submission Server. No written submission is required.
- Edit **lab4.py** that is given to you and submit it to the Submission Server.
- You can submit your solutions to the Submission Server as many times as you wish, but the final submission on the deadline will be taken as your final submission.

Lab 4

Twitter is a social media platform for users to send out a short message (a "tweet") to other users who are registered as her followers. Figure 1 shows a graph. ("Graphs" will be covered after your term break, and you are not required to understand graphs for this lab; it's a very useful illustration for this scenario nonetheless).

The circles labeled "0", "1", "2" etc. are called nodes (or vertices) of the graph, and the arrows connecting the nodes are called edges. This graph shows a small network of Twitter users with the nodes representing users, and the edges show which user is following which other users: ***an arrow means "is following"*** (and not "is followed by"). The graph in figure 1 can hence be interpreted as a network of 11 Twitter users (users 0, 1, 2… 10). Users 0 and 3 are following 1, 1 is following 2, 3 and 4, 2 is following 0 and 3, and so on. By studying this simple graph, it should be obvious that user 3 has the most number of followers (four followers), and is hence the most influential user since a tweet initiated by user 3 will be received by four users.

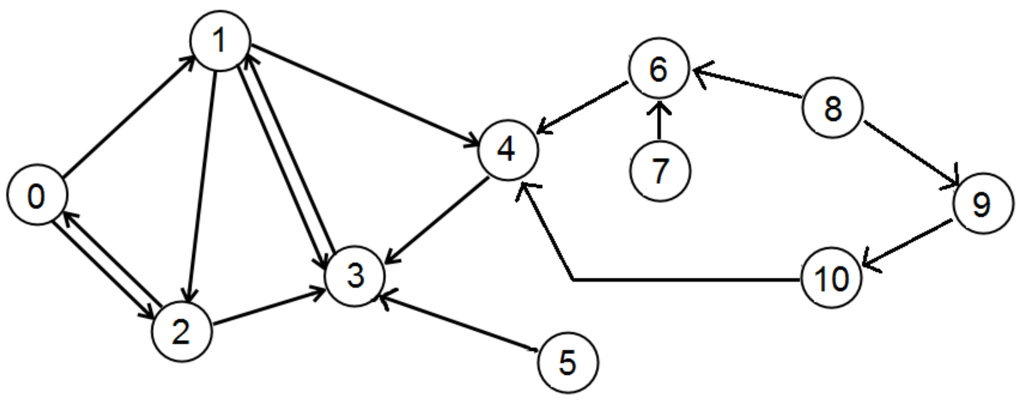


Figure 1. A graph showing a simple Twitter network with 11 users

We would like to start a viral marketing campaign. The approach is to select five Twitter users, and pay them to tweet an advertisement. The problem is to identify which five users to select so that we will reach as many unique followers as possible.

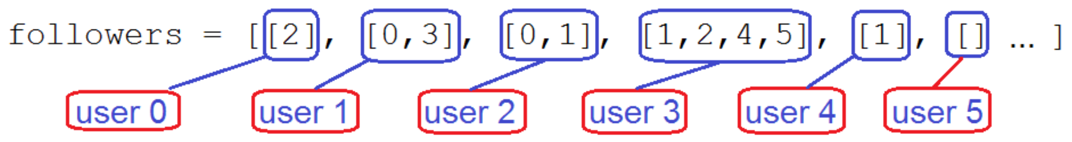
For simplicity, assume that Twitter user IDs are sequential numbers starting from 0. So if there are 11 users, their user IDs will be 0, 1, 2… 10. The Twitter graph above can be represented in the form of a table like this:

User ID	Direct Follower User IDs
0	2
1	0, 3
2	0, 1
3	1, 2, 4, 5
4	1, 6, 10
5	
6	7, 8
7	
8	
9	8
10	9

In Python, this table can be represented as a 2D list like this:

```
followers = [[2], [0,3], [0,1], [1,2,4,5], [1,6,10], [],
[7,8], [], [], [8], [9]]
```

Each element in this list consists of a list of her follower IDs:



```
followers = [[2], [0,3], [0,1], [1,2,4,5], [1], [] … ]
```

If a user has no follower (e.g. users 5, 7 and 8), the list at that respective position (i.e., positions 5, 7 and 8) will be an empty list (i.e. []).

You are given the following files for this exercise:

File name	Description	Comments
lab4.py	Contains the **select_tweeters** function that you will write.	You need to modify and submit this file. This is the only file that you may submit. Do not modify the file name or the function signature in the file.
lab4_main.py	Loads **lab4.py** and calls the **select_tweeters** function using the 4 test cases below.	Do not submit this file; use it to check if your **select_tweeters** function in **lab4.py** works before submitting it to the Submission Server.
lab4_utility.py	Contains function(s) used by **lab4_main.py**.	Do not submit this file. It needs to be in the same folder as **lab4_main.py**. You may want to browse through the contents, but it is not important to understand what the code in this file does.
data folder containing CSV files	Contains text files read by **lab4_main.py**. The 2D list of user IDs used in the test cases are read from CSV files in this folder.	Do not submit these files. **case1.csv** contains the Twitter network shown in figure 1. The other CSV files contain more complex networks of 500 users each. Do take a look at **case2-5.csv** using either Microsoft Excel or a text editor. The test cases on the Submission Server will use data of similar characteristics.

