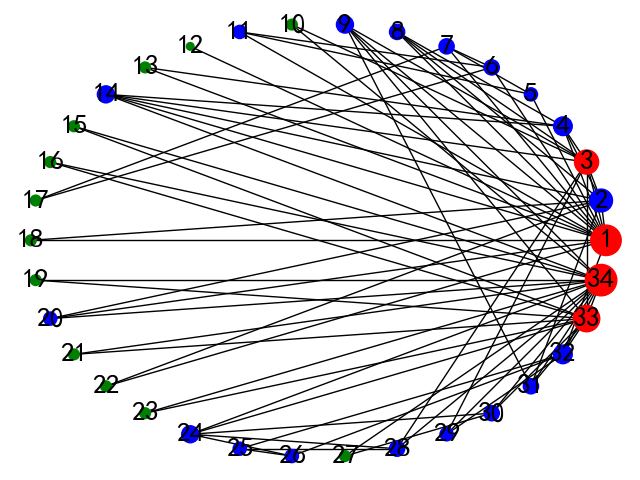
**2018级秋季第一学期复杂网络分析第七组第一次作业**

组员：柳传财，刘炫萍，刘露，曾梓龙，齐志伟

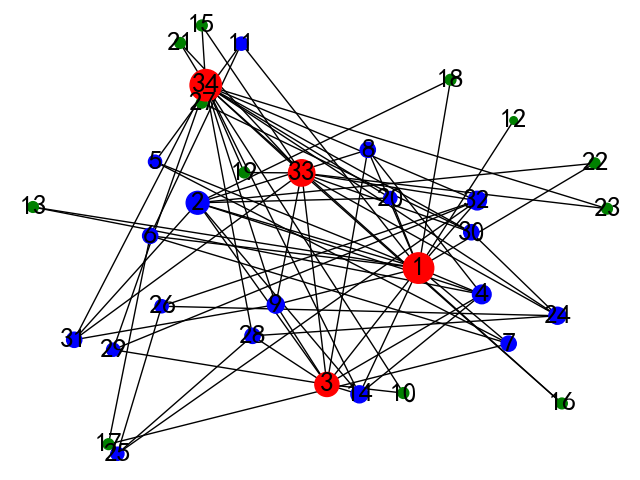
空手道数据结果：

平均集聚系数为0；平均最短路径长度为2；直径为5;

