Write a Python code to generate the scale-free networks of size N and degree exponent gamma

In [1]:

```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import random
```

(a)

In [2]:

```
gamma = 2.2
```

In [3]:

```
N1 = 10**3
N2 = 2*10**3
N3 = 5*10**3
N4 = 10**4
N5 = 2*10**4
N6 = 5*10**4
N7 = 10**5
```

In [4]:

```
#power law degree sequence
def PLds(N) :
    v = np.zeros(N)
    i = 0
    while i < N :
        r1=random.randint(0,N)
        if r1
        val=r1**(-gamma)
        r2=random.randint(0,N)
        if r2 < val :
            v[i] = r1
            i +=1
    return v
```

```
  File "C:₩Users₩LG₩AppData₩Local₩Temp₩ipykernel_7352₩2713376037.py", l
ine 7
    if r1
         ^
SyntaxError: invalid syntax
```

In [6]:

```
def powerlaw_graph(N, gamma, seed):
    sequence = np.array(nx.utils.random_sequence.powerlaw_sequence(N, gamma, seed=seed)).astype(i
    new_seed = seed + 1
    while not nx.is_graphical(sequence):
        sequence = np.array(nx.utils.random_sequence.powerlaw_sequence(N, gamma, seed=new_seed)).
        new_seed += 1
    return nx.configuration_model(sequence, seed=seed)
```

In [8]:

```
G1 = powerlaw_graph(N1, gamma, 0)
G2 = powerlaw_graph(N2, gamma, 0)
G3 = powerlaw_graph(N3, gamma, 0)
G4 = powerlaw_graph(N4, gamma, 0)
G5 = powerlaw_graph(N5, gamma, 0)
G6 = powerlaw_graph(N6, gamma, 0)
G7 = powerlaw_graph(N7, gamma, 0)
```

In [23]:

```
#self loop
print(nx.number_of_selfloops(G1)/G1.number_of_edges())
print(nx.number_of_selfloops(G2)/G2.number_of_edges())
print(nx.number_of_selfloops(G3)/G3.number_of_edges())
print(nx.number_of_selfloops(G4)/G4.number_of_edges())
print(nx.number_of_selfloops(G5)/G5.number_of_edges())
print(nx.number_of_selfloops(G6)/G6.number_of_edges())
print(nx.number_of_selfloops(G7)/G7.number_of_edges())
```

```
0.012035010940919038
0.0076263107721639654
0.01845778354053848
0.006736823565673022
0.0039557132107922175
0.0010097977675281788
0.0033321363897865965
```

In [62]:

```
#multi edges
def number_of_multiedges(G) :
    edges = list(G.edges())
    edges_set = list(set(edges))
    edge_dist = []
    multi_edges = 0

    for i in edges_set :
        edge_dist.append(edges.count(i))
    for j in range(len(edge_dist)) :
        if edge_dist[j] >1 :
            multi_edges += edge_dist[j]

    return multi_edges
```

In [65]:

```
print(number_of_multiedges(G1)/G1.number_of_edges())
print(number_of_multiedges(G2)/G2.number_of_edges())
print(number_of_multiedges(G3)/G3.number_of_edges())
print(number_of_multiedges(G4)/G4.number_of_edges())
print(number_of_multiedges(G5)/G5.number_of_edges())
print(number_of_multiedges(G6)/G6.number_of_edges())
print(number_of_multiedges(G7)/G7.number_of_edges())
```

```
0.14442013129102846
0.15800762631077217
0.2986804972923499
0.18193826779974462
0.1345802429323874
0.06598255142235929
0.1476796670180275
```

(b)

In [66]:

```
gamma = 3
```

In [67]:

```
G1 = powerlaw_graph(N1, gamma, 0)
G2 = powerlaw_graph(N2, gamma, 0)
G3 = powerlaw_graph(N3, gamma, 0)
G4 = powerlaw_graph(N4, gamma, 0)
G5 = powerlaw_graph(N5, gamma, 0)
G6 = powerlaw_graph(N6, gamma, 0)
G7 = powerlaw_graph(N7, gamma, 0)
```

In [68]:

```
#self loop
print(nx.number_of_selfloops(G1)/G1.number_of_edges())
print(nx.number_of_selfloops(G2)/G2.number_of_edges())
print(nx.number_of_selfloops(G3)/G3.number_of_edges())
print(nx.number_of_selfloops(G4)/G4.number_of_edges())
print(nx.number_of_selfloops(G5)/G5.number_of_edges())
print(nx.number_of_selfloops(G6)/G6.number_of_edges())
print(nx.number_of_selfloops(G7)/G7.number_of_edges())
```

```
0.002457002457002457
0.0018610421836228288
0.00047869794159885111
0.00036297640653357535
0.00012208521548040532
0.00021539345203905802
2.4514009756575884e-05
```

In [69]:

```
print(number_of_multiedges(G1)/G1.number_of_edges())
print(number_of_multiedges(G2)/G2.number_of_edges())
print(number_of_multiedges(G3)/G3.number_of_edges())
print(number_of_multiedges(G4)/G4.number_of_edges())
print(number_of_multiedges(G5)/G5.number_of_edges())
print(number_of_multiedges(G6)/G6.number_of_edges())
print(number_of_multiedges(G7)/G7.number_of_edges())
```

```
0.0
0.0012406947890818859
0.00478697941598851l
0.0016938898971566847
0.00036625564644l216
0.0019146084625694046
9.805603902630354e-05
```

In [ ]: