1. (20 pts) Consider the following simple and rather unrealistic model of a network: each of n vertices belongs to one of g groups. The mth group has n_m vertices and each vertex in that group is connected to others in the group with independent probability $p_m = A(n_m - 1)^{-\beta}$, where A and β are constants, but not to any vertices in other groups. Thus, this network takes the form of a set of disjoint groups of communities.

   (a) Calculate the expected degree $\langle k \rangle$ of a vertex in group m.

$$\begin{aligned}\langle k \rangle &= \sum_{m=1}^{\binom{n_m}{2}} \frac{2m}{n_m} * P(m) \\ &= \frac{2}{n_m}\binom{n_m}{2} p_m \\ &= (n_m - 1)p_m \\ &= A(n_m - 1)^{1-\beta}\end{aligned}$$

   (b) Calculate the expected value $\langle C_m \rangle$ of the local clustering coefficient for vertices in group m.

$$\langle C_m \rangle = p_m = A(n_m - 1)^{-\beta}$$

   (c) Hence show that $\langle C_m \rangle \propto \langle k \rangle^{\frac{-\beta}{(1-\beta)}}$. What value would β have to assume for the expected value of the local clustering coefficient to fall off as $\langle k \rangle^{-0.75}$, as has been conjectured by some researchers?

$$\begin{aligned}\langle k \rangle^{\frac{-\beta}{1-\beta}} &\propto \langle C_m \rangle \\ (A(n_m - 1)^{1-\beta})^{\frac{-\beta}{1-\beta}} &\propto A(n_m - 1)^{-\beta} \\ A^{\frac{-\beta}{1-\beta}}(n_m - 1)^{-\beta} &\propto A(n_m - 1)^{-\beta} \\ A^{\frac{-\beta}{1-\beta}} &\propto A\end{aligned}$$

$$\begin{aligned}\langle k \rangle^{\frac{-\beta}{1-\beta}} &= \langle k \rangle^{-0.75} \\ \frac{-\beta}{1-\beta} &= -0.75 \\ -\beta &= -0.75 + 0.75\beta \\ -1.75\beta &= -0.75 \\ \beta &= 0.428\end{aligned}$$

2. (20 pts) Consider the random graph $G(n, p)$ with average degree c.

   (a) Show that in the limit of large n the expected number of triangles in the network is $\frac{1}{6}c^3$. In other words, show that the number of triangles is constant, neither growing nor vanishing in the limit of large n.

   The expected number of triangles in a network is equal to the n choose 3 for all the ways that 3 nodes can be chosen. Since the three nodes need to be connected, this is multiplied by the probability that all three are connected or the probability that two are connected to the power of three (for the three edges). The probability of an edge existing is equal to the average degree divided by the number of nodes minus 1 (the number of possible options for each edge).

$$Triangles = \binom{n}{3}p^3 = \left(\frac{n!}{(n-3)!3!}\right)\left(\frac{c}{n-1}\right)^3 = \frac{n(n-1)(n-2)}{6}\frac{c^3}{(n-1)^3}$$

$$n \to \infty$$

$$= \frac{n^3}{6}\frac{c^3}{n^3} = \frac{c^3}{6}$$

   (b) Show that the expected number of connected triples in the network is $\frac{1}{2}nc^2$

   A connected triple is defined as three nodes with two edges between them. The expected number of connected triples in a network is the same basic structure as the expected number of triangles. Three nodes are chosen using n choose 3. Since only two edges need to exist, this is multiplied by the probability of an edge to the power of two. However, the three nodes can be a triple if any two of the three edges exist meaning this should be multiplied by 3 choose 2.

$$Triples = \binom{n}{3}p^2\binom{3}{2} = \left(\frac{n!}{(n-3)!3!}\right)\left(\frac{c}{n-1}\right)^2\left(\frac{3!}{2!}\right)$$

$$= \frac{n(n-1)(n-2)}{6}\frac{c^2}{(n-1)^2} * 3$$

$$= \frac{n(n-1)(n-2)}{2}\frac{c^2}{(n-1)^2}$$

$$n \to \infty$$

$$= \frac{n^3}{2}\frac{c^2}{n^2} = \frac{n * c^2}{2}$$

   (c) Hence, calculate the clustering coefficient C, as defined in Eq. (7.28) in Networks, and confirm that it agrees for large n with the value given in Eq. (11.11) in Networks.

