Practica_2

March 18, 2020

1 Práctica 2

Álvaro Huertas García

Sara Dorado Alfaro

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

1.1 Estudiar las propiedades de un grafo aleatorio

Usando la función gnp_random_graph(n,p) de NetworkX Implementad una rutina Python L_C_Aleatorio que devuelva valor promedio de los coeficientes C (índice de clusterización promedio) y L (camino característico) de un conjunto de n_graph grafos aleatorios de n nodos y probabilidad p.

[2]:
```
def L_C_Aleatorio(n_graph,n,p):
    mu_C = 0
    mu_L = 0
    for i in range(n_graph):
        G = nx.gnp_random_graph(n, p)
        C = nx.average_clustering(G)
        if nx.is_connected(G):
            L = nx.average_shortest_path_length(G)
        else:
            L = n-1
        mu_C += C/n_graph
        mu_L += L/n_graph

    return mu_L, mu_C
```

[3]:
```
n = 400
n_graph = 20
ar_p = np.array([0,0.0001,0.0003,0.0006, 0.001,0.003,0.006,0.01,0.03,0.06,0.1,0.
↪3,0.6,1.])
```

```python
ar_C = np.zeros(len(ar_p))
ar_L = np.zeros(len(ar_p))

for i, p in enumerate(ar_p):
    print('Probabilidad: ', p)
    ar_L[i], ar_C[i] = L_C_Aleatorio(n_graph,n,p)
```

```
Probabilidad:  0.0
Probabilidad:  0.0001
Probabilidad:  0.0003
Probabilidad:  0.0006
Probabilidad:  0.001
Probabilidad:  0.003
Probabilidad:  0.006
Probabilidad:  0.01
Probabilidad:  0.03
Probabilidad:  0.06
Probabilidad:  0.1
Probabilidad:  0.3
Probabilidad:  0.6
Probabilidad:  1.0
```

[4]:
```python
ar_p = np.array([0,0.0001,0.0003,0.0006, 0.001,0.003,0.006,0.01,0.03,0.06,0.1,0.
↪3,0.6,1.])
```

[5]:
```python
print('Caminos caracteristicos medios: \n', ar_L)
print('\nIndices de clusterizacion medios: \n', ar_C)
```

```
Caminos caracteristicos medios:
 [399.          399.          399.          399.          399.
 399.          399.          399.            2.68336028   2.16219236
   1.91661717   1.70031704   1.39997619   1.        ]

Indices de clusterizacion medios:
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 2.86805556e-04 4.36299603e-03 8.91317641e-03
 2.96267792e-02 6.01628932e-02 1.00270017e-01 2.99652897e-01
 6.00056878e-01 1.00000000e+00]
```

Valor crítico de p, demostración Por un lado, tenemos que $< k >= \frac{2|E|}{|V|}$ y \$ |E| = p|V|(|V|-1)$\overline{2;.\$Despejandodelasdosecuaciones,esfcilverque:}$

$$< k > \simeq p|V| \; .$$

Para un grafo aleatorio, Erdos y Renyi demostraron que la estructura de clusters de un grafo cambia abruptamente cuando $< k >$ se acerca a 1. La probabilidad crítica será por tanto:

$$k \simeq 1 \rightarrow p_c|V| \simeq 1 \rightarrow p_c \simeq \frac{1}{|V|} \; .$$

Para nuestros grafos, que tienen $|V| = 400$ nodos, deberíamos obtener una probabilidad crítica $p_c = \frac{1}{400} = 0.0025$.

En ese valor de p_c se producirá un cambio en las características estructurales de la red, observándose como el índice de clusterización aumenta, dado que la probabilidad de que se establezcan aristas en los nodos crece. En el método de cálculo del camino mínimo que se ha empleado, se ha considrado qe $L = N - 1$ cuando el grafo no es conexo. El hecho de que a partir de p_c cambie el índice de clusterización, pero no el camino mínimo, puede producirse dado que el índice de clusterización es una medida local y, por tanto, no necesita que el grafos sea conexo para su cálculo. Sin embargo, el camino mínimo requiere que sea conexo para poder calcularse, no siendo conexo hasta que $<k> > ln(N)$.

Por tanto, podemos conocer la probabilidad p_l en la que $<k> = ln(N)$, es decir, cuando el grafo pasa a ser conexo y poder calcularse el camino mínimo:

$$p_l = \frac{< k >}{|V|} \rightarrow p_l = \frac{ln(|V|)}{|V|} = \frac{ln(400)}{400} = 0.015$$

Por lo tanto, será a partir de $p_l = 0.015$ cuando el camino característico cambie.

[6]:
```
pc = 1/n
print('Probabilidad crítica (1/n): ', pc)
```

```
Probabilidad crítica (1/n):  0.0025
```

[7]:
```
import math
pc_L = math.log(400)/400
print("La probabilidad en la que el grafo aleatorio pasa a ser conexo es:", pc_L)
```

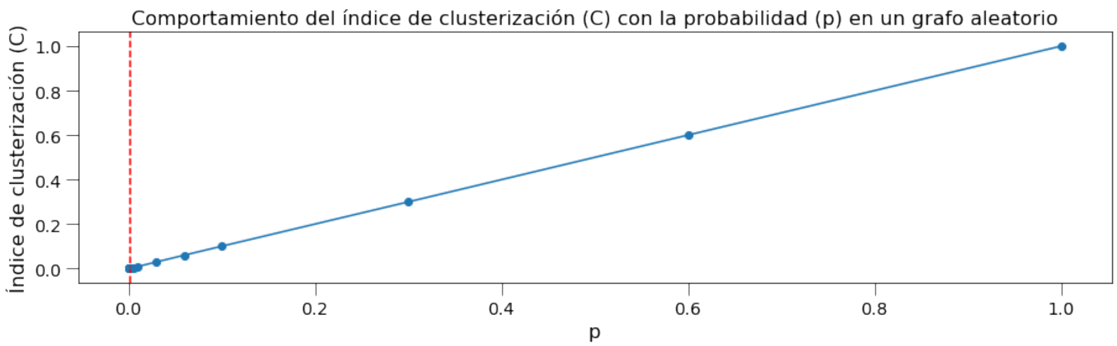
```
La probabilidad en la que el grafo aleatorio pasa a ser conexo es:
0.014978661367769954
```

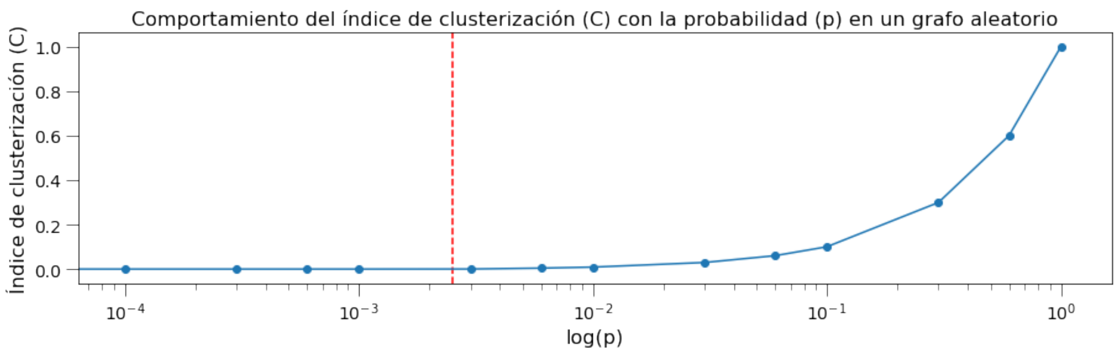
A continuación, se muestra el comportamiento de C y L en función de la probabilidad de conexión p. Se marcan las probabilidades críticas comentadas anteriormente con líneas punteadas verticales: p_c en rojo, p_l en verde.

