cpts591_Mengxiao_hw1

February 18, 2020

[1]:
```
from igraph import *
```

[2]:
```
# At first, we need to read the three networks and generate three Erdos-Renyi␣
↪random networks.
```

[3]:
```
# Political blogs
g_PB = Graph.Read_GML('polblogs.gml')
# Neural network
g_NN = Graph.Read_GML('celegansneural.gml')
# Internet
g_In = Graph.Read_GML('as-22july06.gml')
# With n = 2000, p = 0.01
g_01 = Graph.GRG(2000, 0.01)
# With n = 2000, p = 0.005
g_005 = Graph.GRG(2000, 0.005)
# With n = 2000, p = 0.0025
g_0025 = Graph.GRG(2000, 0.0025)
```

[4]:
```
# Q1. calculate the information for each of the six networks.
```

[5]:
```
g = {}
g['PB']={'model': g_PB, 'Network': 'Political blogs'}
g['NN']={'model': g_NN, 'Network': 'Neural network'}
g['In']={'model': g_In, 'Network': 'Internet'}
g['01']={'model': g_01, 'Network': 'G(2000,0.01)'}
g['005']={'model': g_005, 'Network': 'G(2000,0.005)'}
g['0025']={'model': g_0025, 'Network': 'G(2000, 0.0025)'}
names = ['PB', 'NN', 'In', '01', '005', '0025']
```

[6]:
```
for i in names:
    g[i]['Type'] = g[i]['model'].is_directed()
    g[i]['n'] = g[i]['model'].vcount()
    g[i]['m'] = g[i]['model'].ecount()
    if g[i]['Type']:
        g[i]['c-strong'] = g[i]['model'].components().summary()
        g[i]['c-weak'] = g[i]['model'].components(WEAK).summary()
    else:
```

```
        g[i]['c'] = g[i]['model'].components().summary()
    g[i]['d'] = g[i]['model'].maxdegree()
    g[i]['l'] = g[i]['model'].average_path_length()
    g[i]['L'] = g[i]['model'].diameter()
    g[i]['ccl'] = g[i]['model'].transitivity_avglocal_undirected()
    g[i]['ccg'] = g[i]['model'].transitivity_undirected()
```

[7]:
```
g
```

[7]:
```
{'PB': {'model': <igraph.Graph at 0x7f35cc03ee58>,
  'Network': 'Political blogs',
  'Type': True,
  'n': 1490,
  'm': 19090,
  'c-strong': 'Clustering with 1490 elements and 688 clusters',
  'c-weak': 'Clustering with 1490 elements and 268 clusters',
  'd': 468,
  'l': 3.3901837252152363,
  'L': 9,
  'ccl': 0.3600286522101186,
  'ccg': 0.2259585173589758},
 'NN': {'model': <igraph.Graph at 0x7f35b3fbb048>,
  'Network': 'Neural network',
  'Type': True,
  'n': 297,
  'm': 2359,
  'c-strong': 'Clustering with 297 elements and 57 clusters',
  'c-weak': 'Clustering with 297 elements and 1 clusters',
  'd': 139,
  'l': 3.9918839808408726,
  'L': 14,
  'ccl': 0.30791453707858335,
  'ccg': 0.18071147126607687},
 'In': {'model': <igraph.Graph at 0x7f359a088408>,
  'Network': 'Internet',
  'Type': False,
  'n': 22963,
  'm': 48436,
  'c': 'Clustering with 22963 elements and 1 clusters',
  'd': 2390,
  'l': 3.842426273858345,
  'L': 11,
  'ccl': 0.3499153584893828,
  'ccg': 0.011146383847822162},
 '01': {'model': <igraph.Graph at 0x7f359a0884f8>,
  'Network': 'G(2000,0.01)',
  'Type': False,
```

```
 'n': 2000,
 'm': 633,
 'c': 'Clustering with 2000 elements and 1428 clusters',
 'd': 6,
 'l': 1.4457274826789839,
 'L': 6,
 'ccl': 0.5620967741935483,
 'ccg': 0.5431034482758621},
'005': {'model': <igraph.Graph at 0x7f359a0885e8>,
 'Network': 'G(2000,0.005)',
 'Type': False,
 'n': 2000,
 'm': 153,
 'c': 'Clustering with 2000 elements and 1850 clusters',
 'd': 2,
 'l': 1.0254777070063694,
 'L': 2,
 'ccl': 0.6923076923076923,
 'ccg': 0.6923076923076923},
'0025': {'model': <igraph.Graph at 0x7f359a0886d8>,
 'Network': 'G(2000, 0.0025)',
 'Type': False,
 'n': 2000,
 'm': 34,
 'c': 'Clustering with 2000 elements and 1966 clusters',
 'd': 2,
 'l': 1.0285714285714285,
 'L': 2,
 'ccl': 0.0,
 'ccg': 0.0}}
```

[8]:
```python
# Q2. Plot the degree distribution of each network.
import cairocffi
from igraph import plot
```

[9]:
```python
for i in names:
    plot(g[i]['model'].degree_distribution(),
         g[i]['Network']+"_degree_distribution.png",
         bbox=(300, 300), margin=20)
```

[31]:
```python
# A2. According to the graph, I find that the degree distribution graph of
# the Neural network and Internet is obviously fit the Poisson distribution,
# so that they are the E-R random graph, but the other network look
# like power law distribution, but according to the theory, what our
# build is an E-R random graph, so their degree distribution graph may
# be still a Poisson distribution but just shifted left.
```

[10]:
```
# Q3. Plot the pathlength distribution of each network.
```

[11]:
```
for i in names:
    plot(g[i]['model'].path_length_hist(),
         g[i]['Network']+"_path_length_hist.png",
         bbox=(300, 300), margin=20)
    print("-"*20+g[i]['Network']+"-"*20)
    print(g[i]['model'].path_length_hist())
```

```
********************Political blogs********************
N = 981248, mean +- sd: 3.3902 +- 1.1302
Each * represents 5901 items
[ 1,  2): *** (19022)
[ 2,  3): ******************************** (193830)
[ 3,  4): *********************************************************** (348198)
[ 4,  5): ********************************************** (275702)
[ 5,  6): ****************** (107394)
[ 6,  7): **** (25602)
[ 7,  8): * (10092)
[ 8,  9):  (1371)
[ 9, 10):  (37)
********************Neural network********************
N = 67644, mean +- sd: 3.9919 +- 2.0243
Each * represents 339 items
[ 1,  2): ****** (2345)
[ 2,  3): ********************************** (11767)
[ 3,  4): ************************************************************ (20346)
[ 4,  5): ***************************************** (14133)
[ 5,  6): ****************** (6421)
[ 6,  7): ************ (4323)
[ 7,  8): ********* (3318)
[ 8,  9): ***** (1984)
[ 9, 10): **** (1442)
[10, 11): ** (946)
[11, 12): * (454)
[12, 13):  (142)
[13, 14):  (21)
[14, 15):  (2)
********************Internet********************
N = 263638203, mean +- sd: 3.8424 +- 0.8957
Each * represents 2024692 items
[ 1,  2):  (48436)
[ 2,  3): ***** (11063714)
[ 3,  4): ***************************************** (84546313)
[ 4,  5): ******************************************************** (113382789)
[ 5,  6): ********************** (44950231)
[ 6,  7): **** (8674704)
```

```
[ 7,  8):  (913431)
[ 8,  9):  (56317)
[ 9, 10):  (2214)
[10, 11):  (53)
[11, 12):  (1)
********************G(2000,0.01)********************
N = 866, mean +- sd: 1.4457 +- 0.9085
Each * represents 9 items
[1, 2): **********************************************************************
(633)
[2, 3): **************** (147)
[3, 4): **** (44)
[4, 5): ** (22)
[5, 6): * (15)
[6, 7):  (5)
********************G(2000,0.005)********************
N = 157, mean +- sd: 1.0255 +- 0.1581
Each * represents 2 items
[1, 2):
****************************************************************************
(153)
[2, 3): ** (4)
********************G(2000, 0.0025)********************
N = 35, mean +- sd: 1.0286 +- 0.1690
[1, 2): ********************************** (34)
[2, 3): * (1)
```