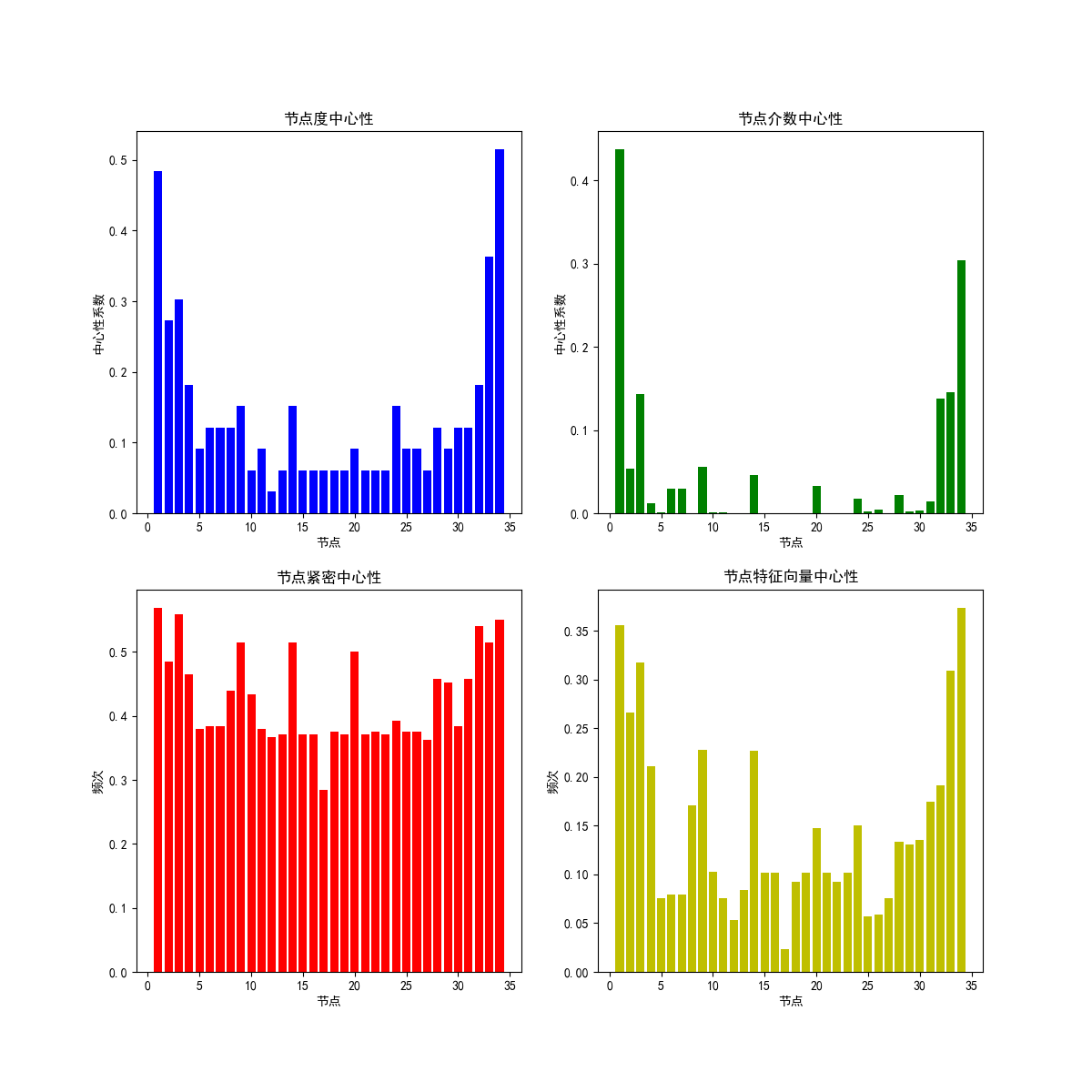
****

**图1 环形图属性**

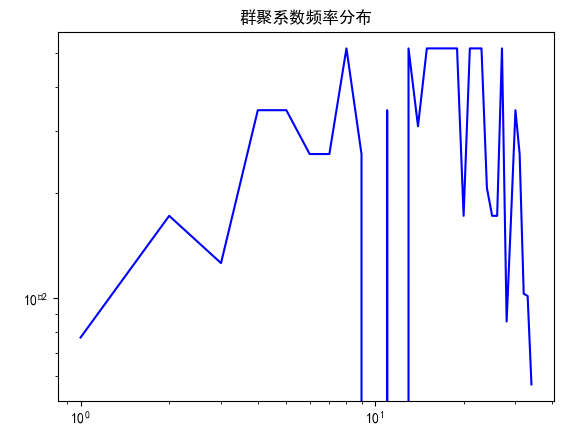
图一中，圆圈大小表示节点重要性（度分布）

****

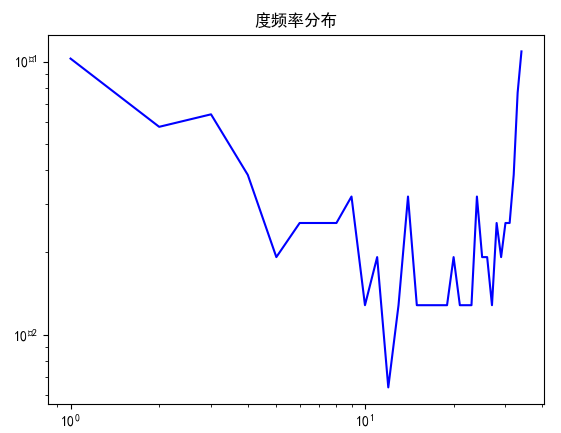
**图2 任意图属性**

****

**图3 节点中心性系数**

****

**图4 集聚系数分布**

****

**图5 度频率分布**

具体实现代码：

import numpy as np

import networkx as nx

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei']

plt.rcParams['font.family']='sans-serif'

def read\_karatetxt(path):

karatedatas=[]

with open(path,'r') as f:

for line in f.readlines():

bat = line.strip().split(' ')

return karatedatas

path=r'C:\Users\Dell\Desktop\karate.txt'

karates=read\_karatetxt(path)

def set\_martix(karates):

len0 = int(max(max(karates))) - 1

kar\_martix = np.zeros([len0, len0])

for i in range(len(karates)):

m = karates[i][0] - 1

n = karates[i][1] - 1

for j in range(len(kar\_martix)):

for k in range(len(kar\_martix)):

if j == m and k == n:

kar\_martix[j][k] = 1

return kar\_martix

print(set\_martix(karates))

