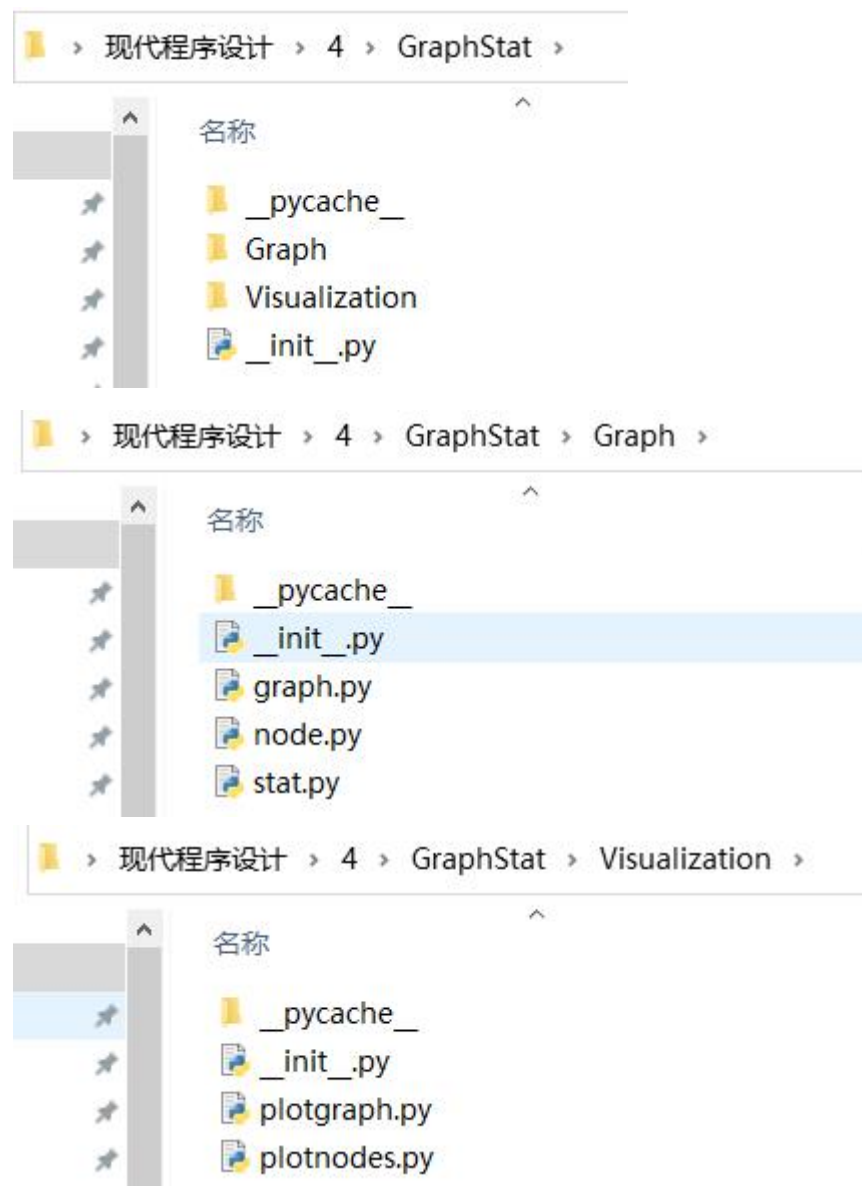


1. 首先建立包，分为图像信息处理和可视化两部分，如下图



2.

(1) 节点信息处理

利用字典，将节点的属性储存在字典中，每个字典是一个节点，将所有的节点以列表的形式储存，实现函数：init\_node ()

```
C: > Users > 86186 > Desktop > 现代程序设计 > 4 > GraphStat > Graph > node.py > init_node

1
2 def init_node(filepath):
3     res=[]
4     with open(filepath,'r',encoding="utf-8") as f:
5         for line in f.readlines():
6             if line == "*Vertices 34282\n":
7                 continue
8             elif line == "*Edges\n":
9                 break
10            else:
11                dict={}
12                infor=list(line.strip().split('\t'))
13                dict["节点id"]=infor[0]
14                dict["节点名称"]=infor[1]
15                dict["节点权重"]=infor[2]
16                dict["节点类型"]=infor[3]
17                dict["节点其他信息"]=infor[4]
18                dict["节点连接信息"]={}
19                res.append(dict)
20    return res
```