$$C = \frac{Triangles * 3}{Triples} = \frac{\frac{c^3}{6} * 3}{\frac{n*c^2}{2}} = \frac{\frac{c}{2}}{\frac{n}{2}} = \frac{c}{n} \approx \frac{c}{n-1}$$

3. (15 pts) Consider an undirected, unweighted network of n vertices that contains exactly two subnetworks of size n_A and n_B, which are connected by a single edge (A, B).

   Show that the closeness centralities C_A and C_B of vertices A and B, as defined by Eq. (7.21) in Networks, are related by

$$\frac{1}{C_A} + \frac{n_A}{n} = \frac{1}{C_B} + \frac{n_B}{n}$$

Defining d_{Aj} as being the geodesic distance between node A and node j. Sum up this value for all nodes to get $\sum_j d_{Aj}$. The goal is to define $\sum_j d_{Bj}$ with this value. Since nodes A and B are connected, the paths from the nodes in n_A to node B will be one greater than the path from the nodes to node A. Thus, $\sum_{j \in n_A} d_B j = \sum j \in n_A(d_A j + 1)$. Similarly, the paths from the nodes in n_B to node B will be one less than the path from the nodes to node A. Thus, $\sum_{j \in n_B} d_B j = \sum j \in n_B(d_A j - 1)$. Since this encompasses all the nodes, sum the two expressions. Then simplify the summations, rearrange and divide by n to put in terms of closeness centralities C_A and C_B. This will result in the above formula as seen below.

$$\begin{aligned}
\sum_{j \in n_A} d_B j + \sum_{j \in n_B} d_B j &= \sum j \in n_A(d_A j + 1) + \sum j \in n_B(d_B j - 1) \\
\sum_j d_B j &= \sum j \in n_A d_A j + n_A + \sum j \in n_B d_A j - n_B \\
\sum_j d_B j &= \sum j d_A j + n_A - n_B \\
\sum_j d_B j + n_B &= \sum j d_A j + n_A \\
\frac{\sum_j d_B j}{n} + \frac{n_B}{n} &= \frac{\sum j d_A j}{n} + \frac{n_A}{n} \\
\frac{1}{C_B} + \frac{n_B}{n} &= \frac{1}{C_A} + \frac{n_A}{n}
\end{aligned}$$

4. (15 pts) Consider an undirected (connected) tree of n vertices. Suppose that a particular vertex in the tree has degree k, so that its removal would divide the tree into k disjoint regions, and suppose that the sizes of those regions are $n_1, \ldots, n_k$. Show that the betweenness centrality b of the vertex is

$$b = n^2 - \sum_{i=1}^{k} n_i^2$$

The total number of geodesic paths in the network is the Cartesian product of all the nodes $n \times n$. This will have a path from every node to every other node. The number of geodesic paths that don't go through the node with degree k can be defined as the number of paths in each of the k regions. For a particular region i, the number of geodesic paths in the region is the Cartesian product of all the nodes in the region $n_i \times n_i$. This will include all the paths from the nodes in the region to other nodes in the region. All other geodesic paths involving the nodes from this region must go through the node with degree k. By doing this for each of the k regions, the sum will be the number of paths that don't go through the node with degree k. By subtracting the number of geodesic paths that don't go through the node from the total number of geodesic paths, the result is number of geodesic paths that go through the node. The betweenness centrality is defined as the number of geodesic paths through the vertex. This results in the following equation:

$$b = n^2 - \sum_{i=1}^{k} n_i^2$$

5. (30 pts) The Medici family was a powerful political dynasty and banking family in 15th century Florence. The classic network-based explanation of their power, offered by Padgett and Ansell in 1993, claims that they established themselves as the most central players within the network of prominent Florentine families.

   (a) Construct a table with four columns, each containing a list of (family, score) pairs for the following centrality score functions:

   - degree centrality,
   - harmonic centrality (Eq. 7.22 in Networks),
   - eigenvector centrality (Eq. 7.2 in Networks),
   - betweenness centrality (Eq. 7.25 in Networks).

