CS-E5740 Complex Networks Submission 4

October 17, 2018

CS-E5740 Complex Networks Adam Ilyas 725819
Percolation, error & attack tolerance, epidemic models
Percolation theory
"order parameter" P: fraction of nodes in the largest connected component (LCC)
Control parameters f: $\frac{\text{number of active links}}{\text{number of all possible links}}$

```
In [1]: import os

        import random

        import networkx as nx
        import numpy as np
        import matplotlib.pyplot as plt
        import scipy.stats

        from percolation_in_er_networks import *
        from error_and_attack_tolerance import *
```

1 Percolation in Erdős-Rényi (ER) networks

Erdős-Rényi networks are random networks where N nodes are randomly connected such that the probability that a pair of nodes is linked is p.

Sparse ER graph: - average degree $\langle k \rangle$ is some fixed small number - the size of the network N is very large. - N large, p large such that $p(N-1) = \langle k \rangle$ stays constant

Percolation properties of ER Graphs.

percolation threshold which is the value of $\langle k \rangle$ where the giant connected component appears

1.1 1a)

Using the idea of branching processes and assumption that large and sparse ER graphs are tree-like,

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. Using this result, justify that the giant component appears in large and sparse ER networks when $\langle k \rangle > 1$.

Hints: – Remember that the degree distribution of an ER network is a Poisson distribution when $N \to \infty$ such that $\langle k \rangle$ is constant.

Hence $\langle k \rangle = \frac{\langle k^2 \rangle}{\langle k \rangle} - 1$

$$
\begin{aligned}
n_d &= \langle q \rangle \cdot n_{d-1} \\
&= (\frac{\langle k^2 \rangle}{\langle k \rangle} - 1) \cdot n_{d-1} \\
&= (\frac{\langle k \rangle^2 + \langle k \rangle}{\langle k \rangle} - 1 \cdot n_{d-1} \\
&= (\langle k \rangle + 1 - 1) \cdot n_{d-1} \\
&= \langle k \rangle \cdot n_{d-1}
\end{aligned}
$$

$$
\begin{aligned}
n_{d-1} &= \langle k \rangle \cdot n_{d-2} \\
n_{d-2} &= \langle k \rangle \cdot n_{d-3} \\
&\vdots \\
n_d &= \langle k \rangle^d
\end{aligned}
$$

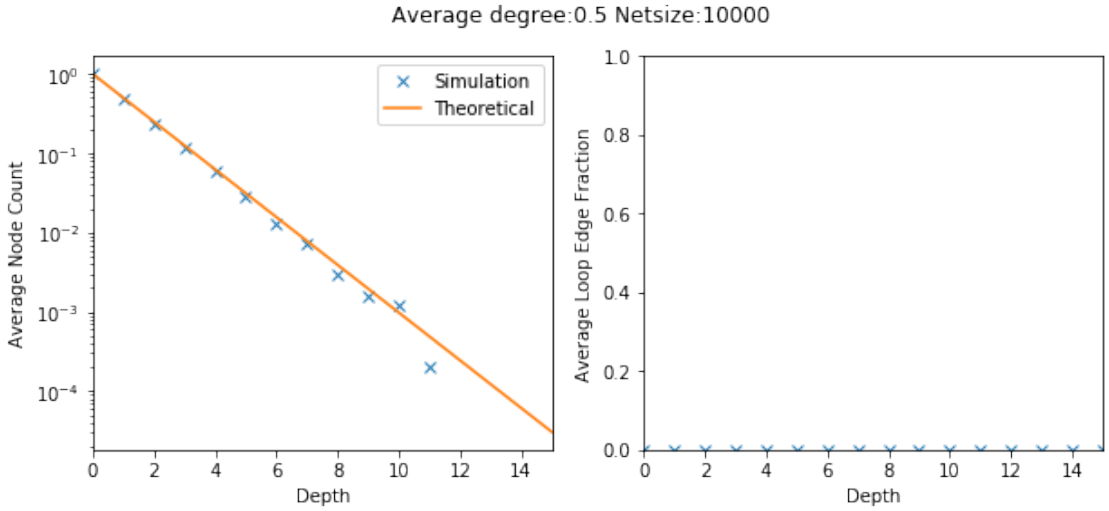
We recurse d times
where $\langle k \rangle$ is average degree, and d is the number of steps

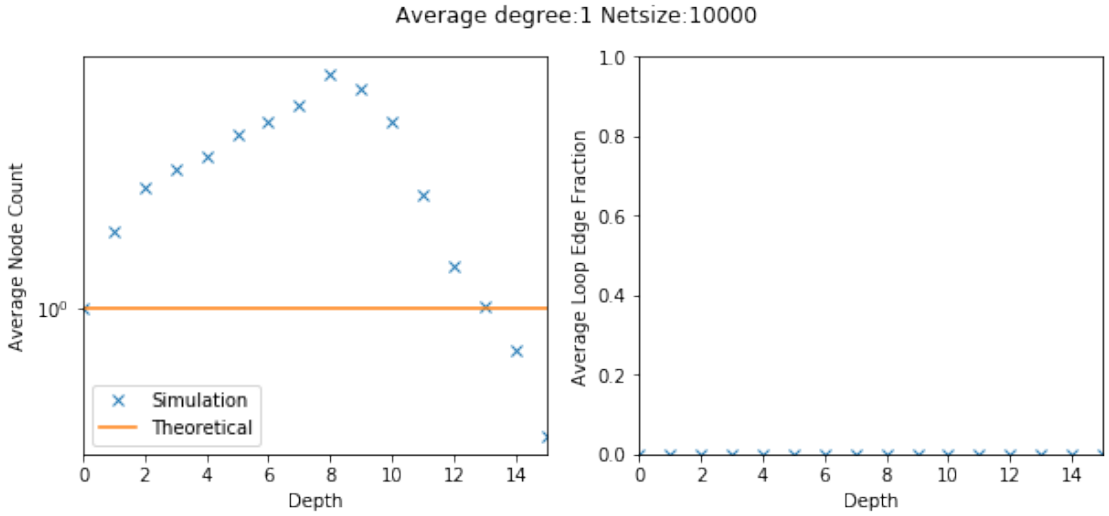
1.2 1b)

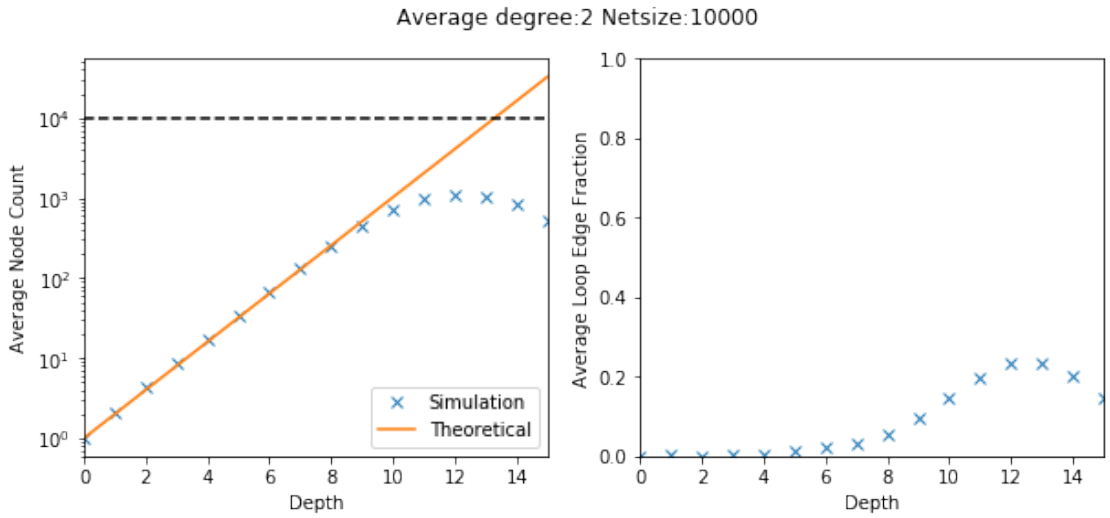
Verify your analytical calculations for n_d using numerical simulations. Calculate the n_d value for $d \in 0...15$, $\langle k \rangle \in \{0.5, 1, 2\}$, and starting from enough randomly selected nodes to get a good estimate for the expected value. Try out two network sizes: $N = 10^4$ and $N = 10^5$ to see how the size affects the calculations.

