CSCI 347
Project 02: Exploring Graph Data

Partner work is allowed on this project.
Choose a data set that you are interested in from one of the following sources:

- SNAP collection: https://snap.stanford.edu/data
- Network Repository: http://networkrepository.com/index.php

Run all the analysis in this project on the largest connected component of the graph. Note that many of these datasets are quite large. If analyzing the data its taking too long, you may pre-process it by taking a sample of the graph first, and then extracting the largest connected component, to get the graph down to a manageable size.

Problem 1: Think about the data

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

- (3 points) Why are you interested in this data set?
- (3 points) Clearly state if/how the data was pre-processed (Was the largest connected component extracted? Was a sample of vertices or edges taken? If so, describe the sampling process that was used.)
- (4 points) Before doing any analysis, answer the question. What characteristics do you expect the vertices with high centrality values to have and why? Specifically, think about non-graph characteristics. For example, in a graph where nodes represent cities and edges are roads between them, we might expect highly central cities to have high populations or to house major industries.

Part 2: Write Python code for graph analysis

Write the following functions in Python. You may assume that the input graph is unweighted, undirected, and simple – has no parallel edges and no loops. Functions provided by networkx can be used within your code, as long as the function does not perform the same task as what you are being asked to implement. For example, you cannot use networkx's `betweenness_centrality` function within your own betweenness centrality function, but you can use networkx's functions for finding shortest paths. You may also assume that vertices are represented as integers (so the pair (1,3) indicates that there is an edge between vertex 1 and 3, for example).

1. (5 points) Number of vertices: A function that takes the following input: a list of edges representing a graph, where each edge is a pair. The output should be the number of vertices.
2. (5 points) Degree of a vertex: A function that takes the following input: a list of edges representing a graph, where each edge is a pair, and a vertex index that is an integer. The output should be the degree of the input vertex.

3. (5 points) Clustering coefficient of a vertex: A function that takes the following input: a list of edges representing a graph, where each edge is a pair, and a vertex index that is an integer. The output should be the clustering coefficient of the input vertex.

4. (5 points) Betweenness centrality of a vertex: A function that takes the following input: a list of edges representing a graph, where each edge is a pair, and a vertex index that is an integer. The output should be the betweenness centrality of the input vertex.

5. (5 points) Adjacency matrix. A function that takes the following input: a list of edges representing a graph, where each edge is a pair. The output should be the dense adjacency matrix of the graph.

6. (10 points) Prestige centrality of vertices: A function that takes the following input: a dense adjacency matrix representation of a graph. The output should be the prestige values for each vertex in the graph. (Note, you may NOT use linear algebra functions in numpy, spicy, or any other library to find eigenvectors but you MAY use linear algebra functions for transpose, matrix-vector multiplication, computing the dot product, the norm, and argmax.)

Part 3: Analyze the graph data

Using tables or figures as appropriate, report the following. You may treat the graph as a simple undirected, unweighted graph. You may use networkx functions in all of Part 3, but you are encouraged to test out your functions from Part 2 on real-world data.

1. (5 points) Produce a visualization of the graph (or graph sample that you used).

2. (3 points) Find the 10 nodes with the highest degree.

3. (3 points) Find the 10 nodes with the highest betweenness centrality.

4. (3 points) Find the 10 nodes with the highest clustering coefficient. If there are ties, choose 10 to report and explain how the 10 were chosen.

5. (3 points) Find the top 10 nodes as ranked by prestige centrality (`eigenvector_centrality` in networkx).

6. (3 points) Find the top 10 nodes as ranked by Pagerank.

7. (3 points) Comment on the differences and similarities in questions Part 3 1-6. Are the highly ranked nodes mostly the same? Do you notice significant differences in the rankings? Why do you think this is the case?

Tips and Acknowledgements

Make sure to submit your answer as a PDF on Gradscope and Brightspace. Make sure to show your work. Include any code snippets you used to generate an answer, using comments in the code to clearly indicate which problem corresponds to which code.

Acknowledgements: Project adapted from assignments of Veronika Strnadova-Neeley.