1. Describe three networks from domains that interest you. What do these networks represent? What are the nodes? What are the edges?
   →
   1) Communication Network
      a. Transportation network → A transportation network is a network or graph in geographic space, describing an infrastructure that permits and constrains movement or flow. Road networks are one of the good network examples.
         i. Nodes: Cities
         ii. Edges: Roads
      b. The Internet → The Internet is called 'Network of Networks' because it is a global network of computers that are linked together by cables and telephone lines making communication possible among them. It can be defined as a global network over a million smaller heterogeneous computer networks.
         i. Nodes: Computers
         ii. Edges: Routers, ethernets cables
      c. The world wide web → The World Wide Web (WWW), commonly known as the Web, is an information system where documents and other web resources are identified by URLs, which may be interlinked by hyperlinks, and are accessible over the Internet
         i. Nodes: Web pages
         ii. Edges: Hyperlinks
   2) Social Network
      a. Transaction Network → Transaction Network Services (TNS) provides global, dedicated real-time data communication networks enabling industry participants to simply and securely interact and transact with other businesses, while connecting to the data and applications they need.
         i. Nodes: Bank, Customers
         ii. Edges: money flows through economic networks
      b. Disease Transmission network → The disease transmission network consists of an agent (pathogen), a susceptible host, and an environment (physical, social, behavioral, cultural, political, and economic factors) that brings the agent and host together, causing infection and disease to occur in the host.
         i. Nodes: Individuals
         ii. Edges: diseases spread by infected individuals in contact with healthy individuals, by air water etc.
      c. Social Networking Sites: → Social networking sites (SNSs) are virtual communities where users can create individual public profiles, interact with real-life friends, and meet other people based on shared interests.
         i. Nodes: People
         ii. Edges: information and ideas flow over friendship/acquaintance networks, such as Instagram
   3) Biochemical Network
      a. Chemical Network → A chemical reaction network consists of a set of chemical reactions and a set of chemical compounds. Each chemical compound is a node of the network. Each chemical reaction is a directed vertex of the network, connecting the

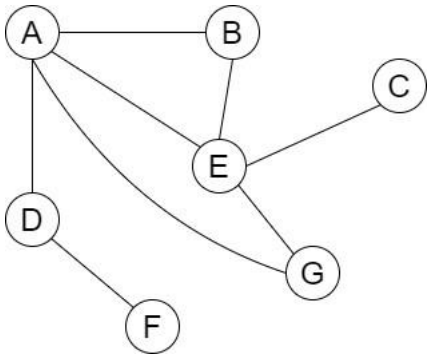
chemical compounds involved in the reaction, from the reactants towards the products.

i. Nodes: molecules
ii. Edges: Chemical Reactions

b. Neural Network → Neural networks, are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

i. Nodes: neurons
ii. Edges: Synaptic connections

Q2.



(a) (5 pts) Produce the adjacency list of this graph.

Vertex

A: [B, D, E, G,]

B: [A, E]

C: [E]

D: [A, F]

E: [A, B, C, G]

F: [D]

G: [A, E]

(b) (5 pts) Produce the adjacency matrix of this graph.

Vertex	A	B	C	D	E	F	G
A	0	1	0	1	1	0	1
B	1	0	0	0	1	0	0
C	0	0	0	0	1	0	0
D	1	0	0	0	0	1	0
E	1	1	1	0	0	0	1
F	0	0	0	1	0	0	0
G	1	0	0	0	1	0	0

(c) (5 pts) Determine the clustering coefficient of each vertex.

Vertex	Clustering Coefficient
A:	K = 4, L = 2, C = 2(2)/4(4-1) => C = 4/12 = 1/3
B:	K = 2, L = 1, C = 2(1)/2(2-1) => C = 2/2 = 1
C:	K = 1, L = 0, C = 2(0)/1(1-1) => C = 0/0 = 0
D:	K = 2, L = 0, C = 2(0)/2(2-1) => C = 0/2 = 0
E:	K = 4, L = 2, C = 2(2)/4(4-1) => C = 4/12 = 1/3
F:	K = 1, L = 0, C = 2(0)/1(1-1) => C = 0/0 = 0
G:	K = 2, L = 1, C = 2(1)/2(2-1) => C = 2/2 = 1

(d) (5 pts) Determine the average degree of the graph.

Degree of each vertex are as below:

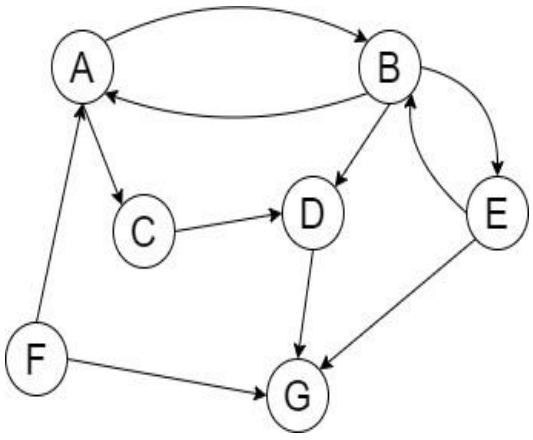
Vertex	Degree
A	4
B	2
C	1
D	2
E	4
F	1
G	2

Average Degree = (4+2+1+2+4+1+2) = 16 /7 = 2.285

OR

Average Degree = 2 * Number of edges/number of vertices
= 2 *(8)/7
= 16/7
= 2.285

Q3



(a) (5 pts) Produce the adjacency list of this graph.

Vertex	
A:	[B, C]
B:	[A, D, E]
C:	[D]
D:	[G]
E:	[B, G]
F:	[A, G]
G:	[]

(b) (5 pts) Produce the adjacency matrix of this graph.

Vertex	A	B	C	D	E	F	G
A	0	1	1	0	0	0	0
B	1	0	0	1	1	0	0
C	0	0	0	1	0	0	0
D	0	0	0	0	0	0	1
E	0	1	0	0	0	0	1
F	1	0	0	0	0	0	1
G	0	0	0	0	0	0	0

(c) (5 pts) Determine the in- and out-degree of each vertex.

Vertex	In-Degree	Out-Degree
A	2	2
B	2	3
C	1	1
D	2	1
E	1	2
F	0	2
G	3	0

Imports used in code

```
#imports Required
import numpy as np
from scipy.stats import binom, poisson
from scipy.special import comb
import networkx as nx
import matplotlib.pyplot as plt
from itertools import combinations
from random import random
import math as math

```

Question 4.

Write a function gnp(n, p) in Python that returns an Erd¨os-R´enyi random graph Gn,p.

Include visualizations of G10,0.5, G200,0.005, and G500,0.05.

```
# function to generate Random graph for n, p
def randomGraphGenerator(n, p):
    V = set([v for v in range(n)])
    E = set()
    for combination in combinations(V, 2):
        a = random()
        if a < p:
            E.add(combination)

    g = nx.Graph()
    g.add_nodes_from(V)
    g.add_edges_from(E)
    return g
```

```
# Function to print/draw random graph
def drawRandomGraph(n, p):
  print("Random Graph Generation G(",n,",",p,")")
  nx.draw_networkx(g, with_labels=True, node_size=100)
  plt.title('$G_{'+str(n)+','+str(p)+'}$', fontsize=24)
  plt.show()
```

