

**杭州电子科技大学**

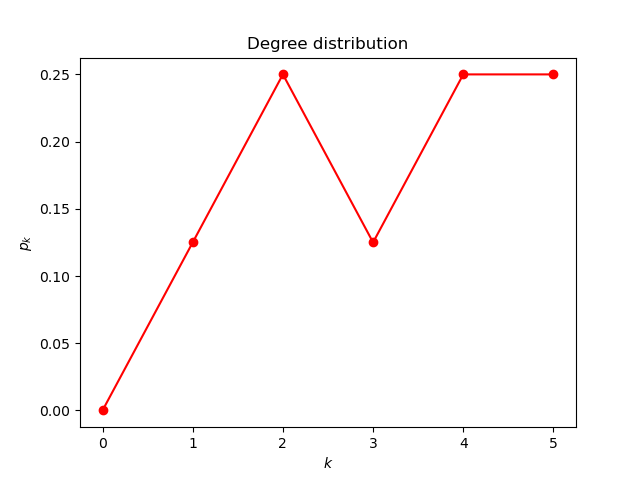
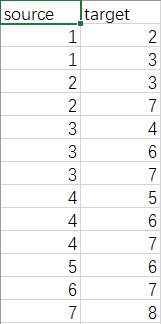
**《创新实践1》期末课程报告**

|  |  |
| --- | --- |
| **学 院** | **理学院** |
| **专 业** |  |
| **姓 名** |  |
| **班 级** |  |
| **学 号** |  |
| **授课教师** | **覃森** |
| **成 绩** |  |

2022年春季《创新实践1》期末检测题

姓名： 学号： 分数：

1. 求解如下网络的度分布、聚类系数与平均最短路径长度，写出程序与相关结果。（20分）



平均最短路径： 1.6785714285714286

平均集聚系数: 0.6124999999999999

以下为程序演示：

import networkx as nx

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv("thefirstquestion.csv")

G = nx.from\_pandas\_edgelist(df,"source","target",create\_using= nx.Graph)

nx.draw(G, with\_labels = True, node\_size = 5,node\_color="y")

plt.show()

#度

d=nx.degree(G)

d=dict(nx.degree(G))

print(d)

n = len(G.nodes)

#获取平均度

d=dict(nx.degree(G))

print("平均度为：",sum(d.values())/len(G.nodes))

#获取所有可能的度值对应的概率

x=list(range(max(d.values())+1))

y=[i/n for i in nx.degree\_histogram(G)]

#绘制度分布

plt.plot(x,y,'ro-')

plt.xlabel("$k$")

plt.ylabel("$p\_k$")

plt.title("Degree distribution")

plt.show()

#求平均最短路径

print("平均最短路径：",nx.average\_shortest\_path\_length(G))

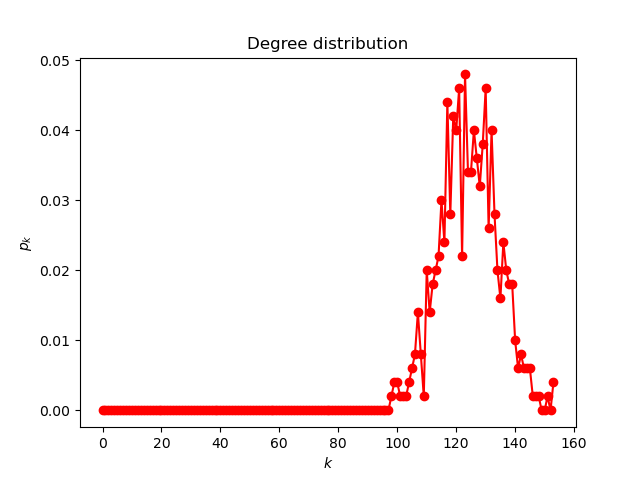
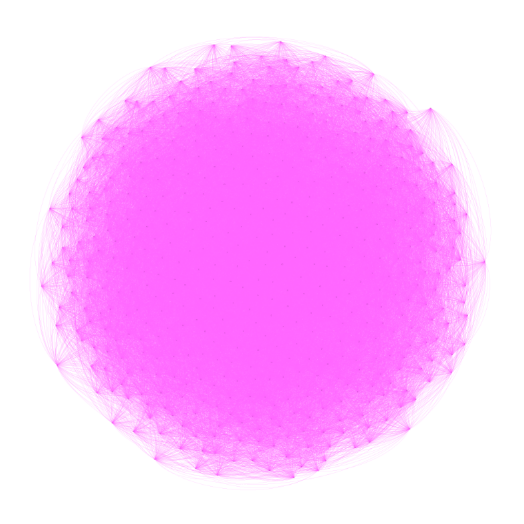
#平均集聚系数

