Assignment+3

February 17, 2020

---

*You are currently looking at **version 1.2** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

---

1 Assignment 3

In this assignment you will explore measures of centrality on two networks, a friendship network in Part 1, and a blog network in Part 2.

1.1 Part 1

Answer questions 1-4 using the network `G1`, a network of friendships at a university department. Each node corresponds to a person, and an edge indicates friendship.

The network has been loaded as networkx graph object `G1`.

```
In [1]: import networkx as nx

        G1 = nx.read_gml('friendships.gml')
```

```
In [15]: #%matplotlib notebook
         #import matplotlib.pyplot as plt
         #nx.draw_networkx(G1)
```

1.1.1 Question 1

Find the degree centrality, closeness centrality, and normalized betweeness centrality (excluding endpoints) of node 100.

This function should return a tuple of floats `(degree_centrality, closeness_centrality, betweenness_centrality)`.

```
In [3]: def answer_one():
            degree_centrality = nx.degree_centrality(G1)[100]
            closeness_centrality = nx.closeness_centrality(G1)[100]
            between_centrality = nx.betweenness_centrality(G1, endpoints=False, normalized=True)
            return degree_centrality, closeness_centrality, between_centrality
        answer_one()
```

```
Out[3]: (0.0026501766784452294, 0.2654784240150094, 7.142902633244772e-05)
```

For Questions 2, 3, and 4, assume that you do not know anything about the structure of the network, except for the all the centrality values of the nodes. That is, use one of the covered centrality measures to rank the nodes and find the most appropriate candidate.

1.1.2 Question 2

Suppose you are employed by an online shopping website and are tasked with selecting one user in network G1 to send an online shopping voucher to. We expect that the user who receives the voucher will send it to their friends in the network. You want the voucher to reach as many nodes as possible. The voucher can be forwarded to multiple users at the same time, but the travel distance of the voucher is limited to one step, which means if the voucher travels more than one step in this network, it is no longer valid. Apply your knowledge in network centrality to select the best candidate for the voucher.

This function should return an integer, the name of the node.

```
In [5]: def answer_two():
            degree = nx.degree_centrality(G1)
            return max(degree.keys(), key=lambda x:degree[x])
        answer_two()
```

```
Out[5]: 105
```

1.1.3 Question 3

Now the limit of the voucher's travel distance has been removed. Because the network is connected, regardless of who you pick, every node in the network will eventually receive the voucher. However, we now want to ensure that the voucher reaches the nodes in the lowest average number of hops.

How would you change your selection strategy? Write a function to tell us who is the best candidate in the network under this condition.

This function should return an integer, the name of the node.

```
In [7]: def answer_three():
            closeness = nx.closeness_centrality(G1)
            return max(closeness.keys(), key=lambda x:closeness[x])
        answer_three()
```

```
Out[7]: 23
```

1.1.4 Question 4

Assume the restriction on the voucher's travel distance is still removed, but now a competitor has developed a strategy to remove a person from the network in order to disrupt the distribution of your company's voucher. Your competitor is specifically targeting people who are often bridges of information flow between other pairs of people. Identify the single riskiest person to be removed under your competitor's strategy?

This function should return an integer, the name of the node.

```
In [8]: def answer_four():
            betweeness = nx.betweenness_centrality(G1)
            return max(betweeness.keys(), key=lambda x:betweeness[x])
        answer_four()

Out[8]: 333
```