Network Properties in Spark GraphFrames

In this project, you will implement various network properties using pySpark and GraphFrames. You may need to look into the documentation for GraphFrames and networkx to complete this project. In addition to implementing these network metrics, you will be required to answer some questions when applying these metrics on real world and synthetic networks.

Submission Requirements

Please create a zip file containing ONLY the following for this project :
0. Do NOT include the data or any extra files.
1. degree.py
2. centrality.py
3. articulations.py
4. A pdf document containing answers to the questions asked in the project description. [Power law questions for Stanford and the 4 random graphs provided in degree.py, answers for the two questions on centrality, answer for the closeness question). Name this as <your_unity_id>.pdf
5. [OPTIONAL] Script for checking power law (if you have written something). Instructions to run in the pdf document which contains the answers to the questions asked in project description.
6. Output for centrality.py saved in a csv file called centrality_out.csv
7. Output for articulations.py saved in a csv file called articulations_out.csv

For articulations problem, you may just run it with networkx approach (5 mins run time) and save the output in the file articulations_out.csv.

We will be evaluating the project using a script. So please make sure you follow the instructions, or your assignment might not get graded.

Degree Distribution

The degree distribution is a measure of the frequency of nodes that have a certain degree. Implement a function **degreedist**, which takes a GraphFrame object as input and computes the degree distribution of the graph. The function should return a DataFrame with two columns: **degree** and **count**.

For the graphs provided to you, test and report which graphs are scalefree, namely whose degree distribution follows a power law, at least asymptotically. That is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as

$$P(k) \sim k^{-\gamma}$$

where γ is a parameter whose value is typically in the range $2 < \gamma < 3$, although occasionally it may lie outside these bounds.

Answer the following questions:

1. Generate a few random graphs. You can do this using networkx's random graph generators. Do the random graphs you tested appear to be scale free? (Include degree distribution with your answer).
2. Do the Stanford graphs provided to you appear to be scale free?

Note that GraphFrames represents all graphs as directed, so every edge in the undirected graphs provided or generated must be added twice, once for each direction. When finding if the degrees follow a power law you can then use either the indegree or outdegree, as they should be equal.

Centrality

Centrality measures are a way to determine nodes that are *important* based on the structure of the graph. Closeness centrality measures the distance of a node to all other nodes. We will define the closeness centrality as

$$CC(v) = 1/\sum_{u \in V} d(u,v)$$

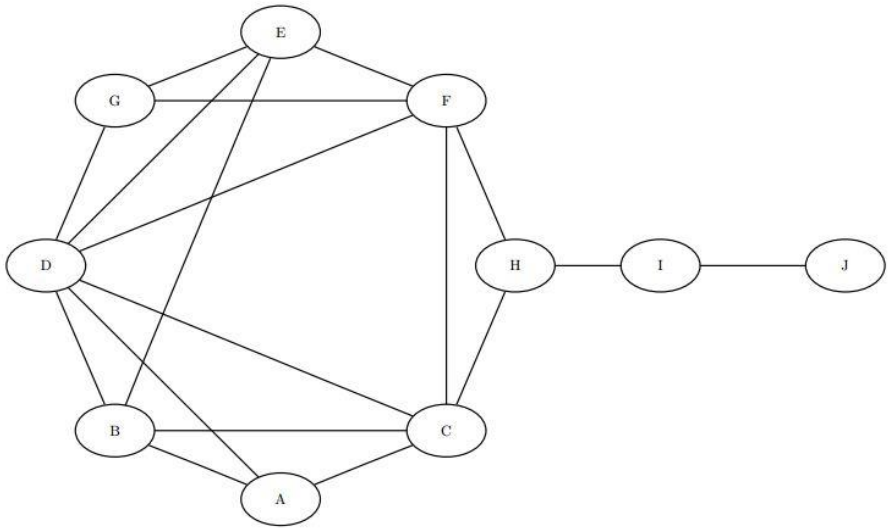
where $d(u, v)$ is the shortestpath distance between u and v.

Implement the function **closeness**, which takes a GraphFrame object as input and computes the closeness centrality of every node in the graph. The function should return a DataFrame with two columns: **id** and **closeness**.

Consider a small network of 10 computers, illustrated below, with nodes representing computers and edges representing direct connections of two machines. If two computers are not connected directly, then the information must flow through other connected machines.

Answer the following questions about the graph:

1. Rank the nodes from highest to lowest closeness centrality.
2. Suppose we had some centralized data that would sit on one machine but would be shared with all computers on the network. Which two machines would be the best candidates to hold this data based on other machines having few hops to access this data?



Articulation Points

Articulation points, or cut vertices, are vertices in the graph that, when removed, create more components than there were originally. For example, in the simple chain 1-2-3, there is a single component. However, if vertex 2 were removed, there would be 2 components. Thus, vertex 2 is an articulation point.

Implement the function **articulations**, which takes a GraphFrame object as input and finds all the articulation points of a graph. The function should return a DataFrame with two columns, **id** and **articulation**, where articulation is a 1 if the node is an articulation point, otherwise a 0.

Suppose we had the terrorist communication network given in the file `9_11_edgelist.txt` . If our goal was to disrupt the flow of communication between different groups, isolating the articulation points would be a good way to do this.

Answer the following questions:
1. In this example, which members should have been targeted to best disrupt communication in the organization?