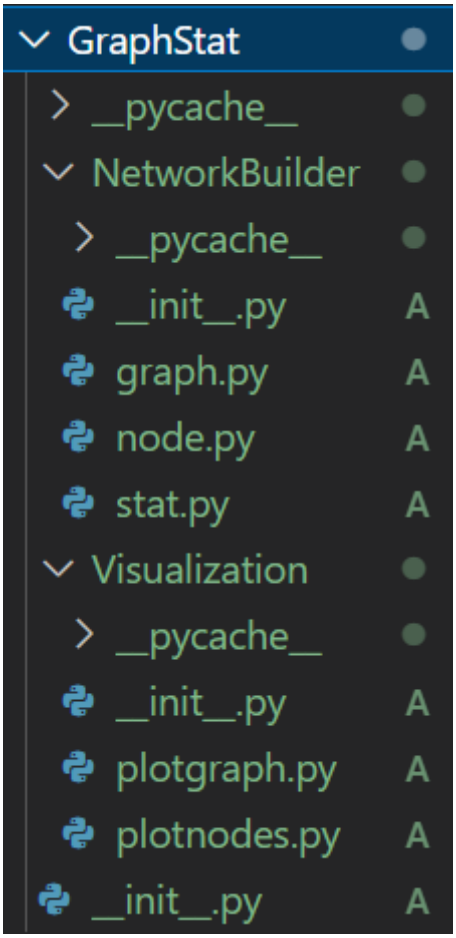
20377199 赵芮箐 第4周作业

图是非常重要的一种数据结构，常用于描述社交网络。本次作业提供了 twitch_gamers 数据集（见在线平台课程资料 twitch_gamers.zip），希望基于该数据，构建一个 python 程序包，并在对应模块中实现：读取并存储节点信息，建立无向的社交网络，以及实现相关统计和可视化功能。

任务一：建立包GraphStat

- 包GraphStat结构



- 包Graph：网络的构建与分析
  - graph模块：实现网络的构建，以及序列化的储存和加载

```
#graph.py

import pickle
import pandas as pd
import networkx as nx

def init_graph(filename):
    """
    从数据文件中加载所有边，构建图结构

    :param filename:数据文件路径
    :return graph:加载后的图结构
```

```
    """
    with open(filename,'r',encoding='utf-8') as f:
        data = pd.read_csv(f)
    edges = data.apply(lambda x: tuple(x), axis=1).values.tolist()

    #构建图结构
    G = nx.Graph()
    G.add_edges_from(edges)

    return G

def save_graph(graph):
    """
    序列化图信息，实现存储
    """
    with open('graph.pkl','wb') as f:
        pickle.dump(graph, f)

def load_graph(path):
    """
    反序列化图信息，实现加载
    """
    with open(path,'rb') as f:
        graph = pickle.loads(f)
    return graph
```

- node模块：加载所有节点及其属性，实现节点度的统计以及打印节点全部属性

```
#node.py

import pandas as pd
import networkx as nx

def init_node(filename):
    """
    从数据文件中加载所有节点及其属性，返回dataframe结构

    :param filename:数据文件路径
    :return node:加载后的所有节点，dataframe结构
    """
    with open(filename,'r',encoding='utf-8') as f:
        data = pd.read_csv(f)
    return data

def get_degree(data,graph):
    """
    获取节点的度，存入data数据表

    :param data:储存节点的数据表
    :param graph:加载的图结构
    :return degree:返回某节点的度
    """
    degree = []
    for i in range(len(data)):
```

```
        de = len(graph[data.iloc[i,5]])# graph第一层结构为以节点编号为key,其相邻节点为value
        degree.append(de)
    data['degree'] = degree
    return data

def print_node(data,node_id):
    """
    输出某节点的全部信息

    :param data:加载后的所有节点
    :param node_id:需要输出的某节点ID
    """
    print("-----ID为{}的节点信息-----".format(node_id))
    print(data.loc[node_id])
```

结果展示：

```
-----ID为0的节点信息-----
views                7879
mature                  1
life_time             969
created_at      2016-02-16
updated_at      2018-10-12
numeric_id               0
dead_account             0
language                EN
affiliate                1
degree                  43
```

