SI608 – Networks PoHeng Chen (pohechen)

Assignment 1 (Due: Jan. 31 by 5:30pm (before class))

Instructions: Please submit as PDF attachment via Canvas. You may turn in python code as supplementary information.
Collaboration policy: You may work with others on this but specific answers need to be your own. Please indicate with whom you worked when you turn in this assignment.

Problem 1) Real World Networks (10 points)

Go to the site http://www.visualcomplexity.com. Select and list two projects describing a network. Answer the following questions about them (this may require going into the source webpage for the project, linked to from the visualcomplexity site).

Project 1: The Interconnected Web of Tech Companies (Mashable, 2011)
Project 2: World Economic Forum (Moritz Stefaner, 2013)

(a) What do the nodes represent?
Project 1: Each node represents a technology company such as Paypal, SpaceX, Tesla, Youtube and etc.
Project 2: Each node represents a kind of potential risks which experts predict the world will be facing such as Global Governance Failure, Interstate Conflict, Terrorist Attack, Liquidity Crisis and etc.

(b) What do the edges represent?
Project 1: Each **directed edge** represents why one company has relationship with another such as Instagram's Co-founded by former Google employee Kevin Systrom.
Project 2: Each **weighted edge** represents the relationship between two potential risks and the weight of it such as Global Governance Failure is connected to Terrorist Attack.

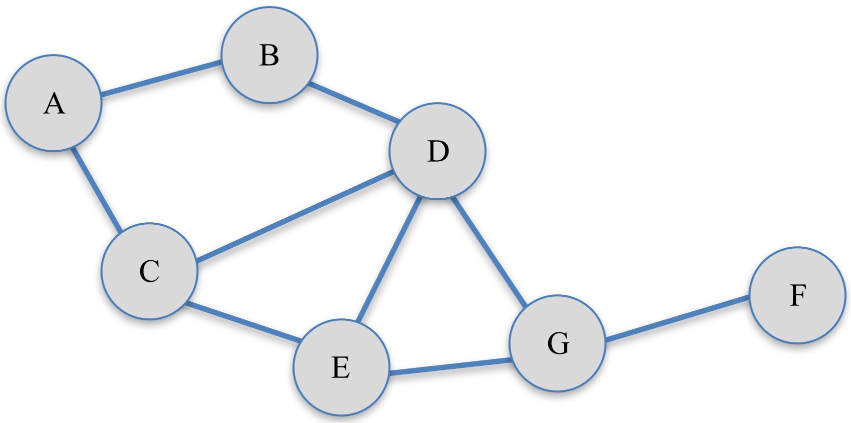
(c) Is the graph directed? Weighted?
Project 1: The graph is directed but not weighted because it is unnecessary to measure the weight between two companies' relationship. If they have relationship, then it is the case.
Project 2: The graph is not directed but weighted because it can take how many experts say the potential risk of A and B as the weight. I also think the graph can be directed because the Global Governance Failure might be caused by Terrorist Attack or Corruption but not likely vice versa.

(d) Can the data be represented as a bipartite graph? If so, what are the two groups of vertices?
Project 1: The data cannot be represented as a bipartite graph because there is an odd cycle among Microsoft, TellMeNetworks and Swipely.

Project 2: The data also cannot be represented as a bipartite graph because there are lots of odd cycles in it. For example, one odd cycle in Fiscal Crisis, Failure of Financial Mechanism or Institute and Liquidity Crisis and another one in Climate Change, Water Crisis and Extreme Weather Events.

Problem 2) Network Basics (25 points)

Below is an undirected network with eight(???) vertices



(a) Please represent this network with an adjacency matrix

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

(b) Find the degree of Vertex A, C, and B
Degree of a node is the number of edges connected to it
Vertex A: 2 degrees, Vertex B: 2 degrees, Vertex C: 3 degrees

(c) Write down the degree sequence of this network
{4, 3, 3, 3, 2, 2, 1}

(d) The deletion of which vertex will make the network unconnected?

It is G.

(e) The deletion of which vertex will make the network a tree?
Tree has no cycle. The deletion of D will make it a tree.

(f) What are the density and the diameter of this network? Explain your answer.
A network density is the number of actual ties over the number of possible ties.
The density is 9 / [(6+1)*6]/2 = 18 / 21 = 3 / 7 = 0.429

The diameter is the largest geodesic distance in the graph.
Is is 4 from A to F.

(g) Calculate the i) closeness, ii) normalized closeness, iii) betweenness, and iv) normalized betweenness for nodes D and G. Do not do this in code and show your work.
i) The closeness is how a node is to other nodes in a network by measuring the sum of the shortest distance (geodesic distance).

$$C(x) = \frac{1}{\sum_y d(y,x)}$$

where d(y,x) is the distance between vertices x and y.

Node D: 1 / (1+2+1+1+1+2) = 1 / 8 = 0.125
Node G: 1 / (1+1+1+2+2+3) = 1 / 10 = 0.1

ii) Normalised closeness: the average length of the shortest path between the node and all other nodes in the graph.
Normalised closeness of D = 6 / 8 = 0.75
Normalised closeness of G = 6 / 10 = 0.6

iii) Betweenness: the number of times a node acts as a bridge along the shortest path between two other nodes.