   Within each centrality, sort the pairs in decreasing order of importance; three decimal places is sufficient detail.

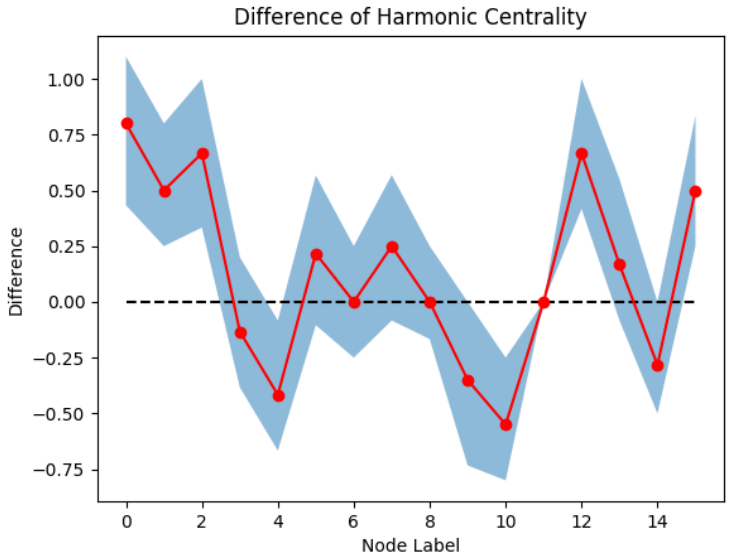
   Discuss (i) the degree to which these scores agree with Padgett and Ansells claim that the Medicis occupied a structurally important position in this network, and (ii) what the scores say about the second most important family.

Degree	Harmonic	Eigenvector	Betweenness
(Medici, 6)	(Medici, 9.5)	(Medici, 0.43)	(Medici, 0.452)
(Guadagni, 4)	(Guadagni, 8.083)	(Strozzi, 0.356)	(Guadagni, 0.221)
(Strozzi, 4)	(Ridolfi, 8)	(Ridolfi, 0.342)	(Albizzi, 0.184)
(Albizzi, 3)	(Albizzi, 7.833)	(Tornabuoni, 0.326)	(Salviati, 0.124)
(Bischeri, 3)	(Strozzi, 7.833)	(Guadagni, 0.289)	(Ridolfi, 0.098)
(Castellani, 3)	(Tornabuoni, 7.833)	(Bischeri, 0.283)	(Bischeri, 0.09)
(Peruzzi, 3)	(Bischeri, 7.2)	(Peruzzi, 0.276)	(Strozzi, 0.089)
(Ridolfi, 3)	(Barbadori, 7.083)	(Castellani, 0.259)	(Barbadori, 0.081)
(Tornabuoni, 3)	(Castellani, 6.917)	(Albizzi, 0.244)	(Tornabuoni, 0.79)
(Barbadori, 2)	(Peruzzi, 6.583)	(Barbadori, 0.212)	(Castellani, 0.048)
(Salviati, 2)	(Salviati, 6.583)	(Salviati, 0.146)	(Peruzzi, 0.019)
(Acciaiuoli, 1)	(Acciaiuoli, 5.917)	(Acciaiuoli, 0.132)	(Ginori, 0)
(Ginori, 1)	(Lamberteschi, 5.367)	(Lamberteschi, 0.089)	(Pucci, 0)
(Lamberteschi, 1)	(Ginori, 5.333)	(Ginori, 0.075)	(Acciaiuoli, 0)
(Pazzi, 1)	(Pazzi, 4.767)	(Pazzi, 0.045)	(Lamberteschi, 0)
(Pucci, 0)	(Pucci, 0)	(Pucci, 0)	(Pazzi, 0)

The Medicis did occupy a structurally important position in the network. They have the highest scores for the 4 measures of centrality meaning that by these 4 methods they are the most central in the network. The second most important family is the Guadagni since they are second in 3 of the 4 scores and fifth in the last. This is better positioning that any other node in the network making them the second most central.

