Node centrality assignment

For this assignment we will be exploring node centrality in an effort to get an intuitive feel for what the various centrality metrics tell us about the nodes in the graph. For this assignment we will be using the Dolphin social network (http://www-personal.umich.edu/~mejn/netdata/dolphins.zip). Download the graph and load it as a networkx graph using the networkx functions (https://networkx.github.io/documentation/networkx-1.10/reference/readwrite.html). If you have any difficulty loading in a graph you can attempt to load it into Gephi and then save it as a .net (pajek) and then load that version into networkx.

In [1]:

```
# Load your graph
import networkx as nx
fname = 'dolphins.net'
g = nx.read_pajek(fname)
g = nx.Graph(g)
print (nx.info(g))
#print (nx.degree(g))
```

```
Name:
Type: Graph
Number of nodes: 62
Number of edges: 159
Average degree:   5.1290
```

Centrality in Networkx

Networkx has several functions available for calculating the centralities of the nodes in the graph. There are functions for eigenvector (http://networkx.readthedocs.io/en/networkx-1.11/reference/generated/networkx.algorithms.centrality.eigenvector_centrality.html), katz (http://networkx.readthedocs.io/en/networkx-1.11/reference/generated/networkx.algorithms.centrality.katz_centrality.html), closeness (http://networkx.readthedocs.io/en/networkx-1.11/reference/generated/networkx.algorithms.centrality.closeness_centrality.html), betweenness (http://networkx.readthedocs.io/en/networkx-1.11/reference/generated/networkx.algorithms.centrality.betweenness_centrality.html), degree (http://networkx.readthedocs.io/en/networkx-1.11/reference/generated/networkx.algorithms.centrality.degree_centrality.html), etc. For a full list you can visit the documentation page (http://networkx.readthedocs.io/en/networkx-1.11/search.html?q=centrality&check_keywords=yes&area=default). The functions take a graph as an argument and return a dictionary with nodes as keys and the centrality as values. This is convenient for us because we can set these as attributes for the nodes in the graph using the `set_node_attributes` (https://networkx.github.io/documentation/networkx-1.9/reference/generated/networkx.classes.function.set_node_attributes.html) function. For example:

In [2]:

```
import networkx as nx

my_graph = nx.erdos_renyi_graph(500, 0.3)

# Get the eigenvector centralities for all the nodes
centralities = nx.eigenvector_centrality(my_graph)

# Set the attributes of the nodes to include the centralities
# The arguments are: <graph> <attribute key> <values>
# Where <values> is a dictionary with keys=nodes
nx.set_node_attributes(my_graph, "eigenvector", centralities)

# Now we can refer to the node's attributes in the graph
print(my_graph.node[3]["eigenvector"])
```

```
0.05075714742875718
```

We want to do this so that we can export our graph as a `gexf` file using networkx's `write_gexf` (http://networkx.readthedocs.io/en/networkx-1.11/reference/generated/networkx.readwrite.gexf.write_gexf.html) function. Gexf is able to contain a lot more information than other graph datatypes like pajek. It can contain information about the node attributes or edge attributes that belong to the graph and then these attributes will be recognized by Gephi for plotting.

Once the graph is saved and you open it in Gephi, you can use the node (or edge) attributes to control node (or edge) size and color. This can be done by clicking the refresh button by the drop-down menu for node/edge sizing and coloring (refer back to the Gephi tutorial if you are unfamiliar with this). Your attributes will be loaded in using whatever name you used as an `<attribute key>`

Choose a visually appealing layout and then arrange your nodes accordingly and then save separate visualizations that only change the node color/size according to your saved attributes. You will be using this ability for the following questions.

What to submit: Turn in a PDF that contains your short responses and the visualizations for each of the following questions. **Keep the node location the same** for your graph visualizations.

Picking the right Dolphins

Answer the following questions:

(1) Popularity contest

We want to know who the top dolphins are in the network, the real centers of attraction. Using what you learned about centrality from the readings and videos, choose an appropriate centrality measure that will tell us who those dolphins are. Justify your decision and list who the important dolphins are.

(2) Relay

Dolphins like passing information around efficiently along the shortest-paths. Among their neighbors who are the most important message relayers in the network? Justify your centrality choice for finding these dolphins.

(3) Gossip

There is a lot smack going around the pod and everyone wants to know if Flipper will be inviting them to the party next week. But gossip takes time travel. Which dolphins are in the best position for getting all the best gossip from around the pod? Justify your centrality choice for finding these dolphins.

In [3]:

```
# (1) Popularity contest
# Get the katz centralities for all the nodes
katz = nx.katz_centrality(g)
nx.set_node_attributes(g, "katz", katz)
print(g.node['SN90']["katz"])
```

```
0.11169814557069989
```

In [4]:

```
# (2) Relay
# Get the betweenness centralities for all the nodes
between = nx.betweenness_centrality(g)
nx.set_node_attributes(g, "betweenness", between)
print(g.node['SN90']["betweenness"])
```

```
0.02325160067018617
```

In [5]:

```
# (3) Gossip
# Get the closeness centralities for all the nodes
close = nx.closeness_centrality(g)
nx.set_node_attributes(g, "closeness", close)
print(g.node['SN90']["closeness"])
```

```
0.2975609756097561
```

In [6]:

```
# save the graph with attributes and work on the visualization
nx.write_gexf(g, 'dolphins w centralities.gexf')
```

In [ ]:

```

```