[32]:
```
# A3. From the graph I can find that all of three real-world network given
# by professor are Poisson distribution and most of the path length are
# 3 to 5. But all of the three E-R random graph built by us still like
# power law distribution and their path length are most 1.
```

[39]:
```
# Q4. Choose one real-world network and do the analysis in (1)-(3).
# I choose the network "Les Miserables" on the web given by professor.
# It contains the weighted network of coappearances of characters in
# Victor Hugo's novel "Les Miserables". Nodes represent characters as
# indicated by the labels and edges connect any pair of characters that
# appear in the same chapter of the book.  The values on the edges are
# the number of such coappearances.

g_LM = Graph.Read_GML('lesmis.gml')
g['LM']={'model': g_LM, 'Network': 'Les Miserables'}
g['LM']['Type']=g_LM.is_directed()
g['LM']['n']=g_LM.vcount()
g['LM']['m']=g_LM.ecount()
if g['LM']['Type']:
    g['LM']['c-strong']=g_LM.components().summary()
```

```python
    g['LM']['c-weak']=g_LM.components(WEAK).summary()
else:
    g['LM']['c']=g_LM.components(WEAK).summary()
g['LM']['d']=g_LM.maxdegree()
g['LM']['l']=g_LM.average_path_length()
g['LM']['L']=g_LM.diameter()
g['LM']['ccl']=g_LM.transitivity_avglocal_undirected()
g['LM']['ccg']=g_LM.transitivity_undirected()
plot(g_LM.degree_distribution(), g['LM']['Network']+"_degree_distribution.png",
bbox=(300, 300), margin=20)
plot(g_LM.path_length_hist(), g['LM']['Network']+"_path_length_hist.png",
bbox=(300, 300), margin=20)
print(g['LM'])
print('*'*20+"Les Miserables"+'*'*20)
print("degree distribution: ")
print(g_LM.degree_distribution())
print("path length hist")
print(g_LM.path_length_hist())
```

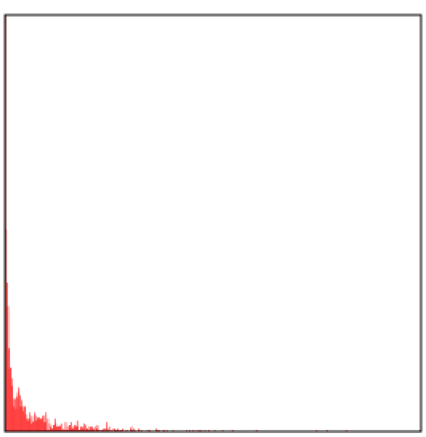
```
{'model': <igraph.Graph object at 0x7f359a0888b8>, 'Network': 'Les Miserables',
'Type': False, 'n': 77, 'm': 254, 'c': 'Clustering with 77 elements and 1
clusters', 'd': 36, 'l': 2.6411483253588517, 'L': 5, 'ccl': 0.735525495746084,
'ccg': 0.49893162393162394}
********************Les Miserables********************
degree distribution:
N = 77, mean +- sd: 6.5974 +- 6.0399
[ 1,  2): ***************** (17)
[ 2,  3): ********** (10)
[ 3,  4): ****** (6)
[ 4,  5): *** (3)
[ 5,  6):  (0)
[ 6,  7): ***** (5)
[ 7,  8): ********** (10)
[ 8,  9): * (1)
[ 9, 10): *** (3)
[10, 11): ***** (5)
[11, 12): ****** (6)
[12, 13): ** (2)
[13, 14): ** (2)
[14, 15):  (0)
[15, 16): ** (2)
[16, 17): * (1)
[17, 18): * (1)
[18, 19):  (0)
[19, 20): * (1)
[20, 21):  (0)
[21, 22):  (0)
```

```
[22, 23): * (1)
[23, 24):  (0)
[24, 25):  (0)
[25, 26):  (0)
[26, 27):  (0)
[27, 28):  (0)
[28, 29):  (0)
[29, 30):  (0)
[30, 31):  (0)
[31, 32):  (0)
[32, 33):  (0)
[33, 34):  (0)
[34, 35):  (0)
[35, 36):  (0)
[36, 37): * (1)
path length hist
N = 2926, mean +- sd: 2.6411 +- 0.8556
Each * represents 19 items
[1, 2): ************* (254)
[2, 3): **************************************************** (995)
[3, 4): ***************************************************************** (1251)
[4, 5): ********************* (399)
[5, 6): * (27)
```

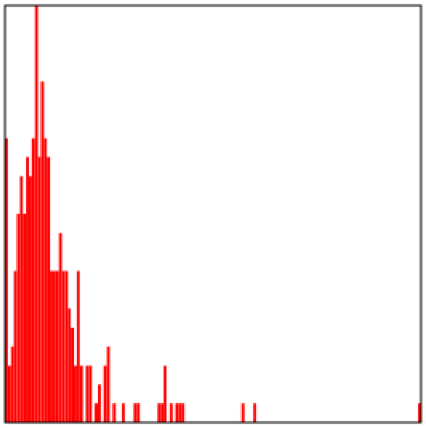
[34]:
```
# A4. According to the graph, we can find the degree distribution of
# the Les Miserables network is still Poisson distribution but it's
# path length hist is clearly not Poisson distribution or power law
# distribution. But there exist a point with 36 degree, this point may
# be very important.
```

——————Degree distribution graph——————

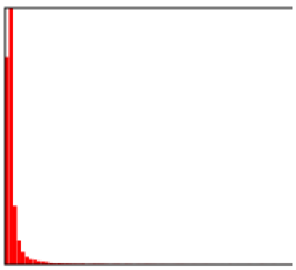
Political blogs:



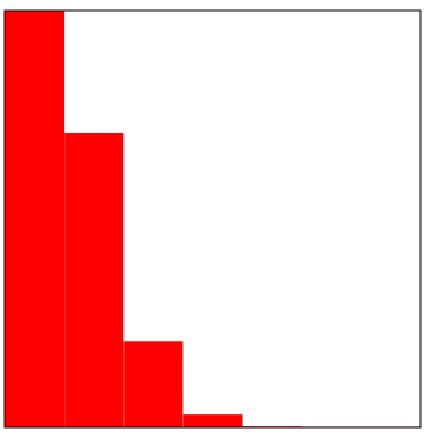
Neural network:



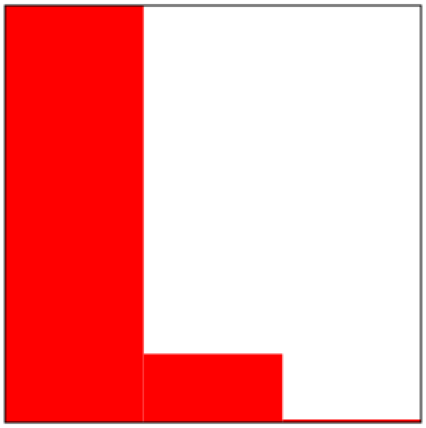
Internet:



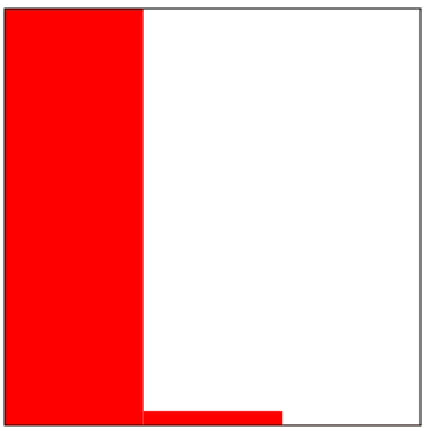
G(2000,0.01)



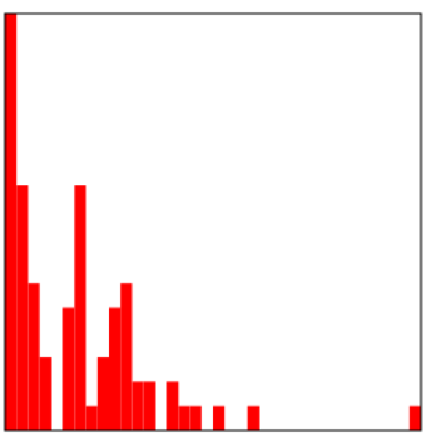
G(2000,0.005)



G(2000,0.0025)

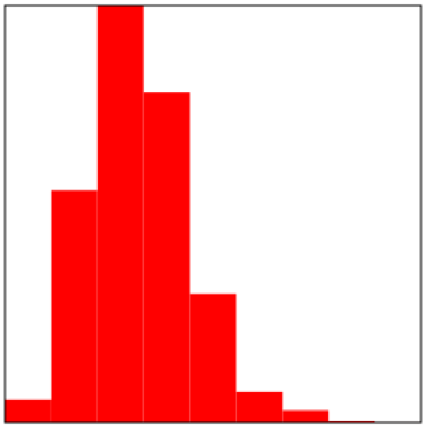


Les Miserables:

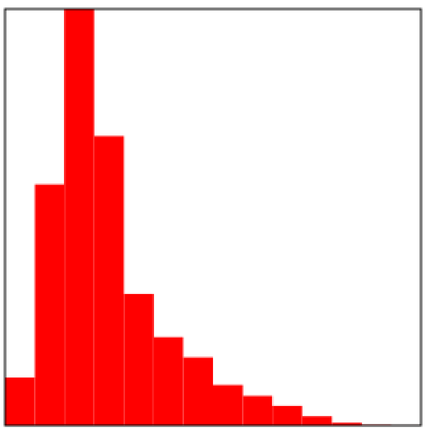


——————path length hist——————

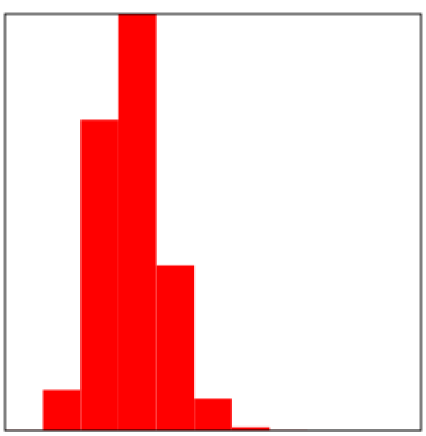
Political blogs:



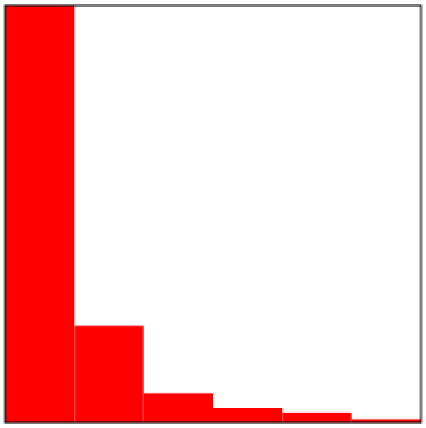
Neural network:



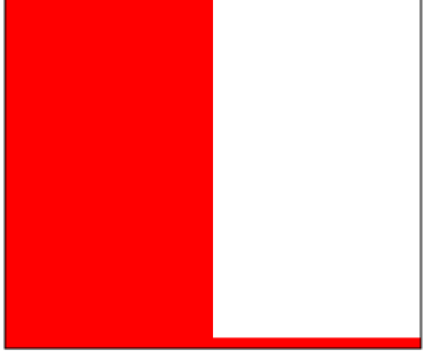
Internet:



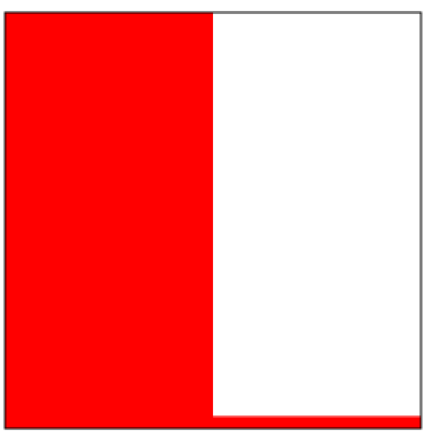
G(2000,0.01)



G(2000,0.005)



G(2000,0.0025)



Les Miserables: