Redes Mundo Pequeno

Nome: Arthur Pontes Nader

Matrícula: 2019022294

Bibliotecas

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

from math import pi
from scipy.io import mmread
```

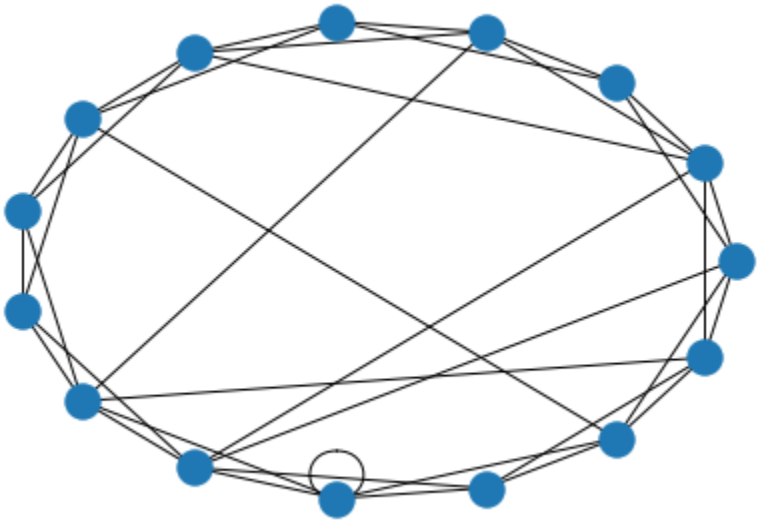
Funções

a) Criação de uma rede

In [2]:
```
def gerar_rede(N, Z, p):

  numero_atalhos = int(p*N*Z/2)

  G = nx.Graph()
  G.add_nodes_from(np.arange(N))

  for i in range(N):

    valor_inicial = (i - Z/2)%N

    adicoes = 0
    while adicoes != Z:

      if valor_inicial == i:
        valor_inicial = (valor_inicial+1)%N
        continue

      G.add_edge(i, valor_inicial)
      valor_inicial = (valor_inicial+1)%N
      adicoes += 1

  for _ in range(numero_atalhos):
    no_1 = np.random.randint(N)
    no_2 = np.random.randint(N)
    G.add_edge(no_1, no_2)

  return G
```

In [3]:
```
def mostrar_rede(G):

  nx.draw_circular(G)
```

In [4]:
```
g = gerar_rede(15, 4, 0.25)
mostrar_rede(g)
```



b) Funções para analisar distribuição do comprimento do caminho

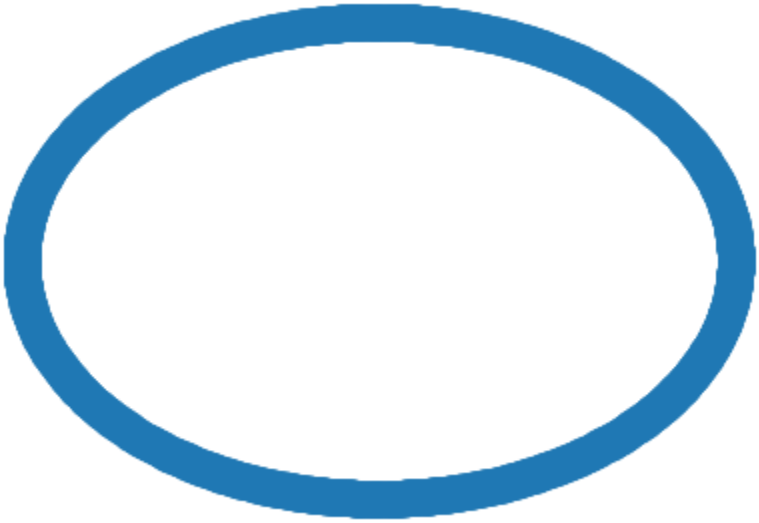
In [5]:
```
def FindPathLengthsFromNode(graph, node):

    distancias = np.full(len(graph), -1)

    distancias[node] = 0
    currentShell = [node]

    while len(currentShell) > 0:

        actual_node = currentShell.pop(0)
        nextShell = []
        vizinhos = graph.neighbors(actual_node)

        for v in vizinhos:
            if distancias[int(v)] == -1:
              distancias[int(v)] = distancias[actual_node] + 1
              nextShell.append(int(v))

        currentShell += nextShell

    return list(distancias)
```

In [6]:
```
def FindAllPathLengths(graph):

    distancias = []

    for node in range(len(graph)):
        d = FindPathLengthsFromNode(graph, node)
        distancias.append(d)

    return distancias
```

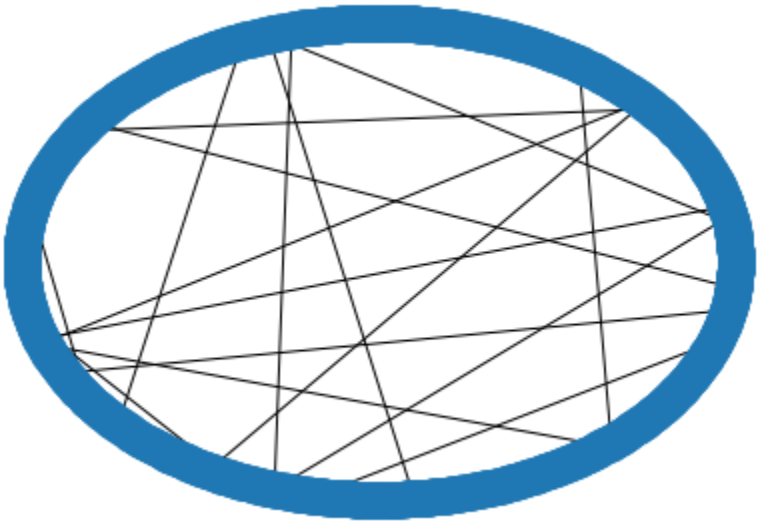
In [7]:
```
def FindAveragePathLength(graph):

    distancias = FindAllPathLengths(graph)