调用部分结果：

```
1 from GraphStat.Graph import *
2 from GraphStat.Visualization import plotgraph
3 from GraphStat.Visualization import plotnodes
4 filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"
5
6 nodelist=node.init_node(filepath)
7 print(nodelist)
```

```
{('节点id': '0', '节点名称': "Ann Blyth", '节点权重': '6035', '节点类型': 'starring', '节点其他信息': '1928 births;living people;American film actors;American musical theatre actors;American child actors;People from Westchester County, New York;', '节点连接信息': {}), ('节点id': '1', '节点名称': "Karen Allen", '节点权重': '7467', '节点类型': 'starring', '节点其他信息': 'American film actors;American stage actors;American video game actors;Bard College at Simon's Rock faculty;Illinois actors;People from Greene County, Illinois;Saturn Award winners;', '节点连接信息': {}), ('节点id': '2', '节点名称': "Mel Tormé", '节点权重': '18868', '节点类型': 'writer', '节点其他信息': '1925 births;1999 deaths;American actor-singers;American jazz singers;American Jews;American male singers;American singers;American television actors;Blue-eyed soul singers;Burials at Westwood Village Memorial Park Cemetery;Chicago musicians;Deaths from stroke;Grammy Award winners;Grammy Lifetime Achievement Award winners;Jewish actors;Jewish American musicians;Jewish singers;Jewish composers and songwriters;Traditional pop music singers;Russian-American Jews;', '节点连接信息': {}), ('节点id': '3', '节点名称': "Jane Anderson", '节点权重': '3355', '节点类型': 'director', '节点其他信息': 'American film actors;American film directors;LGBT directors;California actors;', '节点连接信息': {}), ('节点id': '4', '节点名称': "Lou Myers (actor)", '节点权重': '3288', '节点类型': 'starring', '节点其他信息': '1945 births;African American actors;American film actors;American television actors;Living people;People from Kanawha County, West Virginia;', '节点连接信息': {}), ('节点id': '5', '节点名称': "Mary Ainslee", '节点权重': '2274', '节点类型': 'starring', '节点其他信息': 'American film actors;', '节点连接信息': {}), ('节点id': '6', '节点名称': "Bridgette Wilson", '节点权重': '5465', '节点类型': 'starring', '节点其他信息': '1973 births;American film actors;American soap opera actors;American television actors;Living people;Miss Teen USA 1990 delegates;Miss Teen USA winners;Oregon actors;People from Curry County, Oregon;', '节点连接信息': {}), ('节点id': '7', '节点名称': "George W. Trendle", '节点权重': '12766', '节点类型': 'director', '节点其他信息': '1884 births;1972 deaths;American radio producers;American television producers;', '节点连接信息': {}), ('节点id': '8', '节点名称': "Victor Hugo", '节点权重': '42800', '节点类型': 'writer', '节点其他信息': "Victor Hugo ;Deists;French poets;French novelists;French dramatists and playwrights;French fantasy writers;French-language poets;Romantic poets;Writers who illustrated their own writing;Members of the Académie française;Lycée Louis-le-Grand alumni;People from Besançon;Burials at the Panthéon;Légion d'honneur recipients;1802 births;1885 deaths;Jersey writers;Guernsey writers;Philhellènes;French anti-death penalty activists;", '节点连接信息': {}), ('节点id': '9', '节点名
```

## (2) 节点信息输出

利用字典和 format 函数，将节点信息输出，实现函数：print\_node ()

调用部分结果如下：

```
from GraphStat.Graph import *
from GraphStat.Visualization import plotgraph
from GraphStat.Visualization import plotnodes
filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"

nodelist=node.init_node(filepath)
node.print_node(nodelist)
```

```

id:9520
name:"Arthur J. Nascarella"
weigh:1877
nodetype:starring
otherInformation:American film actors;American television actors;Living people;New York City Police Department officers;Year of birth missing (living people);United States Marines;
id:9521
name:"Alan Davies"
weigh:8701
nodetype:starring
otherInformation:1966 births;Old Bancroftians;Alumni of the University of Kent;English comedians;English actors;English radio personalities;Living people;People from Loughton;English atheists;QI;
id:9522
name:"Arsinée Khanjian"
weigh:3278
nodetype:starring
otherInformation:1958 births|Khanjian, Arsinée;Lebanese Armenians;Armenian Canadians|Khanjian, Arsinée;Armenian people;Canadian film actors|Khanjian, Arsinée;Lebanese Canadians;Gemini Award winners|Khanjian, Arsinée;Genie Award winners for Best Actress|Khanjian, Arsinée;Lebanese immigrants to Canada|Khanjian, Arsinée;Living people|Khanjian, Arsinée;People from Beirut|Khanjian;

```

### 3. 图的信息处理

#### (1) 初始化边信息，将边的信息以列表形式储存

```

C: > Users > 86186 > Desktop > 现代程序设计 > 4 > GraphStat > Graph > graph.py > ...

1  import json
2  def init_edge(filepath):
3      mark=0
4      with open(filepath,'r',encoding="utf-8") as f:
5          mid=[]
6          for line in f.readlines():
7              if line == "Edges\n":
8                  mark = 1
9              elif mark == 1:
10                 mid.append(list(line.strip().split('\t')))
11
12     return mid

```

运行结果：

```

PS C:\Users\86186\Desktop\现代程序设计\第2次> cd 'c:\Users\86186\Desktop\现代程序设计\第2次'; & 'C:\Users\86186\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\86186\vscode\extensions\ms-python.python-2021.10.1336267007\pythonFiles\lib\python\debuggy\launcher' '13709' '-' 'c:\Users\86186\Desktop\现代程序设计\4\图的建立与可视化.py'
[[['0', '4221', '1'], ['0', '4390', '1'], ['0', '2664', '1'], ['0', '6885', '1'], ['0', '989', '1'], ['0', '7387', '1'], ['1', '11814', '1'], ['1', '3869', '1'], ['1', '11348', '1'], ['1', '9023', '1'], ['1', '7353', '1'], ['1', '11190', '1'], ['1', '13214', '1'], ['1', '4562', '1'], ['1', '5465', '1'], ['1', '6106', '1'], ['1', '7922', '1'], ['1', '13175', '1'], ['1', '11814', '1'], ['2', '14185', '1'], ['2', '6938', '1'], ['2', '6938', '1'], ['2', '6255', '1'], ['2', '6826', '1'], ['2', '10597', '1'], ['2', '1870', '1'], ['2', '6172', '1'], ['2', '15364', '1'], ['2', '12753', '1'], ['2', '6938', '1'], ['2', '15364', '1'], ['2', '4388', '1'], ['2', '5924', '1'], ['2', '16173', '1'], ['2', '3898', '1'], ['3', '12931', '1'], ['3', '11371', '1'], ['3', '2748', '1'], ['3', '12206', '1'], ['3', '2498', '1'], ['5', '10462', '1'], ['6', '1985', '1'], ['6', '10341', '1'], ['6', '12388', '1'], ['6', '8410', '1'], ['7', '6466', '1'], ['10', '12853', '1'], ['11', '6513', '1'], ['12', '4599', '1'], ['13', '5279', '1'], ['13', '7705', '1'], ['13', '1121', '1'], ['13', '11326', '1'], ['14', '14271', '1'], ['15', '3685', '1'], ['15', '5696', '1'], ['15', '1426', '1'], ['15', '15551', '1'], ['15', '5758', '1'], ['15', '3685', '1'], ['15', '6327', '1'], ['15', '12835', '1'], ['17', '4063', '1'], ['17', '11576', '1'], ['17', '9090', '1'], ['18', '11926', '1'], ['19', '12669', '1'], ['20', '5002', '1'], ['20', '5196', '1'], ['20', '10246', '1'], ['20', '15689', '1'], ['20', '2502', '1'], ['20', '7007', '1'], ['20', '15815', '1'], ['20', '7723', '1'], ['20', '8559', '1'], ['20', '4891', '1'], ['21', '2507', '1'], ['21', '7232', '1'], ['22', '13393', '1'], ['22', '837', '1'], ['22', '242', '1'], ['24', '1168', '1'], ['24', '992', '1'], ['24', '6718', '1'], ['24', '2007', '1'], ['24', '10653', '1'], ['24', '10403', '1'], ['24', '1980', '1'], ['24', '10511', '1'], ['24', '13170', '1'], ['24', '15719', '1'], ['24', '9859', '1'], ['24', '9859', '1'], ['24', '10511', '1'], ['24', '10278', '1'], ['24', '14077', '1'], ['24', '3070', '1'], ['24', '10511', '1'], ['27', '14857', '1'], ['28', '10014', '1'], ['28', '9336', '1'], ['28', '7375', '1'], ['28', '5234', '1'], ['29', '11118', '1'], ['29', '1555', '1'], ['29', '5012', '1'], ['29', '1046', '1'], ['31', '5151', '1'], ['31', '5837', '1'], ['31', '382', '1'], ['31', '7868', '1'], ['31', '2360', '1'], ['31', '7868', '1'], ['34', '7663', '1'], ['34', '4580', '1'], ['34', '8659', '1'], ['34', '9565', '1'], ['34', '1696', '1'], ['34', '7182', '1'], ['34', '300', '1'], ['34', '14658', '1'], ['34', '4186', '1'], ['34', '4295', '1'], ['34', '14337', '1'], ['34', '873', '1'], ['34', '2979', '1'], ['34', '15390', '1'], ['34', '11904', '1'], ['34', '873', '1'], ['34', '4598', '1'], ['34', '14898', '1'], ['34', '14356', '1'], ['34', '11533', '1'], ['34', '2055', '1'], ['34', '3807', '1'], ['36', '13529', '1'], ['36', '14623', '1'], ['36', '9468', '1'], ['36', '7827', '1'], ['36', '14080', '1'], ['36', '14691', '1'], ['36', '10536', '1'], ['36', '9468', '1'], ['36', '6561', '1'], ['36', '13529', '1'], ['36', '9821', '1'], ['36', '8309', '1'], ['3

```

#### (2) 去掉重复边，计算权重，同时将节点信息和边信息整合在字典中，每个节点增加一个“邻居节点”的属性，以字典方式储存，键为与该节点相邻的节点，值为这两点形成的边的权重。

```
def init_graph(nodelist,edgelist):
    for i in range(len(edgelist)):
        if edgelist[i][1] in nodelist[int(edgelist[i][0])]["节点连接信息"].keys():
            nodelist[int(edgelist[i][0])]["节点连接信息"][edgelist[i][1]] += 1
        else:
            nodelist[int(edgelist[i][0])]["节点连接信息"][edgelist[i][1]] = 1

    return nodelist
```

运行结果：

```
C: > Users > 86186 > Desktop > 现代程序设计 > 4 > 图的建立与可视化.py > ...
1  from GraphStat.Graph import *
2  from GraphStat.Visualization import plotgraph
3  from GraphStat.Visualization import plotnodes
4  filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"
5
6  nodelist=node.init_node(filepath)
7
8
9  edgelist=graph.init_edge(filepath)
10
11 graphlist=graph.init_graph(nodelist,edgelist)
12 print(graphlist)
```

```
PS C:\Users\86186\Desktop\现代程序设计\第2次> c:; cd "c:\Users\86186\Desktop\现代程序设计\第2次"; & "C:\Users\86186\AppData\Local\Programs\Python\Python37\python.exe" "c:\Users\86186\vscode\extensions\ms-python.python-2021.10.1336267807\pythonfiles\lib\python\debugpy\launcher" "13848" "--" "c:\Users\86186\Desktop\现代程序设计\4\图的建立与可视化.py"
[{"节点id": "0", "节点名称": "Ann Blyth", "节点权重": "6035", "节点类型": "starring", "节点其他信息": "1928 births;Living people;American film actors;American musical theatre actors;American child actors;People from Westchester County, New York;", "节点连接信息": {"4221": 1, "4390": 1, "2664": 1, "6885": 1, "989": 1, "7387": 1}}, {"节点id": "1", "节点名称": "Karen Allen", "节点权重": "7467", "节点类型": "starring", "节点其他信息": "American film actors;American stage actors;American video game actors;Bard College at Simon's Rock faculty;Illinois actors;People from Greene County, Illinois;Saturn Award winners;", "节点连接信息": {"11814": 2, "3869": 1, "11348": 1, "9023": 1, "7353": 1, "11190": 1, "13214": 1, "4562": 1, "5465": 1, "6106": 1, "7922": 1, "13175": 1}}, {"节点id": "2", "节点名称": "Mel Tormé", "节点权重": "18868", "节点类型": "writer", "节点其他信息": "1901 births;Living people;American film actors;American musical theatre actors;American child actors;People from Westchester County, New York;"}]
```

### (3) 图信息的序列化储存和反序列化加载

为了方便图信息的储存，设计序列化函数

```
def save_graph(graphlist):
    return json.dumps(graphlist)

def load_graph(graphlist):
    return json.loads(graphlist)
```

## 4. 图的信息统计

### (1) 计算图的平均度

平均度为所有节点度之和除以节点个数，而所有节点度之和等于边的数量的 2 倍，由此可以计算平均度



```
def cal_graphavg(graphlist,edgelist):
    count = len(graphlist)
    edge = 2* len(edgelist)
    avg = edge/count
    return avg
```

运行结果：

```
from GraphStat.Graph import *
from GraphStat.Visualization import plotgraph
from GraphStat.Visualization import plotnodes
filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"

nodelist=node.init_node(filepath)

edgelist=graph.init_edge(filepath)

graphlist=graph.init_graph(nodelist,edgelist)
print([stat.cal_graphavg(graphlist,edgelist)])
```

```
PS C:\Users\86186\Desktop\现代程序设计\newmovies.txt>
8.308899454540152
```

(2) 获取节点的类型分布，单个节点以字典形式储存，所有节点以列表形式输出

```
def typestat(graphlist):
    res=[]
    for i in range(len(graphlist)):
        resdict = {}
        resdict["节点id"] = graphlist[i]["节点id"]
        resdict["节点名称"] = graphlist[i]["节点名称"]
        resdict["节点类型"] = graphlist[i]["节点类型"]
        res.append(resdict)
    return res
```

运行结果：

```
from GraphStat.Graph import *
from GraphStat.Visualization import plotgraph
from GraphStat.Visualization import plotnodes
filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"

nodelist=node.init_node(filepath)

edgelist=graph.init_edge(filepath)

graphlist=graph.init_graph(nodelist,edgelist)
print(stat.typestat(graphlist))
```

[[{"节点id": '0', "节点名称": "Ann Blyth", "节点类型": 'starring'}, {"节点id": '1', "节点名称": "Karen Allen", "节点类型": 'starring'}, {"节点id": '2', "节点名称": "Mel  
Torms", "节点类型": 'writer'}, {"节点id": '3', "节点名称": "Jane Anderson", "节点类型": 'director'}, {"节点id": '4', "节点名称": "Lou Myers (actor)", "节点类型": 'starr  
ing'}, {"节点id": '5', "节点名称": "Mary Ainslee", "节点类型": 'starring'}, {"节点id": '6', "节点名称": "Bridgette Wilson", "节点类型": 'starring'}, {"节点id": '7', "节  
点名称": "George W. Trendle", "节点类型": 'director'}, {"节点id": '8', "节点名称": "Victor Hugo", "节点类型": 'writer'}, {"节点id": '9", "节点名称": "Jeremy Leven", "节  
点类型": 'writer'}, {"节点id": '10", "节点名称": "Rodney Eastman", "节点类型": 'starring'}, {"节点id": '11", "节点名称": "David Greig (dramatist)", "节点类型": 'direct  
or"}, {"节点id": '12", "节点名称": "Emanuel Bernheim", "节点类型": 'writer'}, {"节点id": '13", "节点名称": "John Michael Higgins", "节点类型": 'starring'}, {"节点id":  
'14', "节点名称": "Tim DeKay", "节点类型": 'starring'}, {"节点id": '15", "节点名称": "Jason Miller (playwright)", "节点类型": 'writer'}, {"节点id": '16', "节点名称": "D

### (3) 获取节点的权重分布

```
def weighstat(graphlist):
    res=[]
    for i in range(len(graphlist)):
        resdict = {}
        resdict["节点id"] = graphlist[i]["节点id"]
        resdict["节点名称"] = graphlist[i]["节点名称"]
        resdict["节点权重"] = graphlist[i]["节点权重"]
        res.append(resdict)
    return res
```

运行结果：

```
重': '6881'), {"节点id": '130', "节点名称": "Charles S. Dutton", "节点权重": '8064'}, {"节点id": '131', "节点名称": "David Faustino", "节点权重": '5275'}, {"节点id": '13  
2', "节点名称": "Rain Phoenix", "节点权重": '7650'}, {"节点id": '133', "节点名称": "Victor Rasuk", "节点权重": '3683'}, {"节点id": '134', "节点名称": "Luther Vandross"  
, "节点权重": '26591'}, {"节点id": '135', "节点名称": "Ralph Meeker", "节点权重": '5683'}, {"节点id": '136', "节点名称": "Tony Lip", "节点权重": '799'}, {"节点id": '137'  
, "节点名称": "John Heyer", "节点权重": '17028'}, {"节点id": '138', "节点名称": "George Kirby", "节点权重": '3144'}, {"节点id": '139', "节点名称": "Mark Burton (writer)  
", "节点权重": '314'}, {"节点id": '140', "节点名称": "Doro Merande", "节点权重": '3839"}, {"节点id": '141', "节点名称": "Rick Famuyiwa", "节点权重": '461'}, {"节点id":  
'142', "节点名称": "Mira Sorvin  
PS C:\Users\86186\Desktop\现代程序设计\第2次> ]
```

### (4) 获取节点的自由度分布

```
def degreestat(graphlist):
    res=[]
    for i in range(len(graphlist)):
        resdict = {}
        resdict["节点id"] = graphlist[i]["节点id"]
        resdict["节点名称"] = graphlist[i]["节点名称"]
        degree=sum(list(graphlist[i]["节点连接信息"].values()))
        resdict["节点度"] = degree
        res.append(resdict)
    return(res)
```

运行结果：

"节点度": 11}, {"节点id": "270", "节点名称": "Gracie Allen"}, {"节点度": 12}, {"节点id": "271", "节点名称": "John Bennett (actor)"}, {"节点度": 1}, {"节点id": "272", "节点名称": "Ian Mune"}, {"节点度": 3}, {"节点id": "273", "节点名称": "Wendell Burton"}, {"节点度": 9}, {"节点id": "274", "节点名称": "Leo G. Carroll"}, {"节点度": 4}, {"节点id": "275", "节点名称": "George Melford"}, {"节点度": 3}, {"节点id": "276", "节点名称": "Paul Barresi"}, {"节点度": 17}, {"节点id": "277", "节点名称": "Philip Brophy"}, {"节点度": 1}, {"节点id": "278", "节点名称": "Cliff Owen"}, {"节点度": 1}, {"节点id": "279", "节点名称": "William Asher"}, {"节点度": 8}, {"节点id": "280", "节点名称": "Anton Yelchin"}, {"节点度": 5}, {"节点id": "281", "节点名称": "Jimmy Buffett"}, {"节点度": 6}, {"节点id": "282", "节点名称": "James Robert Baker"}, {"节点度": 3}, {"节点id": "283", "节点名称": "Lisa Morton"}, {"节点度": 0}, {"节点id": "284", "节点名称": "Joi Lansing"}, {"节点度": 15}, {"节点id": "285", "节点名称": "June Travis"}, {"节点度": 10}, {"节点id": "286", "节点名称": "Laurie R. King"}, {"节点度": 10}, {"节点id": "287", "节点名称": "Marilyn vos Savant"}, {"节点度": 10}, {"节点id": "288", "节点名称": "Michael Ondaatje"}, {"节点度": 10}, {"节点id": "289", "节点名称": "Nancy Mitton"}, {"节点度": 10}, {"节点id": "290", "节点名称": "Norman Macdonald"}, {"节点度": 10}, {"节点id": "291", "节点名称": "Oscar Brown Jr."}, {"节点度": 10}, {"节点id": "292", "节点名称": "Pamela Anderson"}, {"节点度": 10}, {"节点id": "293", "节点名称": "Patricia Richardson"}, {"节点度": 10}, {"节点id": "294", "节点名称": "Peter Onorati"}, {"节点度": 10}, {"节点id": "295", "节点名称": "Randy Newman"}, {"节点度": 10}, {"节点id": "296", "节点名称": "Richard Gere"}, {"节点度": 10}, {"节点id": "297", "节点名称": "Rick Warren"}, {"节点度": 10}, {"节点id": "298", "节点名称": "Robert De Niro"}, {"节点度": 10}, {"节点id": "299", "节点名称": "Sally Field"}, {"节点度": 10}, {"节点id": "300", "节点名称": "Shirley Temple"}, {"节点度": 10}, {"节点id": "301", "节点名称": "Steve Buscemi"}, {"节点度": 10}, {"节点id": "302", "节点名称": "Tina Turner"}, {"节点度": 10}, {"节点id": "303", "节点名称": "Tom Cruise"}, {"节点度": 10}, {"节点id": "304", "节点名称": "Uma Thurman"}, {"节点度": 10}, {"节点id": "305", "节点名称": "Vivian Maier"}, {"节点度": 10}, {"节点id": "306", "节点名称": "Warren Beatty"}, {"节点度": 10}, {"节点id": "307", "节点名称": "Will Ferrell"}, {"节点度": 10}, {"节点id": "308", "节点名称": "Yakovlev"}, {"节点度": 10}, {"节点id": "309", "节点名称": "Zoe Lister-Jones"}, {"节点度": 10}], [{"节点id": "270", "节点名称": "Gracie Allen", "x": 100, "y": 100}, {"节点id": "271", "节点名称": "John Bennett (actor)", "x": 100, "y": 150}, {"节点id": "272", "节点名称": "Ian Mune", "x": 100, "y": 200}, {"节点id": "273", "节点名称": "Wendell Burton", "x": 100, "y": 250}, {"节点id": "274", "节点名称": "Leo G. Carroll", "x": 100, "y": 300}, {"节点id": "275", "节点名称": "George Melford", "x": 100, "y": 350}, {"节点id": "276", "节点名称": "Paul Barresi", "x": 100, "y": 400}, {"节点id": "277", "节点名称": "Philip Brophy", "x": 100, "y": 450}, {"节点id": "278", "节点名称": "Cliff Owen", "x": 100, "y": 500}, {"节点id": "279", "节点名称": "William Asher", "x": 100, "y": 550}, {"节点id": "280", "节点名称": "Anton Yelchin", "x": 100, "y": 600}, {"节点id": "281", "节点名称": "Jimmy Buffett", "x": 100, "y": 650}, {"节点id": "282", "节点名称": "James Robert Baker", "x": 100, "y": 700}, {"节点id": "283", "节点名称": "Lisa Morton", "x": 100, "y": 750}, {"节点id": "284", "节点名称": "Joi Lansing", "x": 100, "y": 800}, {"节点id": "285", "节点名称": "June Travis", "x": 100, "y": 850}, {"节点id": "286", "节点名称": "Laurie R. King", "x": 100, "y": 900}, {"节点id": "287", "节点名称": "Marilyn vos Savant", "x": 100, "y": 950}, {"节点id": "288", "节点名称": "Michael Ondaatje", "x": 100, "y": 1000}, {"节点id": "289", "节点名称": "Nancy Mitton", "x": 100, "y": 1050}, {"节点id": "290", "节点名称": "Norman Macdonald", "x": 100, "y": 1100}, {"节点id": "291", "节点名称": "Oscar Brown Jr.", "x": 100, "y": 1150}, {"节点id": "292", "节点名称": "Pamela Anderson", "x": 100, "y": 1200}, {"节点id": "293", "节点名称": "Patricia Richardson", "x": 100, "y": 1250}, {"节点id": "294", "节点名称": "Peter Onorati", "x": 100, "y": 1300}, {"节点id": "295", "节点名称": "Randy Newman", "x": 100, "y": 1350}, {"节点id": "296", "节点名称": "Richard Gere", "x": 100, "y": 1400}, {"节点id": "297", "节点名称": "Rick Warren", "x": 100, "y": 1450}, {"节点id": "298", "节点名称": "Robert De Niro", "x": 100, "y": 1500}, {"节点id": "299", "节点名称": "Sally Field", "x": 100, "y": 1550}, {"节点id": "300", "节点名称": "Shirley Temple", "x": 100, "y": 1600}, {"节点id": "301", "节点名称": "Steve Buscemi", "x": 100, "y": 1650}, {"节点id": "302", "节点名称": "Tina Turner", "x": 100, "y": 1700}, {"节点id": "303", "节点名称": "Tom Cruise", "x": 100, "y": 1750}, {"节点id": "304", "节点名称": "Uma Thurman", "x": 100, "y": 1800}, {"节点id": "305", "节点名称": "Vivian Maier", "x": 100, "y": 1850}, {"节点id": "306", "节点名称": "Warren Beatty", "x": 100, "y": 1900}, {"节点id": "307", "节点名称": "Will Ferrell", "x": 100, "y": 1950}, {"节点id": "308", "节点名称": "Yakovlev", "x": 100, "y": 2000}, {"节点id": "309", "节点名称": "Zoe Lister-Jones", "x": 100, "y": 2050}]]

## 5. 图的可视化包

### (1) 图的自由度分布可视化

统计每个节点自由度的数量，计算出现的频率，以折线图的形式呈现图的自由度分布

```

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

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

def plotdegree(degree):
    degreeelist=[]
    for i in range(len(degree)):
        degreeelist.append(int(degree[i]["节点度"]))
    sumdegree=len(degreeelist)
    x=list(set(degreeelist))
    y=[]
    for i in x:
        caly=degreeelist.count(i)/sumdegree
        y.append(caly)
    x=np.array(x)
    y=np.array(y)
    plt.plot(x,y,ls='-', lw=0.5, color='purple')
    plt.xlabel("节点的度")
    plt.ylabel("频率")
    plt.title("度的分布")
    plt.grid(alpha=0.5,linestyle='-.')
    plt.show()