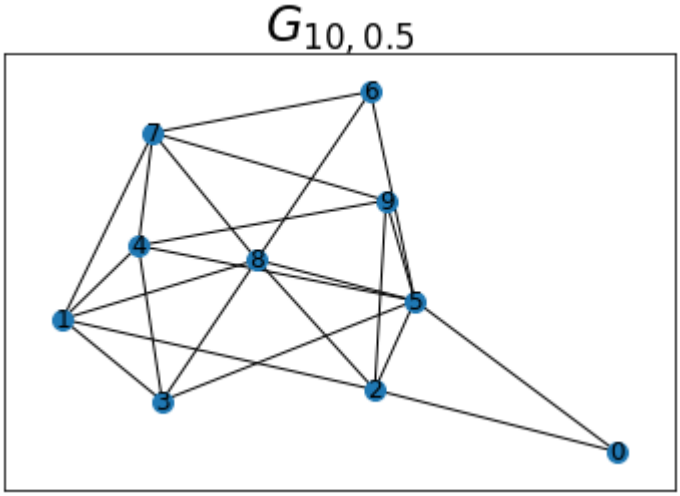
.

.

Random Graph Generation G(10,0.5)

```
#Random Graph Generation G(10,0.5)
n= 10
p = 0.5
g = randomGraphGenerator(n , p)
drawRandomGraph(n, p)
```

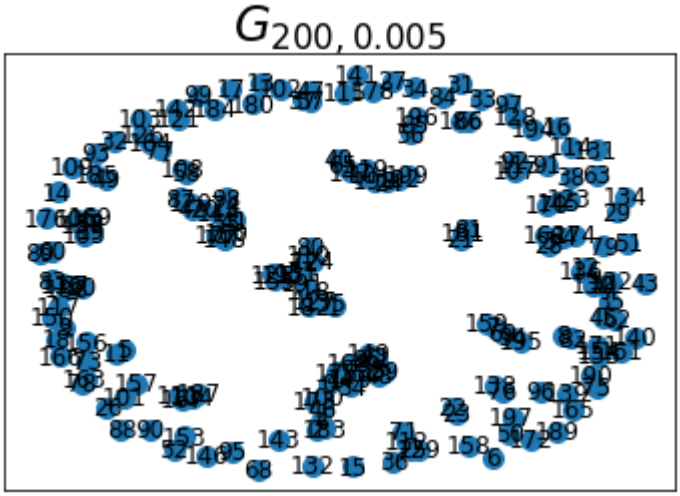
```
Random Graph Generation G( 10 , 0.5 )
```



Random Graph Generation G(200,0.005)

```
#Random Graph Generation G(200,0.005)
n = 200
p = 0.005
g = randomGraphGenerator(n, p)
drawRandomGraph(n, p)
```

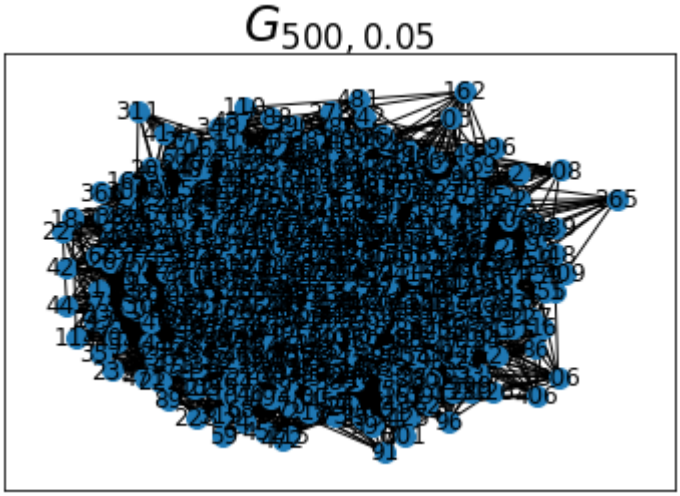
```
Random Graph Generation G( 200 , 0.005 )
```



Random Graph Generation G(500,0.05)

```
1 #Random Graph Generation G(500,0.05)
2 n = 500
3 p = 0.05
4 g = randomGraphGenerator(n, p)
5 drawRandomGraph(n, p)
```

```
Random Graph Generation G( 500 , 0.05 )
```



.

.

.

.

.

.

.

.

.

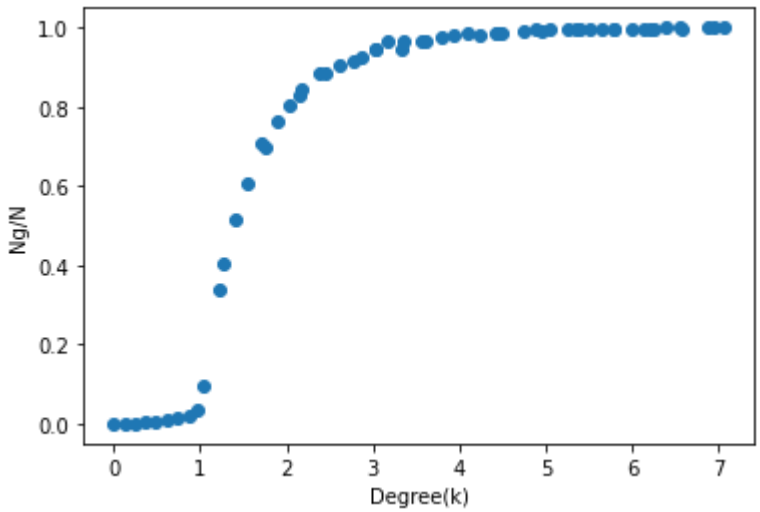
.

Question 5

Generate a figure showing the relative size of the giant component in Erd os-R enyi random graphs as a function of (k), the average degree. In particular, the x-axis should be (k) and the y-axis should be the size of the giant component divided by the total number of vertices in the graph. Identify the subcritical regime, the critical point, the supercritical regime, and the connected regime in this figure.

```
#Function to get the gaint component from random graph
def drawGaintComponent(n,p):
  G = randomGraphGenerator(n, p)
  giantSizes = max(nx.connected_components(G), key=len)
  giantC = G.subgraph(giantSizes)
  avg_degree = 2*G.number_of_edges() / float(G.number_of_nodes())
  return giantC , avg_degree
```

```
n = 2400
X =[]
Y =[]
for p in np.arange(0, 0.003, 0.00005):
  giantC, avg_degree = drawGaintComponent(n,p)
  X.append(avg_degree)
  Y.append(giantC.number_of_nodes()/n)
plt.scatter(X, Y)
plt.xlabel('Degree(k)')
plt.ylabel('Ng/N')
plt.show()
```



.

.

Subcritical Regime: 0 ‹ ‹k› ‹ 1 (p ‹ 1/N)

In a subcritical regime the network has no giant component, only small clusters. In the special case of the network is not connected at all. A random network is in a subcritical regime until the average degree exceeds the critical point, that is the network is in a subcritical regime as long as (k) < 1

In above figure Subcritical Regime is the area between 0 ‹ ‹k› ‹ 1

Critical Point: ‹k› = 1 (p = 1/N)

A critical point is a value of average degree, which separates random networks that have a giant component from those that do not (i.e. it separates a network in a subcritical regime from one in a supercritical regime). Considering a random network with an average degree k the critical point is K=1

In above figure Critical Point: ‹k› = 1

Supercritical Regime: ‹k› › 1 (p › 1/N)

In a supercritical regime, in contrary to the subcritical regime the network has a giant component. In the special case of K = N-1 the network is complete. A random network is in a supercritical regime if the average degree exceeds the critical point, K > 1

In above figure Supercritical Regime: ‹k› › 1

Connected Regime: ‹k› › lnN (p › lnN/N)

Random network theory predicts that for ‹k› › 1 we should observe a giant component, a condition satisfied by all networks we examined. Most networks, however, do not satisfy the ‹k› › lnN condition, implying that they should be broken into isolated clusters

For sufficiently large p the giant component absorbs all nodes and components, hence NG≃ N. In the absence of isolated nodes the network becomes connected. The average degree at which this happens depends on N as ⟨k⟩=lnN

when we enter the connected regime the network is still relatively sparse, as lnN / N → 0 for large N. The network turns into a complete graph only at ‹k› = N - 1.

In this figure connectedness is shown for (k) > ln k

.

Question 6

Pick a real world network from ICON. Let n be the number of vertices and p = (k)/n where (k) is the average degree of vertices in the network. n should be at least 1000.

6 (a) Describe the network you chose and what it represents.

Newman's scientific collaboration network

Resource:

- Details: https://toreopsahl.com/datasets/#newman2001
- Dataset: http://opsahl.co.uk/tnet/datasets/Newman-Cond_mat_95-99-co_occurrence.txt

1. Scientific collaboration network is a social network where nodes are scientists and links are co-authorships.
2. It is an undirected, scale-free network where the degree distribution follows a power law with an exponential cutoff.
3. Most authors are sparsely connected while a few authors are intensively connected.
4. The network has an assortative nature – hubs tend to link to other hubs and low-degree nodes tend to link to low-degree nodes.
5. It is the co-authorship network of based on preprints posted to Condensed Matter section of arXiv E-Print Archive between 1995 and 1999.
6. This dataset can be classified as a two-mode or affiliation network since there are two types of "nodes" (authors and papers) and connections exist only between different types of nodes.

```
#Newman's scientific collaboration network
scientific_collab_graph_data = nx.read_edgelist('/content/Newman-Cond_mat_95-99-co_o
                                        data=[('weight', int)])
print(nx.info(scientific_collab_graph_data))
```

```
Graph with 16264 nodes and 47594 edges
```

Note: I tried to use the dataset url (http://opsahl.co.uk/tnet/datasets/Newman-Cond_mat_95-99-co_occurrence.txt) instead of hardcoded file path, somehow google colab is refusing the connection with URL, Can you please download the data and proceed.

.

.

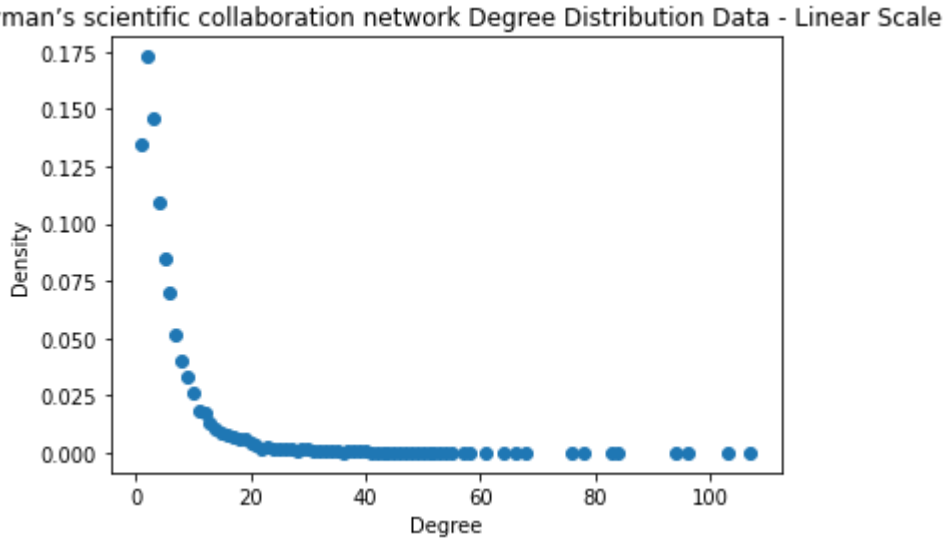
.

6 (b)Generate two scatterplots showing the degree distribution of this network, one with linear axes and one with logarithmic axes. Treat edges in the network as undirected.