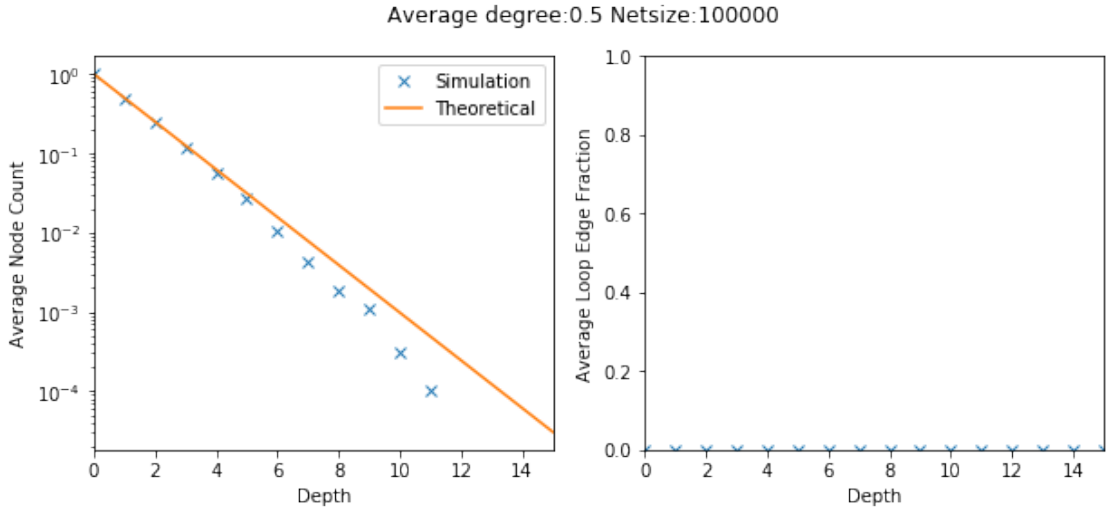
```
In [2]: a_dir = "./assets"
        if not os.path.isdir(a_dir):
            os.mkdir(a_dir)

        #Solution for b)-c):
        fig = ER_breadth_first_search(0.5, 10**4, 10000)
        fig.savefig('./assets/er_breadthfirst_05_10k.pdf')

        fig = ER_breadth_first_search(1, 10**4, 10000)
        fig.savefig('./assets/er_breadthfirst_1_10k.pdf')

        fig = ER_breadth_first_search(2, 10**4, 100, show_netsize=True, max_depth=15)
        fig.savefig('./assets/er_breadthfirst_2_10k.pdf')

        fig = ER_breadth_first_search(0.5, 10**5, 10000)
        fig.savefig('./assets/er_breadthfirst_05_100k.pdf')

        fig = ER_breadth_first_search(1, 10**5, 10000)
        fig.savefig('./assets/er_breadthfirst_1_100k.pdf')
```

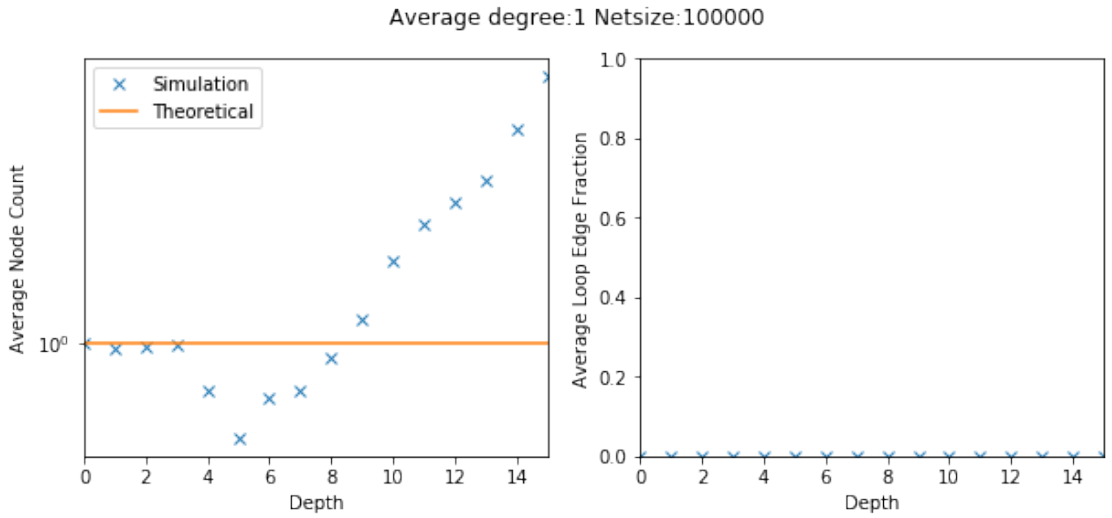
```
fig = ER_breadth_first_search(2, 10**5, 100, show_netsize=True, max_depth=15)
fig.savefig('./assets/er_breadthfirst_2_100k.pdf')
```

