Task 10: Properties of networks

The data

The network, simple_network.edges, is derived from:

http://snap.stanford.edu/data/egonets-Facebook.html

It is described in:

`J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012`

Step 1: Graph visualisation

In this step you are given an undirected, unweighted graph and asked to visualise and calculate various network statistics. To visualise your graph, you will first need to download Gephi. You can find a download for your OS here: https://gephi.org/users/download/. Once downloaded, load the simple_network.edges file and select the undirected option. Use the ForceAtlas 2 layout (see the left side of the screen) to arrange the graph - several clusters should appear in the network.

You will now ask Gephi to determine three statistics for you. Run the network diameter statistic (right hand side) - you should get 8. This is a measure of the longest shortest path between any two vertices i.e., $\max_{u,v} d(u, v)$ where $d(u, v)$ is the shortest path between any two vertices u and v. The next two network statistics are vertex degree and betweenness centrality. The degree of a vertex is the number of edges incident to it. Betweenness centrality is a measure of centrality that will be covered in detail in the next session. Use the appearance pane (top left) to give your vertices a size that depends on their degree attribute and a colour that depends on their betweenness centrality attribute. The resulting graph should give you a good idea of which vertices are the most significant in the network.

How does this correspond to what you were told about how the network was created? Is anything surprising?

The rest of this practical requires you to implement the network diameter and vertex degree statistics yourselves.

Step 2: Get degree of all nodes

Write code to load the network as an undirected graph and calculate the degree of each vertex, i.e., the number of neighbours it has.

Step 3: Find diameter.

Find the longest shortest path in the network. You should implement a simple breadth-first search and iterate it over the nodes. Make sure to take note of visited vertices to prevent looping. Note that there is limited memory on the execution server, so you shouldn't try to store all the distances, just keep the maximum distance you find for each breadth-first search. We're setting an upper bound on the heap space allocation of your program to 50mb. To ensure you don't run over this, you can run your local implementation with a memory restriction e.g., "java -Xmx50m uk.ac.cam.cl.mlrd.testing.Exercise10Tester". You shouldn't need anywhere near 50mb. If you run over this, you're not using memory efficiently in your program.

Starred tick

Two further networks are available as edgelist files on Moodle:

phenomenology.edges, derived from:

http://snap.stanford.edu/data/ca-HepPh.html

theory.edges, derived from:

http://snap.stanford.edu/data/ca-HepTh.html

1. Read the descriptions of the networks on the SNAP site.
2. Investigate the properties of each network. For instance, do they each form a connected graph? If not, is there a **giant component** (see slides)? What is the mean degree of each node?
3. Are there differences between the two networks? If so, what does this suggest about collaboration in these two subareas of physics?