print("平均集聚系数:",nx.average\_clustering(G))

2、编程实现节点数为500个的五种网络：随机图网络（随机连接边的概率为0.25）、WS小世界网络（重连边概率0.25）、NW小世界网络（重连边概率0.25）、BA无标度网络和EBA无标度网络（重连概率为0.25）。画出实现这些网络的一个图形，并计算出这些网络的度分布、平均最短路径长度和聚类系数。（20分）

（注：这里用python中的函数来生成网络，由于python画图不太美观，这里用nx.write\_gexf(G,'导出文件名')生成gephi的文件格式，并用gephi画图）

ER随机网络：



平均集聚系数: 0.249042480652382

平均最短路径： 1.7508617234468937

import networkx as nx

import matplotlib.pyplot as plt

#总计500点，重连边概率0.25

n,p=500,0.25

G=nx.gnp\_random\_graph(n,p)

plt.figure(figsize=(32,32))

nx.draw(G,pos=nx.circular\_layout(G),node\_size=100,node\_color="red",with\_labels=False)

nx.write\_gexf(G,'ER.gexf')

plt.title("G(N,P)")

plt.show()

#度

d=nx.degree(G)

d=dict(nx.degree(G))

print(d)

#获取平均度

d=dict(nx.degree(G))

print("平均度为：",sum(d.values())/len(G.nodes))

#获取所有可能的度值对应的概率

x=list(range(max(d.values())+1))

y=[i/n for i in nx.degree\_histogram(G)]

#绘制度分布

plt.plot(x,y,'ro-')

plt.xlabel("$k$")

plt.ylabel("$p\_k$")

plt.title("Degree distribution")

plt.show()

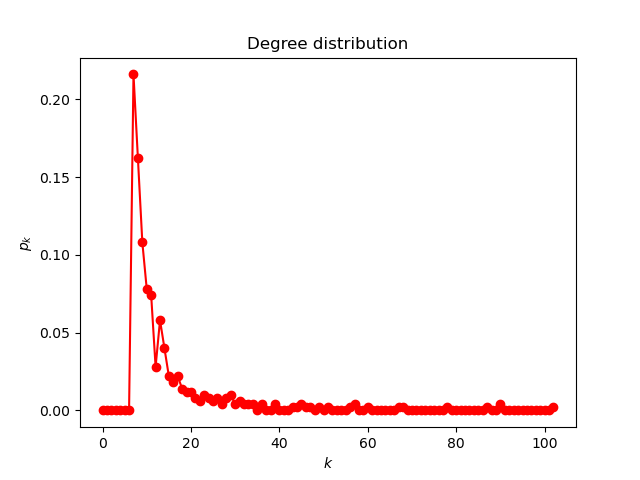
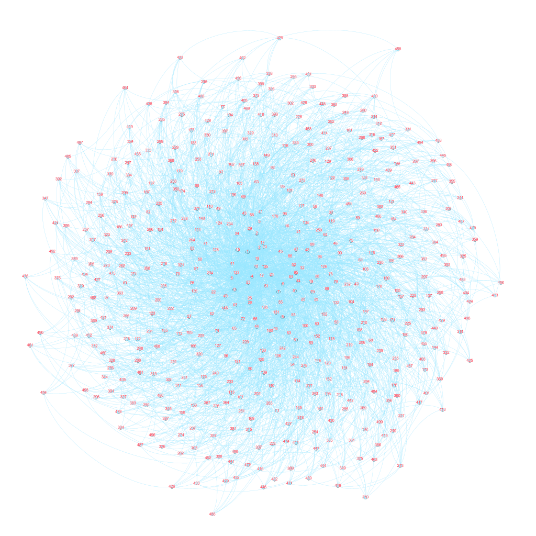
#求平均最短路径