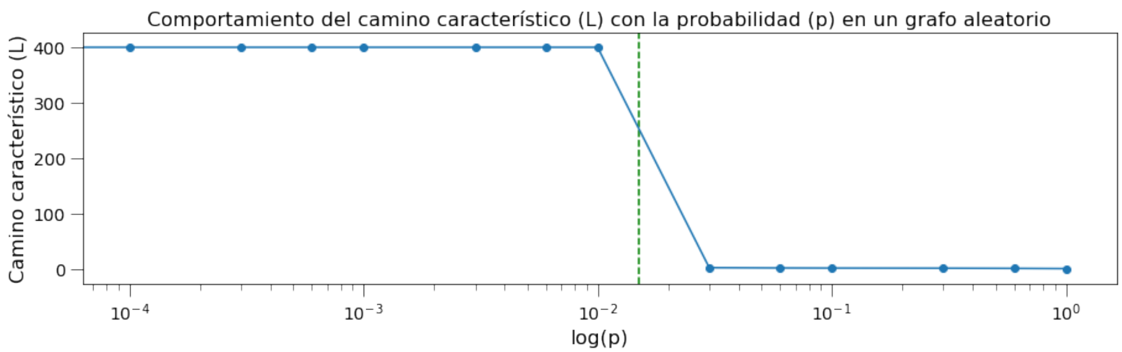
[8]:
```
# Tamaño de la figura
plt.figure(figsize = (15, 15))

# Primer plot
plt.subplot(311)
plt.title("Comportamiento del índice de clusterización (C) con la probabilidad␣
 ↪(p) en un grafo aleatorio",
          fontsize = 16)
# Etiquetas de los ejes
plt.ylabel("Índice de clusterización (C)", fontsize = 16)
```

```
plt.xlabel("p", fontsize = 16)
# Representación de la pribabilidad critica
plt.axvline(x=pc, color='r', ls='--')
# parametros de los ticks
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.tick_params(axis="both", which = "minor", direction="out", length = 5)
plt.tick_params(axis="both", which = "major", direction="out", length = 10)

plt.plot(ar_p, ar_C)
plt.scatter(ar_p, ar_C)

# Segundo plot
plt.subplot(312)
plt.title("Comportamiento del índice de clusterización (C) con la probabilidad␣
 ↪(p) en un grafo aleatorio",
          fontsize = 16)
# Escala de los ejes
plt.xscale("log", nonposx = 'clip')
# Etiquetas de los ejes
plt.ylabel("Índice de clusterización (C)", fontsize = 16)
plt.xlabel("log(p)", fontsize = 16)
# Representación de la pribabilidad critica
plt.axvline(x=pc, color='r', ls='--')
# parametros de los ticks
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.tick_params(axis="both", which = "minor", direction="out", length = 5)
plt.tick_params(axis="both", which = "major", direction="out", length = 10)

plt.plot(ar_p, ar_C)
plt.scatter(ar_p, ar_C)

# tercer plot
plt.subplot(313)
plt.title("Comportamiento del camino característico (L) con la probabilidad (p)␣
 ↪en un grafo aleatorio",
          fontsize = 16)
# Escala de los ejes
plt.xscale("log", nonposx = 'clip')
# Etiquetas de los ejes
plt.ylabel("Camino característico (L)", fontsize = 16)
plt.xlabel("log(p)", fontsize = 16)
# Representación de la pribabilidad critica
plt.axvline(x=pc_L, color='green', ls='--')
# parametros de los ticks
plt.xticks(fontsize = 14)
```

```python
plt.yticks(fontsize = 14)
plt.tick_params(axis="both", which = "minor", direction="out", length = 5)
plt.tick_params(axis="both", which = "major", direction="out", length = 10)

plt.plot(ar_p, ar_L)
# Distancia entre los plots
plt.subplots_adjust(hspace=0.5)
plt.scatter(ar_p, ar_L)


plt.show()
```







1.2 ¿Para qué probabilidades p se producen cambios fase en el comportamiento los parámetros L, C?

En la primera gráfica comprobamos que el índice de clusterización es 0 antes de superarse la p_c=0.0025 (línea roja discontinua), a partir de la cual la estructura de clúster del grafo aleatorio cambia, creciendo de forma lineal (notar que en esta gráfica no usamos la escala logarítmica). Esto corrobora el hecho de que el índice de clusterización de un grafo aleatorio es:

$$C_{aleatorio} = p = \frac{<k>}{N}$$

Por su parte, en el segundo gráfico se muestra el comportamiento del camino característico en escala logarítmica. El camino característico se mantiene en (N-1) para los primeros valores de p. Esto es debido a que todavía el grafo aleatorio no es un grafo conexo y el camino característico es, por tanto, infinito (nosotros consideramos N-1). No obstante, cuando empieza a ser conexo ($p > p_l$ = 0.015, línea verde discontinua) el camino caracteŕistico cae según añadimos nuevas aristas en el grafo. Al final de esta segunda gráfica podemos ver como el camino característico tiende a 1 a medida que aumenta la probabilidad de generarse ramas, es decir, cuando el grafo es denso. Para $p = 1$, todos los nodos están conectados, por lo que sólo necesitamos dar un paso para llegar a cualquier nodo, sea cual sea el origen.

A continuación, vamos a estudiar como cambiaría L en función de la probabilidad de conexión p de una forma distinta. Ahora consideraremos el valor L sobre la componente conexa más grande mientras el grafo no sea conexo. Es decir, antes de $p = p_l$ = 0.015. Para ello vamos a generar una nueva función, L_C_Aleatorio_componente(n_graph,n,p), que tendrá en cuenta esta modificación.

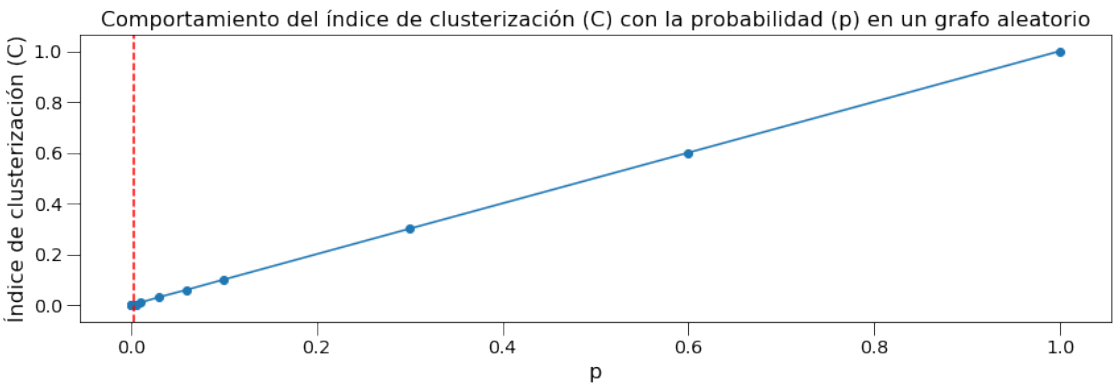
[9]:
```
# Calcular camino característico para componente más grande
def L_C_Aleatorio_componente(n_graph,n,p):
    '''
    Funcion similar a L_C_Aleatorio
    Se calcula el valor medio de L y C para n_graph
    aleatorios. Cuando el grafo NO es conexo se toman
    los valores de L y C para la mayor componente conexa
    '''
    mu_C = np.zeros(n_graph)
    mu_L = np.zeros(n_graph)
    for i in range(n_graph):
        G = nx.gnp_random_graph(n, p)
        C = nx.average_clustering(G)
        if nx.is_connected(G):
            L = nx.average_shortest_path_length(G)
        else:
            Gc = max((G.subgraph(c) for c in nx.connected_components(G)),␣
 ↪key=len)
            L = nx.average_shortest_path_length(Gc)

        mu_C[i] = C
        mu_L[i] = L
```

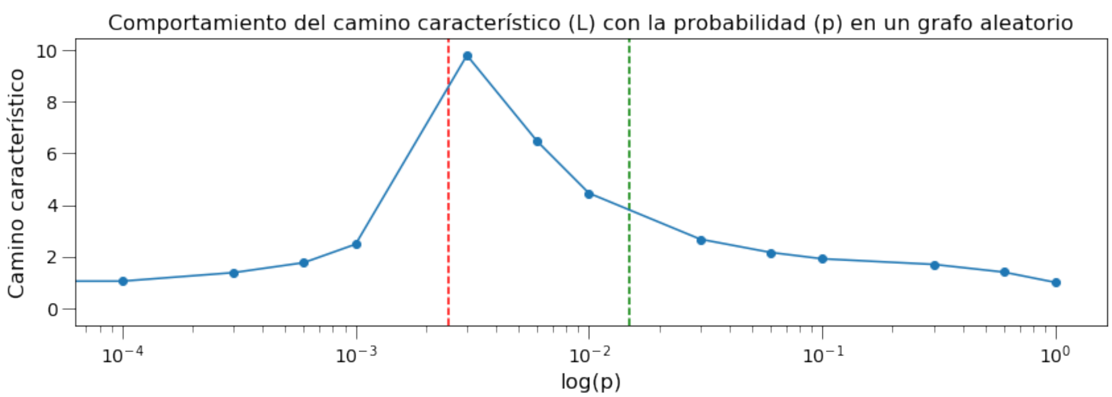
```
    media_L = np.mean(mu_L)
    media_C = np.mean(mu_C)
    des_L = np.std(mu_L)
    des_C = np.std(mu_C)


    return media_L, media_C, des_L, des_C

n = 400
n_graph = 20
ar_p = np.array([0,0.0001,0.0003,0.0006, 0.001,0.003,0.006,0.01,0.03,0.06,0.1, 0.
 ↪3,0.6,1.])
ar_C2 = np.zeros(len(ar_p))
ar_L2 = np.zeros(len(ar_p))

for i, p in enumerate(ar_p):
    print('Probabilidad: ', p)
    ar_L2[i], ar_C2[i], des_L, des_C = L_C_Aleatorio_componente(n_graph,n,p)
    #print("Camino característico", ar_L[i],"con desv", des_L,
    #       "\nÍndice de clusterización", ar_C[i], "con desv", des_C)
    #print()
```

```
Probabilidad:  0.0
Probabilidad:  0.0001
Probabilidad:  0.0003
Probabilidad:  0.0006
Probabilidad:  0.001
Probabilidad:  0.003
Probabilidad:  0.006
Probabilidad:  0.01
Probabilidad:  0.03
Probabilidad:  0.06
Probabilidad:  0.1
Probabilidad:  0.3
Probabilidad:  0.6
Probabilidad:  1.0
```

