Network Analysis and Modeling
CSCI 5352, Fall 2020
Prof. Dan Larremore
Problem Sets 5 and 6
Student: Alex Book; Collaborators: None

1. (15 pts) In a survey of couples in the city of San Francisco in 1992, Catania et al. recorded, among other things, the ethnicity of interviewees and calculated the fraction of couples whose members were from each possible pairing of ethnic groups. The fractions were as follows: Assuming the couples interviewed to be a representative sample of the edges in the undirected network of relationships for the community studied, and treating the vertices as being of four types—black, hispanic, white, and other—calculate the numbers e_{rr} and a_r that appear in Eq. (7.58) in *Networks* [v1: 7.76] for each type. Hence calculate the modularity Q of the network with respect to ethnicity. What do you conclude about homophily in this community?

		Women				
		Black	Hispanic	White	Other	Total
Men	Black	0.258	0.016	0.035	0.013	0.322
	Hispanic	0.012	0.157	0.058	0.019	0.246
	White	0.013	0.023	0.306	0.035	0.377
	Other	0.005	0.007	0.024	0.016	0.052
	Total	0.288	0.203	0.423	0.083	

$e_{rr} = \frac{1}{2m}\sum_{ij} A_{ij}\delta_{g_i,r}\delta_{g_j,r}$
$a_r = \frac{1}{2m}\sum_i k_i \delta_{g_i,r}$

$e_{\text{Black}} = .258$
$e_{\text{Hispanic}} = .157$
$e_{\text{White}} = .306$
$e_{\text{Other}} = .016$

$a_{\text{Black}} = \frac{a_{\text{Black Men}}+a_{\text{Black Women}}}{2} = \frac{(.258+.016+.035+.013)+(.258+.012+.013+.005)}{2} = .305$
$a_{\text{Hispanic}} = \frac{a_{\text{Hispanic Men}}+a_{\text{Hispanic Women}}}{2} = \frac{(.012+.157+.058+.019)+(.016+.157+.023+.0077)}{2} = .2245$
$a_{\text{White}} = \frac{a_{\text{White Men}}+a_{\text{White Women}}}{2} = \frac{(.013+.023+.306+.035)+(.035+.058+.306+.024)}{2} = .4$
$a_{\text{Other}} = \frac{a_{\text{Other Men}}+a_{\text{Other Women}}}{2} = \frac{(.005+.007+.024+.016)+(.013+.019+.035+.016)}{2} = .0675$

$$\begin{aligned} Q = \sum_r (e_r - a_r^2) &= (.258 - .305^2) + (.157 - .2245^2) + (.306 - .4^2) + (.016 - .0675^2) \\ &= (.165975) + (.10659975) + (.146) + (.01144375) \\ &= .4290185 \end{aligned}$$

This leads to the conclusion that, according to this study, people are more likely to become romantically involved with those of the same race as themselves (there is a fair degree of homophily, but not so much as to where interracial couples don't exist - intraracial couples are simply more common).

2. (20 pts total) Consider an undirected "line graph" consisting of n vertices in a single component, with diameter $n-1$, and composed of $n-2$ vertices with degree 2 and 2 vertices with degree 1.

   (a) Show mathematically that if we divide this network into any two contiguous groups, such that one group has r connected vertices and the other has $n-r$, the modularity Q takes the value

$$Q = \frac{3 - 4n + 4rn - 4r^2}{2(n-1)^2} \quad .$$

Group 1 has r vertices:
$e_r = \frac{r-1}{n-1}$
$a_r = \frac{2(r-1)+1}{2(n-1)} = \frac{2r-1}{2(n-1)}$