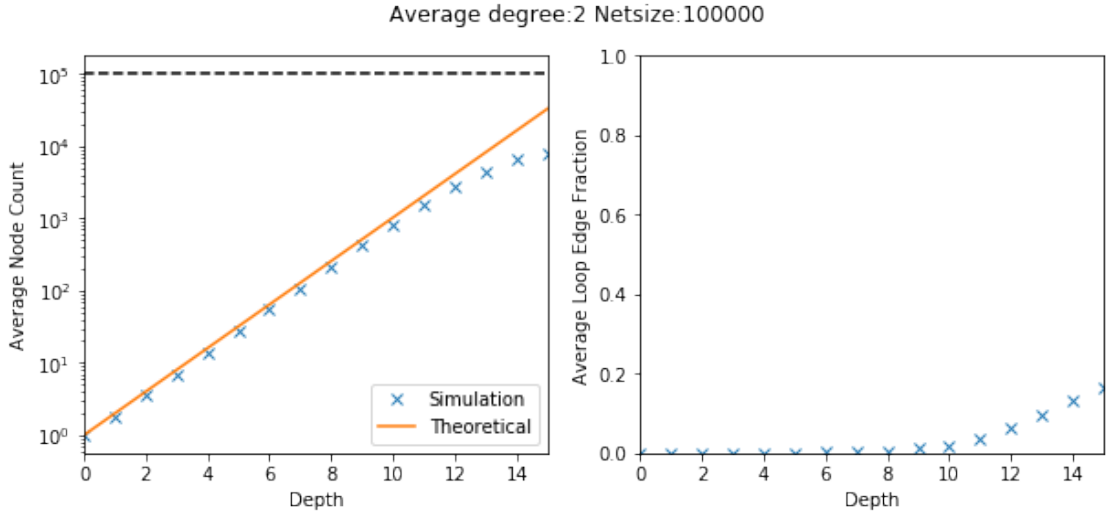





Average degree:2 Netsize:10000
Average Node Count
Depth
Simulation
Theoretical
Average Loop Edge Fraction
Depth


Average degree:0.5 Netsize:100000
Average Node Count
Depth
Simulation
Theoretical
Average Loop Edge Fraction
Depth





1.3 1c)

Explore the range at which the assumption of tree-likeness of the network is valid.

This can be done, for example, 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 the level $d-1$, and

reporting 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. Comment on the results, and their effect on our results to exercise b). What are the other things that make your analytical calculation of n_d to differ from your simulation results?

Ans:

For average_degree = 2, for both $N = 10^4$ and 10^5, we notice from the graph above that we will expect some loops as we go into more depths.

When loops occur, the average node count will not increase as fast as we will encounter the same nodes that we counted before and not count them

Thus, the simulated average node count will not increase as fast as in theory

1.4 1d)

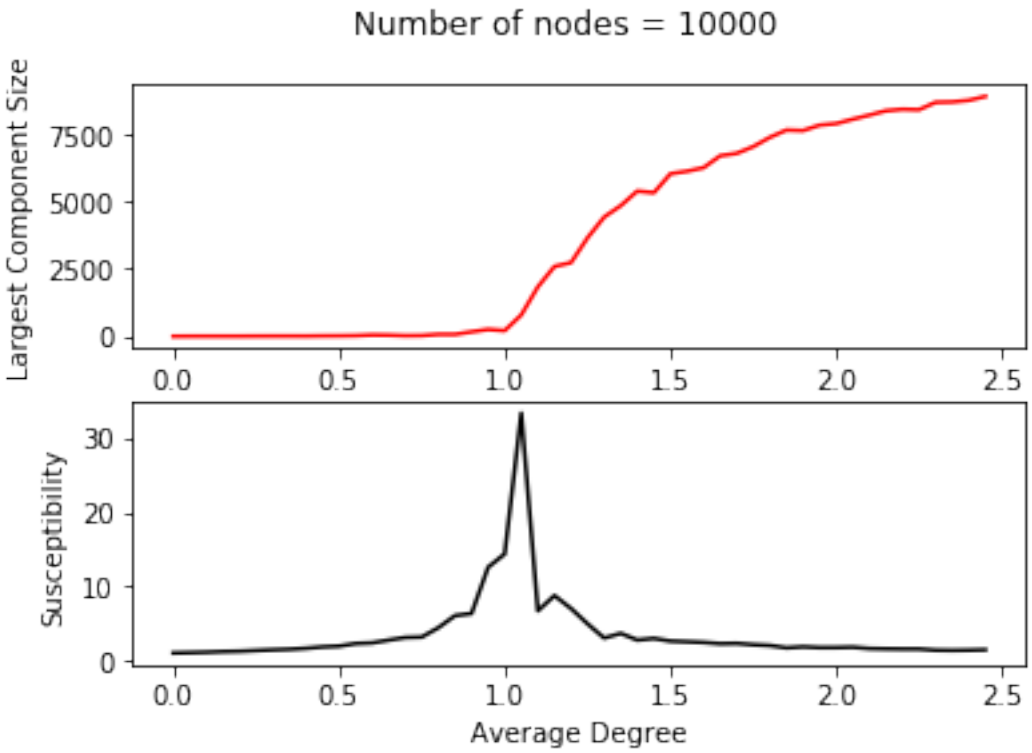
Calculate the component sizes of simulated ER networks,