G=nx.Graph()

point=range(1,35)

G.add\_nodes\_from(point)

G.add\_edges\_from(karates)

def Procrastination(G):

ave\_clt=nx.average\_clustering(G)#平均群聚系数

ave\_path=nx.average\_shortest\_path\_length(G)#平均最短路径长度

dia=nx.diameter(G)

print('平均集聚系数为%d；平均最短路径长度为%d；直径为%d'%(ave\_clt,ave\_path,dia))

def node\_col(G):

node\_clt=nx.clustering(G)#群聚系数

node\_degree=nx.degree(G)# 连接度列表表示，其余为字典表示

node\_decentrality=nx.degree\_centrality(G)#度中心性

node\_cen=nx.betweenness\_centrality(G)#节点介数中心性,指网络中经过某点并连接这两点的最短路径占这两点之间的最短路径线总数之比

node\_clo=nx.closeness\_centrality(G) #紧密中心性,用以衡量最短路径长度

node\_eig=nx.eigenvector\_centrality(G)#特征向量中心性，用以衡量该节点重要性

return node\_clt,node\_degree,node\_decentrality,node\_cen,node\_clo,node\_eig

Procrastination(G)

print(node\_col(G)[1])

def logpp(ll,title):

fig=plt.figure()

x=range(1,len(ll)+1)

ll0=[i[1] for i in ll]

y=[z/float(sum(ll0)) for z in ll0]

plt.loglog(x,y,color='b')

plt.title(title)

plt.show()

ll=[node\_col(G)[2].items(),node\_col(G)[3].items(),node\_col(G)[4].items(),node\_col(G)[5].items()]

def histpp(ll):

fig = plt.figure(figsize=(12,12))

x=np.arange(1,35)

ax1 = fig.add\_subplot(221)

y1= [i[1] for i in ll[0]]

ax1.bar(x,y1,color='b')

ax1.set\_xlabel('节点')

ax1.set\_ylabel('中心性系数')

ax1.set\_title('节点度中心性')

ax2 = fig.add\_subplot(222)

y2= [i[1] for i in ll[1]]

ax2.bar(x,y2,color='g')

ax2.set\_xlabel('节点')

ax2.set\_ylabel('中心性系数')

ax2.set\_title('节点介数中心性')

ax3 = fig.add\_subplot(223)

y3= [i[1] for i in ll[2]]

ax3.bar(x,y3,color='r')

ax3.set\_xlabel('节点')

ax3.set\_ylabel('频次')

ax3.set\_title('节点紧密中心性')

ax4 = fig.add\_subplot(224)

y4= [i[1] for i in ll[3]]

ax4.bar(x,y4,color='y')

ax4.set\_xlabel('节点')

ax4.set\_ylabel('频次')

ax4.set\_title('节点特征向量中心性')

plt.savefig("graph3.png")

plt.show()

logpp(node\_col(G)[0].items(),'群聚系数频率分布')

logpp(node\_col(G)[1],'度频率分布')

histpp(ll)

de=nx.degree(G)

de2=[i[1]\*30 for i in de]

def decolor(de2):

cols=[]

ave=sum(de2)/len(de2)

for i in range(len(de2)):

if de2[i]>2\*ave:

col='r'

cols.append(col)

elif 0.5\*ave<=de2[i]<=2\*ave:

col='b'

cols.append(col)

col='g'

cols.append(col)

return cols

de1=decolor(de2)

fig=plt.figure()

nx.draw(G,pos = nx.circular\_layout(G),node\_color = de1,edge\_color = 'k',with\_labels = True,font\_size =20,node\_size =de2)

plt.savefig("graph1.png")

plt.show()

nx.draw(G,pos = nx.random\_layout(G),node\_color = de1,edge\_color = 'k',with\_labels = True,font\_size =20,node\_size =de2)

plt.savefig("graph2.png")

plt.show()