Nome: Aimê Gomes da Nobrega (nro USP 11882429)

Disciplina: Redes Complexas

In [1]:
```
from numpy import *
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
```

In [2]:
```
from scipy.stats import pearsonr
```

In [3]:
```
def distribution_shortest_path(G):
    sv = [] ##shortest value
    for i in nx.shortest_path_length(G):
        values = np.array(list(dict(i[1]).values()))

#         print(values)
        for j in values[1:]:
            sv.append(j)
    maxk = np.max(sv)
    mink = np.min(sv)
    kvalues = np.arange(0, maxk+1) ## range de valores possíveis
de k (tipo range()) ##intervalo (min, max)
    Pk = np.zeros(maxk +1) ##vetor, q é uma lista sem vírgula
    for k in sv:
        Pk[k] = Pk[k] + 1
    Pk = Pk/sum(Pk) ### sum(Pk) == 1
    return kvalues, Pk

def shannon_entropy_short(G):
    k,Pk = distribution_shortest_path(G)
    H = 0
    for p in Pk:
        if(p > 0):
            H = H - p*math.log(p, 2)
    return H
```

In [4]:
```
def clean_graph(G, remove_edges = True):

    G = G.to_undirected()

    if remove_edges:

        G.remove_edges_from(nx.selfloop_edges(G))

    G_cc = sorted(nx.connected_components(G), key = len, reverse
=True)
    G = G.subgraph(G_cc[0])
    G = nx.convert_node_labels_to_integers(G, first_label = 0 )

    return G

def knnk(G):
    knnk_G = []
    ks_G = []
    knn_G = knn(G)
    vk = np.array(list(dict(G.degree).values()))

    for k in np.arange(vk.min(), vk.max()):
        aux = vk == k
        if len(knn_G[aux]) > 0 :
            average_knn = knn_G[aux].mean()
            knnk_G.append(average_knn)
            ks_G.append(k)

    return ks_G, knnk_G

def knn(G):
    knn_G = np.zeros(len(G.nodes), dtype = float)
    for i in G.nodes:
        aux = nx.average_neighbor_degree(G, nodes = [i])
        knn_G[i] = float(aux[i])

    return knn_G
```

1.Para a rede “Hamsterster”, calcule a média dos menores caminhos e o diâmetro. Use apenas o maior componente da rede e remova ciclos ou auto-conexões.

In [5]:
```
G_hams = clean_graph(nx.read_edgelist('/home/aime/Documents/redes
_complexas/ex_2/data/hamsterster.txt', nodetype = int))
```

In [6]:
```
if nx.is_connected(G_hams) == True:
    l = nx.average_shortest_path_length(G_hams)
    print('Average shortest path lenght of Hamsterster Network:
', '%3.4f'%l)
else:
    print('The graph has more than one connected component')

d = nx.diameter(G_hams)
print('Hamsterster Network diameter: ', d)
```

```
Average shortest path lenght of Hamsterster Network:  3.4526
Hamsterster Network diameter:  14
```

2.Considere a rede “USairport500”e calculea média e variância dos menores caminhos. Use apenas o maior componente da rede e remova ciclos ou auto-conexões.

In [7]:
```
G_US = clean_graph(nx.read_edgelist('ex_2/data/USairport500.txt',
nodetype = int))
```

In [8]:
```
sv = [] ##shortest value
for i in nx.shortest_path_length(G_US):
    values = np.array(list(dict(i[1]).values()))

#         print(values)
    for j in values[1:]:
        sv.append(j)

N = len(G_US)

sv = np.array(sv)
mean_sv = sv.sum()/(N*(N-1))

var = sum((sv - mean_sv)**2)/(N*(N-1))
var == sv.var()
print('Média dos menores caminhos da rede USairport500: ',round(m
ean_sv))
print('Variância dos menores caminhos da rede USairport500: ',rou
nd(var))
```

```
Média dos menores caminhos da rede USairport500:  3.0
Variância dos menores caminhos da rede USairport500:  1.0
```

3.Para a rede “USairport500”, calcule a entropia de Shannon da distribuiçãodos menores caminhos. Use logaritmo na base 2 e considere apenas o maior componente da rede.

In [9]:
```
shannon_US = shannon_entropy_short(G_US)

print('Entropia de Shannon da distribuiçãodos menores caminhos da
rede USairport500: ', shannon_US)
```

```
Entropia de Shannon da distribuiçãodos menores caminhos da rede U
Sairport500:  1.883667007854659
```

4.Calcule o coeficiente de assortatividade da rede Advogato. Considere apenas o maior componente.

In [10]:
```
G_adv = clean_graph(nx.read_edgelist('ex_2/data/advogato.txt'), r
emove_edges = False)
```

In [11]:
```
r = nx.degree_assortativity_coefficient(G_adv)
print('Coeficiente de assortatividade da rede Advogato = ', '%3.4
f'%r)
```

```
Coeficiente de assortatividade da rede Advogato =  -0.0846
```

5.Calcule o coeficiente de correlação de Pearson entre o grau médio dos vizinhos e o grau de cada vértice para a rede “word_adjacencies”. Isto é, entre k e knn(k). Use apenas o maior componente.Considere o exemplo da aula.

In [12]:
```
G_wa = clean_graph(nx.read_edgelist('/home/aime/Documents/redes_c
omplexas/ex_2/data/word_adjacencies.txt', nodetype = int))
```

In [13]:
```
wa_k, wa_Pk = knnk(G_wa)
pearson_kssknnk = np.corrcoef(wa_k, wa_Pk)[0, 1]

print(f'Pearson correlation: {pearson_kssknnk}')
```

```
Pearson correlation: -0.6753041480047248
```