    return np.mean(distancias)
```

Grafos circulares e histogramas

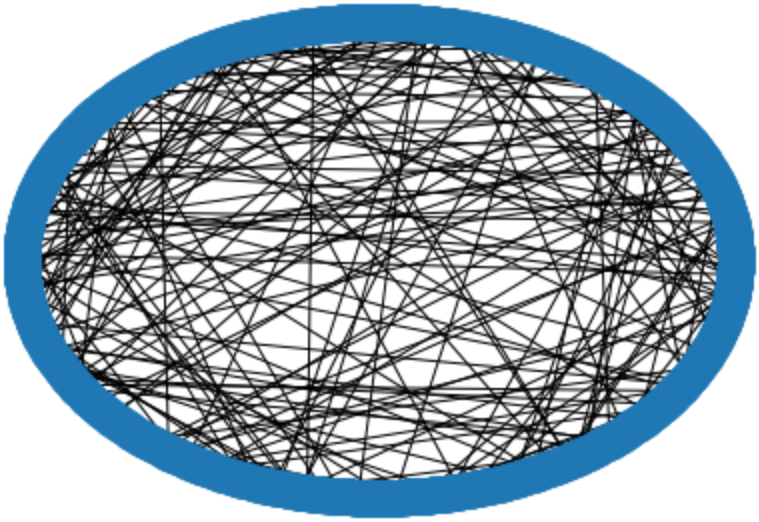
In [ ]:
```
r1 = gerar_rede(1000, 2, 0)
mostrar_rede(r1)
```



In [ ]:
```
r2 = gerar_rede(1000, 2, 0.02)
mostrar_rede(r2)
```

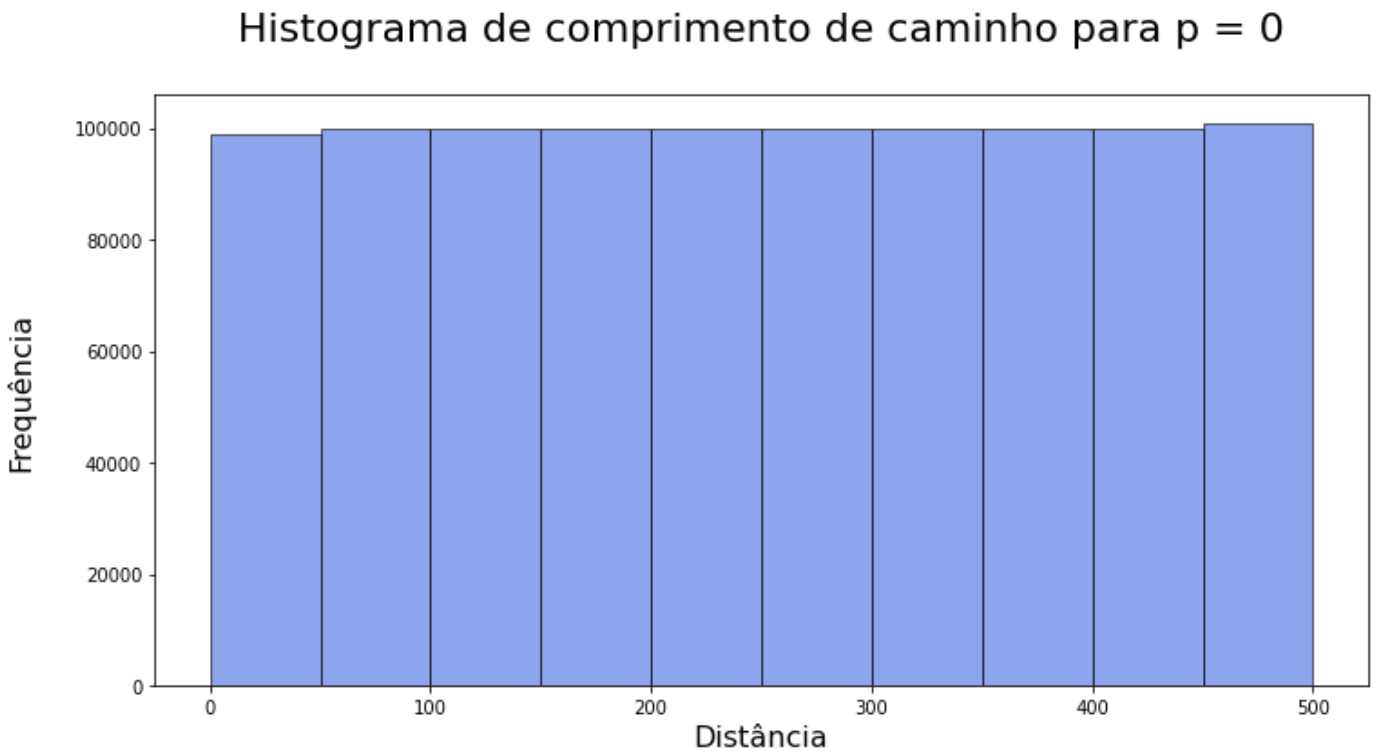


In [ ]:
```
r3 = gerar_rede(1000, 2, 0.2)
mostrar_rede(r3)
```

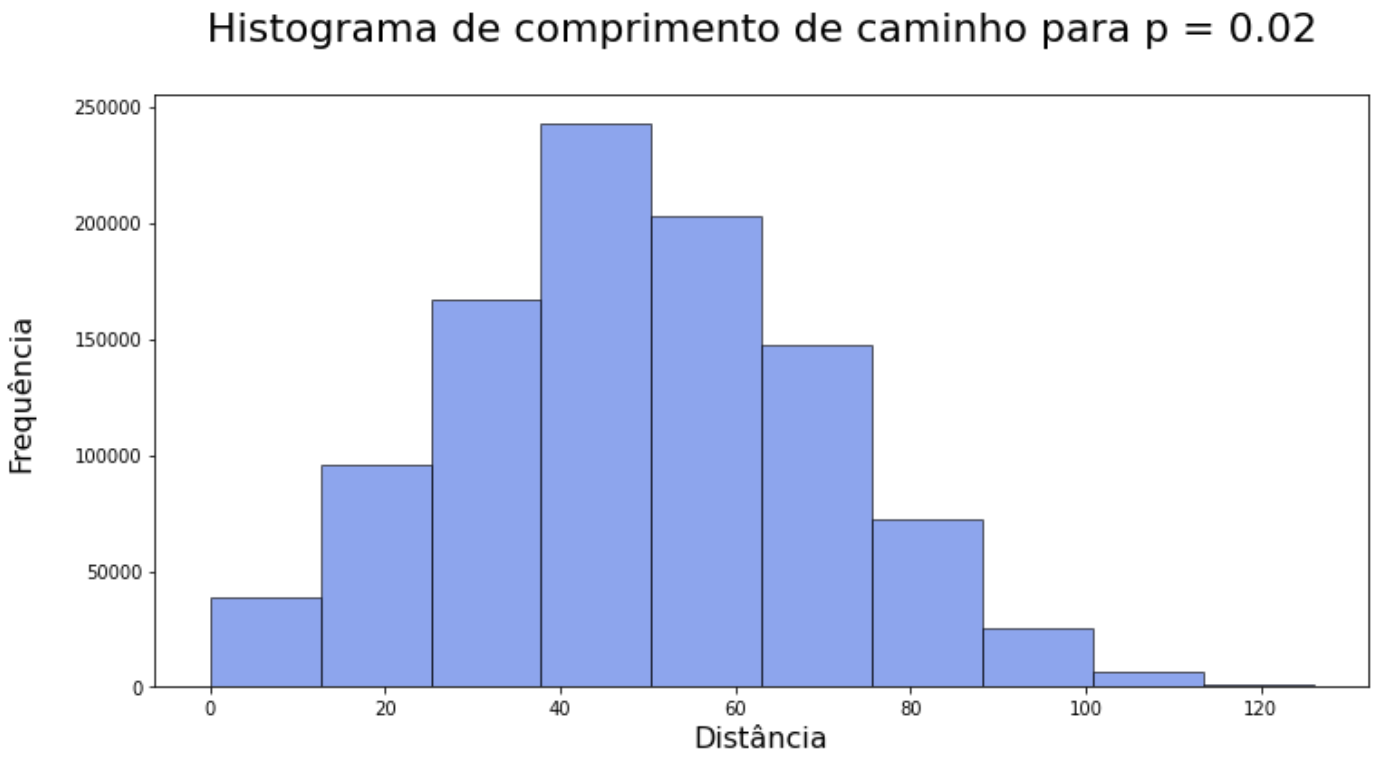


In [8]:
```
def plotar_histograma(rede, p):