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

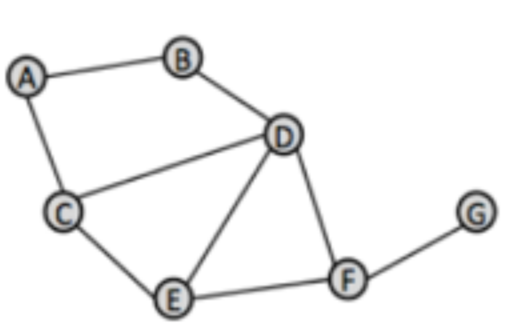
where sigma_st is the total number of shortest paths from node s to node t and sigma_st(v) is the number of those paths that pass through v.

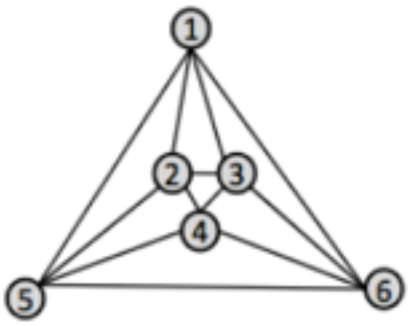
Betweenness of D = 35/6
It is a bit complicated. The cases have: A-F, A-G, B-G, B-F, B-E, B-C, C-G, C-F
Betweenness of G = 5

iv) Normalised betweenness:
Normalised betweenness of D = (35/6)/ 15 = 13/45 = 0.3889

Normalised betweenness of G = 5 / 15 = 0.333

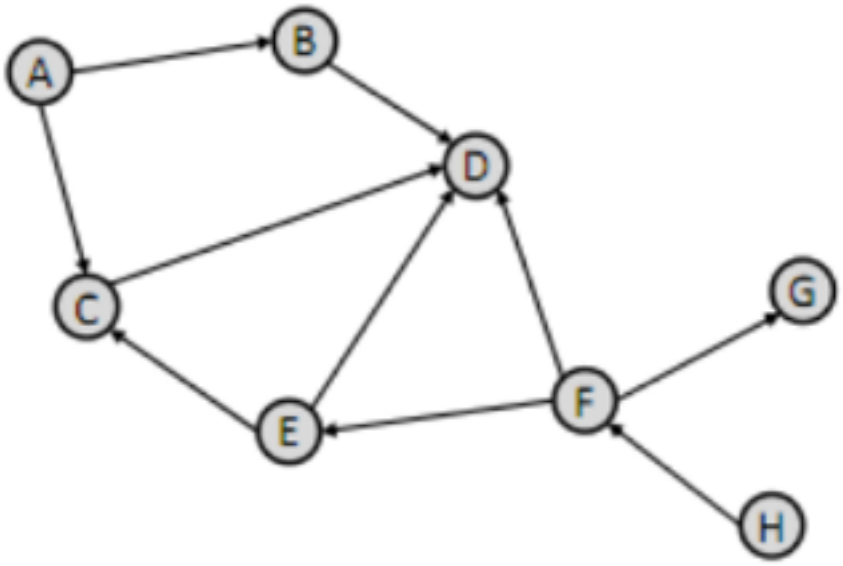
Problem 3) Centralization (10 points)





Which of the two networks has a larger centralization? Left or right? Why?
The left one has a larger centralisation.
The degree of centralisation of left one is: [(4-2)+(4-2)+(4-3)+(4-3)+(4-3)+(4-4)+(4-1)]/[(7-1)(7-2)] = 1/3 = 0.3333
The degree of centralisation of the right one is: [(4-4)*6]/[(6-1)(6-2)] = 0
The right one is more evenly distributed than the left one so it has lower centralisation.

Problem 4) Prestige (20 points)



(a) Label the influence range of the vertex C in the graph above (List the vertices in its input domain). (Hint: see slides for week 3)
The influence range of vertex i is the set of vertices who can reach node i: {A, E, F, H}

(b) Which vertex is in D's input domain but not in C's? Which vertices (or vertex) are in neither of their input domains?
1. D's influence range is vertices of {A, B, C, E, F, H} so {B, C} are not in C's input domain.
2. G is not in neither of them.

(c) Find the proximity prestige of C (show your work).
We compute proximity prestige (PP) of selected vertex by dividing the influence domain of a vertex (expressed as a proportion) by the average distance from all vertices in the influence domain.
Proximity prestige of C = (4/7) / ((1+1+2+3)/4) = 16/49 = 0.327

(d) Find the normalized betweenness of E (show your work).
It is directed case of normalised betweenness:
The cases are: H to C, F to C.
The normalised betweenness of E = 2 / (7*6) = 1/21 = 0.048

Problem 5) Real Data (35 points)

For this problem you will need the file "sj.gml". It is a modified file from this dataset: http://vlado.fmf.uni-lj.si/pub/networks/data/esna/SanJuanSur.htm

Essentially, visits between families in a particular village were tracked. Based on the number of visits, each household was assigned a "status score" ranging from 1 to 14 (there is also a leadership nomination field, but we'll ignore that for now). The question we ask in this exercise is: which centrality metric best correlates to the status score.