print("平均最短路径：",nx.average\_shortest\_path\_length(G))

#平均集聚系数

print("平均集聚系数:",nx.average\_clustering(G))

BA无标度网络：



平均最短路径：2.548304609218437

平均集聚系数: 0.07946759893357544

import networkx as nx

import matplotlib.pyplot as plt

n=500

G = nx.barabasi\_albert\_graph(n, 7)

ps = nx.spring\_layout(G)

nx.draw(G, ps, with\_labels = False, node\_size = 5,node\_color="b")

nx.write\_gexf(G,'BA.gexf')

plt.show()

#度

d=nx.degree(G)

d=dict(nx.degree(G))

print(d)

#获取平均度

d=dict(nx.degree(G))

print("平均度为：",sum(d.values())/len(G.nodes))

#获取所有可能的度值对应的概率

x=list(range(max(d.values())+1))

y=[i/n for i in nx.degree\_histogram(G)]

#绘制度分布

plt.plot(x,y,'ro-')

plt.xlabel("$k$")

plt.ylabel("$p\_k$")

plt.title("Degree distribution")

plt.show()

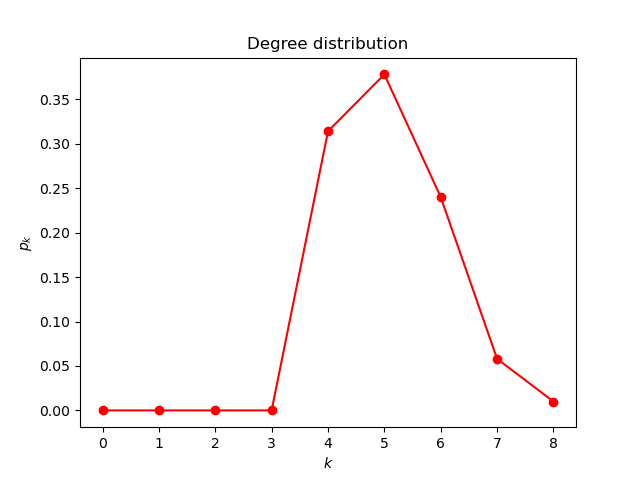
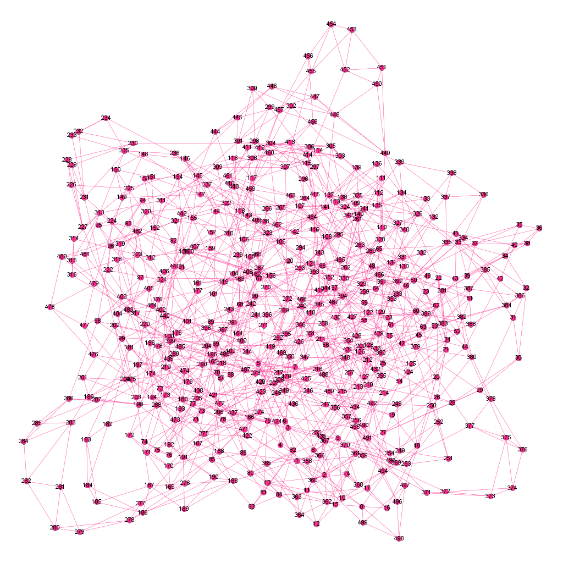
#求平均最短路径

print("平均最短路径：",nx.average\_shortest\_path\_length(G))