(b) Determine whether the relative structural importance of the Medicis can be explained solely by the degree sequence $\{k_i\}$ of the network G.

Finally, (i) discuss your results here in terms of Padgett and Ansells story of the Medici family, and (ii) comment on whether your results from part (5a) have changed.

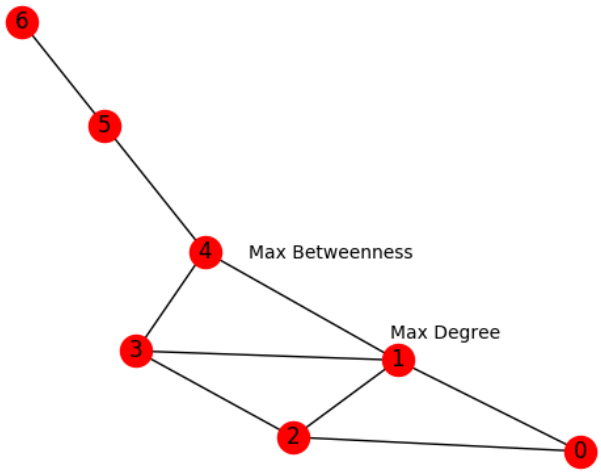


As seen in the graph above, the families have positive differences have a harmonic centrality in the network that is greater than the harmonic centrality for a random network. Node 8 which represents the Medici family has a mean difference of 0. This implies that with any random graph with degree distribution, the score for their harmonic centrality would be about the same as the actual network. This implies that regardless of the connections they made, they would be the same in terms of centrality. This makes Padgett and Ansell's views incorrect. However, since they have the most connections in the network, they may have worked to get those connections which results in their centrality. However, by making this number of connections, they would have been central regardless.

The other most prominent families are nodes 6, 14, 12, and 1. All of these nodes except have a positive or near zero difference meaning that on average, their harmonic centrality would be less than or equal to that of the actual graph. Node 14 (Strozzi) would have a slightly larger harmonic centrality on average. However, since it would have on average a larger score by 0.5, it's average score would be approximately 8.333. This is not greater than the average Medici score of 9.5. Thus, the Medici family is still the most prominent family even in a random network with this degree distribution. However, Strozzi would be the second most prominent family instead of Guadagni on average since the Guadagni's score is 8.083.

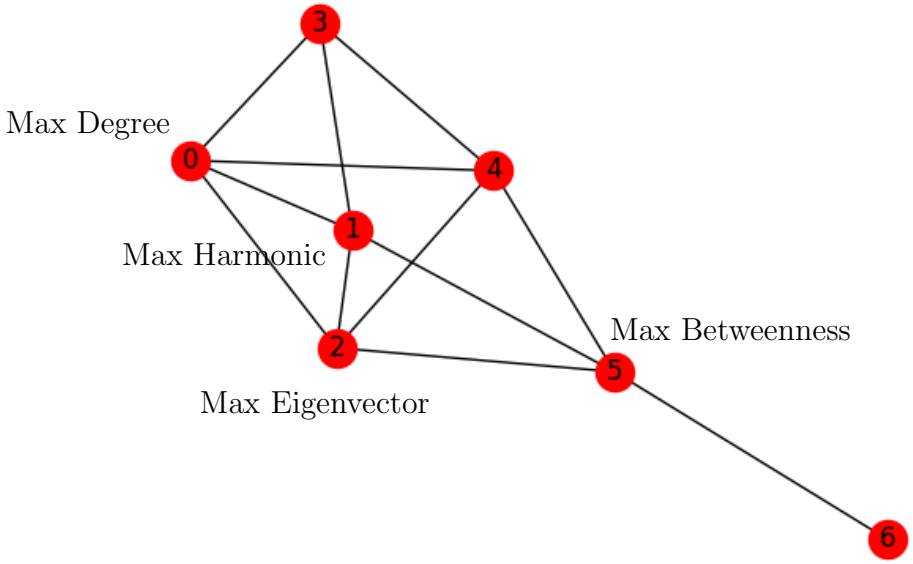
6. (25 pts extra credit) Consider the task of designing a network in which distinct vertices hold the status of highest centrality for some set S of centrality measures. (Ties are prohibited.) If the network is relatively small compared to the number of centrality measures, this can be a very difficult task.