Function to show the degree distribution of Graph on linear and log scale

```
#Function to show the degree distribution of Graph on linear and log scale
def degreeDistrGLinearLogScale(G,linearLog):
  degrees = {}
  degreeList = [G.degree(v) for v in G.nodes()]
  for deg in degreeList:
    degrees[deg] = degrees.get(deg, 0) + 1
  (X, Y) = zip(*[(key, degrees[key]/len(G)) for key in degrees])
  plt.scatter(X, Y)
  plt.title('Newman’s scientific collaboration network Degree Distribution Data - Li
  if linearLog:
      plt.yscale('log')
      plt.xscale('log')
      plt.title('Newman’s scientific collaboration network Degree Distribution Data
  plt.xlabel('Degree')
  plt.ylabel('Density')
  plt.show()
```

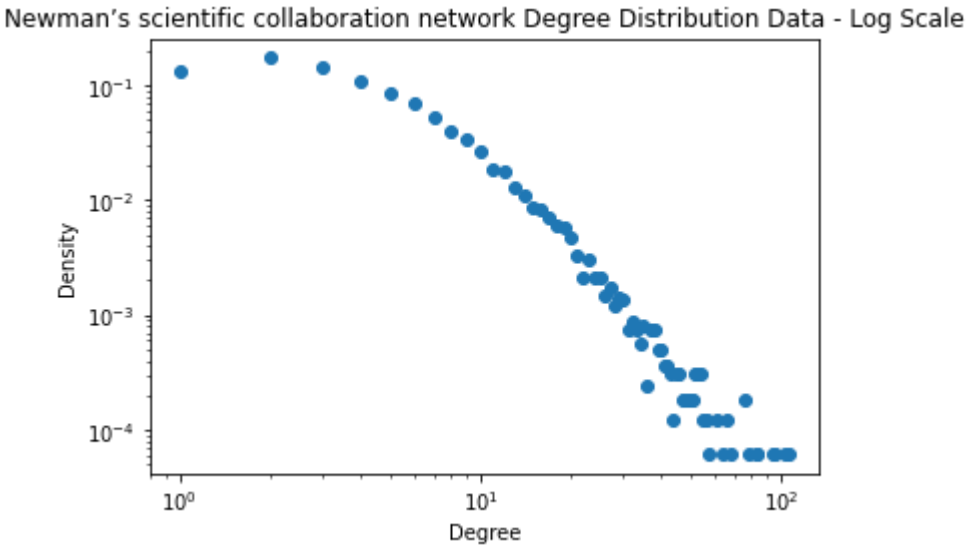
```
#degree distribution on linear scale
degreeDistrGLinearLogScale(scientific_collab_graph_data, False)
```



.

.

```
#degree distribution on log scale
degreeDistrGLinearLogScale(scientific_collab_graph_data, True)
```



6 (c) Generate 10 Erd os-R enyi random graphs Gn;p and use them to construct an empirical degree distribution.

Generate two scatterplots of this distribution, one with linear axes and one with logarithmic axes. Depending on the size of the network you choose, it may take a significant amount of time to generate these graphs.

```
#Function to generate 10 Erd os-R enyi random graphs Gnp and use them
#to construct an empirical degree distribution.

def degreeDistrTenGraph(G,degreeList,linearLog):
  degrees = {}
  for deg in degreeList:
    degrees[deg] = degrees.get(deg, 0) + 1
  (X, Y) = zip(*[(key, degrees[key]/(len(G) *10)) for key in degrees])
  plt.scatter(X, Y)
  plt.title('Newman's scientific collaboration network Degree Distribution Data - Li
  if linearLog:
      plt.yscale('log')
      plt.xscale('log')
      plt.title('Newman's scientific collaboration network Degree Distribution Data
  plt.xlabel('Degree')
  plt.ylabel('Density')
  plt.show()
```

.