#平均集聚系数

print("平均集聚系数:",nx.average\_clustering(G))

NW小世界网络：



平均最短路径： 4.9710060120240485

平均集聚系数: 0.3287476190476189

import networkx as nx

import matplotlib.pyplot as plt

#随机加边

n=500

p=0.25

k=5

G=nx.newman\_watts\_strogatz\_graph(n,k,p)

plt.figure(figsize=(4,4))

nx.draw(G,pos=nx.circular\_layout(G),node\_size=500,node\_color="red",with\_labels=True)

nx.write\_gexf(G,'NW.gexf')

plt.suptitle('NW small\_world')

plt.show()

#度

d=nx.degree(G)

d=dict(nx.degree(G))

print(d)

#获取平均度

d=dict(nx.degree(G))

print("平均度为：",sum(d.values())/len(G.nodes))

#获取所有可能的度值对应的概率

x=list(range(max(d.values())+1))

y=[i/n for i in nx.degree\_histogram(G)]

#绘制度分布

plt.plot(x,y,'ro-')

plt.xlabel("$k$")

plt.ylabel("$p\_k$")

plt.title("Degree distribution")

plt.show()

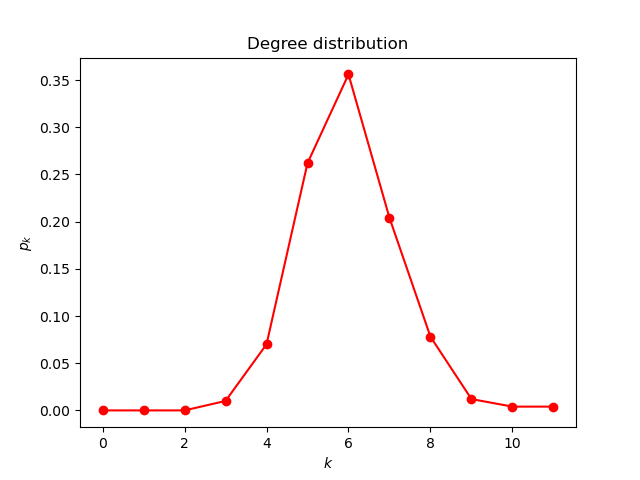
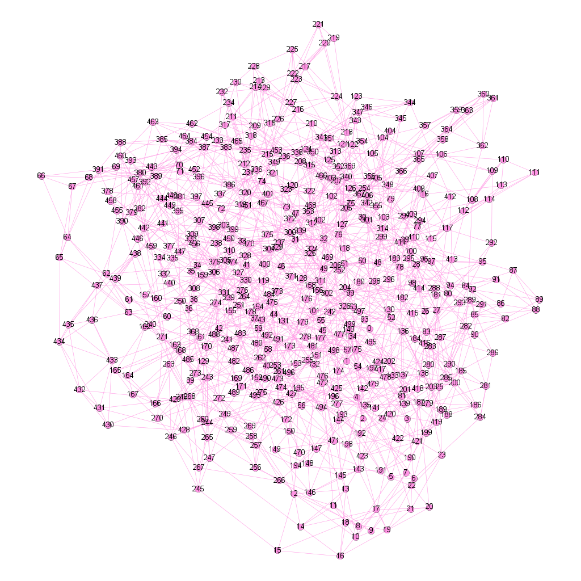
#求平均最短路径

print("平均最短路径：",nx.average\_shortest\_path\_length(G))

#平均集聚系数

print("平均集聚系数:",nx.average\_clustering(G))

WS小世界网络：



平均最短路径： 4.306717434869739

平均集聚系数: 0.2717922077922081

import networkx as nx

import matplotlib.pyplot as plt

#随机重连

n=500

p=0.25

G=nx.watts\_strogatz\_graph(n,6,p)

plt.figure(figsize=(10,10))

nx.draw(G,pos=nx.circular\_layout(G),node\_size=300,node\_color="red",with\_labels=True)

nx.write\_gexf(G,'WS.gexf')

plt.suptitle("WS small\_world")

plt.show()