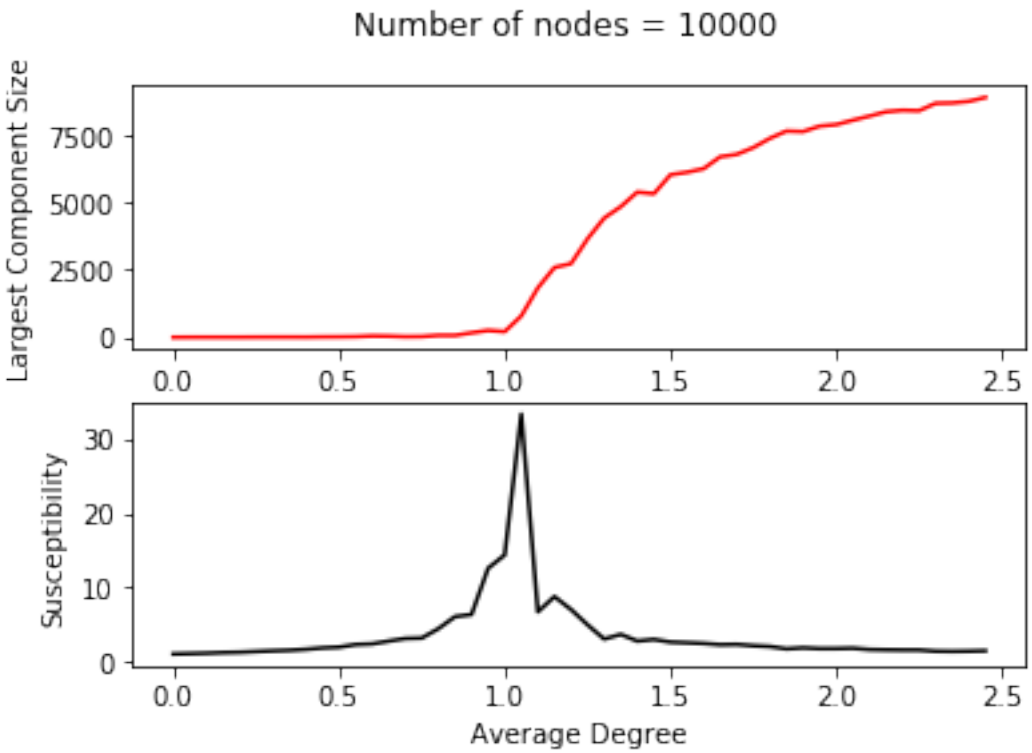
and use this data to (loosely) verify that the percolation threshold of ER networks is at the average degree of $\langle k \rangle = 1$.

That is, for $\langle k \rangle < 1$ the largest connected component is small (size being measured as number of participating nodes), and for $\langle k \rangle > 1$ it quickly reaches the network size. Do this by generating ER networks of size $N = 10^4$ with different average degrees: $\langle k \rangle = [0.00, 0.05, \cdots, 2.45, 2.50]$. For each of the ER networks, compute the size of the largest component and plot it against $\langle k \rangle$

```
In [3]: ER_percolation(10**4, 2.5, 0.05)
```

Out[3]:





1.5 1e)

Another, a more elegant, way to find out when the percolation transition happens is:

to try to find the point at which the possibility for the largest component size growth is the largest when 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 some times called susceptibility, and it should get very large values at the percolation transition point. The susceptibility depends on the size distribution of all the other components, and it can be calculated with the following formula:

$$x = \frac{\sum_s s^2 C(s) - s_m a x^2}{\sum_s s C(s) - s_m a x^2}$$

where $C(s)$ is the number of components with s nodes. Calculate the susceptibility x for each network generated in exercise d), and again plot x as a function of $\langle k \rangle$. Explain the shape of the curve, and its implications.

