江 蘇 大 學

JIANGSU UNIVERSITY

《网络科学基础》

第三次平时作业

学院名称：计算机学院

专业班级：物联网 2303 班

学生姓名：邱佳亮

学生学号：3230611072

教师姓名：熊书明

2024 年 10 月

1. 题一代码

```
import networkx as nx  #导包
import matplotlib.pyplot as plt
import numpy as np
#%%
G=nx.Graph()
n=15 #节点数
H=nx.path_graph(n)#添加节点
G.add_nodes_from(H)
#%%
np.random.seed(10) #定义随机种子
def rand_edge(vi,vj,p=0.6):
    prob=np.random.rand()
    if(prob>p): #随机建立边 权重
        G.add_edge(vi,vj,weight=(prob*10))
i=0
while (i<n):
    j=0
    while j<i:
        rand_edge(i,j) #添加边
        j+=1
    i+=1

pos=nx.circular_layout(G) #可视化
options={
   "with_labels":True,
    "font_size":10,
    "font_weight":"bold",
    "font_color":"white",
    "node_size":200,
    "width":1
}
nx.draw(G,pos,**options)
ax=plt.gca()
ax.margins(0.2)
plt.axis('off')
plt.show()
Prim=nx.minimum_spanning_tree(G=G,algorithm='prim') #调用prim 算法
print(len(Prim.nodes))  #节点个数
print(len(Prim.edges))  #边数
sorted(Prim.edges(data=True)) #边集
```

1.1. 运行结果

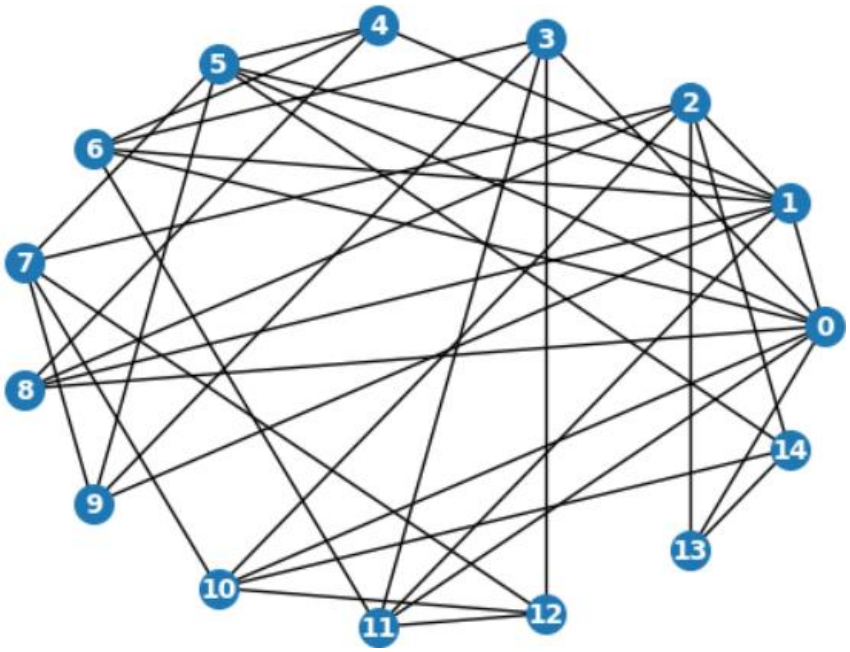

图 1　网络拓扑图

```
Prim=nx.minimum_spanning_tree(G=G,algorithm='prim') #调用prim算法
print(len(Prim.nodes))  #节点个数
print(len(Prim.edges))  #边数
sorted(Prim.edges(data=True)) #边集
[8]
```

```
15
14
[(0, 3, {'weight': 7.488038825386118}),
 (0, 6, {'weight': 6.1252606682938815}),
 (0, 8, {'weight': 6.503971819314672}),
 (0, 10, {'weight': 7.546476915298572}),
 (1, 2, {'weight': 6.336482349262754}),
 (1, 8, {'weight': 6.010389534045444}),
 (2, 7, {'weight': 6.741336150663453}),
 (2, 13, {'weight': 6.055775643937568}),
 (3, 9, {'weight': 6.262871483113925}),
 (4, 6, {'weight': 7.145757833976906}),
 (5, 7, {'weight': 6.177669784693172}),
 (7, 12, {'weight': 6.364911430675447}),
 (11, 12, {'weight': 7.63240587143681}),
 (13, 14, {'weight': 7.819491186191484})]
```