Group 2 has n-r vertices:
$e_r = \frac{n-r-1}{n-1}$
$a_r = \frac{2(n-r-1)+1}{2(n-1)} = \frac{2n-2r-1}{2(n-1)}$

$$\begin{aligned}
Q &= \sum_r (e_r - a_r^2) = (\frac{r-1}{n-1} - (\frac{2r-1}{2(n-1)})^2) + (\frac{n-r-1}{n-1} - (\frac{2n-2r-1}{2(n-1)})^2) \\
&= \frac{r-1}{n-1} - \frac{4r^2-4r+1}{4(n-1)^2} + \frac{n-r-1}{n-1} - \frac{4n^2+4r^2-8rn+4r-4n+1}{4(n-1)^2} \\
&= \frac{n-2}{n-1} + \frac{-4r^2+4r-1-4n^2-4r^2+8rn-4r+4n-1}{4(n-1)^2} \\
&= \frac{4(n-1)(n-2)}{4(n-1)^2} + \frac{8rn-8r^2-4n^2+4n-2}{4(n-1)^2} \\
&= \frac{4n^2-12n+8}{4(n-1)^2} + \frac{8rn-8r^2-4n^2+4n-2}{4(n-1)^2} \\
&= \frac{8rn-8r^2-8n+6}{4(n-1)^2} = \frac{4rn-4r^2-4n+3}{2(n-1)^2}
\end{aligned}$$

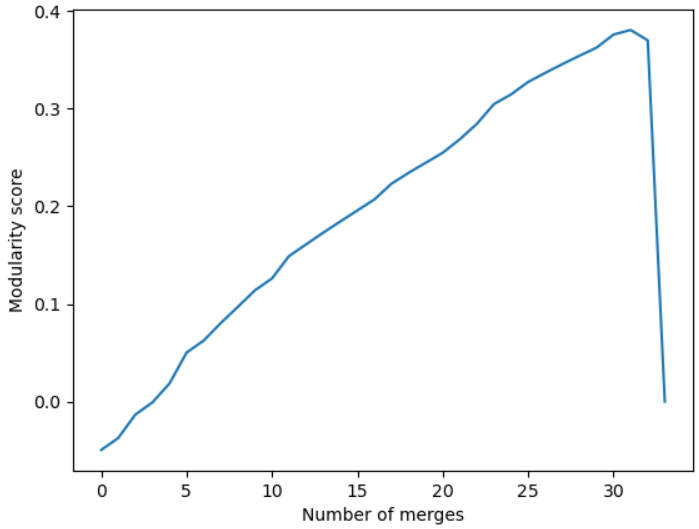
   (b) Considering the same graph, show that when n is even, the optimal division, in terms of modularity Q, is the division that splits the network exactly down the middle, into two parts of equal size.

$$\begin{aligned}
Q &= \frac{4rn-4r^2-4n+3}{2(n-1)^2} \\
\frac{dQ}{dt} &= 4n - 8r \\
0 &= 4n - 8r \rightarrow r = \frac{n}{2}
\end{aligned}$$

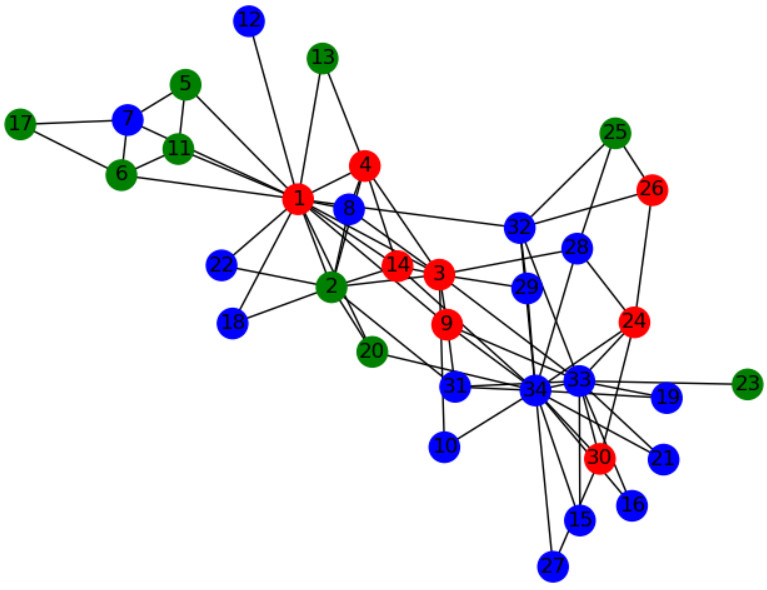
3. (30 pts) Implement the greedy agglomerative algorithm described in the lecture notes for maximizing modularity on an unlabeled simple network. (There is no need to make your algorithm particularly efficient, as we will not apply it to large networks; thus, it is okay to compute ΔQ using the adjacency matrix to derive the e matrix at each step.)