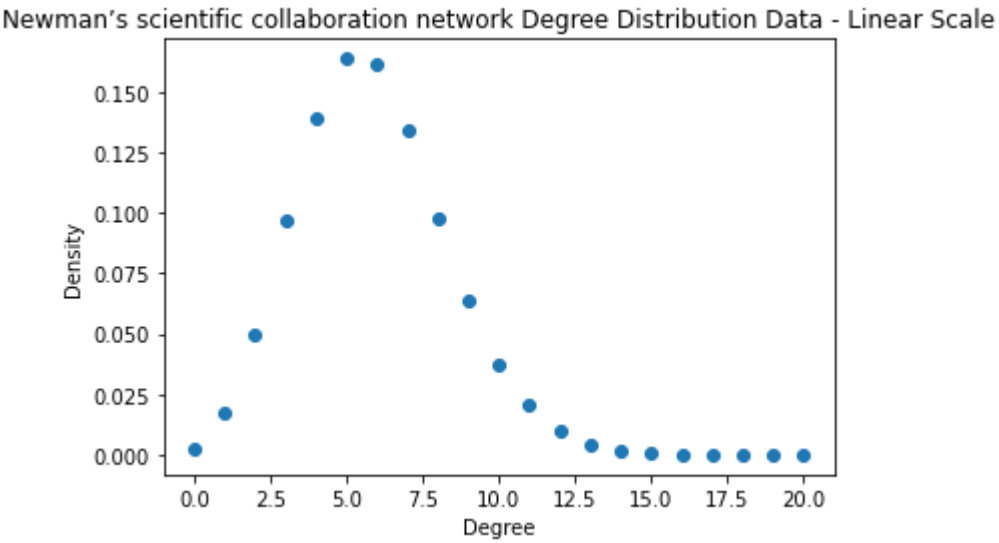
```
degreeListFinal = []
#function to calculate the degrees for 10 random graph
def calculateDegrees(n,p):
  scientific_collab_graph_data_1 = randomGraphGenerator(n, p)
  degreeList = [scientific_collab_graph_data_1.degree(v) for v in scientific_collab_
  degreeListFinal.extend(degreeList)
```

.

```
G = nx.read_edgelist('/content/Newman-Cond_mat_95-99-co_occurrence.txt', data=[('wei
n = G.number_of_nodes()
p = G.number_of_edges()/comb(n, 2)
for x in range(10):
  calculateDegrees(n,p)

```

.

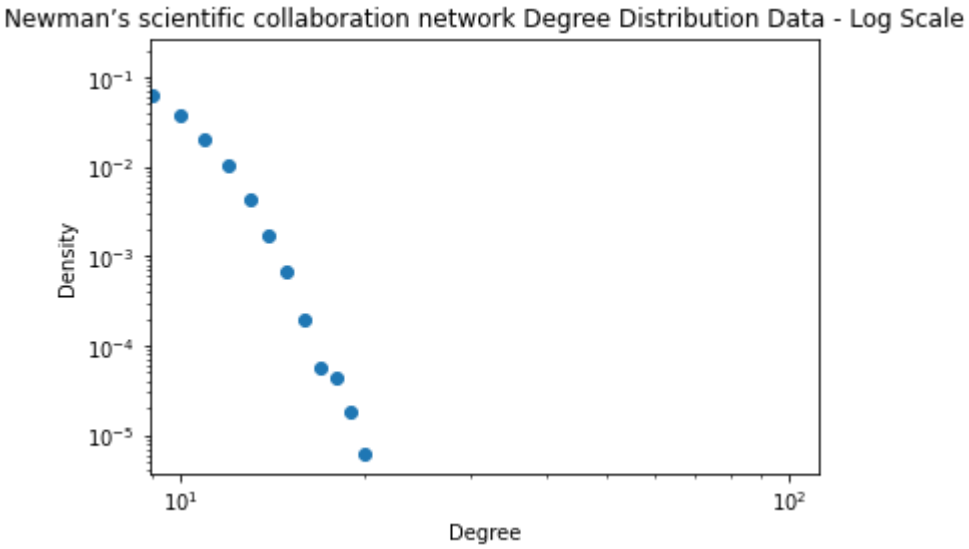
```
#degree distribution on linear scale
degreeDistrTenGraph(G,degreeListFinal, False)

```



.