#度

d=nx.degree(G)

d=dict(nx.degree(G))

print(d)

#获取平均度

d=dict(nx.degree(G))

print("平均度为：",sum(d.values())/len(G.nodes))

#获取所有可能的度值对应的概率

x=list(range(max(d.values())+1))

y=[i/n for i in nx.degree\_histogram(G)]

#绘制度分布

plt.plot(x,y,'ro-')

plt.xlabel("$k$")

plt.ylabel("$p\_k$")

plt.title("Degree distribution")

plt.show()

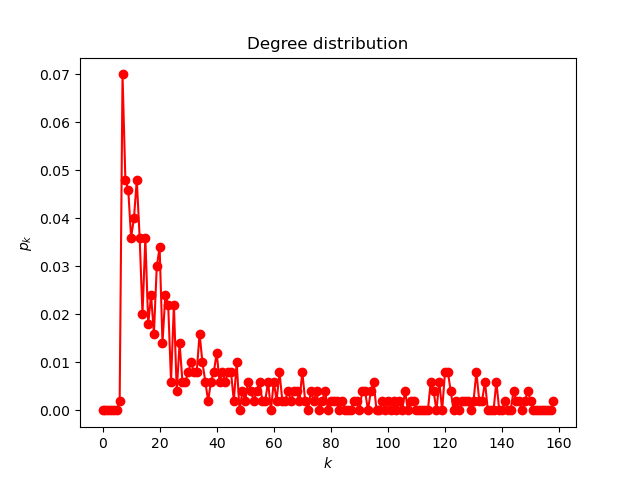
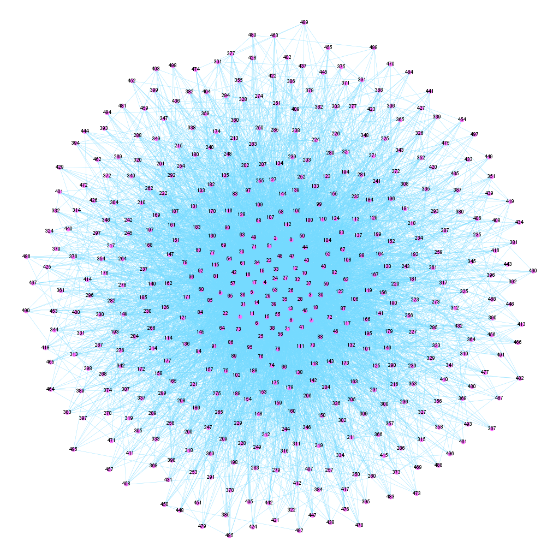
#求平均最短路径

print("平均最短路径：",nx.average\_shortest\_path\_length(G))

#平均集聚系数

print("平均集聚系数:",nx.average\_clustering(G))

EBA无标度网络：



平均最短路径： 2.1296112224448898

平均集聚系数: 0.2931403016565481

import networkx as nx

import matplotlib.pyplot as plt

n=500

m = 7

p = 0.5

q = 0.2

G = nx.extended\_barabasi\_albert\_graph(n,m,p,q)

ps = nx.spring\_layout(G)

nx.draw(G, ps, with\_labels = False, node\_size = 5,node\_color="b")

nx.write\_gexf(G,'EBA.gexf')

plt.show()

#度

d=nx.degree(G)

d=dict(nx.degree(G))

print(d)

#获取平均度

d=dict(nx.degree(G))

print("平均度为：",sum(d.values())/len(G.nodes))

#获取所有可能的度值对应的概率

x=list(range(max(d.values())+1))

y=[i/n for i in nx.degree\_histogram(G)]

#绘制度分布

plt.plot(x,y,'ro-')

plt.xlabel("$k$")

plt.ylabel("$p\_k$")

plt.title("Degree distribution")

plt.show()