  distancias = FindAllPathLengths(rede)
  dados = sum(distancias, [])

  plt.figure(figsize=(12, 6))
  plt.xlabel("Distância\n", fontsize = 16)
  plt.ylabel("Frequência\n", fontsize = 16)
  plt.title("Histograma de comprimento de caminho para p = " + str(p) + "\n", fontsize =
  plt.hist(dados, ec = "k", alpha = .6, color = "royalblue")
  plt.show()
```

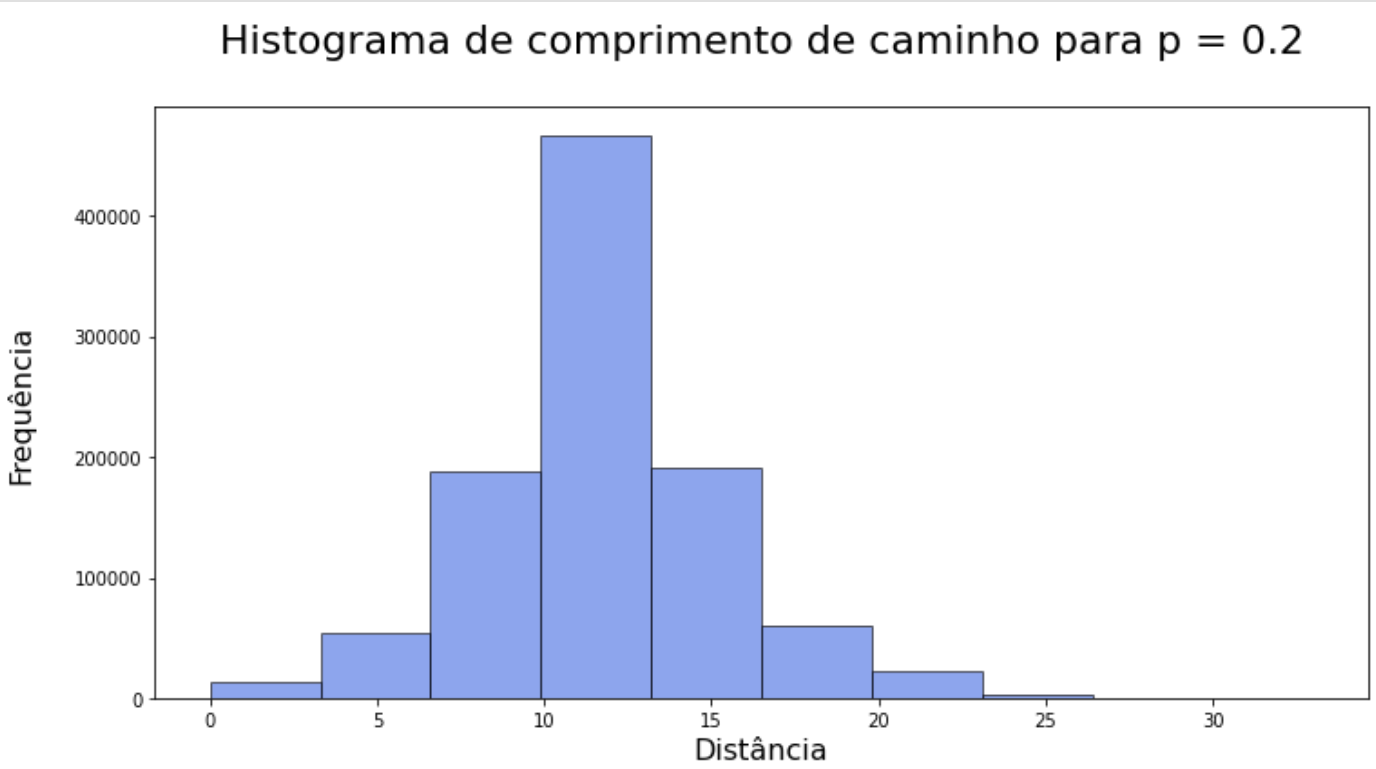
In [ ]:
```
plotar_histograma(r1, 0)
```



In [ ]:
```
plotar_histograma(r2, 0.02)
```



In [ ]:
```
plotar_histograma(r3, 0.2)
```



Cálculo da média

In [ ]:
```
distancias_medias = []

for _ in range(5):
  g = gerar_rede(100, 2, 0.1)
  distancias_medias.append(FindAveragePathLength(g))

