CSE559
Winter 2020

Assignment 1

Mining Large Networks
Dr. Tanmoy Chakraborty

Due: 2^{nd} February 2020 **Total: 150 points**

Instructions

- Institute Plagiarism Policy applies to all HWs.
- It has been made explicit wherever usage of `networkx` or `snap.py` is required. In all other cases, these libraries can only be used for loading and representing the graphs. The implementations need to be done from scratch.
- **Submission Instructions:**
  - All the submissions must be inside a `zip` file named `a1_<your_roll_number>.zip` containing a report named `report.pdf` and a folder named `src` containing all your scripts.
  - All the scripts written must be uploaded in a `.py` format file. Create separate scripts for each question and name them accordingly. For example, the code corresponding to `Problem 1a` should be named `problem_1a.py`
  - All the code must be well documented. Failure to do so will result in a deduction of 2% of the total from the obtained marks in the respective question.
  - All the plots and analyses required should be uploaded in a `PDF` file named `report.pdf`.

Problem 1 (50 points)

Problem 1a (30 points)

We had discussed various Network Analysis tools in class, like `networkx` and `snap.py`. This problem is meant for you to get familiar with these tools.

Use the Wikipedia voting network given in the file `wiki-Vote.txt` for this problem.

Many of you asked the TA how to go about your project. The idea is to model the some data source as a network and perform some task/analysis on it. Let us see an extremely scaled down version of this in this question.

To be rigorous, we will model this network, G as a directed graph, with V as the set of vertices and $E \subset V \times V$ as the set of edges. An edge (a, b) means a user a voted user b. Additionaly, since the graph is directed, E is set of ordered 2-tuples of vertices (tl;dr, $(a, b) \neq (b, a)$).

Using any of the frameworks you are comfortable with, compute and print the following statistics for this `wiki-Vote` network:

1. Total number of nodes
2. Number of nodes having a self-loop
3. Number of edges in the graph that are not self-loops
4. Number of unique pairs of vertices having an edge between them
5. Number of edges in G, such that $(a, b) \in E \iff (b, a) \in E$
6. Number of nodes with zero in-degree
7. Number of nodes with zero out-degree

8. Number of connected components in the network
9. Number of nodes with in-degree > 10
10. Number of nodes with out-degree > 10

Problem 1b **(20 points)**

1. Plot the out-degree distribution of the `wiki-Vote` network on a log-log scale. For plotting on a log scale, you may use take log with base 10.
2. Use linear regression to compute and plot the line that best fits the out-degree distribution on a log-log scale. (Hint: Essentially, you have to compute some m and c, such that $\log_{10} y = m \log_{10} x + c$, where y is the out-degree distribution and x is the number of nodes having out-degree distribution as y.)

Problem 2 **(50 points)**

Problem 2a **(20 points)**

You must have a file named `so.txt` inside your homework bundle. Each line in this file contains ids of two nodes in a network that have a directed edge between them. This network is of logs from StackOverflow. An edge from node a to node b implies that node a upvoted an answer of node b on StackOverflow.

Again using any of the tools that you are comfortable with, compute the following:

1. Number of weakly connected components and number of strongly connected components in the network
2. Number of nodes and edges in the largest weakly connected component
3. Plot the distribution of pagerank scores of nodes in the network. Which node has the highest pagerank score?
4. Run the HITS algorithm on this network and report the top 5 Hubs and top 5 authorities in this network.

Problem 2b **(30 points)**

Implement the HITS algorithm without using HITS functions of any of the libraries. You may load the graph using any of the libraries and use it to facilitate your implementation.

Your implementation should be a function `HITS(G, max_iters, tol)`. The convergence criteria is defined using `max_iters` and `tol`. If the error (L_1 norm of the difference between successive hub scores) is less than `tol` or the algorithm runs for `max_iters` iterations, it converges. Initialise the hub and authority scores with $\frac{1}{\mathtt{n}}$ where `n` is the total number of nodes in `G`.

After implementing, compute the following:

1. Report and compare the top 5 Hubs and top 5 authorities in your implementation with those obtained from the 4th part of `Problem 2a`.
2. Set `max_iters` to 500. Report the L_2 norm of the difference between estimated (implemented by you) and true (library implementation) hub and authority scores.

Problem 3 (50 points)

Problem 3a (5 points)

An example of a Random Network model we studied in class was the Erdős-Rényi model. Use `networkx` or `snap.py` to generate an instance of a random graph `G(N,L)` using the Erdős-Rényi model where `N = 5242` and `L = 14484`, without using their functions from these libraries. Assume all edges to be undirected.

Problem 3b (10 points)

Random networks can be created using a variety of different methods. Given the description below, generate an instance of a random undirected graph taking `N = 5242` nodes -

- Connect the nodes to make the entire graph look like a cycle. This can be done by ordering the nodes and connecting them to their direct neighbours - for example, node x will be connected to nodes $x-1$ and nodes $x+1$.
- Next, connect each node to it's two-hop neighbours - for example, node x will also be connected to nodes $x-2$ and nodes $x+2$.
- Further, randomly draw 4000 edges which are not yet constructed.

Problem 3c (5 points)

Given the undirected network in `arxiv.txt`, read the graph. This graph represents author collaborations on arXiv. An edge exists between authors if they have collaborated on atleast one paper. Eliminate repeats and self-edges from this graph.

Problem 3d (15 points)

Write a function to plot the degree distributions of the graphs from `Problem 3a`, `Problem 3b` and `Problem 3c`. Plot them in the same plot (use the *log-log* scale). Analyse and report differences between the degree distributions of the three graphs.

Problem 3e (15 points)

Implement the average clustering coefficient and report it for all the three graphs from from `Problem 3a`, `Problem 3b` and `Problem 3c` without using any of the libraries. Report the graph which has the highest clustering coefficient and why.