[10]:
```
# Generamos la figura
plt.figure(figsize = (14, 10))

# Creamos el primer grafico
plt.subplot(211)
plt.title("Comportamiento del índice de clusterización (C) con la probabilidad␣
 ↪(p) en un grafo aleatorio",
          fontsize = 16)
# Etiquetas de los ejes
plt.ylabel("Índice de clusterización (C)", fontsize = 16)
```

```python
plt.xlabel("p", fontsize = 16)
# Probabilidad critica de cambio de la estructura de cluster
plt.axvline(x=0.0025, color='r', ls='--')
# parametros de los ticks
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.tick_params(axis="both", which = "minor", direction="out", length = 5)
plt.tick_params(axis="both", which = "major", direction="out", length = 10)

plt.plot(ar_p, ar_C2)
plt.scatter(ar_p, ar_C2)


# Segundo grafico
plt.subplot(212)
plt.title("Comportamiento del camino característico (L) con la probabilidad (p)␣
↪en un grafo aleatorio",
          fontsize = 16)
# Escala de los ejes
plt.xscale("log", nonposx = 'clip')
# Etiquetas de los ejes
plt.ylabel("Camino característico", fontsize = 16)
plt.xlabel("log(p)", fontsize = 16)
# Probabilidad de cambio de cluster y caida de L
plt.axvline(x=0.0025, color='r', ls='--')
plt.axvline(x=pc_L, color='green', ls='--')
# parametros de los ticks
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.tick_params(axis="both", which = "minor", direction="out", length = 5)
plt.tick_params(axis="both", which = "major", direction="out", length = 10)

plt.plot(ar_p, ar_L2)
# Distancia entre los plots
plt.subplots_adjust(hspace=0.5)
plt.scatter(ar_p, ar_L2)
plt.show()
```





En el segundo gráfico, donde se muestra el comportamiento del camino mínimo L calculado para la componente de mayor tamaño en función de la probabilidad de aparición de ramas p, observamos un comportamiento creciente al inicio, alcanzándose un máximo en 9.57, tras el cuál comienza a decrecer tendiendo a 1 (mínimo posible). En el gráfico también se han representado con líneas discontinuas la probabilidad crítica en la que el comportamiento del grafo aleatorio cambia ($p_c = 1/N = 0.0025$) (línea roja discontinua) y la probabilidad en la que el grafo aleatorio pasa a ser conexo (p_l= 0.015)(línea verde discontinua).

Al comienzo de la gráfica las ramas que se generan conectan un número reducido de nodos permitiéndo que el camino característico sea bajo. A medida que aumenta la probabilidad, se añaden más ramas al grafo produciendose un aumento en el número de nodos conectados alargando así la distancia entre ellos. Nótese que al principio, cuando p es pequeña, la mayor componente conexa será, de hecho, pequeña.

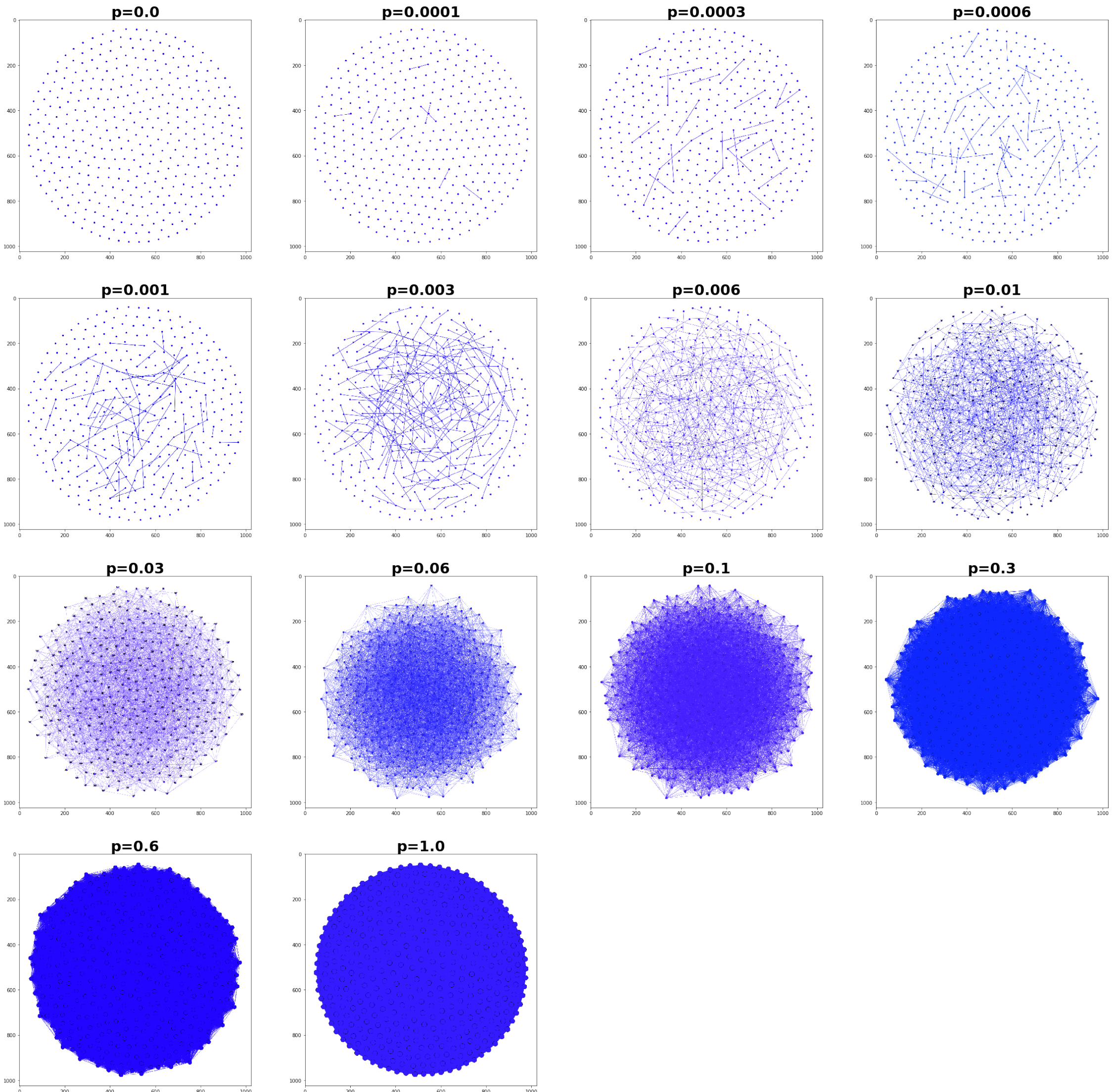
Entre ambas líneas se encontraría la isla gigante, que se transforma en un grafo conexo a medida que aumenta el número de ramas. En esta región de transición, alcanzaríamos el máximo del camino característico debido a que la probabilidad es suficientemente alta como para que se formen ramas que conecten más nodos de la red, pero no lo suficiente como para que esos nodos estén conectados de forma que el camino entre ellos sea corto.

Tras el máximo, se produce un descenso de camino característico debido a que hay que considerar que el número de nodos es constante, por lo que habrá un punto en el que las nuevas ramas que se generen no conectarán nuevos nodos previamente desconectados, sino que se encargarán de aumentar la conexión entre nodos. Esto último se traduce en un aumento de la conectividad y una disminución del camino característico tendiendo a 1 cuando el grafo sea denso.