Susceptibility is large at the point when the percolation transition happens

Where making a minor change to the average degree $\langle$k$\rangle$ lead to big changes in the size of largest component (LCC) in the network

Observation

We see in the curve when $\langle k \rangle \approx 1$, we see a sudden spike in both graph (by definition, both are dependant to another). As such, percolation transition happens when average degree $\langle k \rangle \approx 1$

2 Error and attack tolerance of networks

Error and attack tolerance of networks are often characterized using percolation analysis, where links 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 Facebook-like web-page , and focus on the role of strong and weak links in the network.

now to remove links (one by one) from the network in the order of 1. descending link weight (i.e. remove strong links first), 2. ascending link weight (i.e. remove weak links first), 3. random order 4. descending order of edge betweenness centrality (computed for the full network at the beginning).

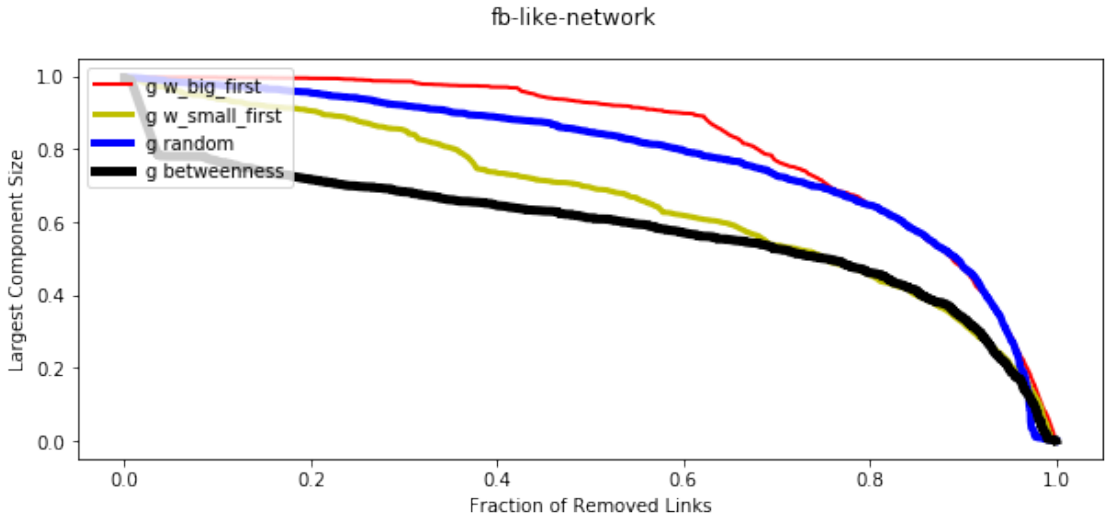
While removing the links, monitor the size of the largest component S as a function of the fraction of removed links $f \in [0, 1]$

2.1 2a)

Visualize S as a function of f in all four cases in one plot. There should be clear differences between all four curves.

```
In [4]: network_path = './OClinks_w_undir.edg' # You may want to change the path to the edge lis
        network_name = 'fb-like-network'

        fig = run_link_removal(network_path, network_name)
```

```
Computing betweenness...
W_BIG_FIRST
W_SMALL_FIRST
RANDOM
BETWEENNESS
```



2.2 2b)

For which of the four approaches is the network most and least vulnerable? In other words, in which case does the giant component shrink fastest / slowest? Or is this even simple to define?

Most vulnerable **Descending order of edge betweenness centrality** Centrality measures how important an edge is. How important meaning that this edge acts as a link between clusters of nodes and by removing this edge, the probability of breaking up a to 2 components is hight

Least vulnerable **descending link weight (i.e. remove strong links first)**

2.3 2c)

When comparing the removal of links in **ascending** and **descending** order strong and weak links first, which ones are more important for the integrity of the network? Why do you think this would be the case?

Ans

The weaker links are more important for the integrity.

Removal of a weak link leads to a higher chance of severing a connection to a hub, Thus, multiple nodes that are dependant on the link will not be able to get to the node easily

2.4 2d)

How would you explain the difference between - random removal strategy - removal in descending order of edge betweenness strategy?

Ans

Removal in descendinng order of edge betweenness follows a predefined rule, by removing edges that are deemed more important to the network first. This method is deterministic and is not random

On the other hand, random removal removes edges randomly and all edges have equal chance of getting removed.