Exercise set #1 (17 pts)

- The deadline for handing in your solutions is September 24th 2018 23:55.
- Return your solutions (one `.pdf` file and one `.zip` file containing Python code) in MyCourses (Assignments tab). Additionally, submit your pdf file also to the Turnitin plagiarism checker in MyCourses.
- Check also the course practicalities page in MyCourses for more details on writing your report.

1. Basic network properties (6 pts, pen and paper)

Define the following quantities, and **calculate** them for the graph $G = (V, E)$ in Figure 1. Note: Providing a mathematical formula alone is not sufficient as a definition, but a brief verbal description or explanation of the variables in a formula is also required.

a) (1 pt) The adjacency matrix A.

b) (1 pt) The edge density ρ of the graph.

c) (1 pt) The degree k_i of each node $i \in V$ and the degree distribution $P(k)$.

d) (1 pt) The mean degree $\langle k \rangle$ of the graph.

e) (1 pt) The diameter d of the graph.

f) (1 pt) The clustering coefficient C_i for each node $i \in V$ that has degree $k_i > 1$, and the average clustering coefficient (averaged over all nodes). For nodes with $k_i = 0, 1$, we define $C_i = 0$.

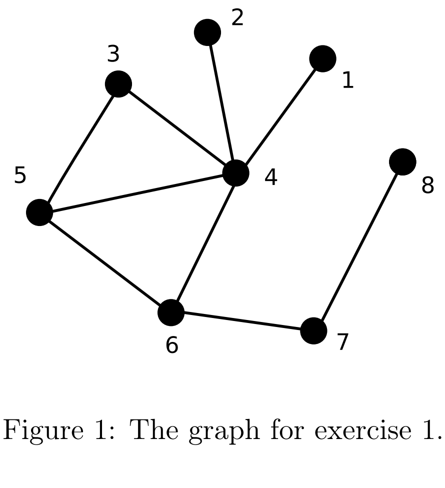

Figure 1: The graph for exercise 1.

2. Computing network properties programmatically (6 pts)

In this exercise, you will get some hands-on experience of NetworkX [1] by calculating some basic network properties. The dataset we use here, the Zachary karate club network [2], is a famous example of a social network, where the club eventually splits into two parts because of a dispute between two leaders. The dataset edge list file (`karate_club_network_edge_file.edg`) can be found in the course MyCourses page. To get you started, you may use the accompanying Python template (`compute_network_properties.py`) but the usage of the **template** is fully **optional**. Then you only need to fill in the required functions. In addition to returning a short report of your results (including the visualizations), return also your commented Python code. Remember to label the axes in your figures!

Hint: Check also the NetworkX online tutorial and index:
`https://networkx.github.io/documentation/networkx-1.10/tutorial/index.html`
`https://networkx.github.io/documentation/networkx-1.10/reference/index.html`

a) (1 pt) Load the edge list and **visualize** the network. The split of the club should be reflected in the shape of the visualized network.
b) (1 pt) Calculate the edge density of the karate club network. First, **write your own code without using** `networkx.density` and then **compare** your result to the output of `networkx.density` (the corresponding NetworkX function).
c) (1 pt) **Calculate** the average clustering coefficient **with your own algorithm and compare** it to the output of the corresponding NetworkX function.
d) (1 pt) **Calculate** the degree distribution $P(k)$ and the complementary cumulative degree distribution 1-CDF(k) of the network. **Visualize** the distributions using `matplotlib.pyplot`[1]. **NOTE:** In this course, we use a slightly non-standard definition of 1-CDF(k): 1-CDF(k) is defined as the probability that a randomly picked node has a degree *larger than or equal to* $\geq k$.
e) (1 pt) **Calculate** the average shortest path length $\langle l \rangle$. Here, you don't need to write your own algorithm. It is sufficient to use the relevant `networkx` function.
f) (1 pt) Using `matplotlib.pyplot` library, **create** a scatter plot of C_i as a function of k_i.

3. Counting number of walks using the adjacency matrix (5 pts, pen and paper)

A lot of network properties can be computed from its adjacency matrix and its (eigenvalue) spectrum. In this exercise, we investigate the relationship between the powers of the adjacency matrix and number of walks between pairs of nodes.

a) (1 pt) **Draw** the *induced subgraph* G^* that is induced by vertices $V^* = \{1 \ldots 4\}$ of network visualized in Figure 1. **Calculate by hand** the number of walks of length two between all node pairs (i, j), $i, j \in \{1, ..., 4\}$ in G^*. The length of a walk is defined as the number of links travelled to get from i to j; a link can be travelled in both directions and the walk can visit a node multiple times. Remember to consider also walks, where $i = j$.

   Then, compute the matrix A^2 (you may do this also using a computer), where A is the adjacency matrix of the network G^*. **Compare your results**; what do you notice?

[1] Check tutorial at `http://matplotlib.org/users/pyplot_tutorial.html`.

b) (1 pt) Compute the number of walks of length three from node 3 to node 4 in G^*. Then, starting from matrices A^2 and A, **compute by hand** the value of $(A^3)_{3,4}$ showing also the **intermediate steps** for computing the matrix element.

c) (3 pts) Now, let's consider a general network with adjacency matrix A. **Show** that the element $(A^m)_{ij}$, $m \in \mathbb{N}$ corresponds to the number of walks of length m between nodes i and j.

Hint: Make use of mathematical induction: Show first that the statement holds for $m = 1$, by analyzing the elements of the matrix A^1. Next, assume that the statement holds for a general m and prove that it holds also for $m + 1$. To do that, consider the element $a_{i,j}^{(m+1)} (= (A^{m+1})_{i,j})$ assuming that $a_{i,j}^{(m)}$ gives the number of walks of length m.

Feedback (1 pt)

To earn one bonus point, give feedback on this exercise set and the corresponding lecture latest two day after the report's submission deadline.
Link to the feedback form: `https://goo.gl/forms/LQcNqDhqFKnUO3Hm2`.

References

[1] [Online]. Available: https://networkx.github.io/

[2] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.