   Visit the Index of Complex Networks (ICON) at `icon.colorado.edu`, and obtain from the "Zachary Karate Club" entry a copy of the 77-edge network data file, along with the associated file that gives the "social partition" of the nodes. Apply your algorithm to this network.

   - Make a plot showing the modularity score Q as a function of the number of merges.



   - Make a visualization of the network itself with vertices labeled according to your maximum modularity partition.



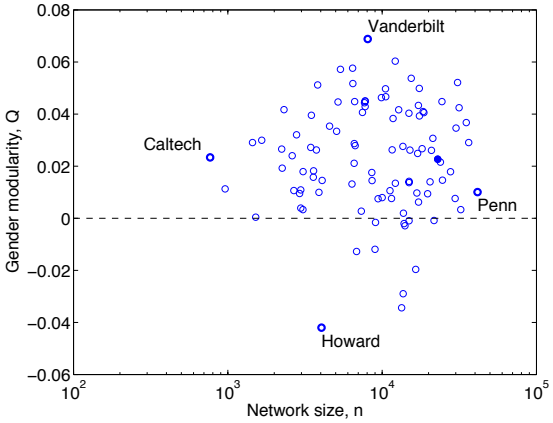
- Then calculate and report the normalized mutual information (NMI) between your partition and the social partition.[1]
  NMI = 0.5646068790944768
- Finally, briefly discuss the agreement or disagreement between the two partitions, and what that agreement/disagreement implies about the utility of modularity maximization inferring good partitions without knowing such labels.
  With an NMI value of roughly .565, the two partitions obviously aren't in total agreement (which is made even more obvious by knowing that the two partitions don't have the same number of groups). However, they also aren't in terrible disagreement. So, it seems that the utility of modularity maximization in inferring good partitions isn't incredibly high, nor is it very low. It's tough to quantify which is socially "optimal," but we can definitively say that the expected behavior according to the maximum modularity isn't necessarily nearly the same as what is observed in reality.

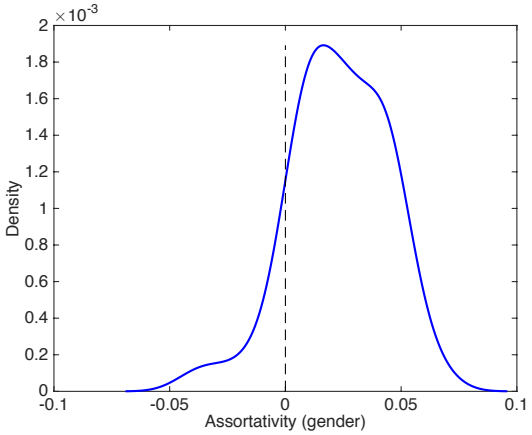
[1]For details of how to do this calculation, see Equation (11) in Karrer, Levina, and Newman, "Robustness of community structure in networks." *Phys. Rev. E* **77**, 046119 (2008), which is available here `http://arxiv.org/abs/0709.2108`.

4. (35 pts) Using the FB100 networks, investigate the assortativity patterns for three vertex attributes: (i) student/faculty status, (ii) major, and (iii) vertex degree. Treat these networks as simple graphs in your analysis.