Como apoyo a la explicación, a continuación se muestra la topología de la red aleatoria para cada p estudiada:

[11]:
```
import glob
import matplotlib.image as mpimg
%matplotlib inline
```

[12]:
```
images = []
paths = []

# Seleccionamos los paths a las imagenes
for img_path in glob.glob('images/aleatorio//*.png'):
    paths.append(img_path)

# Ordenamos los archivos
paths.sort()

# Cargamos las imagenes
for image in paths:
    images.append(mpimg.imread(image))


# ax enables access to manipulate each of subplots
ax = []
ar_p = np.array([0,0.0001,0.0003,0.0006, 0.001,0.003,0.006,0.01,0.03,0.06,0.1, 0.
 ↪3,0.6,1.])
fig = plt.figure(figsize=(40,40))
rows = 4
columns = 4
for i, image in enumerate(images):
    ax.append( fig.add_subplot(rows , columns, i+1) )
    ax[-1].set_title("p="+str(ar_p[i]), fontsize = 30, fontweight = "bold")
    plt.imshow(image)
```



En primer lugar, observamos claramente que los grafos aleatorias son más densos a medida que aumenta la probabilidad de generar ramas p. Igualmente, observamos cómo el grafo pasa a ser conexo entre p=0.01 y p=0.03, de acuerdo con los valores calculado anteriormente (p_l=0.015, línea verde discontinua). También es interesante comprobar cuando $p = 0.3$ aparece la componente de mayor tamaño donde se alcanzaría el mayor camino caracteristico, grafo a partir del cuál disminuye el camino característico dado que la red pasa a estar más conectada en p=0.06 y en los sucesivos grafos.

2 Apartado 2

Usando la función NetworkX watts_strogatz_graph(n, k, p) implementad una rutina Python L_C_small_world(n_graph, n,k,p) que devuelva valor promedio de los coeficientes C (índice de clusterización promedio) y L (camino característico) para un conjunto de n_graph grafos de tipo

Small World o mundo pequeño.

Este algoritmo permite crear redes de mundo pequeño a partir de un grafo regular de N nodos donde cada nodo se encuentra unido con $2m = k$ vecinos (siendo m el número de vecinos a izquierda o derecha del nodo), modificando la probabilidad p con la que un nodo se una a otro nodo cualquiera, generándose así atajos dentro del grafo. De este modo, cuando p=0, el grafo constituirá un anillo regular en el que todos los nodos tienen el mismo grado, mientras que si p=1 el grafo se comportará como un grafo aleatorio, es decir, la distribución de las aristas entre los distintos nodos será aleatoria.

Las redes regulares se caracterizan por un coeficiente de agrupamiento C muy alto (cercano a 1) y una longitud de camino mínimo promedio L también grande, sobre todo para k<<N. Siendo de hecho $L \sim N/k$.

Las redes de mundo pequeño comparten características tanto con los grafos regulares, como con los grafos aleatorios (donde C y L es bajo). En consecuencia, las redes de mundo pequeño se encuentran en la región de transición entre ambos extremos, presentando un valor de índice de clusterización C alto y un camino mínimo L bajo.

A continuación se muestra un ejemplo de un grafo regular con algunos de sus parámetros más significativos.

[13]:
```
G_wst = nx.watts_strogatz_graph(400, 8, 0)
print(nx.info(G_wst))
L = nx.average_shortest_path_length(G_wst)
print("Camino mínimo medio", L)
C = nx.average_clustering(G_wst)
print("Índice de clusterización", C)
```

```
Name:
Type: Graph
Number of nodes: 400
Number of edges: 1600
Average degree:   8.0000
Camino mínimo medio 25.43859649122807
Índice de clusterización 0.6428571428571402
```

A modo de comprobación, podemos calcular el valor teórico de C, L, número de ramas ($|E|$) y el número de atajos generados para un grafo regular con p=0, mediante las siguientes fórmulas:

$$C(p) = \frac{3(k/2-1)}{2 \times (k-1)} \times (1-p)^3 \rightarrow C(0) = \frac{3(8/2-1)}{2 \times (8-1)} = 0.643$$

$$L(N,0) = \frac{N(N+k-2)}{2k(N-1)} \sim \frac{N}{2k} \rightarrow L(400,0) \sim \frac{400}{2 \times 8} \sim 25$$

$$|E| = \frac{N \times k}{2} = \frac{400 \times 8}{2} = 1600$$

$$\text{Número de atajos}(p=0) = p \times N \times k = 0$$

Donde:

- N es el número de nodos.
- k es el grado medio de los nodos (igual al grado de cada nodo en un grafo regular).
- $|E|$ es el número de aristas.
- p es la probabilidad de reordenar una arista de forma aleatoria.

A continuación, presentamos la función pedida en el ejercicio.

[14]:
```python
def L_C_Small_World(n_graph,n,k,p):
    mu_C = 0
    mu_L = 0
    for i in range(n_graph):
        G = nx.watts_strogatz_graph(n, k, p)
        C = nx.average_clustering(G)
        if nx.is_connected(G):
            L = nx.average_shortest_path_length(G)
        else:
            L = n-1
        mu_C += C/n_graph
        mu_L += L/n_graph

    return mu_L, mu_C
```

[15]:
```python
n = 400
n_graph = 20
k = 8
ar_p = np.array([0,0.0001,0.0003,0.0006, 0.001,0.003,0.006,0.01,0.03,0.06,0.1,0.
 ↪3,0.6,1.])

ar_C = np.zeros(len(ar_p))
ar_L = np.zeros(len(ar_p))

for i, p in enumerate(ar_p):
    print('Probabilidad: ', p)
    ar_L[i], ar_C[i] = L_C_Small_World(n_graph, n, k, p)
```

```
Probabilidad:  0.0
Probabilidad:  0.0001
Probabilidad:  0.0003
Probabilidad:  0.0006
Probabilidad:  0.001
Probabilidad:  0.003
Probabilidad:  0.006
Probabilidad:  0.01
Probabilidad:  0.03
Probabilidad:  0.06
Probabilidad:  0.1
Probabilidad:  0.3
```

```
Probabilidad:  0.6
Probabilidad:  1.0
```

2.1 ¿Para qué probabilidades p se producen cambios fase en el comportamiento los parámetros L y C?

Para un grafo de mundo pequeño se comprueba que el camino característico disminuye bruscamente (cambio de régimen) para el valor teórico:

$$p = \frac{1}{k|V|} \,.$$

En nuestro caso, para $k = 8$ y $|V| = 400$, tendríamos que la probabilidad crítica

$$p_l = \frac{1}{8 \times 400} = 0.0003 \,.$$

La probabilidad crítica en la que se produce la transición del índice de clusterización puede calcularse de forma aproximada mediante el cálculo del punto de intersección entre las dos regiones con distinta pendientes que muestra la gráfica (ver abajo). De este modo, tenemos una región inicial donde la pendiente es nula (m_1=0), y una segunda región donde la pendiente es negativa (m_2 < 0). Asignando el punto P_1 a la primera región con las coordenadas (0, 0.64); y los puntos P_2, P_3 a la segunda región con coordenadas (0.3, 0.2) y (0.6, 0.03), respectivamente, se puede calcular el punto de corte de las dos regiones resolviendo el siguiente sistema:

$$a)y = 0.64$$
$$b)y = -1.76x + 0.73$$

Resolviendo, se obtiene que:

$$x = 0.05 \to p_c = 0.05$$

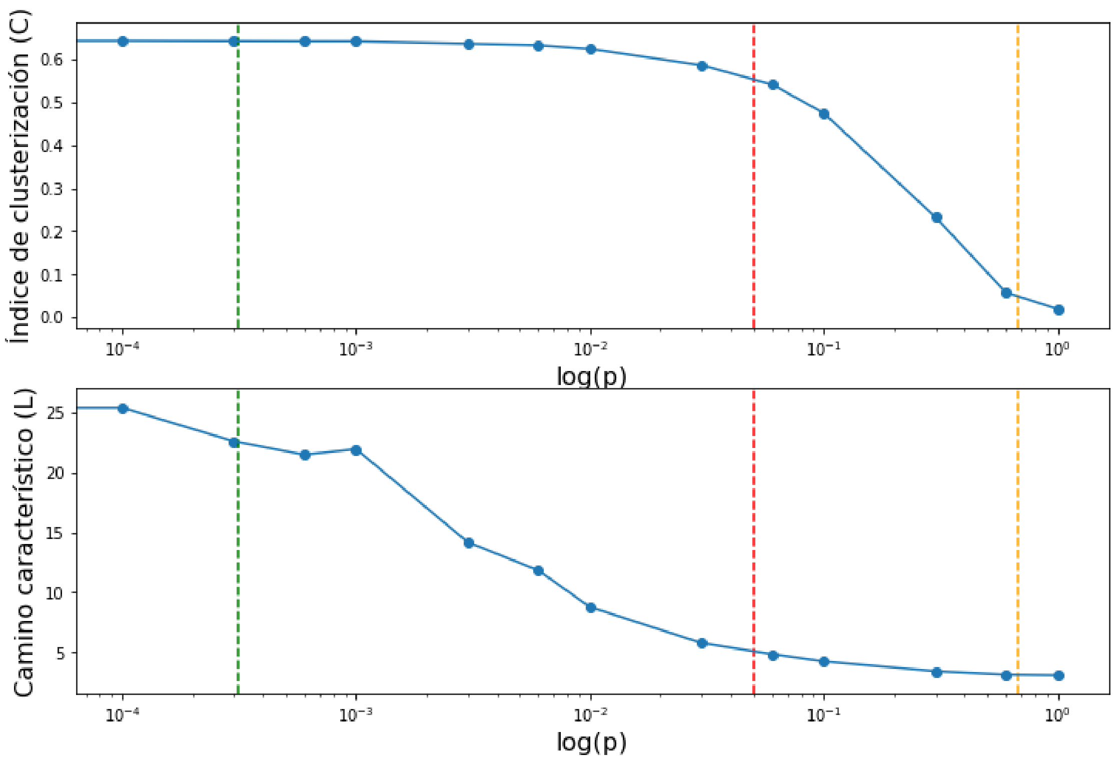
[16]:
```python
pl = 1/(k*n)
print('Probabilidad crítica (1/(k*n)): ', pl)
```

```
Probabilidad crítica (1/(k*n)):  0.0003125
```

[17]:
```python
pc = 0.05 # probabilidad critica donde ocurre el cambio de fase de C
pc_R = 0.68 # Margen derecho region small-world, se explica despues
```

[18]:
```python
plt.figure(figsize = (12, 8))
plt.subplot(211)
plt.xscale("log", nonposx = 'clip')
plt.ylabel("Índice de clusterización (C)", fontsize = 16)
plt.xlabel("log(p)", fontsize = 16)
plt.axvline(x=pc, color='r', ls='--')
plt.axvline(x=pl, color='green', ls='--')
```

```
plt.axvline(x=pc_R, color='orange', ls='--')
plt.plot(ar_p, ar_C)
plt.scatter(ar_p, ar_C)
plt.subplot(212)
plt.xscale("log", nonposx = 'clip')
plt.ylabel("Camino característico (L)", fontsize = 16)
plt.xlabel("log(p)", fontsize = 16)
plt.axvline(x=pc, color='r', ls='--')
plt.axvline(x=pl, color='green', ls='--')
plt.axvline(x=pc_R, color='orange', ls='--')
plt.plot(ar_p, ar_L)
plt.scatter(ar_p, ar_L)
plt.show()
```



En ambos gráficos hemos señalado la probabilidad crítica teórica p_c con una línea vertical de color rojo. Obervaciones:

- La probabilidad crítica teórica p_c para el camino característico se corresponde a la obtenida con los resultados empíricos, ya que marca un cambio de tendencia en la gráfica obtenida para el camino característico L (línea verde punteada).
- A medida que aumenta la probabilidad de reconectar una arista observamos que el índice de clusterización disminuye. En un watts_strogatz los nodos se unen con sus k vecinos. Por lo tanto, a medida que se reconectan aristas esta propiedad se diluye y el índice de cluterización

se comporta en mayor medida como un grafo aleatorio.

2.2 ¿Entre que valores de p dirías que está la zona de Small-World para estos valores de N y k?

En los gráficos anteriores se muestra el comportamiento del índice de clusterización (C) y el camino mímino característico (L) en función de la probabilidad de reconexión p [2].

Vemos que la función del camino característico dependiente de p decrece a valores bajos de p, mientras que el índice de clusterización decrece con valores más altos de p. El efecto de la reconexión sobre el camino característico (L) es altamente no-lineal pues puede disminuir considerablemente con muy pocos atajos ("shortcuts"). Sin embargo, la reconexión de unas pocas aristas en un grupo de nodos muy agrupados solo tiene un efecto lineal sobre ese grado de agrupamiento. Por esto puede haber una región de valores de p para la que el índice de clustericación (C) apenas se ve afectado mientras que L pasa rápidamente de valores altos a pequeños [3].

Las redes de mundo pequeño se encontrarían en torno a la región delimitada por las probabilidades críticas de los parámetros L y C. No obstante, el margen derecho de la región, delimitado por la p_c de C, puede extenderse algo más dado que la caída del índice de clusterización no es tan abrupto como el del camino característico. La extensión puede calcularse conociendo qué índice de clusterización tendría un grafo aleatorio con el mismo número de nodos ($N = 400$) y mismo número de ramas ($N \times k/2$=1600); y calculando posteriormente la probabilidad p del modelo watts_strogatz_graph asociado a ese índice de clusterización.

A continuación se muestra el grafo aleatorio con el mismo número de nodos y ramas que el grafo regular, y el valor del índice de clusterización para 20 simulaciones:

[19]:
```
#Con la funcion del apartado 1
n_graph = 20
e_al = 1600
p = (2*e_al) / (n*(n-1))
print("Probabilidad p aleatoria: ", p)

mu_L, mu_C = L_C_Aleatorio(n_graph,n,p)

print("Camino característico en %d redes aleatorias: "%n_graph, mu_L)
print("Índice clusterización en %d redes aleatorias: "%n_graph, mu_C)
```

```
Probabilidad p aleatoria:  0.020050125313283207
Camino característico en 20 redes aleatorias:  62.4887731829574
Índice clusterización en 20 redes aleatorias:  0.01976174549135074
```

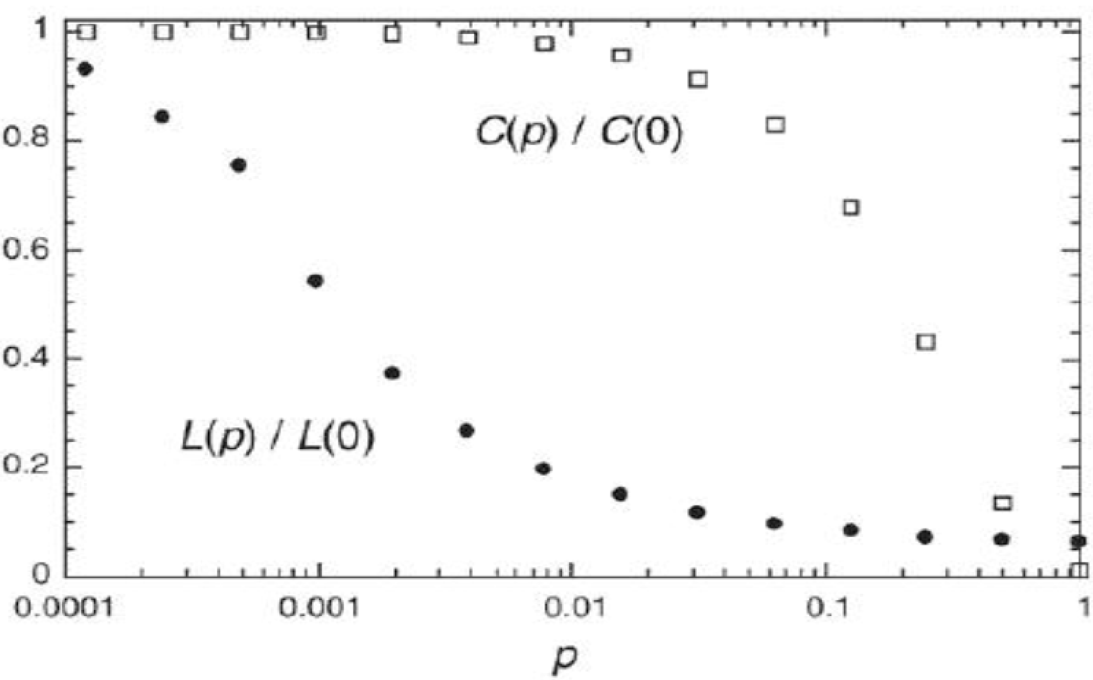
Para una $C \simeq 0.02$, la probabilidad de reordenamiento de aristas del grafo regular p, sería $p = 0.68$. Por lo que la región de mundo pequeño se extendería desde la probabilidad crítica del camino mínimo ($p_c = 0.0003$) hasta la probabilidad donde se obtiene un índice de clusterización igual al de un grafo aleatorio con el mismo número de ramas y nodos ($p = 0.68$). En las gráficas, sería la región entre las líneas verticales punteadas de color verde y naranja.

