Aalto University
CS-E5740 Complex Networks, Autumn 2023
Hiraoka, Kivelä, Korhonen
Exercise set #6

Exercise set #6 (16 pts)

- The deadline for handing in your solutions is October 30th 2023 23:59.
- There are multiple ways for submitting your solutions. Check each question for details.
- Return also one `.zip` file containing all your Python code of the round in MyCourses.
- Check also the course practicalities file in MyCourses for more details on submitting your solutions.

Please submit all your solutions for exercise round 6 to MyCourses as a single pdf report.

1. Percolation in Erdős-Rényi networks (8 pts)

Erdős-Rényi (ER) networks are random networks where N nodes are randomly connected such that the probability that any pair of nodes is linked is p. In network science, ER networks are important because they provide the simplest reference to which one can compare real-world networks. In this exercise, we will analyze the percolation properties of ER networks. We will especially focus on the *percolation threshold*: the value of average degree $\langle k \rangle$ of the ER network for which a *giant component* appears. A giant component exists if the size of the largest connected component, $S_{\max}$, also grows in proportion to N, or in other words, $S_{\max}/N \to s_{\max} > 0$ for $N \to \infty$.

You can use the Python template provided in GitHub to get started.

a) (1 pt) We start by visualizing the percolation process in a relatively small ER network. Create five ER networks, each with $N = 500$ nodes and average degree $\langle k \rangle = [0.7, 0.9, 1.1, 1.3, 1.5]$, respectively. **Visualize** each network with the nodes in the largest component in red and the other nodes in grey.

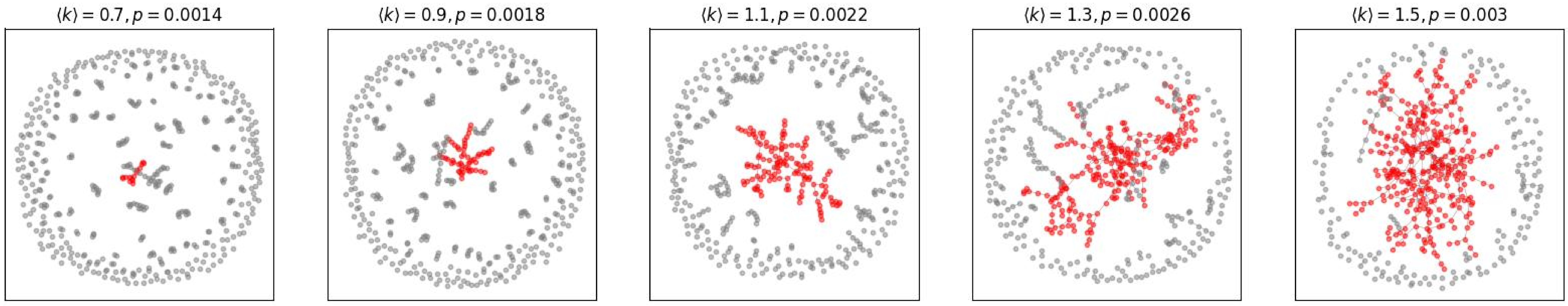
b) (2 pt) To understand how the percolation properties depend on the average degree, and how they behave differently for different network sizes, generate ER networks with $N = [50, 500, 5000, 50000]$ nodes. For each network, **make a plot** of the fraction of nodes in the largest component, $s_{\max}$, as a function of the average degree $\langle k \rangle$ scanned between 0 and 2.5 in steps of 0.1, i.e., $\langle k \rangle = [0, 0.1, 0.2, \ldots, 2.5]$. By looking at the plots, **estimate** the percolation threshold, i.e., the value of $\langle k \rangle$ for which a giant component starts to appear.

c) (2 pt) Another, more elegant way to numerically identify the percolation threshold is to find the point at which the probability for the largest component size growth is the highest as the control parameter (here $\langle k \rangle$ or p) is changed very little. Think about the situation where $\langle k \rangle$ is changed so slightly that a single link is added between the largest component and a randomly selected node that is not in the largest component. The expected change in the largest component size in this situation is sometimes called *susceptibility*, and its value should have a peak at the percolation threshold. The susceptibility depends on the size distribution of all the other components, and it can be calculated with the following formula:

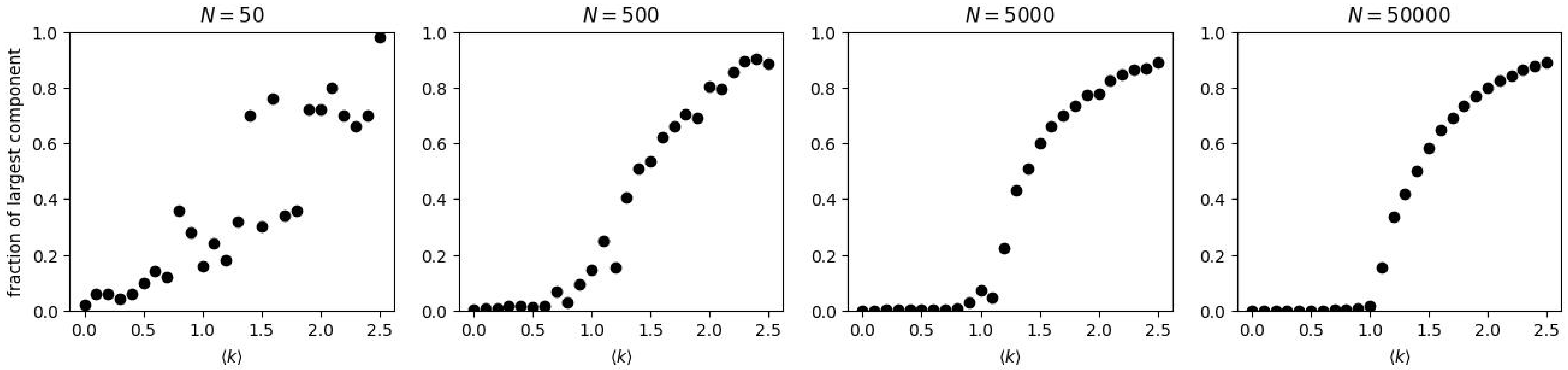
$$\chi = \frac{\left(\sum_i i^2 C(i)\right) - S_{\max}^2}{\left(\sum_i i C(i)\right) - S_{\max}}, \tag{1}$$

where $C(i)$ is the number of components with i nodes.

a)



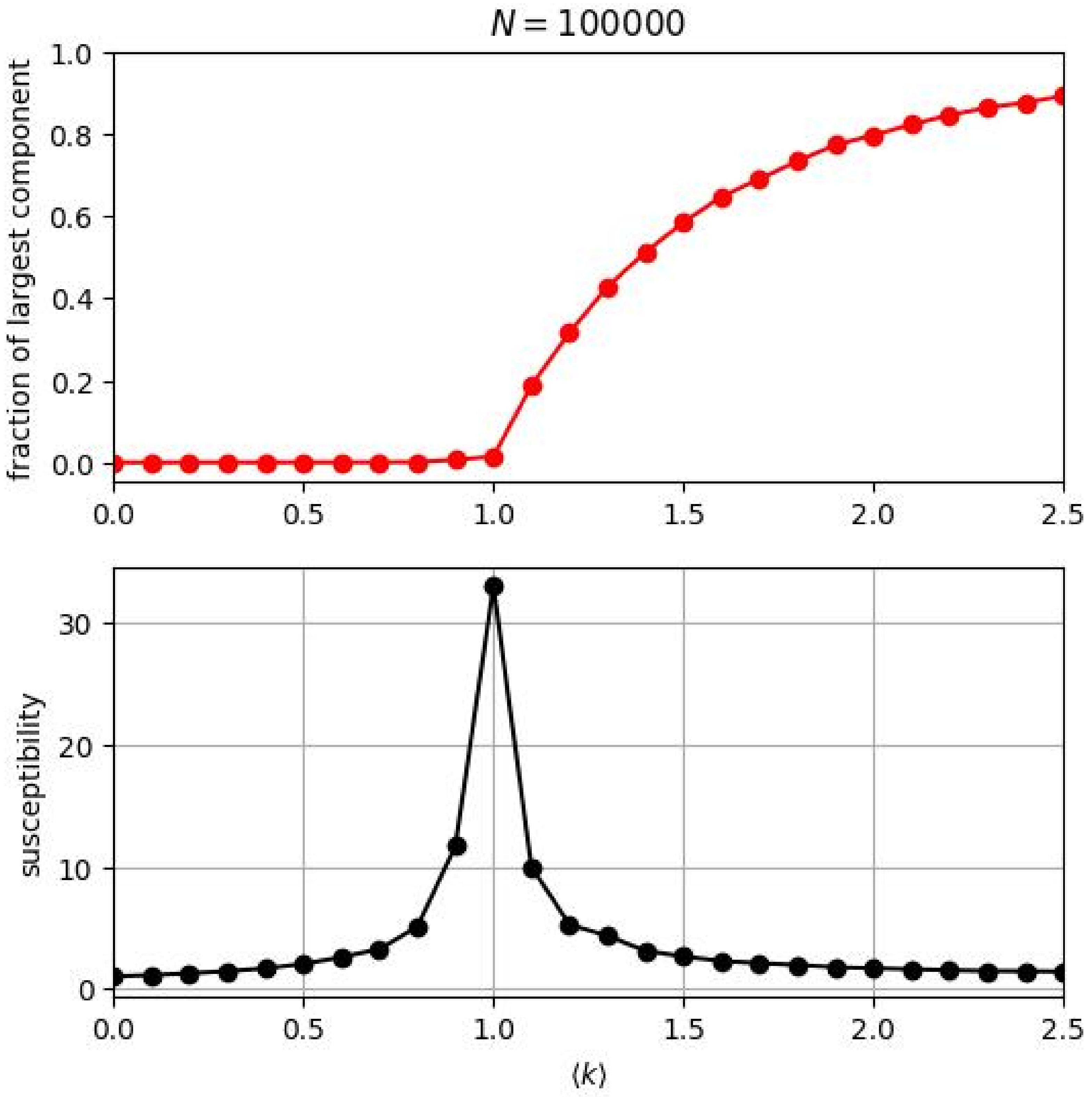
b)



Looking at the plots it is possible to notice that the percolation threshold is $\langle k \rangle = 1$, and this value is very clear expecially in the plot with $N = 50000$.

The is in agreement with the results found by Erdős and Rényi in thei paper. They predicted that the condition for the emergence of the giant component is $\langle k \rangle = 1$. In other words, we have a giant component when each node has on average one link.

c)



Also this plot, looking at the peak of the susceptibility confirms the result obtained in the previous plot, i.e. the percolation threshold is $\langle k \rangle = 1$.

Calculate the susceptibility χ for networks with $N = 100000$ and $\langle k \rangle = [0, 0.1, 0.2, \ldots, 2.5]$, and **plot** χ as a function of $\langle k \rangle$. **Estimate** the percolation threshold by observing the peak of the susceptibility.

d) (pen and paper, 3 pt) The percolation threshold can also be calculated analytically by using the fact that large and sparse ER graphs are locally tree-like, meaning that there are few cycles of any size and short cycles are especially rare. For the purposes of this exercise, we will assume that large and sparse ER graphs behave the same as trees. Under this assumption, use the idea of branching processes and the concept of excess degree (presented in the lecture) to **calculate** the expected number of nodes at d steps away, n_d, from a randomly selected node in an ER network as a function of $\langle k \rangle$ and d. Use this result to **justify** your estimate of the percolation threshold based on numerical observations from parts b) and c).