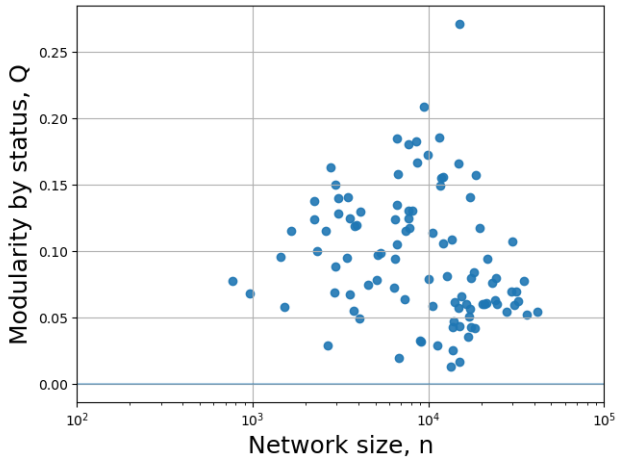
   For each vertex attribute, make a scatter plot showing the assortativity versus network size n, on log-linear axes, for all 100 networks, *and* a histogram or density plot showing the distribution of assortativity values. In both figures, include a line indicating no assortativity. Briefly discuss the degree to which vertices do or do not exhibit assortative mixing on each attribute, and speculate about what kind of processes or tendencies in the formation of Facebook friendships might produce this kind of pattern.

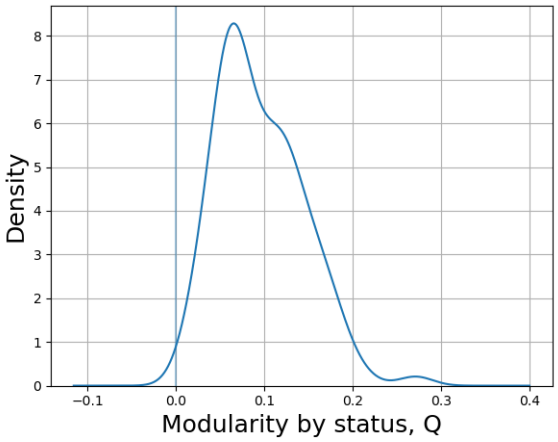
   For example, below are figures for assortativity by gender on these networks. The distribution of points spans the line of no assortativity, with some values nearly as far below 0 as others are above 0. However, the gender attributes do appear to be slightly assortative in these social networks: although all values are within 6% in either direction of 0, the mean assortativity is 0.02, which is slightly above 0. This suggests a slight amount of homophily by gender ("like links with like") in the way people friend each other on Facebook, although the tendency is very weak. In some schools, we see a slight tendency for heterophily ("like links with dislike"), as one might expect if the networks reflected heteronormative dating relationships.