2.3 ¿Corresponde el valor de p observado con el valor de p teórico para el cambio de fase en el comportamiento del camino característico?

Analizando la gráfica que muestra como el camino mímino característico () cambia en función de la probabilidad de reconexión p, podemos observar que el valor $p_c = 0.0003$ teórico y calculado anteriormente, coincide con la transición de L desde valores altos a bajos.

2.4 Comparar las gráficas obtenidas de C y L con las correspondientes gráficas teóricas

A continuación se muestra la gráfica del comportamiento teórico de L y C en función de la probabilidad de reconexión p:



Cabe mencionar que ambas gráficas son comparables, dado que los valores de C y L han sido normalizados, siendo divididos por el valor de estos parámetros cuando $p = 0$.

Comprobamos que se produce el mismo cambio que hemos observado en nuestras gráficas obtenidas, siendo el parámetro L quien cambia de fase en primer lugar, y C posteriormente. Igualmente, se observa en la gŕafica teórica que la p_c de L se sitúa en torno a 0.003 y 0.005, obteniéndose el mismo resultado que el calculado anteriormente. Lo mismo ocurre para el índice de clusterización C, que comienza a decrecer en torno al valor $p_c = 0.05$. De este modo, la región de mundo pequeño podría extenderse en el margen derecho como en el caso anterior estudiado.

2.5 Con los datos que obtuviste en la práctica 1. ¿Dirías que la red de interacción de proteínas del Caernobidis Elegans puede ser considerada una red de mundo pequeño (Razona la respuesta)?

En la práctica anterior se estudió el índice de clusterización y el camino característico para el grafo de CaernoElegans, obteniéndose estos resultados:

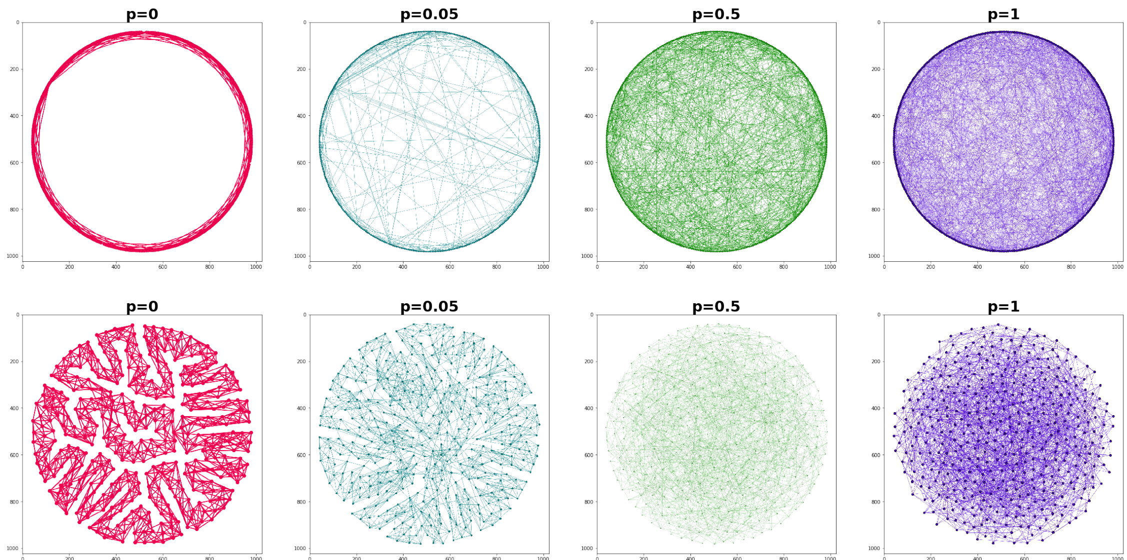
	CaernoElegans	Aleatorio
L	7.922564	7.967503 ± 0.102507
C	0.075708	0.000939 ± 0.0011093
Degree_Centrality	0.001714	0.001714 ± 1.362e-18
Closeness_Centrality	0.071142	0.097581 ± 0.00165
Betweness_Centrality	0.002567	0.003852 ± 0.000130
Max k-core	6	2 ± 0.0
Dispersión/Densidad	0.00171	—

En primer lugar, observamos que la red de proteínas de CaernoElegans no es una red aleatoria, y que presenta un camino característico L igual que el grafo aleatorio, y un índice de clusterización C superior en un orden de magnitud al del grafo aleatorio. En concordancia con lo que se ha establecido anteriormente en este apartado, podemos concluir que la red de proteínas de Caernobidis Elegans es una red de mundo pequeño y libre de escala (esto último se comprobó en la práctica anterior).

A modo de explicación visual, se muestra a continuación la topología que muestran redes generadas con el modelo watts_strogatz_graph con diferentes valores de p: p=0; p=0.05; p=0.5; p=1. Siendo el primero un claro grafo regular, y el último un claro grafo aleatorio. Los dos grafos restantes (p=0.05; p=0.5) pertenecerían a la región definida como mundo pequeño, dado que se encuentran dentro de la región (p= 0.0003, p=0.68).

[20]:
```
images = []
paths = []

# Seleccionamos los paths a las imagenes
for img_path in glob.glob('images/circular/*.png'):
    paths.append(img_path)

# Ordenamos los archivos
paths.sort()

# Cargamos las imagenes
for image in paths:
    images.append(mpimg.imread(image))


# ax enables access to manipulate each of subplots
ax = []
titulos = ["p=0", "p=0.05", "p=0.5", "p=1", "p=0", "p=0.05", "p=0.5", "p=1"]
```

```
fig = plt.figure(figsize=(40,20))
rows = 2
columns = 4
for i, image in enumerate(images):
    ax.append( fig.add_subplot(rows , columns, i+1) )
    ax[-1].set_title(titulos[i], fontsize = 30, fontweight = "bold")
    plt.imshow(image)
```



Se muestran en los grafos en orden ascedente de p (p=0; p=0.05; p=0.5; p=1), en la primera fila con disposición circular, y en segunda fila con disposición Fruchterman Reingold.

En la secuencia de imágenes, se observa claramente cómo es la transición de grafo regular a un grafo aleatorio, apareciendo una región de transición correspondiente a los grafos de mundo pequeño. Analizando ambos tipos de disposiciones se puede observar cómo cambia la topología a medida que aumenta la probabilidad de aparecer atajos (p).

3 Apartado 3

Generar una red libre de escala con 10000 nodos y que añade 10 nodos en cada paso (utilizad la rutina barabasi_albert_graph(n, m)). Calculad los valores de L, C y la distribución de grado de los nodos. Generad una red aleatoria que tenga el mismo número de nodos y ramas que la red que habéis creado, calculad sus valores de L, C y la distribución del grado de los nodos y comparad los valores obtenidos.

[21]:
```
#Creamos la red libre de escala
n = 10000
m = 10
G_BA = nx.barabasi_albert_graph(n, m)
nx.write_edgelist(G_BA, 'barabasi_albert.txt')
```