图 2　运行结果

2. 题二代码

```
Kruskal = nx.minimum_spanning_tree(G=G, algorithm='kruskal')
print(len(Kruskal.nodes))  #节点个数
print(len(Kruskal.edges))  #边数
sorted(Kruskal.edges(data=True))  #边集
```

2.1. 运行结果

```
Kruskal = nx.minimum_spanning_tree(G=G, algorithm='kruskal')
print(len(Kruskal.nodes))  #节点个数
print(len(Kruskal.edges))  #边数
sorted(Kruskal.edges(data=True))  #边集
[9]

 15
 14
 [(0, 3, {'weight': 7.488038825386118}),
  (0, 6, {'weight': 6.1252606682938815}),
  (0, 8, {'weight': 6.503971819314672}),
  (0, 10, {'weight': 7.546476915298572}),
  (1, 2, {'weight': 6.336482349262754}),
  (1, 8, {'weight': 6.010389534045444}),
  (2, 7, {'weight': 6.741336150663453}),
  (2, 13, {'weight': 6.055775643937568}),
  (3, 9, {'weight': 6.262871483113925}),
  (4, 6, {'weight': 7.145757833976906}),
  (5, 7, {'weight': 6.177669784693172}),
  (7, 12, {'weight': 6.364911430675447}),
  (11, 12, {'weight': 7.63240587143681}),
  (13, 14, {'weight': 7.819491186191484})]
```

图 3　运行结果

3. 题三代码

```
from time import time #导包
import datetime
#%%
def get_time_prim(G): #获取 Prim 算法时间
    t=[]
    for i in range(10):
        time0=time() #当前时间
        Prim=nx.minimum_spanning_tree(G=G,algorithm='prim')
        t.append(time()-time0) #计算时间差
     return np.mean(t)
def get_time_kruskal(G):
    t=[]
```

```
    for i in range(10):
        time0=time() #当前时间
        Kruskal=nx.minimum_spanning_tree(G=G,algorithm='kruskal')
        t.append(time()-time0) #计算时间差
    return np.mean(t)
def rand_edge(vi, vj, p=0.5):
    np.random.seed(10) #定义随机种子
    prob = np.random.rand()
    if (prob > p):
        G.add_edge(vi, vj, weight=(prob * 10))
def init_graph(n): #创建图
    G = nx.Graph()
    H = nx.path_graph(n)  #添加节点
    G.add_nodes_from(H)
    i = 0
    while (i < n):
        j = 0
        while (j < i):
            rand_edge(i, j)
            j += 1
        i += 1
    time_p=get_time_prim(G)
    time_k=get_time_kruskal(G)
    time_prim[str(n)]=time_p #键值对 存入字典
    time_kruskal[str(n)]=time_k
    return time_kruskal,time_prim #返回字典
#%%
#3min
time_prim={}
time_kruskal={}
i=100
while i<=11000:
    init_graph(i)
    i=i+2000 #规模每次增加2000
print(time_prim)
print(time_kruskal)
#%%
plt.plot(time_prim.keys(),time_prim.values(),label='Prim') #可视化
plt.plot(time_kruskal.keys(),time_kruskal.values(),label='Kruskal')
plt.legend(loc='best')
plt.xlabel("N")
plt.ylabel("t/s")
plt.grid()
plt.show()
```

3.1. 运行结果

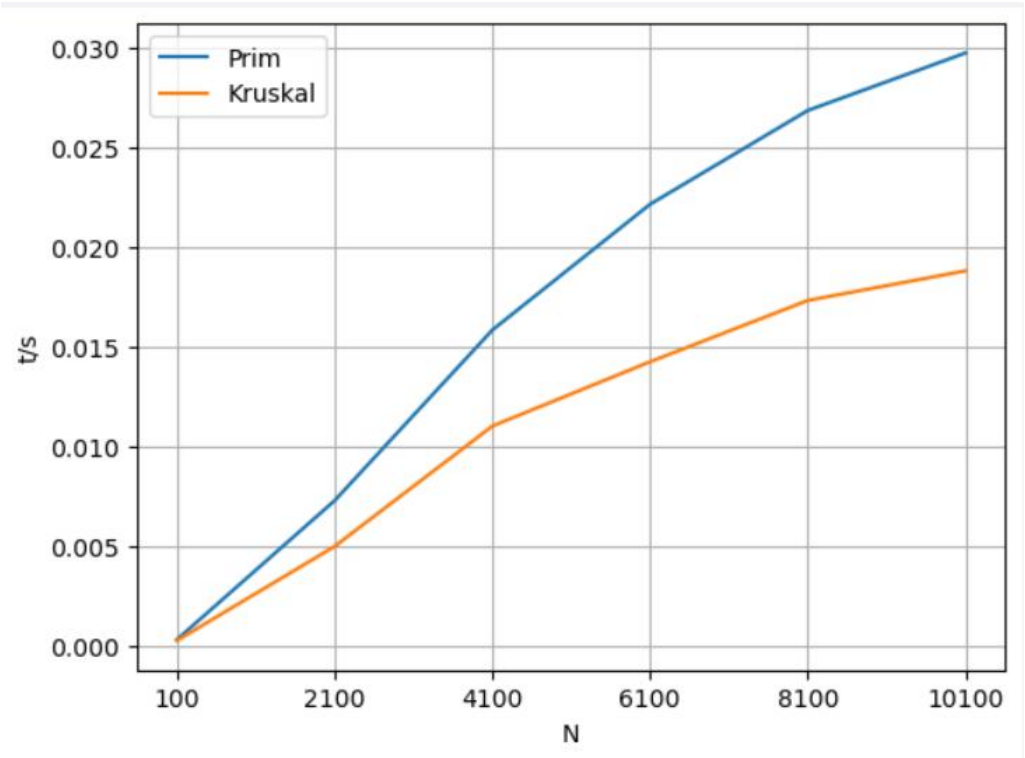

图 4　运行结果

4. 阅读论文

这篇论文的主要工作是研究加权无标度网络中的最小生成树。通过研究节点间权重分配与网络拓扑结构之间的关系，研究者们研究了这些关系如何影响复杂的网络的组织结构。如果权重分布与网络拓扑结构相关联，那么最小生成树的度分布要么是无标度的，要么是指数的。相反，如果权重和拓扑之间没有相关性，那么最小生成树的度分布遵循幂律分布，并且与权重分布无关。

首先，研究者们分析了各种真实系统的权重与网络结构之间的关系。然后，他们创建了一个模型，其权重分布模仿了真实网络的统计特征。他们仔细研究了权重分布对最小生成树结构的影响，使用各种权重分配方法。当权重分布与网络拓扑结构相关时，研究者发现最小生成树的度分布要么是指数的，要么是无标度的。最小生成树的度分布会呈现指数特性，如果节点度的最大值或权重与节点度

的乘积成正比。因此，这些最小生成树主要由低度节点组成，而枢纽节点在最小生成树中发挥的作用被边缘化。研究人员发现，当最小生成树的度分布与节点度的最小值成正比或与节点度的乘积成反比时，最小生成树的度分布呈现幂律特征。即使在拓扑结构与其权重分布相关的情况下，其权重分布仍然可能遵循幂律分布。在没有相关性的情况下，最小生成树的度和权重分布都依赖于幂律分布。

此外，该论文讨论了弱链接在网络中的作用。在某些加权网络中，最弱的链接可能对整个网络的性能产生重大影响。

最后，论文提出了未来研究的方向，包括权重的时间依赖性以及权重之间可能存在的联系，这些都可能对网络的动态行为产生影响。