- stat模块：实现图的基础信息统计，包括：节点数、边数、平均度、度分布以及views属性分布

```
#stat.py

import numpy as np
import pandas as pd

def get_node_number(graph):
    """
    计算节点数
    """
    nodes_num = len(graph)
    return nodes_num

def get_edge_number(graph):
    """
    计算边数
    """
    edges = 0
    for node in graph:
        edges += len(graph[node])
    edges_num = edges / 2
    return edges_num

def cal_average_dgree(graph):
    """
    计算网络中的平均度
    """
    nodes_num = len(graph)
```

```
    edges_num = 0
    for node in graph:
        edges_num += len(graph[node])
    return edges_num/nodes_num

def cal_dgree_distribution(graph):
    """
    计算网络的度分布

    :return degree_dis:返回度分布，series类型
    """
    degree = []
    for key in graph:                           # 统计每个节点的度，存成列表
        de = len(graph[key])
        degree.append(de)
    degree_df = pd.DataFrame(degree)            # 将列表转化成dataframe储存
    degree_dis = degree_df.value_counts()       # 用dataframe的函数完成统计
    return degree_dis

def cal_views_distribution(data):
    """
    计算views属性的分布

    :return views_dis:返回views分布，series类型
    """
    views_dis = data['views'].value_counts()
    return views_dis
```

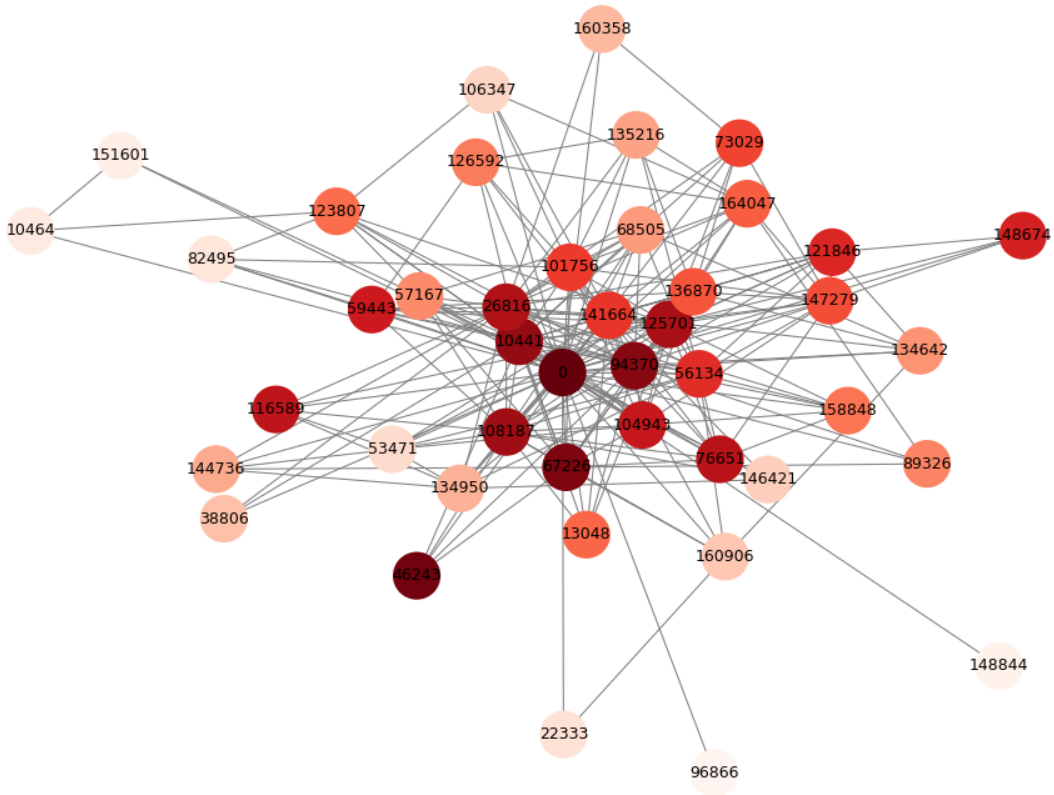
```
-----图的基本统计信息-----
nodes number:168114
edges_num:6797557
average degree:80.87
```