Hints:

- What is the relationship between n_d and n_{d-1}? Can this relationship be re-written as a closed-form expression, that is, a non-recursive relationship that does not involve n_{d-1}?
- How does this closed-form expression simplify for the special case of an ER network, given the properties of the degree distribution? Remember that the degree distribution of an ER network is a Poisson distribution when $N \to \infty$ such that $\langle k \rangle$ is constant. A property of the Poisson distribution is that the mean and the variance are equal, in other words, $\langle k \rangle = \langle k^2 \rangle - \langle k \rangle^2$.

Challenge exercises (3 pts)

If you use the Python template, please note that it reports the plots for e) and f) as one set for easier comparison.

e) (2 pt) Using numerical simulations, **calculate** the n_d value for $d \in \{0 \ldots 15\}$, $\langle k \rangle \in \{0.5, 1, 2\}$, starting from a set of randomly selected nodes and taking averages to get a good estimate for the expected value. Try out two network sizes: $N = 10^4$ and $N = 10^5$ to see how size affects your calculations. **Present a plot** of mean n_d as a function of d for different values of $\langle k \rangle$.

Hint: A reasonable number of starting nodes can be anywhere from hundreds to thousands based on how fast your computer and your simulation code is. With a moderately recent computer and an efficient implementation of the algorithm, you should be able to simply use the parameters provided in the template. (Same for the next exercise as well.)

f) (1 pt) In this exercise, we will explore the limits to the assumption of local tree-likeness of large, sparse Erdős-Rényi networks. More specifically, we will explore how far from any node you would have to look to see non-tree-like behavior. We will do this by calculating the number of edges that nodes at depth d have that go back to some earlier level, in addition to the single edge that connects each node to level $d - 1$. **Calculate** the average fraction of such edges to all edges that go from depth d to earlier levels/depths. In a perfect tree, this fraction is exactly 0. **Plot** this fraction as a function of d.