```

运行结果：

```

from GraphStat.Graph import *
from GraphStat.Visualization import plotgraph
from GraphStat.Visualization import plotnodes
filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"

odelist=node.init_node(filepath)

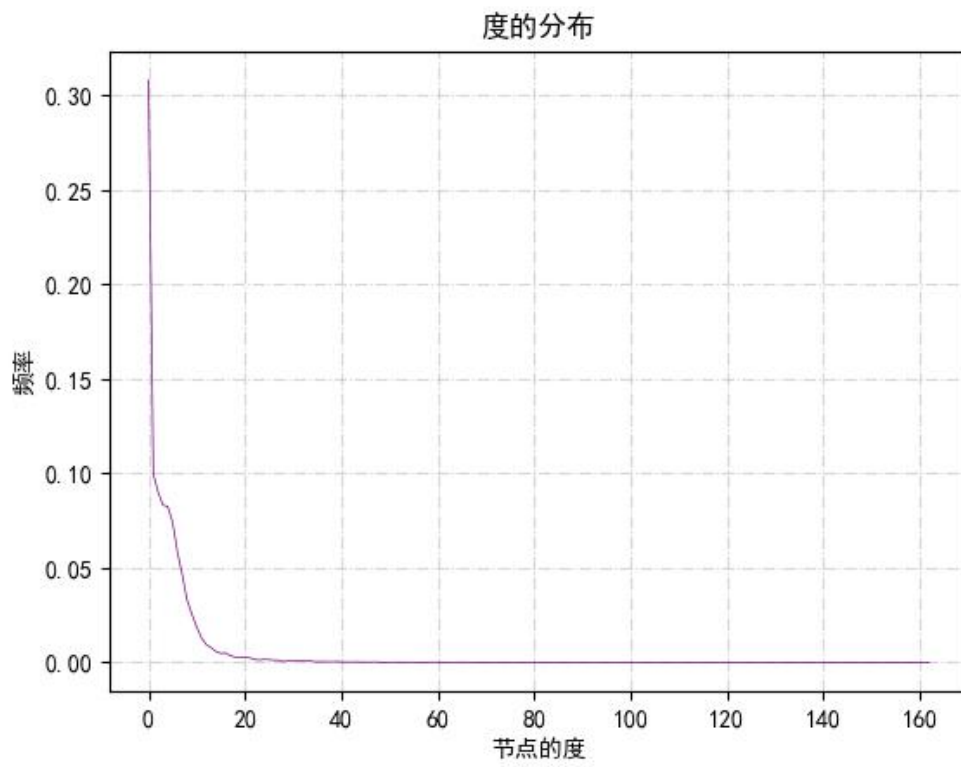
edgelist=graph.init_edge(filepath)

graphlist=graph.init_graph(odolist,edgelist)

degree=stat.degreestat(graphlist)
plotgraph.plotdegree(degree)

```





(2) 节点的权重分布可视化

统计不同权重出现的次数，以柱状图的形式呈现结果

```

import numpy as np
import matplotlib.pyplot as plt

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

def plotweigh(weigh):
    weighlist=[]
    for i in range(len(weigh)):
        weighlist.append(int(weigh[i]["节点权重"]))
    x=list(set(weighlist))
    y=[]
    for i in x:
        caly=weighlist.count(i)
        y.append(caly)
    x=np.array(x)
    y=np.array(y)
    print(x,y)
    plt.bar(x,y,color='purple')
    plt.xlabel("节点的权重")
    plt.ylabel("次数")
    plt.title("节点权重的分布")
    plt.grid(alpha=0.5,linestyle='-.')
    plt.show()

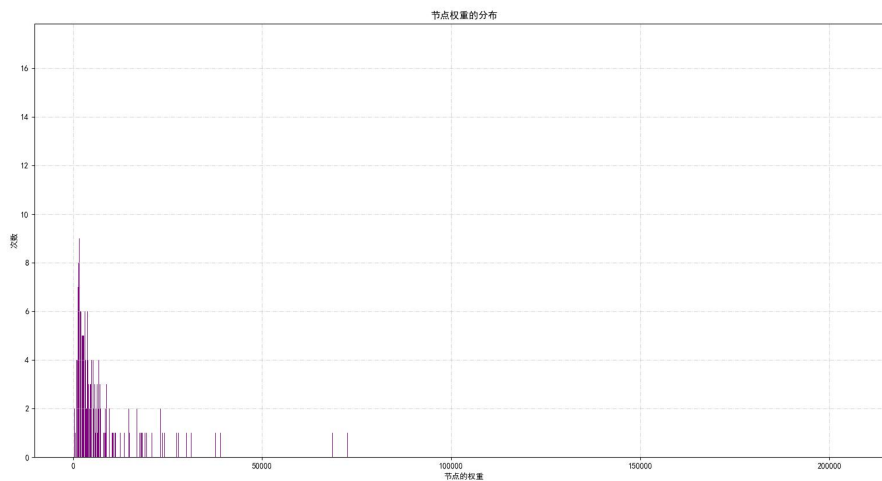
```

运行结果：

```

1  from GraphStat.Graph import *
2  from GraphStat.Visualization import plotgraph
3  from GraphStat.Visualization import plotnodes
4  filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"
5
6  nodelist=node.init_node(filepath)
7
8
9  edgelist=graph.init_edge(filepath)
10
11 graphlist=graph.init_graph(nodelist,edgelist)
12
13 #degree=stat.degreestat(graphlist)
14 #plotgraph.plotdegree(degree)
15 weigh=stat.weighstat(graphlist)
16 |plotnodes.plotweigh(weigh)

