CS 7280: Network Science

Assignment-1

Learning Objectives

The objective of this first assignment is to learn basic operations of network analysis, mostly covered in Lecture 2. The assignment can be divided into 5 parts.

- Introduction to NetworkX
- Directed Graphs
- Undirected Graphs
- Bipartite Graphs
- Directed Acyclic Graphs

NetworkX

Please spend a couple of hours going through the different functionalities provided by NetworkX (https://networkx.github.io/documentation/stable/reference/index.html). You will be using this library extensively in this course. Other than NetworkX, we recommend you use Numpy and Matplotlib for programming and visualization.

Submission

Please submit your Jupyter Notebook **Assignment1-YOURLASTNAME.ipynb** with **requirements.txt** so that we may be able to replicate your Python dependencies to run your code as needed.

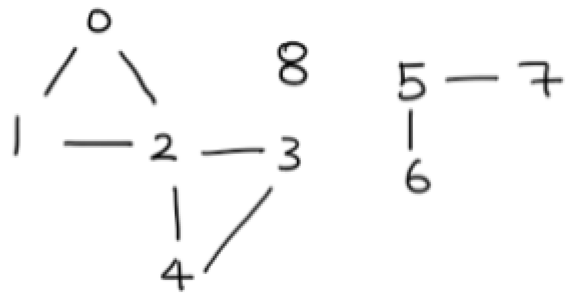
With Anaconda, you can do this by running:

```
conda list -e > requirements.txt
```

Ensure all graphs and plots are properly labeled with unit labels and titles for x & y axes – there may be point deductions if plots are not properly labeled.

Part I. Introduction to NetworkX (20 Points)

1. (5 points) Create the graph shown below using NetworkX. Use the "nx.draw(with_labels=True)" function to get a graph visualization.



2.
   a) (5 points) Load "Assignment1_Part1-2_cities_data.graphml" dataset (Check R/W functions in NetworkX) and parse it into a Graph object *G*. Use the functions "G.nodes(data=True)" and "G.edges(data=True)" to figure out what the nodes, edges and weights represent. How many nodes and edges are in graph G? How many pairs of cities have a weight between them less than 50?

   b) (5 points) Implement a function "*cities_within_100"* to do the following:

      def cities_within_100(*G*, *city_list*):

      Input
      G: graph
      city_list: list of strings (names of cities in G)

      Output
      S: subgraph of G that only contains edges between cities in "city_list" and directly neighboring cities that are less than 100 miles away.

   For example, if you run *cities_within_100(G, ["Rochester, NY","Tacoma, WA"])*, the function should return a subgraph *S* whose nodes are 'Seattle, WA', 'Tacoma, WA', 'Syracuse, NY', and 'Rochester, NY'. In this case, *S* has only two edges: one edge - ('Seattle, WA', 'Tacoma, WA', {'weight': 30}) and another - ('Syracuse, NY', 'Rochester, NY', {'weight': 88}).

   Visualize the subgraph *S* for *cities_within_100(G, ["Savannah, GA", "Santa Rosa, CA", "San Francisco, CA", "San Jose, CA"]).*

3) (5 points) Create the following three toy 10-node networks (check graph generators in NetworkX): a cycle network, a clique (complete) network, and a star network. Then compute the leading eigenvalue of the adjacency matrix for each network. *(Hint: to_numpy_matrix() of NetworkX*

and/or linalg.eig() of Numpy). Compare the largest eigenvalue of the three networks and explain the relationship between the value of that eigenvalue and the maximum and average degrees of each network.

Part II. Undirected Graphs (20 Points)

"Assignment1_Part2_lesmis_data.gml" contains the weighted network of character co-appearances in Victor Hugo's novel "Les Miserables". Nodes represent characters as indicated by the labels and edges connect any pair of characters that appear in the same chapter of the book. The values on the edges are the number of such co-appearances.

1. (3 points) Is this undirected graph connected or not?
2. (7 points) Show the distribution of the shortest path lengths between every pair of nodes in the graph. For this part, you should not use the weight of the edges (i.e., the cost of every edge should be set to 1). Plot the distribution (histogram) of the shortest path lengths across all node pairs. (Check barplot or histogram functions provided by Matplotlib)
3. (10 points) This part of the assignment relates to random walks on graphs. Suppose you initially meet Jean Valjean. Then you randomly move on the graph to meet other characters based on the co-appearance edge weights. If you are currently at node u and (u,v) is an edge with node v, then:

   Probability of visiting v from u = the weight of edge(u,v) / sum of all weights adjacent to u

   For example, the total weight of all edges of Valjean is 158 and the weight of the edge between Cosette and Valjean is 38. In this case, the probability of moving to Cosette in the next step is 31/158. Who are the top three characters in the novel based on the three highest node probabilities in the random walk stationary distribution?

Part III. Directed Graphs (20 Points)

"Assignment1_Part3_drosophila_medulla_data.graphml" is the dataset of synapses among neurons in the Drosophila optic medulla. The nodes represent neurons and the edges are chemical synapses from a neuron to the other. Parse the dataset into a directed graph. Check functions for connected components and shortest paths provided by NetworkX.

1. (5 points) Compute the weakly connected components of this network. What is the percentage of nodes in the largest weakly connected component? (Round to second decimal place)
2. (5 points) Compute the strongly connected components of this network. What is the percentage of nodes in the largest strongly connected component? (Round to second decimal place)
3. (10 points) Compute the shortest path length between every pair of nodes in the largest strongly connected component using Dijkstra's algorithm ignoring the link weights. Show the distribution of shortest path lengths (histogram).

Part IV. Bipartite graphs (20 Points)

“Assignment1_Part4_github_data.txt” represents a bipartite user-project membership network for Github. An edge means that the corresponding user has a membership in the corresponding project. The format of each line in the text file consists of two numbers separated by a space. The left number corresponds to the user ID and the right number corresponds to the project ID.

(10 points) Load the text file and parse the related bipartite graph. First, check that there are no edges between nodes of the same type. Then, compute the two one-mode projections using the adjacency matrix.

CAUTION: There are several different ways of matrix multiplication but, If you want to use “@” operator, your python version should be equal or higher than 3.5 *(HINT : bipartite.projected_graph(), bipartite.biadjacency_matrix() from networkX)*

1. (5 points) Find the pair of users that share most Github projects.
2. (5 points) Find the pair of projects that share most users.

Part V. Directed Acyclic Graphs (20 Points)

“Assignment1_Part5_language_data.txt” represents the relationships of influence among computer programming languages. Each line in the file consists of two programming languages separated by a space, and it means that the programming language at the left is influenced by the programming language at the right. Check functions for directed acyclic graphs in NetworkX.

1. (5 points) Parse this dataset into a directed graph and then check whether it is a DAG or not.
2. (5 points) If the graph is not a DAG, find all cycles in the graph and remove the first edge from each cycle in alphabetical order. For example, if a cycle consists of the edges ('Basic', 'COBOL') and (‘COBOL’,’Basic’), remove (`Basic’, `COBOL’).
3. (10 points) Perform topological sorting on the resulting DAG. How many “source” languages are there in this DAG? Which source language had the highest influence, either direct or indirect?