Please turn in working code for this as well as the specific answers

a) Load the gml file into networkx
(Hint: http://networkx.lanl.gov/reference/readwrite.gml.html)

code:

```
## for the series of Problem 5, I put my code in a .py file named
## hw1_pohechen.py for your reference

import networks as nx
```

g = nx.read_gml('sj.gml')

note that this should create a directed graph for you (rather than an undirected one). You shouldn't have to do anything special for this, since it's specified in the GML file.

b) Calculate betweenness centrality for this network (see lab 2 for help if you can't remember). Once you have that, calculate the pearson or spearman correlation between the status field and the betweenness centrality. This isn't strictly "kosher" from the stats point of view, but it'll do for now. Report the correlation and p-values. Happily, Python will do most of the work for you. Once you have two arrays (one with the status scores and one with the centrality scores) you should be able to run:

```
# load the scipy statistics package
import scipy.stats
# calculate the correlation and report the p-value
scipy.stats.pearsonr(array1,array2)

g.between = nx.betweenness_centrality(g)
```

cores and on

```
g.between = nx.betweenness_centrality(g)

g.id = []
g.status = []
g.betweenness = []
for i in g.nodes(True):
    g.id.append(i[0])
    g.status.append(i[1]['status'])
    g.betweenness.append(g.between[i[0]])

import scipy.stats
scipy.stats.pearsonr(g.status, g.betweenness)

# In[199]: scipy.stats.pearsonr(g.status, g.betweenness)
# Out[199]: (0.048755484710749537, 0.67785625033568242)
```

The pearson correlation is 0.049
The p-value for it is 0.678

c) Repeat b for some other centrality metrics. Again, report the correlation and p-values

```
In[221]: scipy.stats.pearsonr(g.status, g.degree_centrality)
Out[221]: (0.36669299388965881, 0.0012129232372535951)

In[222]: scipy.stats.pearsonr(g.status, g.closeness)
Out[222]: (-0.2489858659672779, 0.031231350625960184)
```

d) Unfortunately, networkx doesn't implement proximity prestige so we'll need to do it ourselves. Luckily it shouldn't take more than a dozen lines of code. Recall that a person's prestige is the ratio of the percent of the graph that can communicate with that node divided by the average number of steps.

Thankfully. Networkx takes care of most of this. First, calculate the shortest paths between all nodes in the network (hint: http://networkx.lanl.gov/reference/generated/networkx.algorithms.shortest_paths.weighted.all_pairs_dijkstra_path.html)

```
g.sum_shortest_paths = {}
g.all_pairs = nx.all_pairs_dijkstra_path_length(g)
for i in g.all_pairs:
    for j in g.all_pairs[i]:
        if j in g.sum_shortest_paths:
            g.sum_shortest_paths[j][0] += g.all_pairs[i][j]
            g.sum_shortest_paths[j][1] += 1
        else:
            g.sum_shortest_paths[j] = [g.all_pairs[i][j], 0]
```

Note: you may find it easier to think about the graph in reverse (changing the directions of the edges. You can do this pretty easily: http://networkx.lanl.gov/reference/generated/networkx.DiGraph.reverse.html)

e) Write code to compute the number of nodes that can "reach" any given node in our network. Divide that by the total number of nodes in the network to get the percent. If you did (d) correctly, this is probably 3-4 lines of code

f) Expand (e) so that you calculate the average number of hops between a particular node and all reachable nodes and then calculate the ratio between the percent in (e) and the average number of hops.

```
# I combined the codes for (e) and (f) together.
g.prestige = []
for i in g.sum_shortest_paths:
    if g.sum_shortest_paths[i][1] == 0:
        g.prestige.append(0)
    else:
```

```
denominator = g.sum_shortest_paths[i][0]*(len(g.sum_shortest_paths)-1)
numerator = g.sum_shortest_paths[i][1]**2
g.prestige.append(numerator/denominator)
```

g) Calculate the correlation between this new prestige score and the status field.

```
In[376]: scipy.stats.pearsonr(g.status, g.prestige)
Out[376]: (0.26436821133298227, 0.021906612208094371)
```

h) Do you have any intuition about why the best centrality metric was the best?
The best centrality metric should be status scores versus degrees of centrality.
The status scores represent the number of visits from other village to that household. It is very similar to the degree of a node which means how many edges a node has. That's the intuition reason why degree of centrality has the highest correlation with status scores.

Reference

Betweenness Centrality: https://en.wikipedia.org/wiki/Betweenness_centrality
Closeness Centrality: https://en.wikipedia.org/wiki/Closeness_centrality
Networkx: https://networkx.github.io/documentation/networkx-1.9/index.html
Proximity Prestige (page.9): http://mrvar.fdv.uni-lj.si/sola/info4/uvod/part4a.pdf