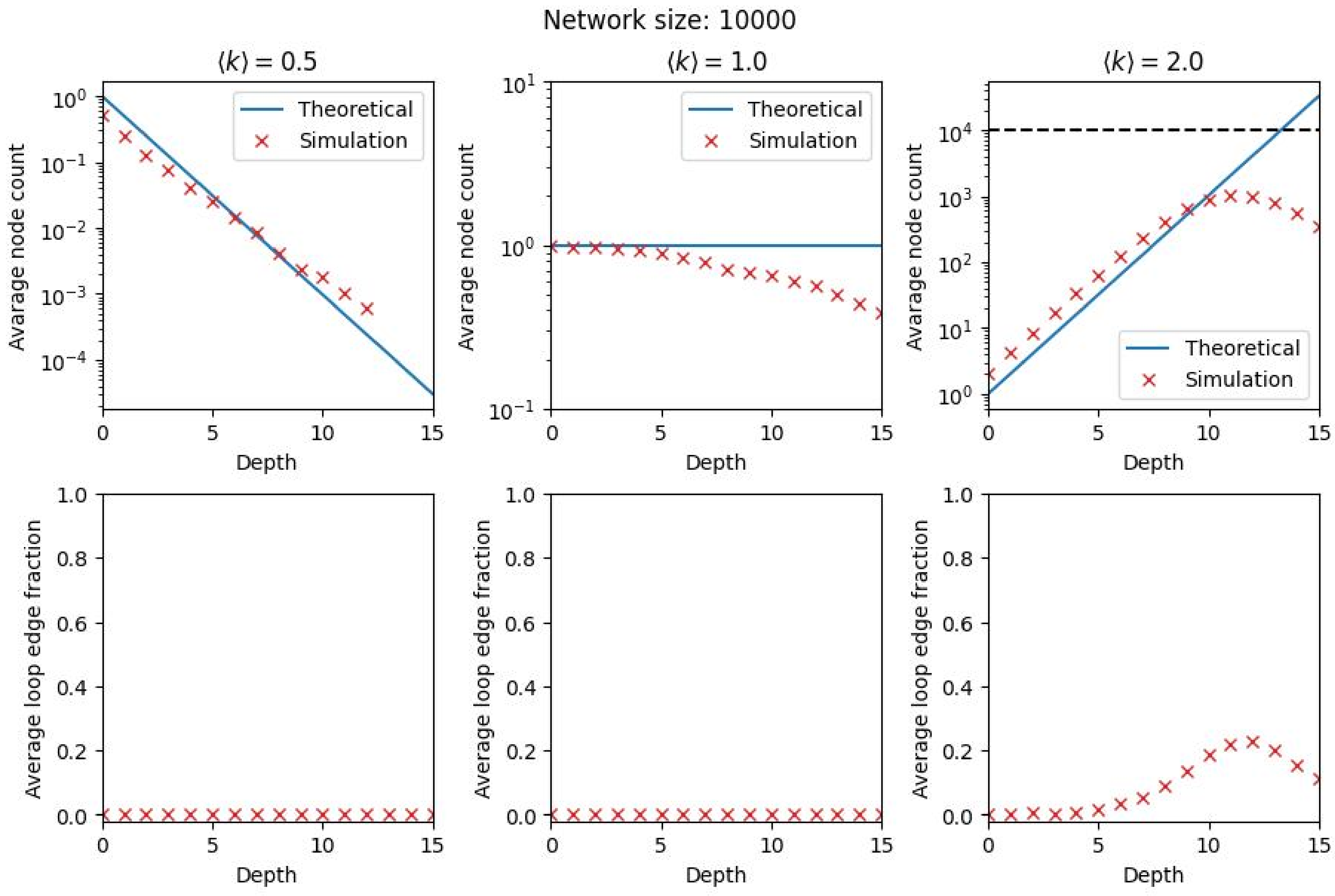
d) Referring to the branching process, we know that each node has on average $\langle q\rangle$ children, and that there are m_d nodes at step d, s.t. $m_{d+1} = \langle q\rangle m_d$.

we can write
$$m_d = \langle q\rangle m_{d-1}$$
$$m_{d-1} = \langle q\rangle m_{d-2}$$
$$m_{d-2} = \langle q\rangle m_{d-3}$$
$$\vdots$$
$$m_0 = 1$$
If we substitute, in an iterative way, the values for $m_{d-i} = \langle q\rangle m_{d-i-1}$