.

```
#degree distribution on log scale
degreeDistrTenGraph(G,degreeListFinal, True)
```



6 (d) What is the expected and approximate degree distribution for G(n,p) in this case?

```
#Function to calculate expected (Binomial) and approximate degree distribution
#(Poisson approximation) for G(n,p) in this case
def expectedApproxDegreeDistribution(n, p):
  X = range(int(n*p-3*np.sqrt(n*p*(1-p))), int(n*p+3*np.sqrt(n*p*(1-p))), 1)
  print('Expected Degree distribution ',[binom.pmf(x, n, p)for x in X])
  print('Approximate Degree distribution ',[poisson.pmf(x, n*p)for x in X])
  plt.plot(X, [binom.pmf(x, n, p) for x in X], 'b-', label='Binom')
  plt.plot(X, [poisson.pmf(x, n*p) for x in X], 'r-', label='Poisson')
  plt.title('n: '+str(n)+', p: '+str(p))
  plt.ylabel('P(X = k)')
  plt.xlabel('k')
  plt.legend(loc='upper right')
  plt.show()

```

.

.

.

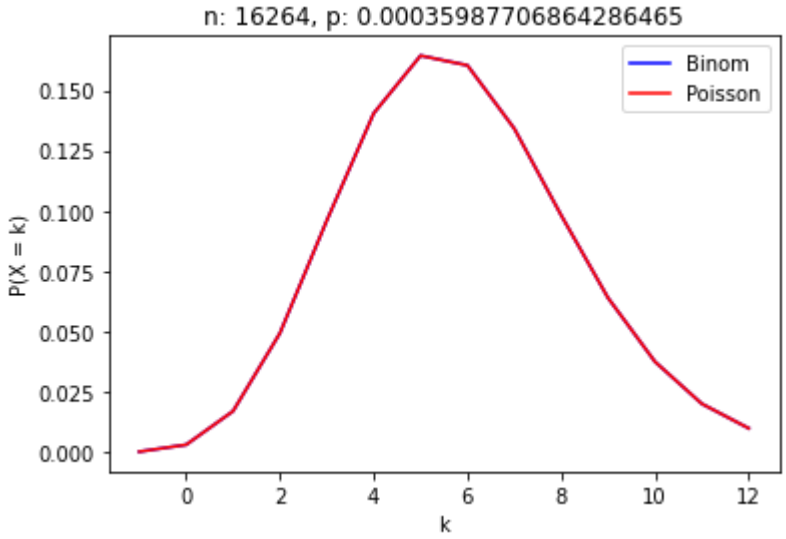
.

.

Calculate Expected and approximate degree distribution

```
1 expectedApproxDegreeDistribution(n, p)
```

```
Expected Degree distribution  [0.0, 0.0028681327098338885, 0.016793340862041702, 0.04916
Approximate Degree distribution  [0.0, 0.002871155708416042, 0.01680499105778185, 0.0491
```



In this case n tends to large(infinity) and p tends to 0 while np remains constant, the binomial distribution tends to the Poisson distribution.

Here When λ is very large i.e., $\lambda \to \infty$ poisson distribution tends to normal distribution

.

.

.

.

.

.

..

.

6 (e) Does this match the degree distribution of the real world network you chose? Why or why not? Discuss any discrepancies in the context of the network domain.

The figure in question 6b, the real world degree distribution doesn't match clearly and there is huge deviation in terms of networks with higher degree distribution value with the 10 random graphs degree distribution.

I can see degree distribution ranging from 0-100 for real world data (figure for 6b) and whereas distribution is limited to 0-20 for 10 random graphs degree distribution (figure 6c).

The real world networks (figure for 6b) represents connected regime where networks are part of single giant component with distant coupling for few networks. Random graph distribution (figure for 6c) represents subcritical regime as expected from random graph distribution compared to real world data.

The real world networks usually have very different degree distributions. In a real world network, most nodes have a relatively small degree, but a few nodes will have very large degree, being connected to many other nodes.

While in (c) the (Erdős–Rényi model) random graph, in which each of n nodes is independently connected (or not) with probability p (or 1 − p), has a binomial distribution of degrees k. Most networks in the real world, however, have degree distributions very different from this.