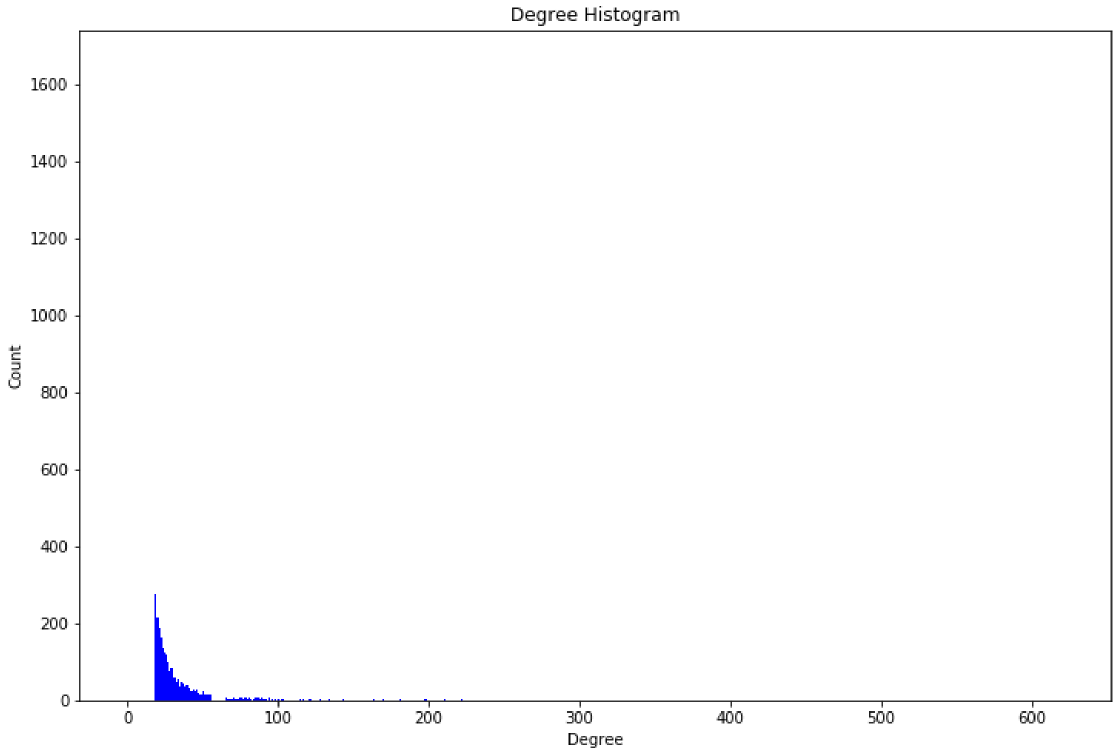
```
nx.write_gexf(G_BA, 'barabasi_albert.gexf')

print(nx.info(G_BA))
```

```
Name:
Type: Graph
Number of nodes: 10000
Number of edges: 99900
Average degree:  19.9800
```

[22]:
```
#L y C
C = nx.average_clustering(G_BA)
L = nx.average_shortest_path_length(G_BA)

print("Camino caracteristico red libre de escala: ", L)
print("Indice clusterizacion red libre de escala: ", C)
```

```
Camino caracteristico red libre de escala:  3.055652085208521
Indice clusterizacion red libre de escala:  0.011823099259453514
```

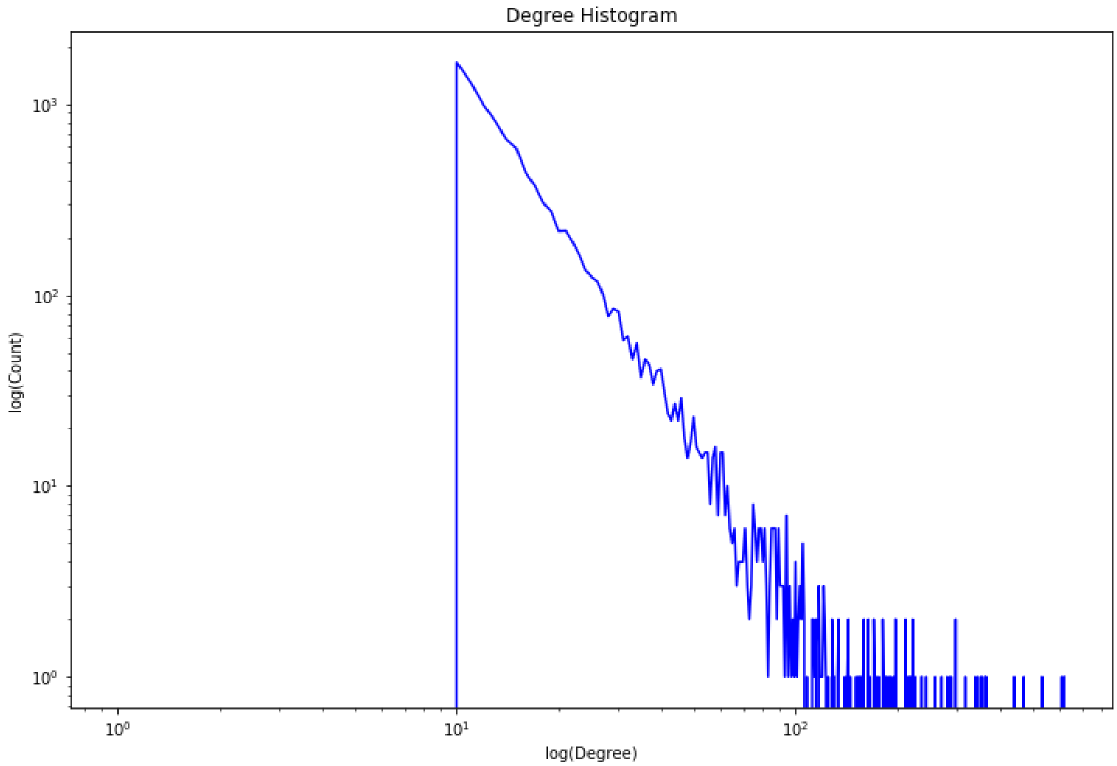
[23]:
```
dh=nx.degree_histogram(G_BA)
plt.figure(figsize = (12, 8))
plt.bar(range(len(dh)),dh, width=0.80, color='b')
plt.title("Degree Histogram")
plt.ylabel("Count")
plt.xlabel("Degree")
plt.show()
print('Grado medio red libre de escala: ',np.mean(list(dict(nx.degree(G_BA)).
 ↪values())))
```




```
Grado medio red libre de escala:  19.98
```

[24]:

```python
plt.figure(figsize = (12, 8))
plt.plot(range(len(dh)),dh, color='b')
plt.title("Degree Histogram")
plt.ylabel("log(Count)")
plt.xlabel("log(Degree)")
plt.xscale("log", nonposx = 'clip')
plt.yscale("log", nonposy = 'clip')
plt.show()
```



El modelo de Barabási y Albert es un modelo empleado para generar redes libres de escala mediante la adición preferencial de nodos. La distribución de grado de las redes libres de escala no se corresponde a una distribución de Poisson característica de redes aleatorias donde la forma de la distribución cae de manera exponencial a medida que nos alejamos del valor máximo, $<k>$ [2]. Tampoco se asemeja a la distribución de los grafos regulares que presentan un Delta en el valor k, que es igual para todos los nodos. Las redes libres de escala se denominan así porque su distribución cae de forma más gradual que una exponencial siguiendo una ley de potencias, lo que permite la existencia de algunos nodos de grado muy alto y que no se concentren alrededor de una media (escala) [2].

Todo lo comentado anteriormente puede comprobarse en las gráficas anteriores. En la primera gráfica donde se representa la frecuencia de nodos en función del grado, observamos como algunos nodos están altamente conectados, aunque el grado de conexión de casi todos los nodos es bastante bajo. Igualmente, en la segunda gráfica se observa que la ley de potencias se comporta de forma lineal en escala logarítmica, apreciendo una cola en el extremo derecho de la ǵrafica correspondiente a la desaparición de nodos durante la creación de la red libre de escala de Barabasi y Albert.

Comparación con grafos aleatorios. Primero realizamos una comparación con un único grafo aleatorio generado con la función gnm_random_graph, donde estudiaremos L, C y la distribución de grados.

Después utilizamos la función del apartado 1 de esta práctica, para comparar con valores más robustos en grafos aleatorios de L y C.