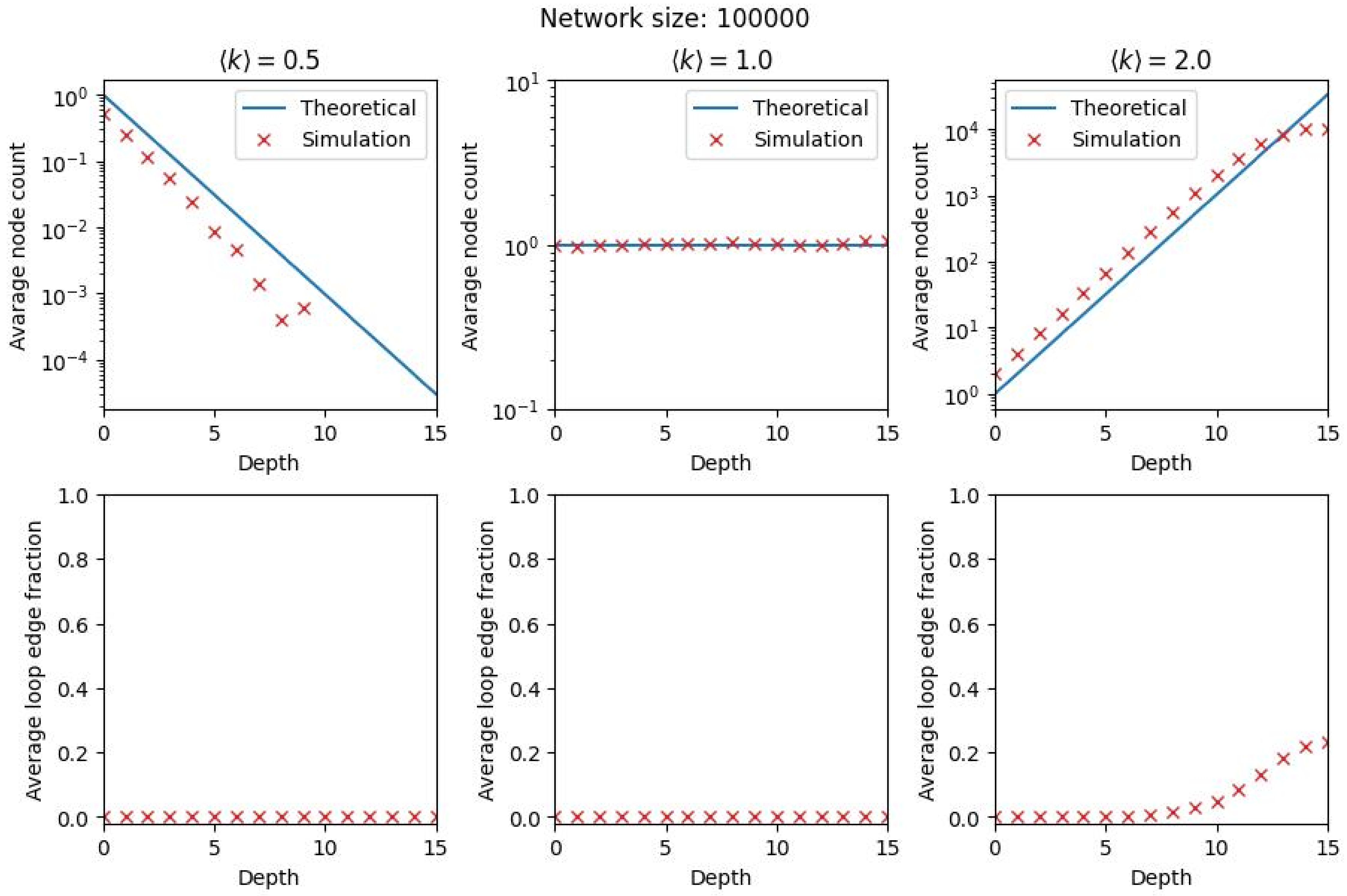
we can obtain that $m_d = \langle q\rangle^d$. Remembering that the expected excess degree $\langle q\rangle = \frac{\langle k^2\rangle}{\langle k\rangle} - 1$ and also that the degree distribution of an ER network is a Poisson distribution when $N \to \infty$, we can write $\langle k\rangle = \langle k^2\rangle - \langle k\rangle^2$, and from this $\langle q\rangle = \frac{\langle k^2\rangle}{\langle k\rangle} - 1 = \frac{\langle k\rangle + \langle k\rangle^2}{\langle k\rangle} - 1 = \langle k\rangle$. So we have $\underline{m_d = \langle k\rangle^d}$

Starting from this result, $m_d = \langle k\rangle^d$, we can say that if $\langle k\rangle < 1$, there is a decay process, so no possibilities for a giant component. But, if $\langle k\rangle > 1$, there is an exponential growth, and from this we can obtain a giant component. This is in line with we have found in the exercises b) and c), for which a giant component starts to appear if $\langle k\rangle > 1$

e)



f)



Aalto University
CS-E5740 Complex Networks, Autumn 2023
Hiraoka, Kivelä, Korhonen
Exercise set #6