print("Média = %.4f \nVariância = %.4f" %(np.mean(distancias_medias), np.var(distancias_
```

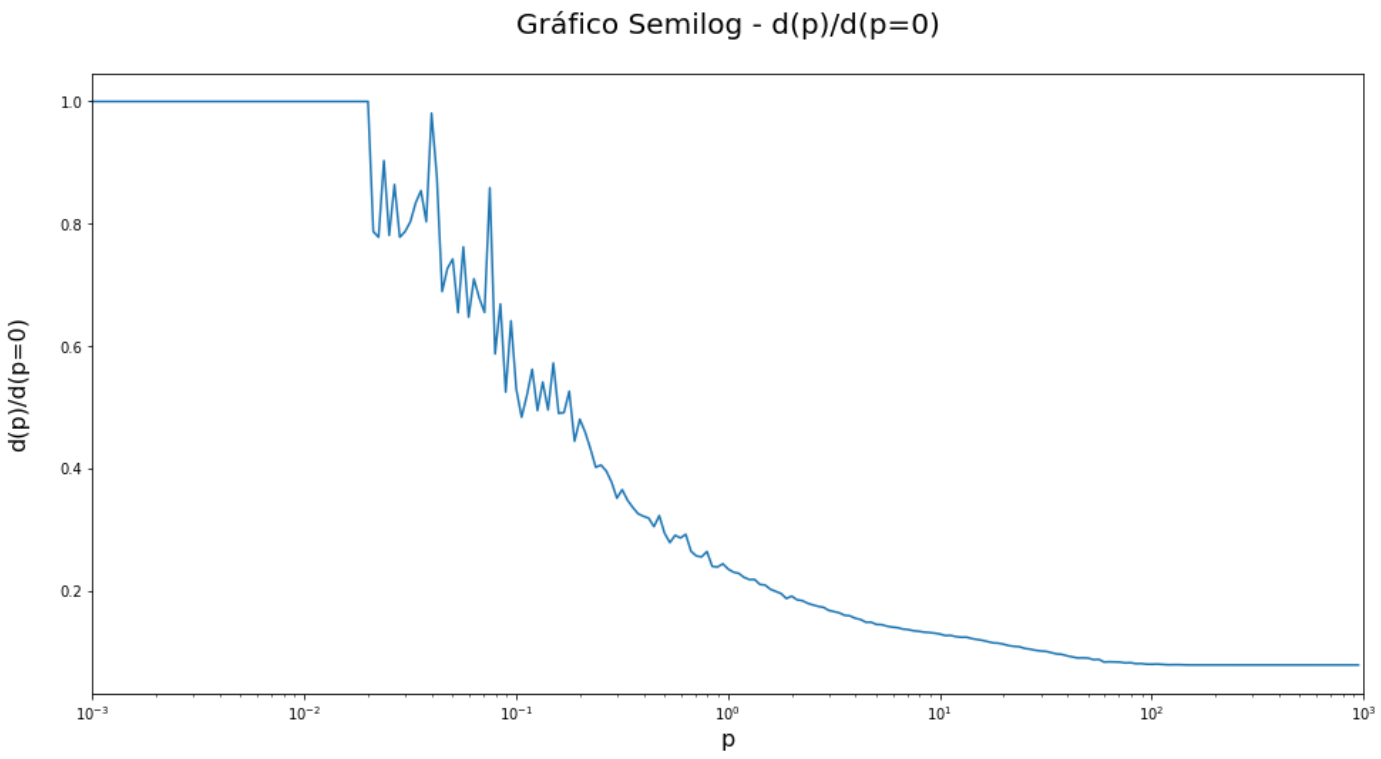
```
Média = 9.2387
```

```
Variância = 0.4503
```

c) Gráfico do comprimento médio do caminho entre os nós d(p) dividido por d(p = 0)

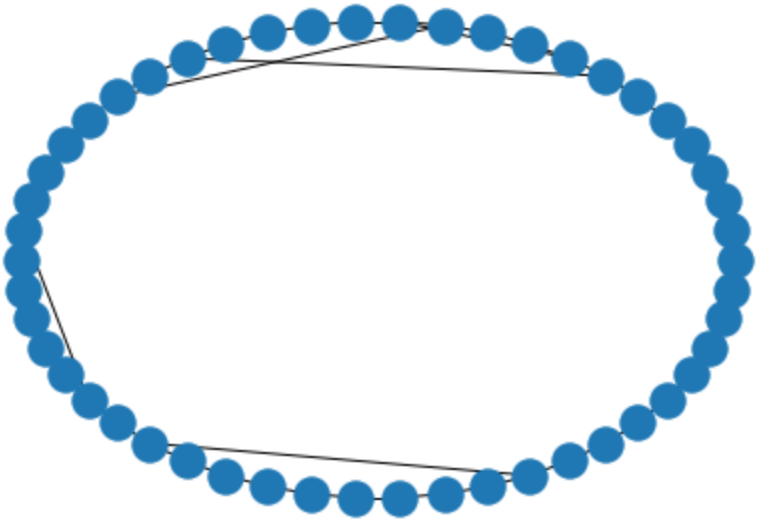
In [ ]:
```
grafo_inicial = gerar_rede(50, 2, 0.001)
l_inicial = FindAveragePathLength(grafo_inicial)