- 包Visualization，实现图及统计结果的可视化
  - plotgraph模块：

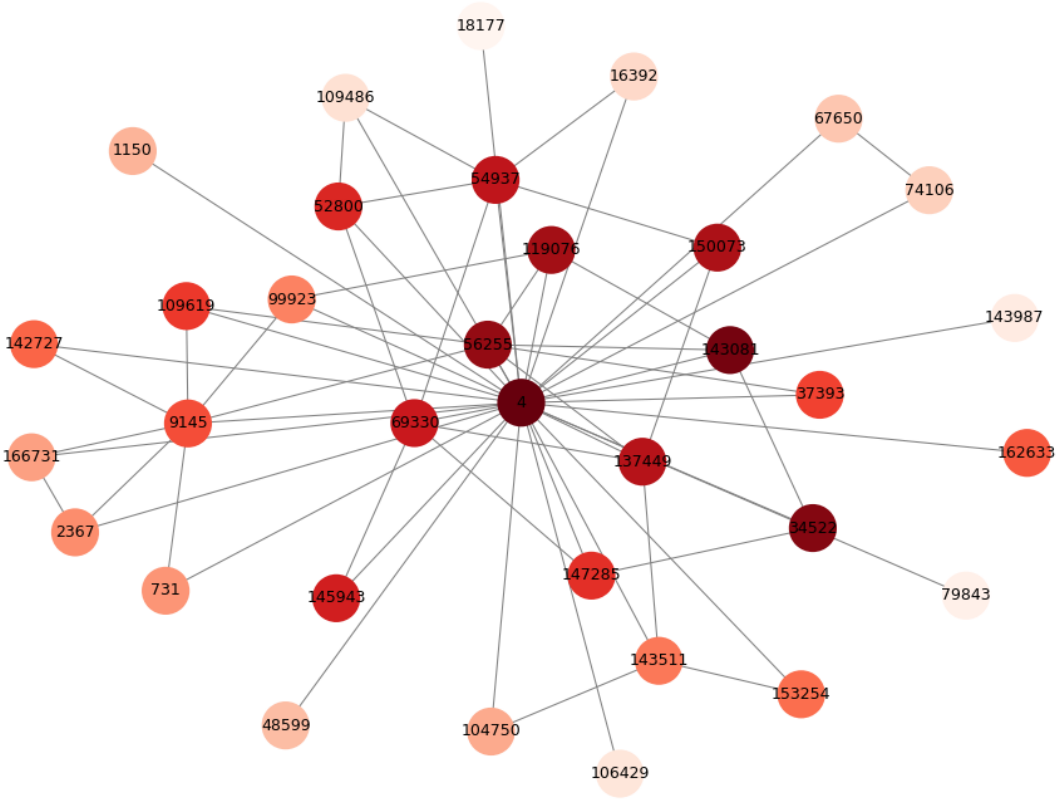
```
#plotgraph.py

import networkx as nx
import matplotlib.pyplot as plt

def plot_ego(graph,node_id):
    """
    绘制某节点的局部网络
    """
    ego = nx.Graph()
    #ego = nx.ego_graph(graph, node_id)

    #构建ego network
    for nei in graph[node_id]:
        ego.add_edge(node_id, nei)              # 把该节点的所有边加进去
        for nei_nei in graph[nei]:
            if nei_nei in graph[node_id]:       # 把该节点邻居之间的边加进去
                ego.add_edge(nei, nei_nei)
    #绘图
```

```
    pos = nx.spring_layout(ego)              # 节点中心布局
    colors = range(len(ego[node_id])+1)
    nx.draw(ego, pos, with_labels=True, font_size = 6, node_color=colors,
cmap=plt.cm.Reds_r, edge_color='gray', width=0.5)
    plt.show()

def plotdegree_distribution(graph):
    """
    绘制度的分布直方图
    """
    degree = []
    for key in graph:
        de = len(graph[key])
        degree.append(de)
    y, bins, patches = plt.hist(degree, bins=7, log=True, align='left')
    for i in range(len(y)):
        plt.text(bins[i], y[i]*1.02, int(y[i]), fontsize=12,
horizontalalignment="center") # 打标签
    plt.title('degree distribution')     # 标题
    plt.xlabel('degree')                 # x轴名
    plt.ylabel('frequency')              # y轴名
    #plt.savefig('度分布直方图' + '.png')
    plt.show()
```

