CSCI 347: Project 2: Exploring Graph Data
Due date: March 2, 2020

This project is to be done with a partner. If you are having trouble finding a partner, please use the discussion boards on D2L and/or email the instructor to help you find one.

**** You are highly encouraged to work with a new partner for this project. ****

Choose a data set that you are interested in from the SNAP collection:

https://snap.stanford.edu/data/

Note that some of these datasets are quite large. If analyzing the data is taking too long, you may pre-process it by one or more of the following methods:

- Extract the largest connected component, and use the vertices and edges of this subgraph in your analysis. In networkx, you can use the connected_component_subgraphs function to extract the largest connected component.
- Take a sample of the edges, and use the subgraph induced by the vertices and edges in this edge set in your analysis. In networkx, the subgraph function will return the subgraph of G that is induced on the vertices that are passed in as input.

Part 1: Think about the data

In a well-written paragraph, answer the following questions:

1. Why are you interested in this data set?
2. Clearly state if/how the data was pre-processed (was a sample taken of the vertices and edges? Was the largest connected component extracted?).
3. Before doing any analysis, answer the following questions:
    1. What characteristics do you expect the vertices with high centrality values to have and why? Specifically think about non-graph characteristics. For example, we might expect highly central cities to have high population or to house major industries in a graph where cities are vertices and roads are edges, or highly central people to have a high presence on social media in a graph where people are vertices and edges represent follows or friendships of these people online.
    2. Do you think that the degree distribution will exhibit a power law? Why or why not?
    3. Do you think that the graph will have the small-world property? Why or why not?

Part 2: Write functions for graph analysis in Python

Write the following functions in Python. You may assume that the input graph is simple — that is, it is undirected, unweighted, and has no parallel edges or loops. Do not use the functions provided by networkx within your code (though you may use these functions to test your code):

4. A function that takes list of edges representing a graph, where each edge is a pair, as input, and outputs the degree of a vertex.
5. A function that takes list of edges representing a graph, where each edge is a pair, as input, and outputs the clustering coefficient of a vertex.
6. A function that takes list of edges representing a graph, where each edge is a pair, as input, and outputs the clustering coefficient for the graph
7. A function that takes list of edges representing a graph, where each edge is a pair, as input, and outputs the closeness centrality of a vertex
8. A function that takes list of edges representing a graph, where each edge is a pair, as input, and outputs the betweenness centrality of a vertex. For this function, you may use networkx's shortest_path function to compute the shortest path between two vertices.

9. A function that takes list of edges representing a graph, where each edge is a pair, as input, and outputs the average shortest path length μ_L of the graph.

Part 3: Use your functions to analyze the graph

Report the following, using tables or figures as appropriate. Treat the graph as an undirected, unweighted graph with no loops or parallel edges, in order to be able to use the functions written for Part 2.

Include a visualization of the graph in your report.

Use your functions to report the top 10 nodes of your chosen graph by the following metrics:
- betweenness centrality
- closeness

Use network to report the top 10 nodes of your chosen graph by the following metrics:
- eccentricity
- eigenvector centrality
- pagerank

Use your code to report the clustering coefficient of the 5 nodes with highest betweenness centrality, and the 5 nodes with highest degree. If there are ties, choose 5 and report how these 5 were chosen.

Use your code to compute the clustering coefficient of the graph.

Use your code to compute the average shortest path distance in the graph. Does the graph exhibit small-world behavior?

Plot the degree distribution of the graph on a log-log scale: does it exhibit power law behavior? Include the plot and the code used to generate it in your submission.

Create a log-log plot with the logarithm of node degree on the x-axis and the average clustering coefficient of nodes with that degree on the y-axis. Does the clustering coefficient exhibit power law behavior? Include the plot and the code used to generate it in your submission.