resultados = []
valores_p = []

for n in np.arange(-3, 3, 0.025):
    p = 10 ** n
    grafo = gerar_rede(50, 2, p)
    l = FindAveragePathLength(grafo)
    resultados.append(l/l_inicial)
    valores_p.append(p)
```

In [ ]:
```
plt.figure(figsize=(16, 8))
plt.semilogx(valores_p, resultados)
plt.xlim([0.001,1000])
plt.title('Gráfico Semilog - d(p)/d(p=0)\n', fontsize = 20)
plt.xlabel('p\n', fontsize = 16)
plt.ylabel('d(p)/d(p=0)\n', fontsize = 16)
plt.show()
```

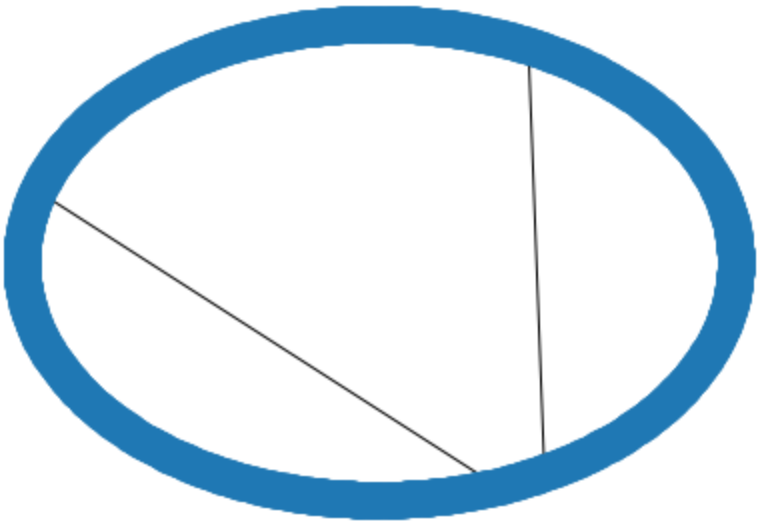


d) Geometria de Watts e Strogatz

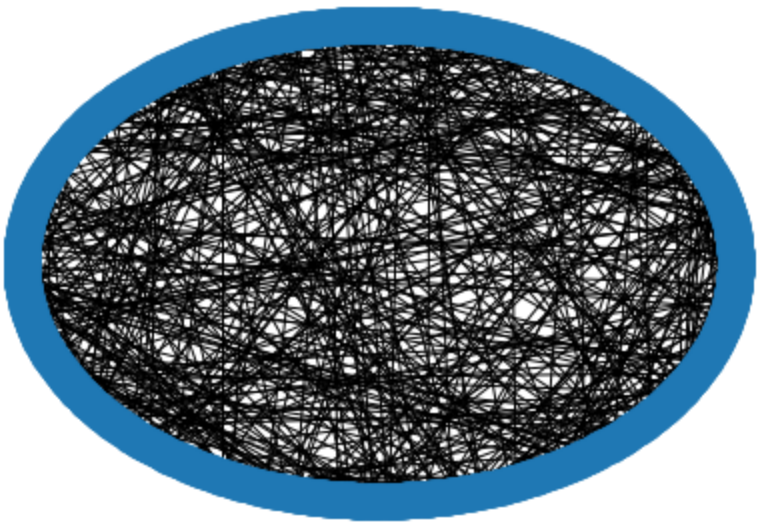
In [ ]:
```
grafo = gerar_rede(50, 2, 0.1)
mostrar_rede(grafo)
```



In [ ]:
```
watts_strogatz1 = nx.watts_strogatz_graph(1000,10,0.001)
mostrar_rede(watts_strogatz1)
```



In [ ]:
```
watts_strogatz2 = nx.watts_strogatz_graph(1000,10,0.1)
mostrar_rede(watts_strogatz2)
```

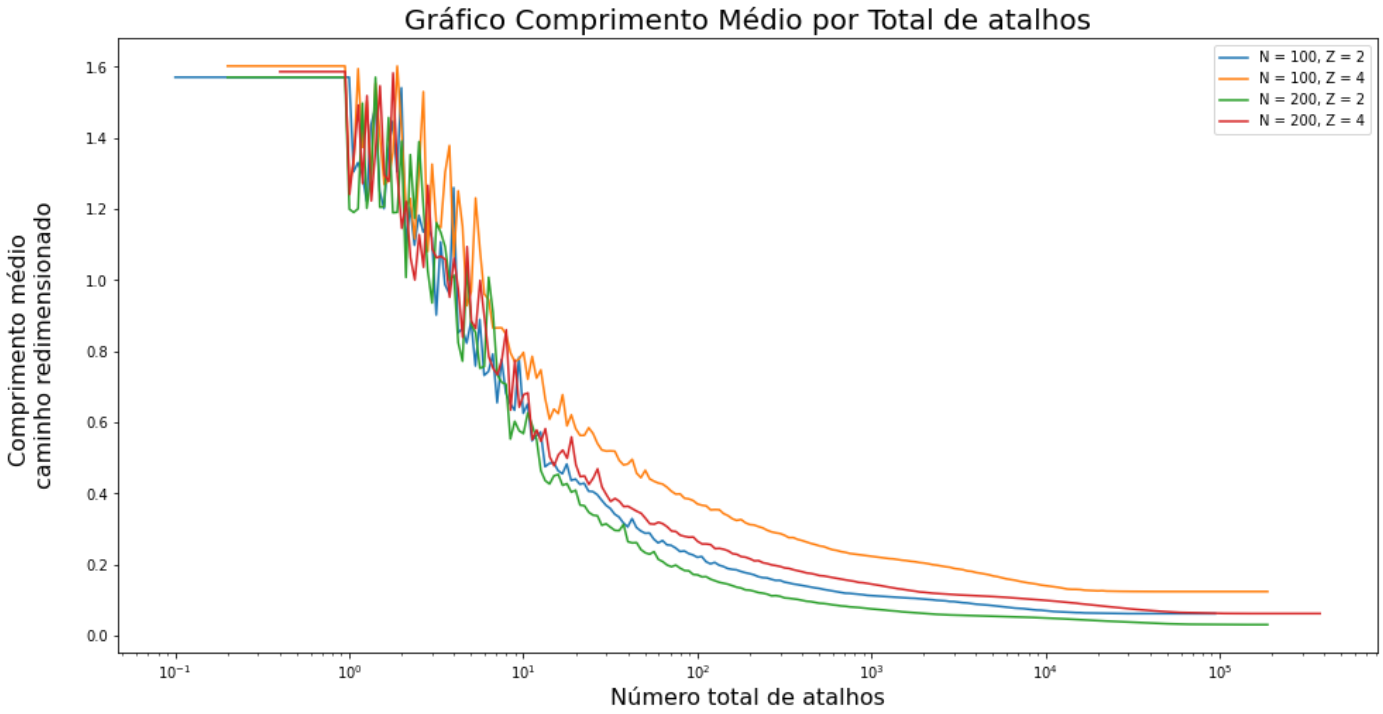


In [ ]:
```
def gerar_dados(N, Z):