```



### (3) 节点类型分布可视化

统计不同类型节点的数目，以柱状图形式呈现

```
def plottype(type):
    typelist=[]
    for i in range(len(type)):
        typelist.append(type[i]["节点类型"])
    x=["writer","starring","director","movie"]
    y=[]
    for i in x:
        caly=typelist.count(i)
        y.append(caly)
    x=np.array(x)
    y=np.array(y)
    plt.bar(x,y, color='purple')
    plt.xlabel("节点的类型")
    plt.ylabel("次数")
    plt.title("节点类型的分布")
    plt.grid(alpha=0.5,linestyle='-.')
    plt.show()
```

运行结果：

```

from GraphStat.Graph import *
from GraphStat.Visualization import plotgraph
from GraphStat.Visualization import plotnodes
filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"

odelist=node.init_node(filepath)

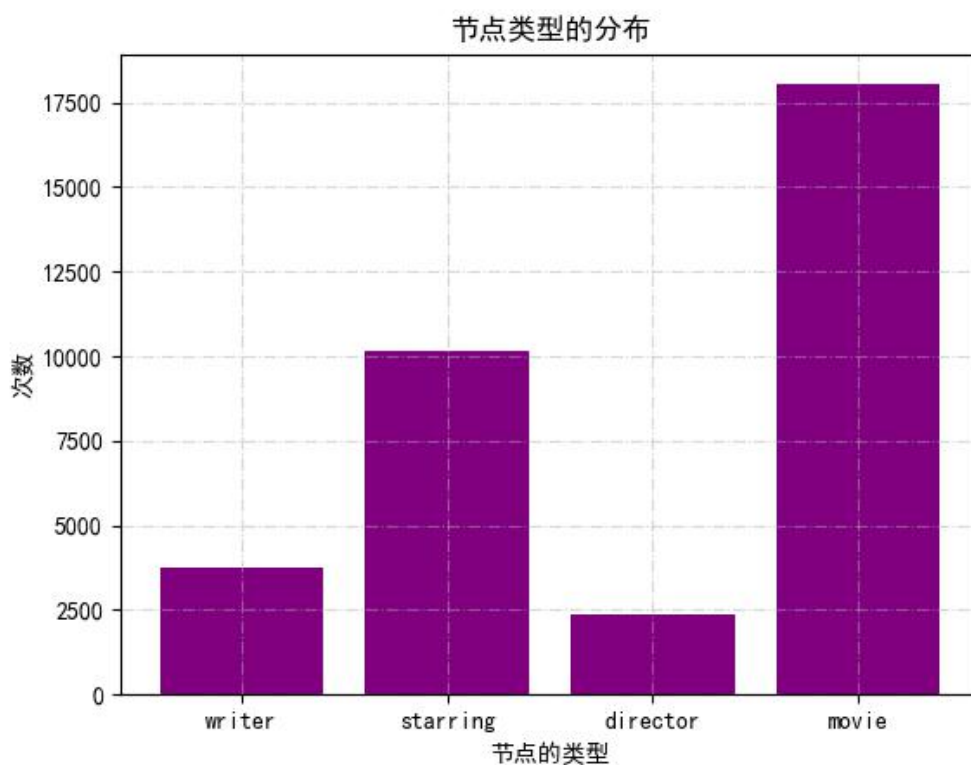
edgelist=graph.init_edge(filepath)

graphlist=graph.init_graph(odolist,edgelist)

#degree=stat.degreestat(graphlist)
#plotgraph.plotdegree(degree)
#weigh=stat.weighstat(graphlist)
#plotnodes.plotweigh(weigh)

type=stat.typestat(graphlist)
plotnodes.plottype(type)

```



6. 利用 networkx 库，画出部分节点之间的网络结构



此处取前 1000 个节点，按照有权无向图画出关系，并且标出节点 id

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import networkx as nx
4
def plotgraph(filepath):
    mark=0
    with open(filepath,'r',encoding="utf-8") as f:
        mid=[]
        for line in f.readlines():
            if line == "*Edges\n":
                mark = 1
            elif mark == 1:
                mid.append(tuple(map(int,line.strip().split('\t'))))
    G=nx.Graph()
    G.add_weighted_edges_from(mid[:1000])
    nx.draw(G,with_labels=True)
    plt.show()
```

运行结果：

```
from GraphStat.Graph import *
from GraphStat.Visualization import plotgraph
from GraphStat.Visualization import plotnodes
filepath = r"C:\Users\86186\Desktop\现代程序设计\newmovies.txt"

nodelist=node.init_node(filepath)

edgelist=graph.init_edge(filepath)

graphlist=graph.init_graph(nodelist,edgelist)

#degree=stat.degreestat(graphlist)
#plotgraph.plotdegree(degree)
#weigh=stat.weighstat(graphlist)
#plotnodes.plotweigh(weigh)
#type=stat.typestat(graphlist)
#plotnodes.plottype(type)
plotgraph.plotgraph(filepath)
```