   - As a warm up, design by hand a small network ($n \leq 7$) that has this property with respect to S = { degree, betweenness } centralities. Include a visualization of the network (label all the vertices, and indicate which ones win at which centrality measure) and produce a table in which the columns show the ranked vertices and scores for these centrality measures.



Degree	Betweenness
(1, 4)	(4, 0.533)
(2, 3)	(1, 0.333)
(3, 3)	(5, 0.333)
(4, 3)	(3, 0.1)
(0, 2)	(2, 0.033)
(5, 3)	(0, 0)
(6, 1)	(6, 0)

- Now, write a program to search the space of all possible connected, non-isomorphic networks of a given size n to find a network with this property for $S = \{$ degree, harmonic, betweenness, eigenvector $\}$ centralities, or report that no such network exists for that size. Run this program for $n = 3, 4, 5, \ldots$ and report the smallest value of n for which at least one such network exists. Visualize that network, label all the vertices, and indicate which ones win at which centrality measure; and produce a table in which the columns show the ranked vertices and scores for these centrality measures. Comment briefly about what insights you gained about how to build networks that have distinct winners for different measures of centrality.



The smallest n for which this network exists is 7. Networks of this type have a cluster of nodes that are more connected and a tail (or couple of nodes not very connected to the others). This will allow for some nodes to have more paths through them, others to have greater degrees, and others to be the most central.

Degree	Eigenvector	Betweenness	Harmonic
(0, 4)	(2, 0.439)	(5, 0.350)	(1, 5)
(1, 4)	(0, 0.434)	(1, 0.133)	(2, 5)
(2, 4)	(1, 0.422)	(4, 0.133)	(4, 5)
(4, 4)	(4, 0.422)	(2, 0.061)	(5, 5)
(5, 4)	(5, 0.369)	(0, 0.039)	(0, 4.833)
(3, 3)	(3, 0.341)	(3, 0.017)	(3, 4.333)
(6, 1)	(6, 0.098)	(6, 0)	(6, 3.167)