  numeros_atalhos = []
  resultados = []

  for n in np.arange(-3, 3, 0.025):
      p = 10 ** n
      grafo = gerar_rede(N, Z, p)
      l = FindAveragePathLength(grafo)

      resultados.append(pi * Z * l/ N)
      numeros_atalhos.append(p * N * Z/2)

  return numeros_atalhos, resultados
```

In [ ]:
```
def plotar_grafico(dados):

  plt.figure(figsize=(16, 8))
  plt.xlabel("Número total de atalhos", fontsize = 16)
  plt.ylabel("Comprimento médio \ncaminho redimensionado\n", fontsize = 16)
  plt.title("Gráfico Comprimento Médio por Total de atalhos", fontsize = 20)

  labels = ('N = 100, Z = 2','N = 100, Z = 4','N = 200, Z = 2','N = 200, Z = 4')
  id_label = 0

  for num_atalhos, resultados in dados:
    plt.semilogx(num_atalhos, resultados, label=labels[id_label])
    id_label +=1

  plt.legend()
  plt.show()
```

In [ ]:
```
atalhos_1, r_1 = gerar_dados(100, 2)
atalhos_2, r_2 = gerar_dados(100, 4)
atalhos_3, r_3 = gerar_dados(200, 2)
atalhos_4, r_4 = gerar_dados(200, 4)
```

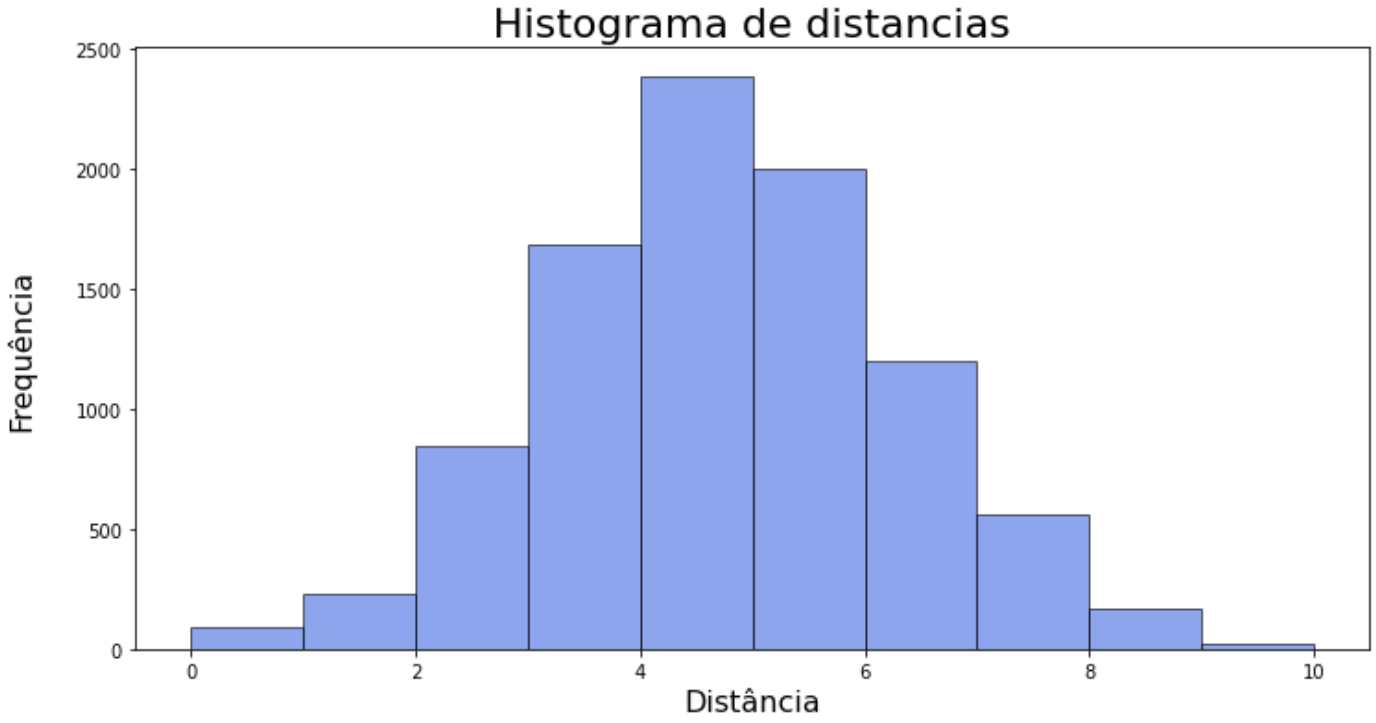
In [ ]:
```
dados = ((atalhos_1, r_1), (atalhos_2, r_2), (atalhos_3, r_3), (atalhos_4, r_4))
plotar_grafico(dados)
```



e) Redes Reais

In [12]:
```
matrix = mmread('rt-retweet.mtx')
grafo = nx.from_scipy_sparse_matrix(matrix)
```

In [14]:
```
distancias = FindAllPathLengths(grafo)
dados = sum(distancias, [])

plt.figure(figsize=(12, 6))
plt.xlabel("Distância\n", fontsize = 16)
plt.ylabel("Frequência\n", fontsize = 16)
plt.title("Histograma de distancias", fontsize = 22)
plt.hist(dados, ec = "k", alpha = .6, color = "royalblue")
plt.show()
```



In [13]:
```
distancia_media = FindAveragePathLength(grafo)
print("Distância Média:", distancia_media)
```

```
Distância Média: 4.265625
```

Análise dos Resultados e Conclusão

Os histogramas de comprimento de caminho gerados permitem concluir que a medida que se aumenta o valor de p, e consequentemente, aumenta-se a quantidade de arestas, diminui-se a distância entre os nós. Isso ocorre pois as arestas adicionadas podem fazer com que nós que anteriormente estavam distantes fiquem pertos.

Ja no gráfico do comprimento médio do caminho entre os nós d(p) dividido por d(p = 0), nota-se que com valor de p pequeno, poucas ou nenhuma arestas foram adicionadas ao grafo, de forma que as distâncias permanecem similares as de p = 0. Isso explica o comportamento do gráfico fixado para valores pequenos de p. Percebeu-se também que a medida que o valor de p cresce, o gráfico tende a um valor constante.

Na parte relacionada a Geometria de Watts e Strogatz, o grafo gerado por watts_strogatz_graph com p = 0.001 ficou mais semelhante com o gerado usando a função criada. Ambos apresentaram uma quantidade

bem menor de arestas ligando nós quando comparados com o grafo gerado por watts_strogatz_graph com p = 0.1

A rede real que foi analisada é baseada no Twitter e pode ser encontrada no seguinte link: https://networkrepository.com/rt.php. Pela análise do histograma gerado, pode-se perceber que houveram nós separados por uma distância maior que 6.

In [ ]: