Questionário

Modelos de grafos aleatórios: Erdos-Renyi e Small-Worlds de Watts e Strogatz

OBS: Sempre aproxime para uma casa decimal e considere o valor mais próxima da sua solução, pois pode haver pequenas variações devido à natureza estocástica dos modelos.

1 - Gere um grafo aleatório com N=1000 e p = 0.1. Qual o valor do grau médio, segundo momento do grau e coeficiente de aglomeração médio (average clustering coefficient)?

In [11]:
```
N = 1000

p = 0.1
av_degree = 0.1*(N - 1)

GER = nx.gnp_random_graph(N, p, seed = 42, directed = False)
```

In [12]:
```
GER_N = len(GER.nodes)
GER_M = len(GER.edges)

GER_mm1 =  momment_of_degree_distribution(GER, 1)
GER_mm2 =  momment_of_degree_distribution(GER, 2)

GER_cc = nx.average_clustering(GER)
```

In [13]:
```
print(f'Números de nós: {GER_N}')
print(f'Número de conexões: {GER_M}')

print('\n')

print(f'Grau médio: {GER_mm1:.1f}')
print(f'Segundo momento: {GER_mm2:.1f}')

print('\n')

print(f'Coeficiente de aglomeração médio: {GER_cc:.1f}')
```

```
Números de nós: 1000
Número de conexões: 49929


Grau médio: 99.9
Segundo momento: 10062.9


Coeficiente de aglomeração médio: 0.1
```

2 - Gere um small-world com N=1000, grau médio igual 10 e p = 0.1. Qual o valor do grau médio, segundo momento do grau e coeficiente de aglormeração médio (average clustering coefficient)?

In [28]:
```
N = 1000

k = 10
p = 0.1

GWS = nx.watts_strogatz_graph(N, k, p, seed = 42)
```

In [29]:
```
GWS_N = len(GWS.nodes)
GWS_M = len(GWS.edges)

GWS_mm1 =  momment_of_degree_distribution(GWS, 1)
GWS_mm2 =  momment_of_degree_distribution(GWS, 2)

GWS_cc = nx.average_clustering(GWS)
```

In [30]:
```
print(f'Números de nós: {GWS_N}')
print(f'Número de conexões: {GWS_M}')

print('\n')

print(f'Grau médio: {GWS_mm1:.1f}')
print(f'Segundo momento: {GWS_mm2:.1f}')

print('\n')

print(f'Coeficiente de aglomeração médio: {GWS_cc:.1f}')
```

```
Números de nós: 1000
Número de conexões: 5000


Grau médio: 10.0
Segundo momento: 100.9


Coeficiente de aglomeração médio: 0.5
```

3 - Considere uma rede aleatória (Erdos-Renyi) com N=1000 vértices. Qual o valor da entropia de Shannon do grau para $\langle k \rangle$ = 5, $\langle k \rangle$ = 10,$\langle k \rangle$ = 50.

In [31]:
```
N = 1000

av_degrees = [5, 10, 50]

ger_k = []

for k in av_degrees:

    p = k/(N - 1)

    ger_k.append(nx.gnp_random_graph(N, p, seed = 42, directed = False))
```

In [32]:
```
ger_entropy = [shannon_entropy(G) for G in ger_k]

_ = [print(f'Entropia de Shannon para K = {av_degrees[i]}: {ger_entropy[i]:.1f}') for i in np.arange(3)]
```

```
Entropia de Shannon para K = 5: 3.2
Entropia de Shannon para K = 10: 3.6
Entropia de Shannon para K = 50: 4.7
```

4 - Para o modelo small-world, calcule o valor da menor distância média (average shortest path) para p=0; p=0.01; p=0.05 e p=0.1. Considere grau médio igual a 10.

In [95]:
```
N = 100

k = 4
ps = [0, 0.01, 0.05, 0.1]
```

In [99]:
```
sam_av_shortest_path_k = []

for i in range(30):
    gws_k = [nx.watts_strogatz_graph(N, k, p, seed = i) for p in ps]
    av_shortest_path_k = [nx.average_shortest_path_length(G) for G in gws_k]

    sam_av_shortest_path_k.append(av_shortest_path_k)

sam_av_shortest_path_k = np.array(sam_av_shortest_path_k)
```

In [100...
```
av_shortest_path_k = [np.mean(sam_av_shortest_path_k[:,i]) for i in np.arange(4)]
```

In [101...
```
_ = [print(f'O menor caminho médio para P = {ps[i]}: {av_shortest_path_k[i]:.1f}') for i in np.arange(4)]
```

```
O menor caminho médio para P = 0: 12.9
O menor caminho médio para P = 0.01: 10.2
O menor caminho médio para P = 0.05: 6.2
O menor caminho médio para P = 0.1: 4.9
```

5 - Considere o modelo de Erdos-Renyi. Gere redes com grau médio igual a 5, 10 e 50 e N=1000. Qual o valor da assortatividade?

In [102...
```
N = 1000

av_degrees = [5, 10, 50]

ger_k = []

for k in av_degrees:

    p = k/(N - 1)

    ger_k.append(nx.gnp_random_graph(N, p, seed = 42, directed = False))
```

In [105...
```
degree_assortativity_k = [nx.degree_assortativity_coefficient(G) for G in ger_k]
```

In [106...
```
_ = [print(f'O valor da assortatividade para k = {av_degrees[i]}: {degree_assortativity_k[i]:.5f}') for i in np.arange(3)]
```

```
O valor da assortatividade para k = 5: -0.01343
O valor da assortatividade para k = 10: -0.01305
O valor da assortatividade para k = 50: -0.00309
```

6 - Considere o modelo small-world. Gere redes com grau médio 10 e N=1000. Qual o valor da assortatividade para p=0.01; 0.05 e 0.2?

In [107...
```
N = 1000

k = 10
ps = [0.01, 0.05, 0.2]

gws_k = [clean_graph(nx.watts_strogatz_graph(N, k, p, seed = 42)) for p in ps]
```

In [108...
```
degree_assortativity_k = [nx.degree_assortativity_coefficient(G) for G in gws_k]
```

In [112...
```
_ = [print(f'O valor da assortatividade para P = {ps[i]}: {degree_assortativity_k[i]:.1f}') for i in np.arange(3)]
```

```
O valor da assortatividade para P = 0.01: 0.0
O valor da assortatividade para P = 0.05: -0.0
O valor da assortatividade para P = 0.2: 0.0
```

Bibliotecas

In [1]:
```
import numpy as np
from numpy import linalg as al
```

In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
import networkx as nx
```

Funções

In [4]:
```
def shannon_entropy(G):

    k, Pk = degree_distribution(G)

    H = 0

    for p in Pk:

        if p > 0:

            H -= p*np.log2(p)

    return H
```

In [5]:
```
def momment_of_degree_distribution(G, m):

    k, Pk = degree_distribution(G)

    M = sum((k**m)*Pk)

    return M
```

In [6]:
```
def degree_distribution(G):

    vk = np.array(list(dict(G.degree()).values()))

    maxk = vk.max()
    mink = vk.min()

    kvalues = np.arange(maxk + 1)
    Pk = np.zeros(maxk + 1)

    for k in vk:

        Pk[k] += 1

    Pk = Pk/Pk.sum()

    return kvalues, Pk
```

In [7]:
```
def clean_graph(G, remove_edges = True):

    G = G.to_undirected()
    nodes_original = G.nodes

    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
```

Constantes

In [8]:
```
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']
```

Configurações

In [9]:
```
%matplotlib inline
```

In [10]:
```
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=14)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=13)    # fontsize of the tick labels
plt.rc('ytick', labelsize=13)    # fontsize of the tick labels
plt.rc('legend', fontsize=13)    # legend fontsize
plt.rc('font', size=13)          # controls default text sizes
plt.rc('figure', figsize = (8,8)) # Set the figure size
```

In [ ]:
```

```

In [ ]:
```

```