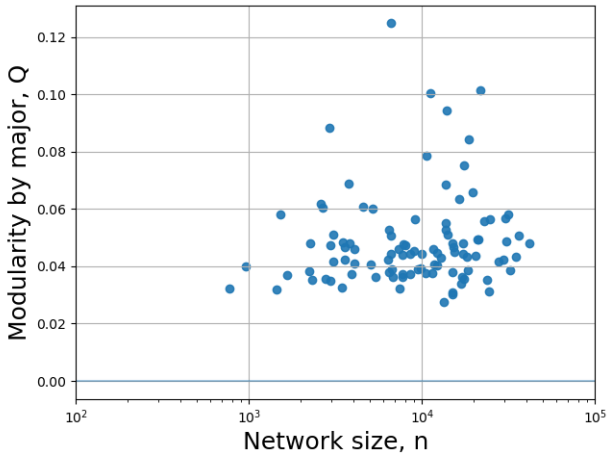
Modularity by status

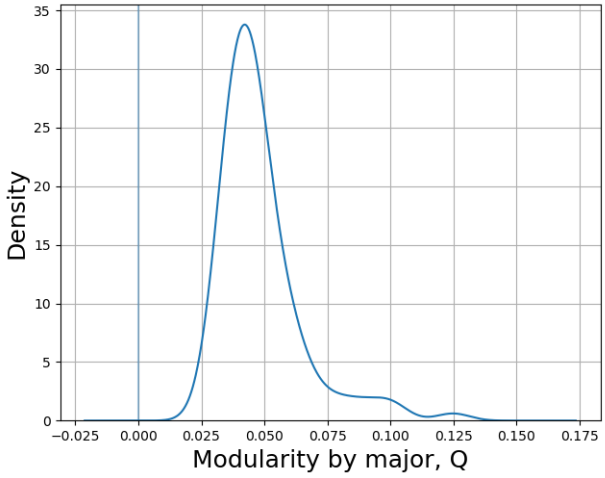




Above are figures for assortativity by student/faculty status on these networks. The distribution of points is entirely above the line of no assortativity, and the average assortativity is roughly .075. Therefore, the status attributes appear to be slightly assortative in these networks. This suggests a fair amount of homophily by status (most students link with other students more than with faculty) in the way people friend each other on Facebook, although the tendency isn't absolutely strong. This makes sense, as students are much more likely to friend their peers on Facebook, as opposed to friending their instructors/superiors, and instructors are more likely to friend fellow instructors (people are more likely to friend their educational equals). It's surprising that the assortativity isn't higher, but this may be able to be explained by the nature of certain faculty members that joined Facebook in its early stages (the tech-savvy/internet-savvy faculty members were likely more popular with students than their technologically inept counterparts, so many students likely friended the few faculty members that were on Facebook at the time).

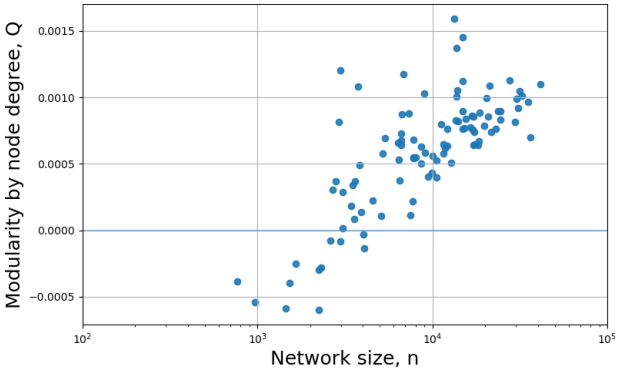
Modularity by major

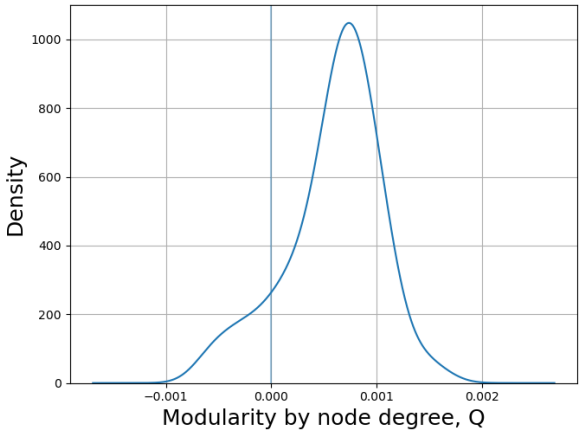




Above are figures for assortativity by major on these networks. The distribution of points is entirely above the line of no assortativity, and the average assortativity is roughly .04. Therefore, the major attributes appear to be slightly assortative in these networks. This suggests a fair amount of homophily by status (most students link with other same-major more than with other different-major students) in the way people friend each other on Facebook, although the tendency isn't very strong. This makes sense, as students should be more likely to meet and friend other students in their same major, as they'll have more interactions with them on average (likely taking some classes together). It's a bit surprising that the assortativity isn't higher, but this may be able to be explained by the existence of various student organizations and living accomodations (dorms, most clubs, Greek life, etc.) that aren't related to major, as they would serve the purpose of bringing together a diverse group of students, allowing people to meet (and subsequently friend) other people with different majors than their own.

Modularity by node degree





Above are figures for assortativity by node degree on these networks. The distribution of points spans the line of no assortativity, and the average assortativity is roughly .00075. Therefore, the node degree attributes appear to be neither assortative nor disassortative in these networks. There is, however, a clear trend in that larger schools tend to be slightly assortative, while smaller schools tend to be slightly disassortative. I'm not entirely sure on the exact reasons for this, but my best guess is that at smaller schools, there are not many "popular" people, and those that are may not be linked to other popular people; whereas at bigger schools, there are a fair abundance of popular people, and they're more likely to know each other (possibly due to the higher number of student organizations at larger schools). For example, those in Greek life are more likely to know other people than those not in Greek life. Bigger schools are also more likely to have higher amounts of Greek life organizations, giving more opportunities for more students to know people with similar levels of popularity.

5. (10 pts extra credit) As described in Section 13.2 of *Networks*, the configuration model can be thought of as the ensemble of all possible matchings of edge stubs, where vertex i has k_i stubs. Show that for a given degree sequence, the number Ω of matchings is

$$\Omega = \frac{(2m)!}{2^m m!} \ ,$$

which is independent of the degree sequence.

6. (15 pts extra credit) Using the configuration model, investigate the set of random graphs in which all vertices have degree 1 or 3.

   - Calculate via computer simulation the mean fractional size of the largest component for a network with $n = 10^4$ vertices, and with $p_1 = 0.6$, $p_3 = 1 - p_1$, and $p_k = 0$ for all other values of k.
   - Now make a figure showing the mean fractional size of the largest component for values of p_1 from 0 to 1 in steps of 0.01. Show that this allows you to estimate the value of p_1 for the phase transition at which the giant component disappears.

     Hint: The more smooth your line, the better the figure. The more independent instances you average over, the smoother your line.
   - Do your results depend on which graph space you choose for your configuration model?