Your task:

- Write a function called **select_tweeters** that takes in 1 parameter:
  - **followers** (a 2D list of Twitter users)

  and returns the user IDs of five Twitters (as an list of integers) selected by your algorithm who will be selected to broadcast the advertisement.
- The objective of **select_tweeters** is to return the IDs of five selected users who will tweet the advertisement, so that as many other users get the tweet as possible.
- Edit **lab4.py** provided to meet the requirement above.
  You are provided with a few CSV files in the data folder that you can use to evaluate your function in **lab4_main.py**. When **lab4_main.py** runs, you will be prompted to enter the file name of one of the CSV files. The Submission Server may use a different data set of similar characteristics as those provided in your **data** folder.

Your solution's quality will be calculated by the following function: **get_unique_followers**. It takes in an list of five user IDs (returned by **select_tweeters**) and returns a sorted list of unique users who follow the five selected Twitters.

Example

Assuming that the **select_tweeters** function has been written to always return the first five users (0, 1, 2, 3 and 4), here's what will happen:

```
>>> followers = [[2], [0,3], [0,1], [1,2,4,5], [1,6,10], [],
[7,8], [], [], [8], [9]]
```

```
>>> s = select_tweeters(followers)
[0, 1, 2, 3, 4]

>>> unique_followers = get_unique_followers(s)
[5, 6, 10]

>>> len(unique_followers)
3
```

In this example, arbitrarily choosing the first five users is clearly a poor choice, because if an advertisement is tweeted by these five users, only three other users (users 5, 6 and 10) will get the message.

In another case, if you improve the **select_tweeters** function so that it returns users 1, 3, 4, 6 and 10:

```
>>> s = select_tweeters(followers)
[1, 3, 4, 6, 10]

>>> unique_followers = get_unique_followers(s)
[0, 2, 5, 7, 8, 9]

>>> len(unique_followers)
6
```

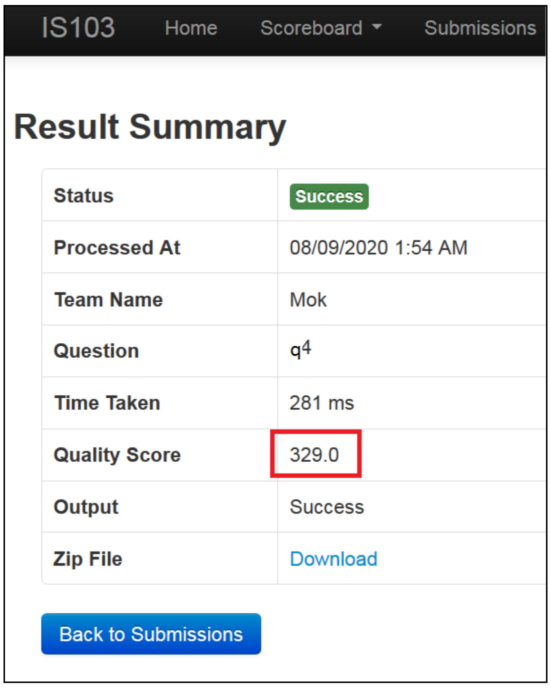
In this instance, six other users will receive the tweet if it is broadcast by users 1, 3, 4, 6 and 10. This solution turns out to be the best for this case, since all eleven users in the network will either be one of the selected Twitters, or get the message from one of them.

To submit:

- **lab4.py** (at submission server). Edit the comments at the top of your Ruby file to indicate your name and section.

Assessment:

- This exercise is not graded but submission of a working answer is mandatory.
- It is trivial to arrive at a "correct" solution for this exercise – a "correct" solution is any solution that returns a list of five user IDs (integers). What matters for this exercise is the "Quality Score".
- The "Quality Score" is the number of unique users who will be reached by the five Twitters selected by your algorithm for the data set on the Server. It follows that the higher your quality score, the better is your algorithm. For example, the following screenshot shows a quality score of 329 for the five selected users that my **select_tweeters** function returned:



Result Summary

Status	Success
Processed At	08/09/2020 1:54 AM
Team Name	Mok
Question	q4
Time Taken	281 ms
Quality Score	329.0
Output	Success
Zip File	Download

- You may ignore the "Time Taken" value for this lab, but your submitted code must complete running **within 1 minute** on the server. If your **select_tweeters** function takes more than 1 minute to return, your submission will be marked as "Failed".

Appendix A: Best solutions for test cases provided in data folder

Remember that your solution is NOT expected to return the optimal solution, but a good algorithm will return a near-optimal solution. In other words, it is definitely OK if your algorithm does not result in the "real best solutions" below.

The real best solutions for case1-5.csv:

case1.csv: 11
case2.csv: 337
case3.csv: 342
case4.csv: 337
case5.csv: 336

Examples of selected users that will produce the best solutions (there could be more than one "best solution" for each test case):

case1.csv [0,3,4,6,9]
case2.csv [53,146,170,262,486]
case3.csv [22,41,225,256,314]
case4.csv [79,81,104,117,154]
case5.csv [61,250,259,312,476]

~End