#求平均最短路径

print("平均最短路径：",nx.average\_shortest\_path\_length(G))

#平均集聚系数

print("平均集聚系数:",nx.average\_clustering(G))

3、试采用两种社区发现算法（如基于拉普拉斯普分解、KL算法、GN算法、Newman快速算法等），编程计算出US Airline网络（332个节点和2461条边）的社区结构，并指出最优的社区个数。数据可在<http://vladowiki.fmf.uni-lj.si/doku.php?id=pajek:nets:old>中下载。（20分）

第一种：基于louvain的社区发现

社团数目： 6

社团划分： [[118, 126, 127, 148, 76, 78, 163, 70, 114, 77, 67, 161, 137, 90, 130, 140, 82, 51, 106, 136, 71, 100, 84, 160, 43, 191, 93, 73, 120, 134, 139, 135, 57, 111, 158, 171, 69, 79, 52, 44, 98, 154, 164, 102, 107, 88, 155, 122, 145], [201, 258, 47, 144, 263, 203, 65, 253, 183, 169, 274, 250, 248, 219, 197, 246, 251, 213, 242, 81, 153, 168, 166, 245, 214, 244, 178, 185, 236, 45, 75, 175, 50, 41, 264, 56, 74, 86, 104, 59, 54, 66, 46, 55, 58, 83, 85, 97, 181, 188, 199, 205, 200, 224, 108, 151, 116, 87, 62, 63, 40, 61, 173, 53, 39, 208, 228, 42, 138, 124, 193, 231, 48, 49, 60, 64, 165], [182, 297, 216, 261, 293, 232, 296, 172, 276, 225, 133, 239, 128, 254, 275, 198, 308, 240, 286, 271, 281, 267, 309, 206, 220, 290, 266, 226, 270, 142, 241, 278, 279, 291, 247, 304, 184, 280, 257, 269, 288, 256, 283, 272, 196, 285, 262, 233, 229, 303, 105, 294, 282, 121, 268, 277, 180, 211, 209, 195, 207], [152, 162, 301, 221, 321, 306, 307, 212, 92, 68, 91, 125, 189, 305, 202, 222, 80, 273, 204, 186, 129, 150, 311, 146, 147, 112, 299, 177, 310, 123, 292, 167, 94, 284, 218, 255, 109, 230, 179, 174, 176, 131, 217, 119, 159, 95, 260, 157, 237, 322, 249, 298, 287, 192, 300, 234, 223, 132, 295, 215, 265, 289, 324, 99, 143, 96, 156, 252, 235, 243, 238, 227, 89, 141, 302, 325, 101, 170, 149, 72, 259, 113, 117, 110, 210, 190, 115, 103, 187, 323, 194, 320], [313, 316, 331, 312, 329, 318, 319, 314, 317, 326, 315, 327, 330, 328, 332], [8, 26, 4, 13, 23, 3, 2, 5, 36, 16, 6, 35, 28, 30, 33, 29, 31, 22, 1, 32, 7, 21, 18, 17, 12, 20, 19, 10, 9, 14, 15, 27, 24, 38, 37, 25, 34, 11]]

代码：

import networkx as nx

import pandas as pd

import community.community\_louvain

def FindsameValue(dict):

values\_list = list(set(dict.values()))

club = []

for i in range(len(values\_list)):

club.append([])

for k,v in dict.items():

club[values\_list.index(v)].append(k)

print('社团数目：', len(club))

print('社团划分：', club)

df = pd.read\_csv('demo.csv')

G = nx.from\_pandas\_edgelist(df,source = "source",target = 'target')

partition = community.community\_louvain.best\_partition(G)

FindsameValue(partition)、

第二种：基于GN算法的可控制数量的社团划分

结果：

社团数量： 6

[[39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 331], [327, 328, 329, 330, 332], [1, 2, 3, 4, 5, 8, 16, 22, 23, 24, 27, 28, 30, 34, 35, 36, 37, 38], [25, 26, 29, 31, 32, 33], [6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21], [155, 171]]