2. Error and attack tolerance of networks (5 pts)

The error and attack tolerance of networks is often characterized using percolation analysis, where edges are removed from the network according to different rules. Typically this kind of analyses are performed on infrastructure networks, such as power grids or road networks. In this exercise, we will apply this idea to a network of the users of Facebook-like website[1], and focus on the role of strong and weak edges in the network. In this network, each node corresponds to a user and an edge between two nodes represents contact, in the form of messages, between the two users. Additionally, an integer "weight" value is assigned to each edge that corresponds to the number of messages exchanged between the users. As opposed to a non-weighted network where nodes are either connected or disconnected, here a higher or lower weight on an edge shows a stronger or weaker relation between pairs of users.

In the file `OClinks_w_undir.edg`, the three entries of each row describe a weighted edge:
`(node_i node_j w_ij)`,
where the last entry `w_ij` is the weight of the edge between nodes `node_i` and `node_j`.

Your task is to remove edges one by one from the network in the order of:

(i) descending edge weight (i.e. remove strong edges first),

(ii) ascending edge weight (i.e. remove weak edges first),

(iii) random order.

In addition, remove nodes one by one in the order of:

(iv) descending node strength (i.e. remove nodes with high strength first),

(v) ascending node strength (i.e. remove nodes with low strength first),

(vi) random order.

When you remove a node, remove also all edges of the removed node.

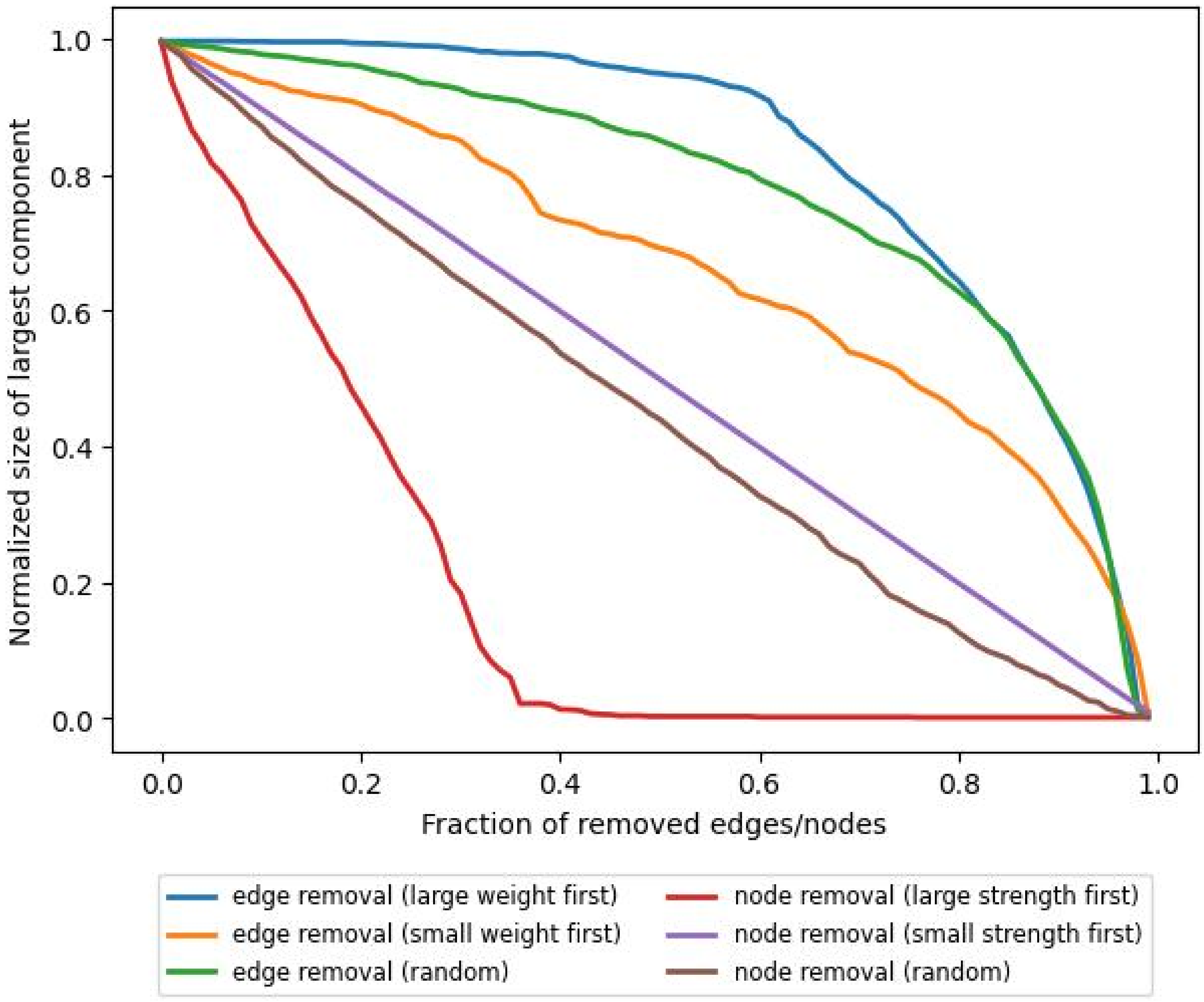
While removing edges/nodes, monitor the fraction of nodes in the largest component S as a function of the fraction of removed edges/nodes $f \in [0, 1]$.

