Network Properties in Spark GraphFrames

<u>Degree Distribution</u>

<u>Degree Distribution</u>

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?

Ans 1. Checking for graphs that are scalefree (1-4)
(Also checking for the Stanford graphs)

Filename	Scalefree
gnm1.csv	True
gnm2.csv	False
gnp1.csv	False
gnp2.csv	False
amazon.graph.small.csv	True
amazon.graph.large.csv	True
dblp.graph.small.csv	True
dblp.graph.large.csv	True
youtube.graph.small.csv	True
youtube.graph.large.csv	True

Ans2. The Stanford graphs are all scale free as can be seen from the above table.

amazon.graph.small.csv	True
amazon.graph.large.csv	True
dblp.graph.small.csv	True
dblp.graph.large.csv	True
youtube.graph.small.csv	True
youtube.graph.large.csv	True

<u>Centrality</u>

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?

Ans 1. To rank the nodes from highest to lowest closeness centrality, we use output of centrality.py

id	closeness
C	0.071429
F	0.071429
D	0.066667
H	0.066667
B	0.058824
E	0.058824
A	0.055556
G	0.055556
I	0.047619
J	0.034483

Ans 2. The best candidates to hold this data based on other machines having few hops to access would be the machines with highest value in closeness, which in our table generated in answer 1 shows that ids - : C,F have highest closeness(0.071429, 0.071429) columns

Articulation

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?

Ans 1. The members that should have been targeted to best disrupt the communication in the organization are the ones with articulation value '1'. We can retrieve their id's using articulation.py and output generated from it will be in table format like:

	id	articulation
0	Mohamed Atta	1
1	Usman Bandukra	1
2	Mamoun Darkazanli	1
3	Essid Sami Ben Khemais	1
4	Djamal Beghal	1
5	Nawaf Alhazmi	1
6	Raed Hijazi	1