Code for problem 3

```
import networkx as nx
import matplotlib
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import networkx.algorithms.community as nx_comm
from pprint import pprint

def e_r(G, r, p):
    ret = 0
    for i in G.nodes():
        for j in G.nodes():
            if G.has_edge(i,j):
                if groupDict[i] == r and groupDict[j] == p:
                    ret += 1
    return ret/(2*len(G.edges()))

def a_r(G, r):
    ret = 0
    for i in G.nodes():
        if groupDict[i] == r:
            ret += G.degree[i]
    return ret/(2*len(G.edges()))

def modularity(G):
    ret = 0
    for group in set(groupDict.values()):
        ret += e_r(G, group, group) - a_r(G, group)**2
    return ret

def prob_x_y(x, y, partition1, partition2):
    # x and and y are ints representing groups in p1 and p2,
    # partition1 and partition2 are dicts of node,group pairs in p1 and p2
    # probability of a given node being in group x of one partition
    # and group y of another
    matches = 0
    for node in partition1.keys():
        if partition1[node] == x and partition2[node] == y:
            matches += 1
    return matches/len(partition1)

def prob_x(x, partition):
    # probability of a given node being in group x of a partition
```

```
    size_x = 0
    for node,group in partition.items():
        if group == x:
            size_x += 1
    return size_x/len(partition)

def entropy(partition):
    ret = 0
    for g in set(partition.values()):
        size = 0
        for node,group in partition.items():
            if group == g:
                size += 1
        p_i = size/len(partition)
        ret += -p_i*np.log(p_i)
    return ret

def nmi(c, cprime):
    i_c_cprime = 0
    for x in set(c.values()):
        for y in set(cprime.values()):
            numerator = prob_x_y(x, y, c, cprime)
            denominator = prob_x(x, c)*prob_x(y, cprime)
            # print('########################')
            # print(numerator, denominator)
            if denominator != 0 and numerator != 0:
                i_c_cprime += prob_x_y(x, y, c, cprime)*np.log(numerator/denominator)
    return (2*i_c_cprime)/(entropy(c)+entropy(cprime))

if __name__ == '__main__':
    with open('karate_edges_77.txt') as f:
        G = nx.read_edgelist(f)

    # key is vertex, value is what group that vertex is part of
    groupDict = {
        '1' : '1',
        '2' : '2',
        '3' : '3',
        '4' : '4',
        '5' : '5',
        '6' : '6',
        '7' : '7',
        '8' : '8',
        '9' : '9',
        '10' : '10',
```

```
    '11' : '11',
    '12' : '12',
    '13' : '13',
    '14' : '14',
    '15' : '15',
    '16' : '16',
    '17' : '17',
    '18' : '18',
    '19' : '19',
    '20' : '20',
    '21' : '21',
    '22' : '22',
    '23' : '23',
    '24' : '24',
    '25' : '25',
    '26' : '26',
    '27' : '27',
    '28' : '28',
    '29' : '29',
    '30' : '30',
    '31' : '31',
    '32' : '32',
    '33' : '33',
    '34' : '34'
}

Q = modularity(G)

# key is number of merges, value is modularity
modDict = {
    0 : Q
}
merges = 0

Q_max = -np.inf
Q_max_groups = None

while len(set(groupDict.values())) > 1:
    merges += 1
    # analyze all possible merges, perform best one, add new modularity to modDict
    delta_Q = -np.inf
    for u in set(groupDict.values()):
        for v in set(groupDict.values()):
            if u != v:
                currentDelta = 2*(e_r(G, u, v) - a_r(G, u)*a_r(G, v))
```

```
                if currentDelta > delta_Q:
                    delta_Q = currentDelta
                    lowerMerged = min(int(u), int(v))
                    upperMerged = max(int(u), int(v))

    for key in groupDict.keys():
        if int(groupDict[key]) == upperMerged:
            groupDict[key] = str(lowerMerged)
        if int(groupDict[key]) > upperMerged:
            groupDict[key] = str(int(groupDict[key])-1)

    Q += delta_Q
    modDict[merges] = Q

    if Q > Q_max:
        Q_max = Q
        Q_max_groups = dict(groupDict)

# plot showing the modularity score as a function of the number of merges
lists = sorted(modDict.items())
x, y = zip(*lists)
plt.plot(x, y)
plt.xlabel("Number of merges")
plt.ylabel("Modularity score")
plt.show()

# visualization showing the grouping with max modularity
color_map = []
for i in sorted(Q_max_groups):
    if Q_max_groups[i] == '1':
        color_map.append('red')
    elif Q_max_groups[i] == '2':
        color_map.append('green')
    elif Q_max_groups[i] == '3':
        color_map.append('blue')
nx.draw(G, node_color=color_map, with_labels=True)
plt.show()

# nmi the two partitions
karate_groups_dict = {}
with open('karate_groups.txt') as f:
    for line in f:
        arr = line.strip().split('\t')
        karate_groups_dict[arr[0]] = arr[1]
print(nmi(Q_max_groups, karate_groups_dict))
```