Visualize S as a function of f in all six cases **in one plot**. There should be clear differences between all six curves. **Explain** the results in five to ten sentences. To which of the six approaches is the network most and least vulnerable? In other words, in which case does the giant component shrink the fastest/slowest? What is the difference between node removal and edge removal and between different removal strategies? What does it imply about the structure of the network?

Hints:

- In the exercise, `networkx.connected_components(G)` may turn out handy. It returns a list of the components of the network, each of them presented as a list of nodes belonging to the component.
- Let `components` be the outcome from `networkx.connected_components(G)`. For getting the largest component, you can use `max(components, key=len)`.

[1] Data originally from `http://toreopsahl.com/datasets/`



The network is most vulnerable when nodes with high strength are removed first: this is because nodes with high strength tend to be network hubs (they have many connections to other nodes) so removing hubs fragments the network quickly because their many ties hold the network together, and when the hubs are removed, the network breaks down into many disconnected components.

On the other hand, the network is least vulnerable with respect to edge removal in descending order of edge weight: when strong ties are removed first, the remaining weak ties can still hold the network together, even if many edges are removed, but when weak ties are removed first, the network quickly falls apart into disconnected subgraphs.

This is because strong edges connect nodes that are highly active and engaged with the network, so removing them does not cause as many nodes to become isolated.

Weak ties also act as bridges between communities, so removing them cuts connections between communities and fragments the network much faster.

Basically, the difference between node removal and edge removal is that when a node is removed, all of its edges are also removed: this means that removing a node can have a cascading effect, disrupting the connectivity of other nodes in the network, while when an edge is removed, only the two nodes connected by the edge are affected.

The difference between different removal strategies is that some strategies remove more important nodes or edges than others. For example, removing strong edges or nodes with high strength is more disruptive than removing weak edges or nodes with low strength.

The results of the percolation analysis imply that the network has a core of highly active and engaged users who are connected by strong edges, and removing these users can have a significant impact on the connectivity of the network.

- Edges of the present network are tuples of three values. For sorting them based on their weight, `sorted` function with `key` parameter can be useful. For more information, check `https://wiki.python.org/moin/HowTo/Sorting`.
- The strength of a node is defined as the sum of the weights of the edges connected to the node.
- The overall running time of this simulation can take up to a minute but not orders of magnitudes more.

Feedback (1 pt)

To earn one bonus point, give feedback on this exercise set and the corresponding lecture latest two days after the exercise round's submission deadline. You can find the feedback form in MyCourses.

References