Mini-Project

Announcement Date: July 4, 2020
Due: Towards the end of the semester. Exact dates to be announced.

Note: The mini-project includes designing and implementing algorithms for graphs. It is organized into tasks. The description of some of these tasks intentionally lacks too many details: part of what you have to do is figure out what is the appropriate approach, algorithm and data structure, for each task. You can do that in interaction with me, or your classmates. But you will be solely responsible for your code.

Deliverable: You must deliver your code via Colab link, or repl.it if you prefer a language other than Python. The code should contain instructions on how to run it. Besides the code, you are expected to also deliver a short writeup that provides documentation for your code, including a discussion of your algorithmic choices. During the next weeks I will elaborate more on the deliverables, when needed.

Evaluation: Part of the evaluation of the mini-project will possibly be a short and informal meeting with me, during which you will show me your work. You should be ready to answer questions about it!

Milestones: To help you with planning, we will have a number of milestones. These milestones will identify sub-problems and tasks that will be eventually composed to solve the bigger problems. I will not be checking your progress on these individual milestones, as they are only intended to keep you in pace. The mini-project document will be continuously updated, so please check back for updates.

Tasks in Social Network Analysis

1. The Data

Social networks are represented as graphs. A Facebook-like network where friendship is mutual is represented as an undirected graph. A Twitter-like network where following is a one-way relationship is represented as a directed graph.

There exist many open data sets coming from social networks. Here is a good source:
https://snap.stanford.edu/data/

Task 1. Spend some time reviewing the data sets in the above link. Pick 3 of them, that look interesting to you. One of them must be relatively small (at most 5K nodes), and one of them must be a directed graph. Try to understand their format: these are text files, but how are these text files organized in order to signify nodes and edges?

Task 2. Find or write a small interface that you can use to import these data sets into your language environment. The graph must be encoded as a list of edges, where each edge is a pair of nodes.

Milestone 1. Start your writeup with a discussion of the characteristics of the data sets you picked, i.e. size, where they come from etc. (5 days)

2. Finding Influencers.

Task 3. One interesting aspect of social networks is the existence of 'influencers'. In a graph these are the nodes with high in-degree. For this task you will have to write a function that finds the 100 of the nodes with highest in-degree and report them, along with the number of their followers.

3. Cleaning-up the network.

In a social network people and bots keep opening fake accounts. Such accounts are often 'isolated': they are disconnected from the bulk of the network. In graph terms, they may be either non-connected nodes, or small groups of nodes that have some connections to each other but they are not connected to the 'main' part of the network.

Task 4. Write code that will find and report these smaller components of the network. More specifically, you should write code that reports the (simple) connected components of a graph. The output must be an array C with n entries, where n is the number of the nodes in the graph. Each connected component should have an id number. Then, $C[i]$ must be equal to the ID number of the connected component that contains node i. Your code should also output the list of edges comprising the biggest connected component. Note: You should treat the graph as undirected in this part.

Milestone 2. Complete Tasks 3-4. (10 days.)

4. Closed Subnetworks.

In a Twitter-like social network there are subsets of people forming 'closed subnetworks'. Such people do not follow anyone else outside their subnetwork, and only follow other people within their subnetwork (but not all of them). People outside that subnetwork do follow people that are part of the subnetwork. For example, famous people on Twitter tend to not follow too many other people, except other famous people. So, they may be forming subnetworks of famous people.

Task 5. Write code that finds and reports all closed subnetworks of a Twitter-like network. More specifically, write code that computes the decomposition of the network into strongly connected components. The output must be an array S with n entries, where n is the number of the nodes in the graph. Each strongly connected component should have an id number. Then, $S[i]$ must be equal to the ID number of the strongly connected component that contains node i.

4. A recommendation engine.

Three persons P_1, P_2, P_3 form a length-3 following chain if a person P_1 follows person P_2, and person P_2 follows P_3. Given a length-3 following chain, it may make sense to recommend to P_1 to follow person P_3. It may make even more sense doing that if there are several 3-following chains between P_1 and P_3.

Task 6. Write code that takes as input two persons P and P' and a number k, finds the **number** of length-k following chains that connect P to P'.

Milestone 3. Complete tasks 5-6. (10 days)

Graidng Notes: Milestone 1 and 2 should be completed by all. We have already essentially done these tasks. For milestone 3, students in the alternative track can use Python package `networkx`. This should be relatively easier, as the only thing required is to read the manual of some ready-made functions, and apply them on your data.

For other students, task 5 requires writing code for finding a DFS of a graph. If you want to avoid that, you can also use `networkxx` for a deduction of 2/20 points. But the rest of the implementation should be yours.