Code for problem 4

```
import networkx as nx
import matplotlib
from matplotlib import pyplot as plt
import pandas as pd
import networkx.algorithms.community.quality as nx_qual
from pprint import pprint
import os
from os import listdir

# Using the FB100 networks, investigate the assortativity patterns for
# three vertexattributes:
# (i) student/faculty status, (ii) major, and (iii) vertex degree.
# Treat these networks as simple graphs in your analysis.
if __name__ == '__main__':
    path = 'facebook100txt'
    edge_list_filepaths = []
    for f in listdir(path):
        if ('_attr' not in f) and ('readme' not in f) and ('pdf' not in f) and
        ('.DS_Store' not in f):
            fp = os.path.join(path, f)
            if os.path.isfile(fp):
                edge_list_filepaths.append(fp)
    edge_list_filepaths = sorted(edge_list_filepaths)

    path = 'facebook100txt'
    attr_filepaths = []
    for f in listdir(path):
        if '_attr' in f:
            fp = os.path.join(path, f)
            if os.path.isfile(fp):
                attr_filepaths.append(fp)
    attr_filepaths = sorted(attr_filepaths)

    modList = []

    for i in range(len(edge_list_filepaths)):
        print(attr_filepaths[i])

        with open(edge_list_filepaths[i]) as f:
            G = nx.read_edgelist(f, nodetype=int)

        attrDF = pd.read_csv(attr_filepaths[i], sep='\t')
        attrDF.index += 1
```

```
    degreeList = sorted(list(G.degree()))
    degreeList = [i for _,i in degreeList]
    attrDF['degree'] = degreeList

    statuses = attrDF['status'].unique()
    status_set = []
    for j in statuses:
        status_set.append(attrDF.index[attrDF['status'] == j].tolist())
    Q_status = nx_qual.modularity(G, status_set)

    majors = attrDF['major'].unique()
    major_set = []
    for j in majors:
        major_set.append(attrDF.index[attrDF['major'] == j].tolist())
    Q_major = nx_qual.modularity(G, major_set)

    degrees = attrDF['degree'].unique()
    degree_set = []
    for j in degrees:
        degree_set.append(attrDF.index[attrDF['degree'] == j].tolist())
    Q_degree = nx_qual.modularity(G, degree_set)

    network_size = len(attrDF.index)

    modList.append([attr_filepaths[i].split('_')[0].split('\\')[1],
    Q_status, Q_major, Q_degree, network_size])

df = pd.DataFrame(modList, columns=['School', 'Modularity by status',
'Modularity by major', 'Modularity by degree', 'Network size'])
df.to_csv('df_out.csv')

dfNew = pd.read_csv('df_out.csv')

fig, ax = plt.subplots()
ax.scatter(dfNew['Network size'], dfNew['Modularity by status'], alpha = .9)
ax.hlines(0, 1e2, 1e5, alpha = .4)
ax.grid(True)
ax.set_xlim(1e2, 1e5)
ax.set_xlabel('Network size, n', fontsize=18)
ax.set_ylabel('Modularity by status, Q', fontsize=18)
ax.set_xscale('log')
plt.show()

fig, ax = plt.subplots()
```

```
dfNew['Modularity by status'].plot(kind='density')
ax.grid(True)
ax.set_xlabel('Modularity by status, Q', fontsize=18)
ax.set_ylabel('Density', fontsize=18)
ax.set_ylim(0, ax.get_ylim()[1])
ax.vlines(0, -1000, 10000, alpha = .4)
plt.show()

fig, ax = plt.subplots()
ax.scatter(dfNew['Network size'], dfNew['Modularity by major'], alpha = .9)
ax.hlines(0, 1e2, 1e5, alpha = .4)
ax.grid(True)
ax.set_xlim(1e2, 1e5)
ax.set_xlabel('Network size, n', fontsize=18)
ax.set_ylabel('Modularity by major, Q', fontsize=18)
ax.set_xscale('log')
plt.show()

fig, ax = plt.subplots()
dfNew['Modularity by major'].plot(kind='density')
ax.grid(True)
ax.set_xlabel('Modularity by major, Q', fontsize=18)
ax.set_ylabel('Density', fontsize=18)
ax.set_ylim(0, ax.get_ylim()[1])
ax.vlines(0, -1000, 10000, alpha = .4)
plt.show()

fig, ax = plt.subplots()
ax.scatter(dfNew['Network size'], dfNew['Modularity by degree'], alpha = .9)
ax.hlines(0, 1e2, 1e5, alpha = .4)
ax.grid(True)
ax.set_xlim(1e2, 1e5)
ax.set_xlabel('Network size, n', fontsize=18)
ax.set_ylabel('Modularity by node degree, Q', fontsize=18)
ax.set_xscale('log')
plt.show()

fig, ax = plt.subplots()
dfNew['Modularity by degree'].plot(kind='density')
ax.grid(True)
ax.set_xlabel('Modularity by node degree, Q', fontsize=18)
ax.set_ylabel('Density', fontsize=18)
ax.set_ylim(0, ax.get_ylim()[1])
ax.vlines(0, -1000, 10000, alpha = .4)
plt.show()
```