代码：

from networkx.algorithms.community import girvan\_newman

import pandas as pd

import networkx as nx

import itertools

df = pd.read\_csv('demo.csv')

G = nx.from\_pandas\_edgelist(df, source="source", target='target')

k = 6

comp = girvan\_newman(G)

limited = itertools.takewhile(lambda c: len(c) <= k, comp)

for communities in limited:

a = list(tuple(sorted(c) for c in communities))

print("社团数量：", len(a))

print(a)

4、利用理论推导和编程实现两种方法，求解如下网络的生成树的个数（假定每条边的权重均为1）。（20分）



生成树个数：103.99999999999996



1/9\*4\*27/4\*1\*1/12\*1/9\*4\*1/16\*9\*10\*4\*1\*2\*13/5\*8 = 104.0

以下为程序展示：

import numpy as np

# 先输入矩阵A

A = np.array([[0,1,0,1,1,0],

[1,0,1,0,1,1],

[0,1,0,0,1,1],

[1,0,0,0,1,0],

[1,1,1,1,0,1],

[0,1,1,0,1,0]], dtype=int)

AD = np.sum(A, axis=0)

# print(AD)

L = np.diag(AD) - A

# print(L)

E, V = np.linalg.eig(L)

L1 = L[:len(L) - 1, :len(L) - 1]

E1, V1 = np.linalg.eig(L1)

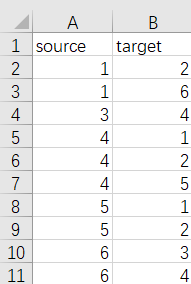
s = 1

for i in range(len(E1)):

s = s \* E1[i]

print(s)

5、利用PageRank算法计算如下有向图表示的网页页面的等级（可用其稳态概率来表示。）（20分）



当X和R的欧式距离小于0.01时，取到以下结果：（注:这里将A,B,C等映射为1,2,3等）

A:0.85833333

B:1.28333333

C:0.575

D:1.425

E:0.43333333

F:0.575

以下为代码展示：

import numpy as np

import pandas as pd

#该函数用来求网站数目，参数为通过read\_csv读入的数据，即df

def NumofWeb(date):

#获取source和target两列的点的集合

s1 = set(list(df.iloc[:,0]))

s2 = set(list(df.iloc[:,1]))

#合并两个集合

s = s1|s2

return len(s)

#该函数用于获取数据中边的数量,参数同NumofWeb函数

def NumofLine(date):

return len(list(df.iloc[:,0]))

#该函数用于求概率转移矩阵

def ProbabilityTransitionMatrix(date):

#通过NumofWeb函数获得数据中网站的个数并创建一个零矩阵

n = NumofWeb(date)

p = np.zeros((NumofWeb(date),NumofWeb(date)))

for i in range(NumofLine(date)):

p[df.iloc[i,0]-1,df.iloc[i,1]-1] = 1

p = np.mat(p)

#定义一个一维矩阵，获得p每行值的和

Nump = np.sum(p,axis = 1)

for i in range(n):

for j in range(n):

# 这里判断p中每一行的和是否为0

if Nump[i,0]:

p[i, j] = p[i, j] / Nump[i,0]

return p.transpose()

def PR(A,X,R):

c = float(input("请输入阈值："))

while float(np.linalg.norm(np.array(X-R)))>c:

(X,R) = (R,A\*X)

else:

print(R)

df = pd.read\_csv("demolist.csv")

#阻尼系数

q = 0.85

X = np.mat(np.ones((NumofWeb(df),1)))

eet = np.mat(np.ones((NumofWeb(df),NumofWeb(df))))/NumofWeb(df)

p=ProbabilityTransitionMatrix(df)

A = q\*p + (1-q)\*eet

R = A\*X

PR(A,X,R)