Social Network Analysis for Computer Scientists
Fall 2022 — Assignment 1

`https://liacs.leidenuniv.nl/~takesfw/SNACS`

Deadline: October 3, 2022

This document contains two exercises that each consist of various numbered questions that together form Assignment 1 of the Social Network Analysis for Computer Scientists course taught at Leiden University.

For each question, the number of points awarded for a 100% correct answer is listed between parentheses. In total, you can obtain 100 points and 10 bonus points. Your assignment grade is computed by dividing your number of points by 10. Please do not be late with handing in your work. You have to hand in the solutions to these exercises **individually**. Discussing the harder questions with fellow students is allowed, but writing down identical solutions is not.

Clearly and concisely describe how you obtained each answer. Write down any nontrivial assumptions that you make. For the exercises that require programming, you can use any programming language, scripting language or toolkit. In any case, always clearly describe which tools and languages you used and how you obtained your answer using these tools. When asked for an algorithm, use simple and consistent pseudo-code. Include relevant source code, for example, in an Appendix that you reference in your answers. Please use the `listings` package for including source code. If you write code in an interative notebook, you can use `nbconvert` to covert the notebook to regular source code, i.e., a `script`.

Submission. Hand in your solutions via Brightspace, in one `.pdf` file, typeset using LaTeX. Remember to specify your name and student ID (ULCN number) on top of your assignment. Do not copy the full text of each question into your document. Just stating the question number and then your answer, is sufficient. If you really need to submit multiple files, please attach them all in one submission. If you want to make a new submission, replacing your previous submission, make sure to again include all the files in that submission. Thank you for taking this into consideration.

Questions or remarks? Ask your questions during one of the weekly lectures or lab sessions, or send an e-mail. Good luck!

Exercise 1: Neighborhoods (40p)

A directed network $G = (V, E)$ consists of a set of nodes V and a set of directed links E. For the number of nodes $|V|$ we use n, and the number of links $|E|$ will be denoted by m. The neighborhood $N(v)$ of a node $v \in V$ is defined as the set of nodes to which v links:

$$N(v) = \{w \in V : (v, w) \in E\}$$

Similarly, the reverse neighborhood $N'(v)$ can be defined as the set of nodes that link to node v:

$$N'(v) = \{u \in V : (u, v) \in E\}$$

The notion of a neighborhood can be extended by defining it for a *set* of nodes W as:

$$N(W) = \{w \in V : v \in W \wedge (v, w) \in E\}$$

For convenience, for a node $v \in V$ we say that $N(v) = N(\{v\})$. Next, we say that the k-neighborhood $N_k(W)$ is defined as all nodes that are between 0 and k steps away from nodes in W. For the case $k = 0$ we have $N_0(W) = W$. Then for $k > 0$ we have:

$$N_k(W) = N(N_{k-1}(W)) \cup N_{k-1}(W)$$

Essentially, the k-neighborhood allows us to apply the neighborhood function to a set of nodes k times. Using these notions, it is possible to define other measures, procedures and algorithms.

(2p) Question 1.1 Give a formal definition of the *indegree* and *outdegree* of a node using the notion of a (reversed) neighborhood.

(3p) Question 1.2 In a directed network, the *combined degree* of a node is the number of neighbors connected to that node through either an incoming or an outgoing link. Give a formal definition of the combined degree using the notion of a (reversed) neighborhood.

(3p) Question 1.3 Logically combine the notions of a k-neighborhood and reversed neighborhood to that of a *reversed k-neighborhood*, and briefly explain what it measures.

(6p) Question 1.4 Give an algorithm that determines whether a given set of nodes $S \subseteq V$ is a strongly connected component of the network, using the notion of a (reversed)-(k)-neighborhood.

Assume from now on that the network has a symmetric edge set, modeling that it is undirected. Also assume there is one connected component.

(4p) Question 1.5 Give a formal definition of the density of a node's ego network using the notion of a (k-)neighborhood.

(4p) Question 1.6 If there exists non-empty strict subsets $S, T \subset V$ such that $N(S) = T$, $N(T) = S$, $S \cup T = V$ and $S \cap T = \emptyset$, what type of graph are we dealing with? What can you say about the length of circular (round trip) paths in such graphs?

(4p) Question 1.7 Assortativity is typically defined as the statistical correlation between the degree values of connected nodes. We now wish to create a node-based measure of "local" assortativity, measuring the extent to which a node connects to nodes of a similar degree. Give a formal definition of such a measure using the notion of a neighborhood, capturing the ratio between a node's degree value and the average of its neighbor's degree values.

(6p) Question 1.8 Give a formal definition of the network *diameter* using the notion of a (k-)neighborhood.

(8p) Question 1.9 Give an algorithm that counts the number of squares (cycles of length 4) in a given network, using the notion of a (k-)neighborhood. What is the time complexity of your algorithm?

Exercise 2: Mining An Online Social Network (60p + 10p bonus)

This is a practical exercise. Two social network datasets can be found at
`https://liacs.leidenuniv.nl/~takesfw/SNACS/medium.tsv` and
`https://liacs.leidenuniv.nl/~takesfw/SNACS/large.tsv`.
Each file contains a list of social network friendships in edge list format, so of the form
`userA[tab]userB[newline]`
A line thus represents one directed link from a person identified by `userA` to a person identified by `userB`. Assume that these identifiers are integers that fit a 4-byte `signed int` in C++.

Questions about visualization can likely best be done using GEPHI, whereas computing measures and distributions requires the use of programming and packages such as NETWORKX.

Answer each of the following six questions for `medium.tsv` and `large.tsv` (hence, up to Question 2.6, points are also given 2×). Remember to write down how you obtained your answer, for example by including pointers to relevant Appendix source code. Display your referenced diagrams with neat, properly scaled, readable, labelled axes and captions; the latter also holds for tables. Plots can for example be generated using GNUPLOT or MATPLOTLIB.

(2×2p) Question 2.1 How many directed links does this network have?

(2×2p) Question 2.2 How many users (nodes) does this social network have? Hint: a node counts as a node if it is a source or a target of a link.

(2×4p) Question 2.3 Give the indegree and outdegree distributions of this network using a proper diagram.

(2×4p) Question 2.4 How many weakly connected components and strongly connected components are there? How many nodes and links do the largest strongly and largest weakly connected component have? (6 answers per network)

(2×3p) Question 2.5 Give the exact or approximated average clustering coefficient of this network.

(2×7p) Question 2.6 Give the exact or approximated distance distribution of the largest weakly connected component of this network as a diagram.

(16p) Question 2.7 Visualize the social network in `medium.tsv`, for example using GEPHI. Give the size and the color of a node a sensible meaning based on node centrality, and describe your choices. State which visualization algorithm you used and how you have chosen its parameters. Include your visualization as a proper full-page A4 vector graphic PDF in your report.

(10p, bonus) Question 2.8 This dataset contains over 5 million nodes and 1 billion edges:
`/vol/share/groups/liacs/scratch/SNACS/huge.tsv`
`/data/SNACS/huge.tsv`
The files are identical, where the first is in the university-provided (remote) Linux environment and the second in the LIACS data science lab environment (see `https://rel.liacs.nl` for access instructions).
Answer Questions 2.1 through 2.6 above for this dataset. You will need to use approximation and/or a more advanced software package and environment (e.g., you should investigate GRAPH-TOOL, IGRAPH or SNAP), or write efficient code yourself.