source code

```
# -*- coding: utf-8 -*-
"""
Created on Fri Dec 18 15:17:32 2020
@author: pdsu_szj
import networkx as nx
import pylab
import random
import math
import openpyxl
import pathlib
import matplotlib.pyplot as plt
import operator
from itertools import groupby
path = 'TEST.txt'
name = 'TEST.xlsx'
G = nx.Graph()
with open(path) as file:
    for line in file:
        head, tail = [int(x) for x in line.split()]
        G.add_edge(head, tail)
nums = G.number_of_nodes()
print('总节点数',nums)
G.remove_edges_from(nx.selfloop_edges(G))
edgs=G.number_of_edges()
print('总边数',edgs)
print('总节点数',nums)
A=edgs*2/(nums*(nums-1))
print('图密度',A)
print('总边数',edgs)
print('总节点数',nums)
k_shell=nx.core_number(G)
def calculation_shortest_path()：
    i = 0
    for node in G.nodes():
        print("{node}节点写入 xlsx,已经计算了{num}".format(node=node,num=i))
        dict = {}
        for otherNode in G.nodes:
            if node == otherNode:
                continue
            try:
                distance = nx.shortest_path_length(G, node, otherNode)
                r =distance
```

```
                    if r not in dict.keys():
                        dict[r] = []
                    dict[r].append(otherNode)
                except:
                    print("{node}和{otherNode}没有边".format(node=node,otherNode=otherNode))
        for index in dict.keys():
            ws.cell(node+1,index).value = str(dict[index]).replace("[",'').replace("]",'').replace(' ','')
        i = i + 1
    wb.save(name)
if  pathlib.Path(name).exists():
    wb = openpyxl.load_workbook(name)
    ws = wb.active
else:
    wb = openpyxl.Workbook()
    wb.save("./"+name)
    ws = wb.active
    calculation_shortest_path()
def get_shortest_path(node,r):
    value =  ws.cell(node+1,r).value
    if value!= None :
        return  set(map(int,str(value).lstrip(',').rstrip(',').split(",")))
    return value
def get_all_shortest_path(node,r):
    setu = set()
    while r>=1:
        value = get_shortest_path(node+1,r)
        if value != None :
            setu = setu | value
        r= r - 1
    return setu
ec = nx.eigenvector_centrality(G, max_iter=5000)
print('ec',ec)
maxEc=max(ec.values())
minEc=min(ec.values())
z=maxEc-minEc
def calculation_Q (node):
    k2=ec[node]
    r = 1
    nodes = get_shortest_path(node,r)
    sums = 0;
    while nodes != None :
        lists = []
        for otherNode in list(nodes):
                    k3=k2-ec[otherNode]
```

```
                    T1= (G.degree(node))*math.exp(k3)
                    print('节点，T1',otherNode,T1)
                    k5=G.degree(otherNode)/(3.1415926**r)
                    print('节点，k5',otherNode,k5)
                    Q = k5*A*T1/r
                    print('节点，Q',otherNode,Q)
                    lists.append(Q)
            sums = sums + sum(lists)*1.0
            r = r + 1
            nodes = get_shortest_path(node,r)
        return sums
def algorithm1():
        dicts = {}
        i = 0
        for node in G.nodes():
            print("{node}节点正在计算,已经计算了{num}".format(node=node,num=i))
            dicts[node] = calculation_Q(node)
            i = i+1
        return dicts
res = algorithm1()
print("res=",res)
nodelist=[]
sortNum = sorted(res.items(), key=lambda x: x[1], reverse=True)
for key in sortNum:
        nodelist.append(key.__getitem__(0))
print(nodelist)
print('图密度',A)
print('总边数',edgs)
print('总节点数',nums)
f =open('outputdata\\rcd-8_'+path, "w+")
for key,val in sortNum:
    f.write(str(key)+'\t'+str(val)+"\n")
f.close()
nodelist1=[]
sortNum1 = sorted(res.items(), key=lambda x: x[0], reverse=False)
for key in sortNum1:
        nodelist1.append(key.__getitem__(0))
print(nodelist1)
f =open('outputdata\\rcd-8_1'+path, "w+")
for key,val in sortNum1:
    f.write(str(key)+'\t'+str(val)+"\n")
f.close()
```