Algorithm 1: Python Code for Problems 5 and 6

```
import networkx as nx
import copy
import random
import numpy as np
import matplotlib.pyplot as plt

# QUESTION 5a
# read adjacency list
G = nx.read_adjlist("adj.adjlist")
print(G.nodes())
print(len(G.edges()))

# calculate eigenvector centrality
eigenvector = nx.eigenvector_centrality(G)
print(sorted(((v, '{:0.3f}'.format(c)) for v, c in eigenvector.items()), key=lambda x: x[1]

# calculate betweenness centrality
betweenness = nx.betweenness_centrality(G)
print(sorted(((v, '{:0.3f}'.format(c)) for v, c in betweenness.items()), key=lambda x: x[1]

# calculate harmonic centrality
harmonic = nx.harmonic_centrality(G)
print(sorted(((v, '{:0.3f}'.format(c)) for v, c in harmonic.items()), key=lambda x: x[1],re

# QUESTION 5b
# extract degree sequence
degreeSeq = []
for i in G.nodes():
    for j in range(G.degree[i]):
        degreeSeq.append(i)
harmonicDist = [[] for i in range(len(G.nodes))]

# do 1000 instances
for i in range(1000):
    # copy the degree sequence and shuffle until there are no loops or multiedges
    deg = copy.deepcopy(degreeSeq)
    shuffle = True
    while shuffle:
        random.shuffle(deg)
        shuffle = False
        tuples = zip(*[deg[i::2] for i in range(2)])
        for t in tuples:
            if tuples.count(t) + tuples.count((t[1],t[0])) > 1:
                shuffle = True
                break
            if t[0] == t[1]:
                shuffle = True
                break
```

```
    # create a network from the configuration model
    tuples = zip(*[deg[i::2] for i in range(2)])
    Gp=nx.Graph()
    Gp.add_edges_from(tuples)
    Gp.add_nodes_from(G.nodes)

    # calculate the harmonic centrality
    harmon = nx.harmonic_centrality(Gp)
    for j in Gp.nodes():
        harmonicDist[int(j)].append(harmon[j])

# calculate the quartiles
quart1 = []
quart2 = []
quart3 = []
x = range(len(G.nodes))
for i in x:
    h = harmonic[str(i)]
    quart1.append(h - np.quantile(harmonicDist[i], .25))
    quart2.append(h - np.quantile(harmonicDist[i], .50))
    quart3.append(h - np.quantile(harmonicDist[i], .75))

# plot the quartiles for the harmonic centrality differences
plt.figure()
plt.plot(x, [0 for i in range(len(x))], '--k')
plt.plot(x, quart2, 'ro-')
plt.fill_between(x, quart1, quart3, alpha=0.5)
plt.xlabel("Node Label")
plt.ylabel("Difference")
plt.title("Difference of Harmonic Centrality")
plt.savefig("harmonic_dist.png")

# PROBLEM 6a
# create graph from edge list
edgeList = [(1,0), (1,2), (1,3), (1,4), (2,3), (3,4), (5,6), (4,5), (0,2)]
G = nx.Graph()
G.add_edges_from(edgeList)

# calculate betweenness
betweenness = nx.betweenness_centrality(G)
print(sorted(((v, '{:0.3f}'.format(c)) for v, c in betweenness.items()), key=lambda x: x[1]

# plot and label graph
plt.figure()
nx.draw_networkx(G)
plt.draw()
plt.annotate("Max Degree", (0.15, -0.2))
plt.annotate("Max Betweenness", (0,0.1))
plt.axis('off')
plt.savefig("net.png")
```

```
# QUESTION 6b
def findNetWithDiffCentralities(n):
    # get list of all possible edges
    edges = list(combinations(range(n), 2))

    # start with blank assignment, edge 0 and no backtracking
    assignment = []
    i = 0
    backtracking = False
    # iterate until the assignment is blank and backtracking is true
    while assignment or not backtracking:
        # is the assignment hasn't gone through all the variables
        if i < len(edges):
            if backtracking:
                # if backtracking with edge, remove edge, stop backtracking
                if assignment[-1] == edges[i]:
                    assignment.pop()
                    backtracking = False
                    i += 1
                # otherwise continue backtracking
                else:
                    i -= 1
            # add edge to assignment
            else:
                assignment.append(edges[i])
                i += 1
        # if all edges have been considered
        else:
            # check if the centralities are different, if so return assignment
            if diffCentralities(assignment, n):
                return assignment
            # otherwise begin backtracking
            else:
                backtracking = True
                i -= 1
    # all assignments have been considered, indicate none have been found correct
    return None


def diffCentralities(assign, n):
    # create a graph from the assignment
    G = nx.Graph()
    G.add_nodes_from(range(n))
    G.add_edges_from(assign)

    # calculate each centrality and add the max node to an array
    arr = []
    degree = nx.degree_centrality(G)
    degree = sorted(((v, '{:0.3f}'.format(c)) for v, c in degree.items()), key=lambda x: x
```

```
    arr.append(degree[0][0])

    eigenvector = nx.eigenvector_centrality(G)
    eigenvector = sorted(((v, '{:0.3f}'.format(c)) for v, c in eigenvector.items()), key=la
    arr.append(eigenvector[0][0])

    betweenness = nx.betweenness_centrality(G)
    betweenness = sorted(((v, '{:0.3f}'.format(c)) for v, c in betweenness.items()), key=la
    arr.append(betweenness[0][0])

    harmonic = nx.harmonic_centrality(G)
    harmonic = sorted(((v, '{:0.3f}'.format(c)) for v, c in harmonic.items()), key=lambda x
    arr.append(harmonic[0][0])

    # return if the array has no duplicate elements
    return len(set(arr)) == len(arr)

# start with 4 nodes, iterate until found
n = 4
while True:
    # check for valid graph, if so break, otherwise increment n
    assign = findNetWithDiffCentralities(n)
    if assign:
        break
    n += 1

# create graph from assignment
G = nx.Graph()
G.add_nodes_from(range(n))
G.add_edges_from(assign)

# plot graph
plt.figure()
nx.draw_networkx(G)
plt.draw()
plt.axis('off')
plt.savefig("netN.png")
```