Questionário: Barabási-Albert

Nome (Nro. USP): Aimê G. da Nobrega (11882429)
Disciplina: Redes Complexas (SME0130)
Docente: Francisco A. Rodrigues
USP São Carlos - ICMC

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

In [9]:
```
def degree_distribution(GER):
    vk = dict(GER.degree())
    vk = list(vk.values()) # we get only the degree values
    maxk = np.max(vk)
    mink = np.min(min)
    kvalues = arange(0, maxk+1) # possible values of k
    Pk = np.zeros(maxk +1) #P(k)
    for k in vk:
        Pk[k] = Pk[k] +1
    Pk = Pk/sum(Pk) # sum of elements of P(k)
    return kvalues, Pk

def momment_of_degree_distribution(G, m):
    k, Pk = degree_distribution(G)
    M = sum((k**m)*Pk)
    return M

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

1) Calcule a média do coeficiente aglomeração e segundo momento do grau para uma rede BA com grau médio igual a 8 e N=1000.

In [30]:
```
N = 1000
av_degree = 8
m = int(av_degree/2)
GBA = nx.barabasi_albert_graph(N, m)

k2 = momment_of_degree_distribution(GBA, 2)
# print(k2)
avc = nx.average_clustering(GBA)
# print(avc)

print(f'Média do coeficiente aglomeração: {avc}')
print(f'Segundo momento do grau: {k2}')
```

```
Média do coeficiente aglomeração: 0.0362242944276952
Segundo momento do grau: 138.144
```

2) Considere uma rede aleatória (Erdos-Renyi) e uma rede BA com N=1000 vértices e grau médio 10. Qual o valor da entropia de Shannon da distribuição do grau para essas redes?

In [8]:
```
N = 1000
av_degree = 10
p = av_degree/(N-1)
m = int(av_degree/2)
GBA = nx.barabasi_albert_graph(N, m)
GER = nx.gnp_random_graph(N, p, seed=42, directed = False)
```

In [29]:
```
shannon_gba = shannon_entropy(GBA)
shannon_ger = shannon_entropy(GER)
# print(shannon_gba)
# print(shannon_ger)

print(f'Shannon entropy for BA Network: {shannon_gba}')
print(f'Shannon entropy for Erdos Renyi Network: {shannon_ger}')
```

```
Shannon entropy for BA Network: 3.5752912442894815
Shannon entropy for Erdos Renyi Network: 3.6379483175410465
```

3) Considere o modelo de Barabási-Albert com N=1000 e grau médio igual a 10. Calcule o coeficiente de correlação de Pearson (rho) entre o grau e a medida eigenvector centrality. O que esse valor indica?

In [31]:
```
N = 1000
av_degree = 10
m = int(av_degree/2)
GBA = nx.barabasi_albert_graph(N, m)
```

In [32]:
```
EC = dict(nx.eigenvector_centrality(GBA, max_iter = 1000))
# print('Eigenvetor centrality: ', EC)
EC = list(EC.values())
# print(EC)
av_EC = np.mean(EC)
# print('Average eigenvector centrality', av_EC)

d = dict(GBA.degree())
dv = list(d.values())


pearson=np.corrcoef(EC , dv)[0,1]
print(f'Pearson correlation entre grau e medida eigenvector centr
ality: {pearson}')
```

```
Pearson correlation entre grau e medida eigenvector centrality:
0.9409079966700954
```

4) Calcule a correlação entre a medida betweeness centrality e o grau para uma rede BA. Considere N=500 e grau médio 10.

In [33]:
```
N = 500
av_degree = 10
m = int(av_degree/2)
GBA = nx.barabasi_albert_graph(N, m)
```

In [34]:
```
B = dict(nx.betweenness_centrality(GBA))
Bv = list(B.values())
d = dict(GBA.degree())
dv = list(d.values())

pearson=np.corrcoef(Bv , dv)[0,1]
print(f'Pearson correlation entre betweeness centrality e grau:
{pearson}')
```

```
Pearson correlation entre betweeness centrality e grau: 0.9538689
897365547
```

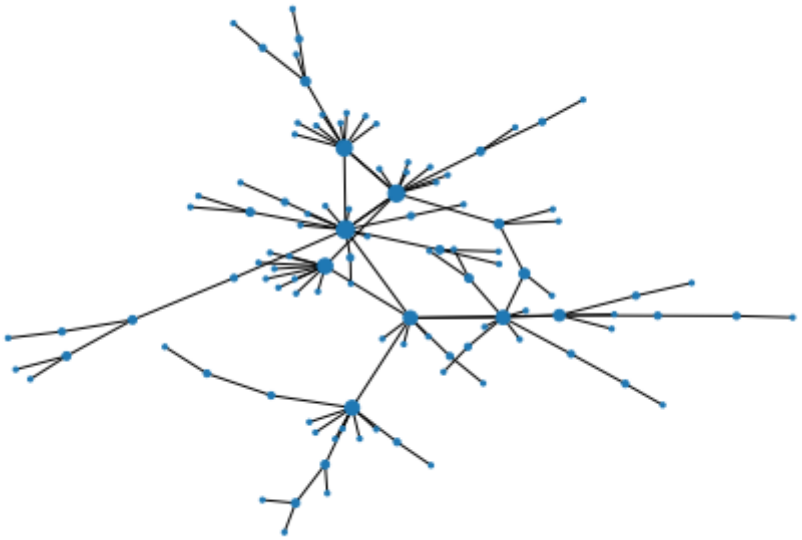
5) Calcule o segundo momento do grau para o modelo de configuração com a=3 (coeficiente da lei de potência (Zipf)). Considere N=500 e o valor mais próximo, pois os valores podem variar de uma simulação para outra.

In [55]:
```
N = 500
a = 3
seq = np.random.zipf(a, N) #Zipf distribution
#seq = np.random.poisson(10, N) #Poisson distribution
#print(seq)

if(sum(seq)%2 != 0): # the sum of stubs have to be even
    pos = randint(0, len(seq))
    seq[pos] = seq[pos]+ 1
#print(sum(seq))

G=nx.configuration_model(seq)
#get the largest component

Gcc = sorted(nx.connected_components(G), key=len, reverse=True)
G = G.subgraph(Gcc[0])
d = dict(G.degree())
nx.draw(G, nodelist=d.keys(), node_size=[v * 5 for v in d.values
()])
plt.show()
```



In [56]:
```
k2 = momment_of_degree_distribution(G, 2)
# print(k2)
print(f'Segundo momento do grau: {k2}')
```

```
Segundo momento do grau: 11.284403669724771
```