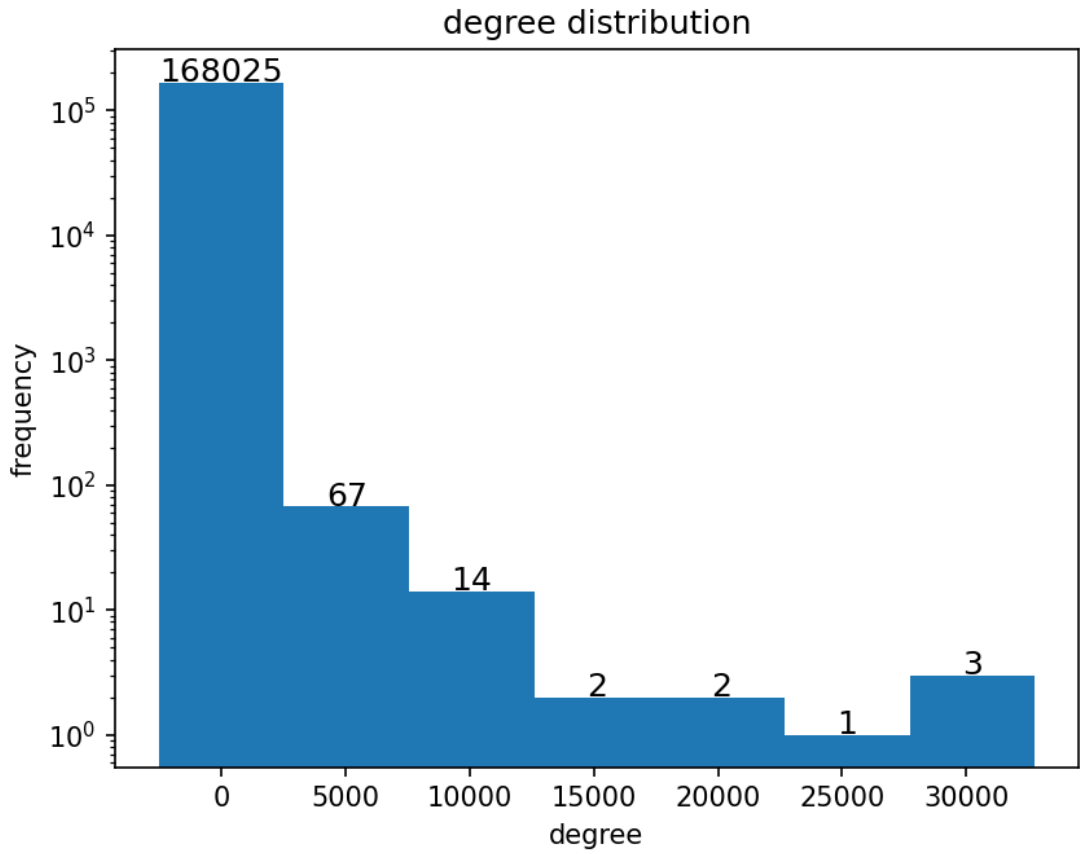
ID为0的节点的ego network:



ID为4的节点的ego network:



度的分布图:

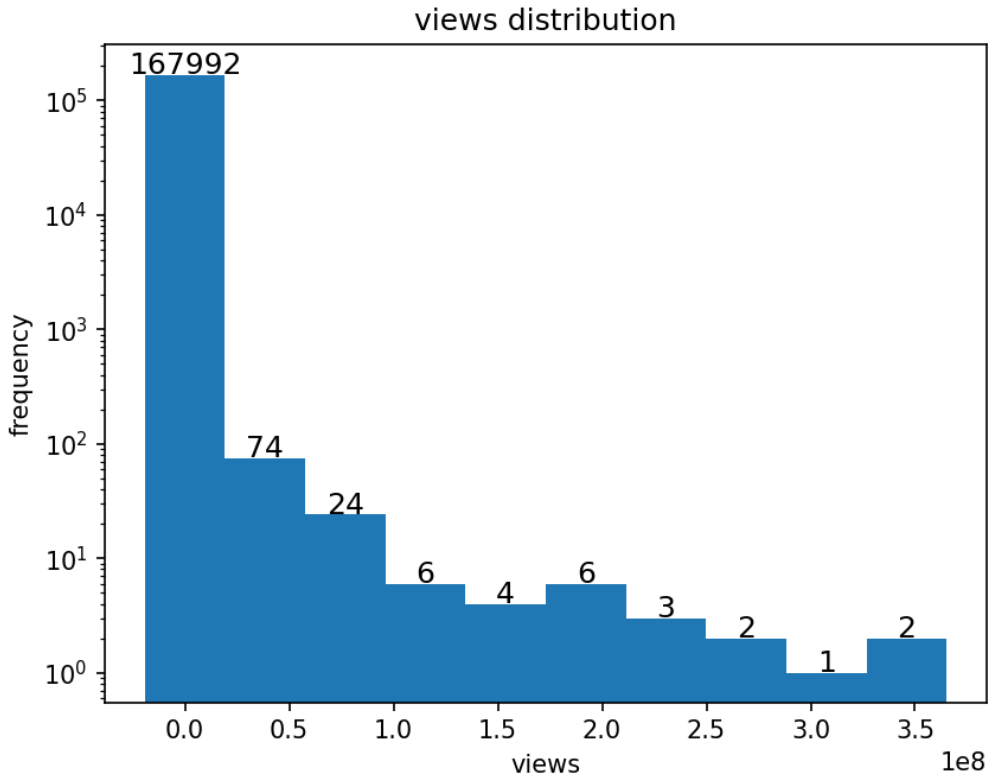


- plotnodes模块:

```
#plotnodes.py

import numpy as np
import matplotlib.pyplot as plt

def plot_nodes_attr(data, attr):
```

```
    """
    绘制图中节点属性的统计结果
    """
    attr_list = np.array(data[attr]).tolist()
    y, bins, patches = plt.hist(attr_list, bins=10, log=True,
align='left')
    for i in range(len(y)):
        plt.text(bins[i], y[i]*1.02, int(y[i]), fontsize=12,
horizontalalignment="center")  # 打标签
    plt.title(attr +' distribution')        # 标题
    plt.xlabel(attr)                        # x轴名
    plt.ylabel('frequency')                 # y轴名
    #plt.savefig('attr'+'分布直方图' + '.png')
    plt.show()
```

views属性的分布图:



> 所有的分布图也都用seaborn试着画过了，感觉大差不差。这个数据因为分布的不是很好看，画出来的图不如教程好看呜呜呜

任务二：（附加）对网络统计数据应用的思考

观察所构建的网络在平均度，度分布，甚至局部结构上的结果和形态，讨论如何用这些数据来对节点进行排序或者挑选，比如假设你想在这个网络上营销一个新游戏，应该找那些节点来“试用”，以快速地产生口碑？了解一些节点重要性的一些常见指标。

- 对节点进行排序或者挑选：

  比如如果我想在网络营销一个新游戏，我肯定会选择那些度很高的节点来实用，因为他们和更多的人有关联，具有良好的推广效果，从而快速的产生口碑

- 节点重要性的一些常见指标：

  度中心性：这个节点在网络中所处的位置，位置越中心的节点其价值也越大

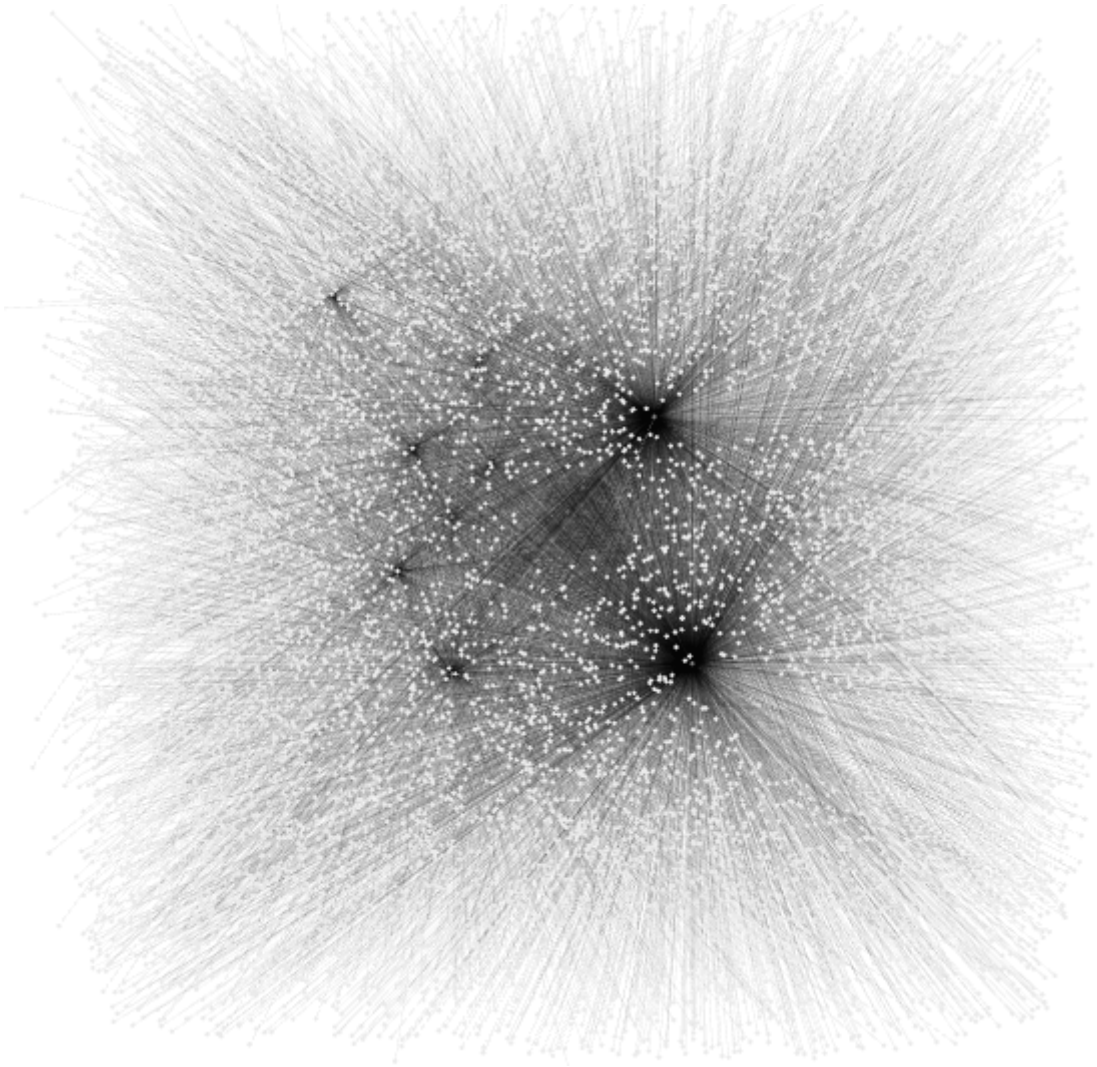
  介数中心性：就是如果要传输数据的话，那些最繁忙的点，即最常需要经过的点

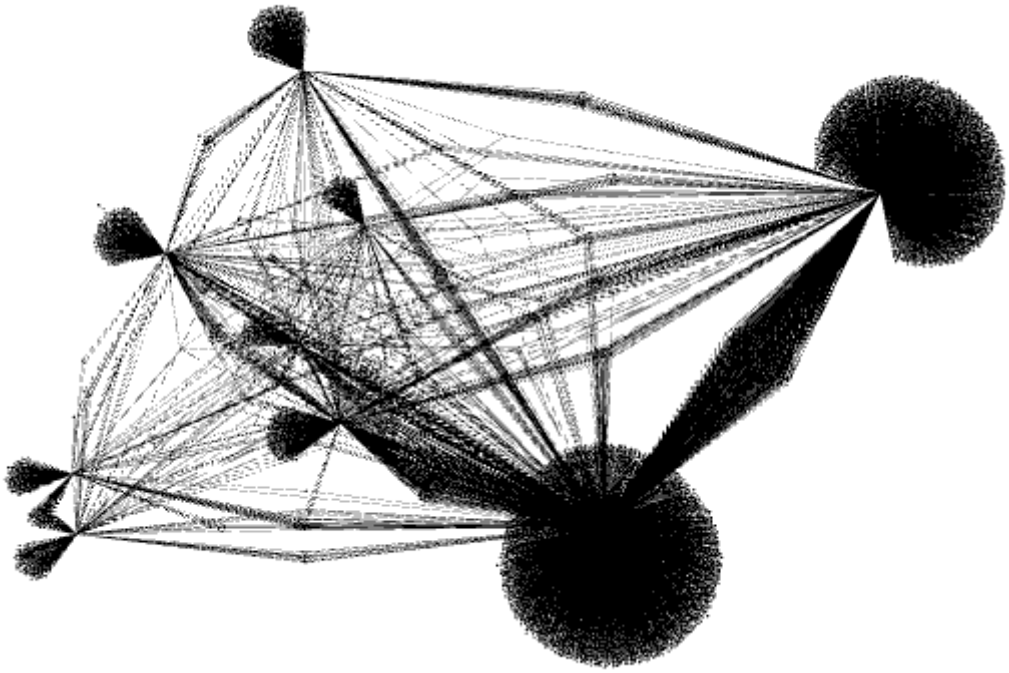
接近中心性：判断节点 i是否更接近其他节点

https://www.cnblogs.com/dong973711/p/14001306.html

任务三：了解并使用Gephi工具，尝试网络结构的可视化与社团分析等

跑全部数据差点没把我电脑跑卡死，已发朋友圈吐槽！！！故就跑了10000条边哈哈哈。





的确是一个很好的网络可视化工具，可以做的很漂亮，能够很清晰地看到社群的划分。

代码：

https://github.com/rachhhhing/MP2022/tree/master/week4/GraphStat

Ref:

> 本周作业特别鸣谢：hzh在周三让我发现了更多很好用很快捷的函数

- networkx

  5 内部存储结构：

  networkx将图的存储结构封装到了一个类中，这个类只提供了最简单的结点遍历、结点边的属性获取等操作，一个该类的实例就是一个图，所有的图数据存储于其中。networkx使用邻接表来存储图，使用一个三重的python dict来实现：

  - 第一重dict的key是结点，value是一个dict表示它的邻居结点，由于直接使用python的dict所以这里只要是可以hash的值都可以作为结点，不过一般实际中使用一个结点id来作为key；
  - 第二重dict就是上面的邻居结点，它的key是每个邻居结点，同理也是可以hash的类型，value是边上的多个属性；
  - 第三重dict就指边上的所有属性，key是边的属性名，value就是属性的值了；

  图类型中的成员方法直接访问这个底层结构，而networkx的api是通过图类中的方法来访问数据的，这样做的好处是以后可以将三重dict的存储方式替换成其它方式

  函数：https://blog.csdn.net/u012856866/article/details/116458059

  绘图：https://blog.csdn.net/fyfugoyfa/article/details/107830112

- pickle：http://c.biancheng.net/view/5736.html
- plt.hist：https://blog.csdn.net/u010916338/article/details/105663074
- seaborn：https://zhuanlan.zhihu.com/p/49035741
- 绘图配色总结：https://blog.csdn.net/qq_45759229/article/details/125440016
- Gephi：https://zhuanlan.zhihu.com/p/350447192