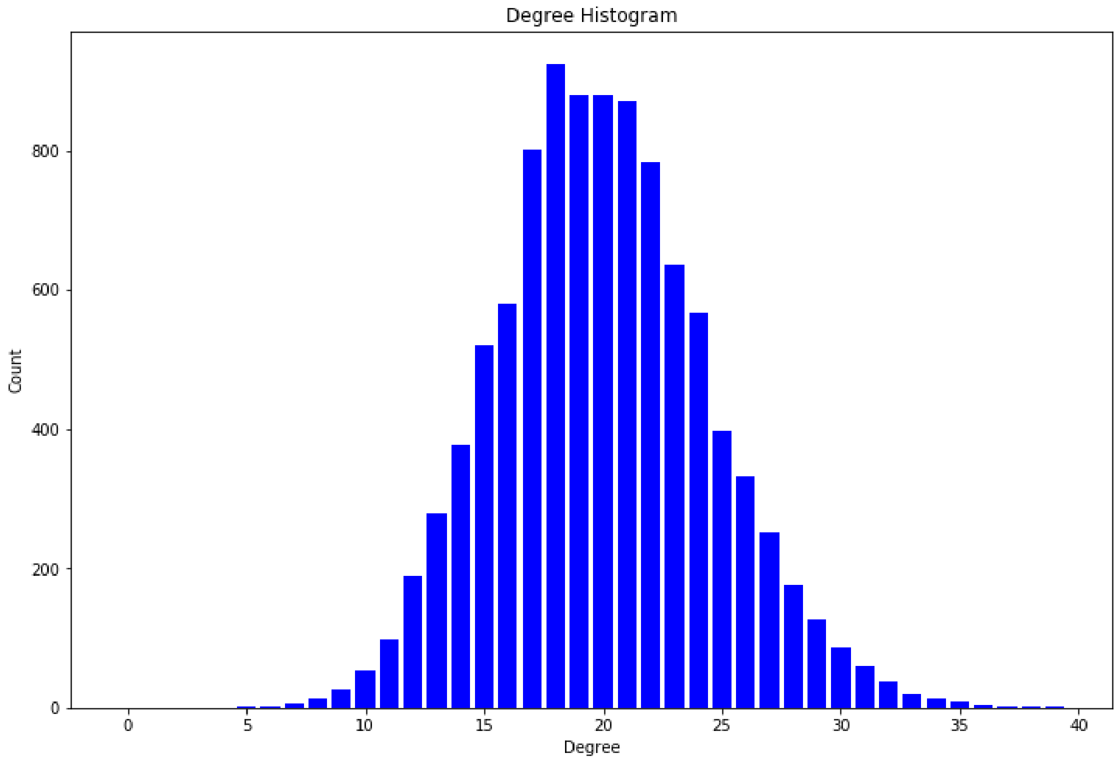
[25]:
```
# Vemos el numero de ramas que tiene el grafo barabasi
# En la info hemos visto que son 99900
e_al = 99900 #from info
G_AL = nx.gnm_random_graph(n, e_al)
nx.write_edgelist(G_AL, 'aleatorio.txt')
nx.write_gexf(G_AL, 'aleatorio.gexf')
print(nx.info(G_AL))
```

```
Name:
Type: Graph
Number of nodes: 10000
Number of edges: 99900
Average degree:  19.9800
```

[26]:
```
C = nx.average_clustering(G_AL)
L = nx.average_shortest_path_length(G_AL)

print("Camino caracteristico red aleatoria: ", L)
print("Indice clusterizacion red aleatoria: ", C)
```

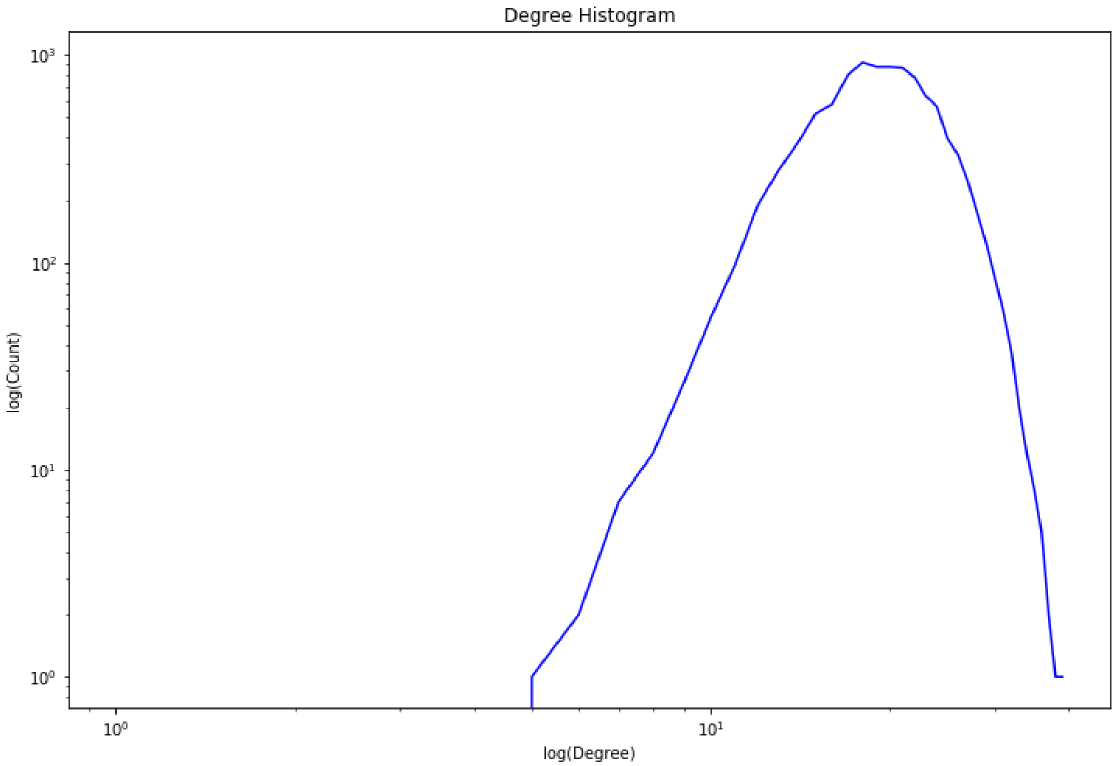
```
Camino caracteristico red aleatoria:  3.4038095409540956
Indice clusterizacion red aleatoria:  0.0021233614402428633
```

[27]:
```
dh=nx.degree_histogram(G_AL)
plt.figure(figsize = (12, 8))
plt.bar(range(len(dh)),dh, width=0.80, color='b')
plt.title("Degree Histogram")
plt.ylabel("Count")
plt.xlabel("Degree")
plt.show()
print('Grado medio del grafo aleatorio: ', np.mean(list(dict(nx.degree(G_AL)).
 ↪values())))
```



Grado medio del grafo aleatorio: 19.98

[28]:
```
plt.figure(figsize = (12, 8))
plt.plot(range(len(dh)),dh, color='b')
plt.title("Degree Histogram")
plt.ylabel("log(Count)")
plt.xlabel("log(Degree)")
plt.xscale("log", nonposx = 'clip')
plt.yscale("log", nonposy = 'clip')
plt.show()
```



Comparando la distribución de grado de redes libre de escala con redes aleatorias, comprobamos que la ley de potencia propia de las redes libres de escala implica que todos los grados entre k_{min} y k_{max} pueden aparecer en la red, a diferencia de las redes homogéneas, como las redes aleatoias de Ërdos y Renyi, donde casi todos los nodos tienen el mismo grado y la probabilidad de que aparezca un nodo con un grado muy diferente de la media decae rápidamente a cero [1]. Igualmente, la representación de la distribución de grado en escala logarítmica de las redes aleatorias muestra un claro comportamiento no-lineal.

Para comparar los parámetros de índice de clusterización C y camino característica L de la red libre de escala con una red aleatoria es interesante usar la función del **apartado 1**. Para ello necesitamos calcular la probabilidad p de generar una arista entre dos nodos en un grafo aleatorio. Sabemos que nuestra red libre de escala tiene: * Número de nodos | V | = 10000 * Número de aristas | E | = 99900 (ver info de G_BA)

Con la ecuación:

$$|E| = p\frac{|V|(|V|-1)}{2}$$

Podemos despejar p, obteniendo:

$$p = \frac{2|E|}{|V|(|V|-1)} = 0.01998$$

Pudiendo generarse así un grafo aleatorio equivalente con el mismo número de nodos y ramas que el grafo libre de escala:

[29]:
```
#Con la funcion del apartado 1
n_graph = 3
p = (2*e_al) / (n*(n-1))
print("Probabilidad p aleatoria: ", p)

mu_L, mu_C = L_C_Aleatorio(n_graph,n,p)

print("Camino caracteristico en %d redes aleatorias: "%n_graph, mu_L)
print("Indice clusterizacion en %d redes aleatorias: "%n_graph, mu_C)
```

```
Probabilidad p aleatoria:  0.001998199819981998
Camino caracteristico en 3 redes aleatorias:  3.4007502150215023
Indice clusterizacion en 3 redes aleatorias:  0.0020345744116506475
```

Como indica la teoría, tanto el camino característico como el índice de clusterización es muy similar entre grafos aleatorios y redes libre de escala. Observando los valores obtenidos para 3 redes aleatorias con el mismo número de nodos y aristas que nuestra red libre de escala, simplemente observamos que el camino característico es ligeramente menor en la red libre de escala (3.40 > 3.06). Lo mismo ocurre para el índice de clusterización, que es superior en la red libre de escala (0.011 vs 0.002). La gran diferencia está en la distribución de grados, como se comentó anteriormente:

- La red aleatoria tiene una distribución de grados que sigue una Poisson, como podemos observar claramente en el histograma generado para este tipo de grafos.
- Sin embargo, en una red libre de escala la distribución de grados sigue una ley de potencias. Por eso, al pintarlo en escala logarítmica, aparece una recta.