ÖNSZERV HF3

```
%pylab inline
from scipy.optimize import curve_fit # Az illesztéshez használt
függvény
from numpy.fft import *              # Fourier-analízishez használt
rutinok
from scipy.signal import spectrogram # Spektrogramm készítő függvény
from scipy.interpolate import interp1d # Interpoláció
import scipy.integrate as integrate # Integrálás
import random
import imageio
```

```
Populating the interactive namespace from numpy and matplotlib
```

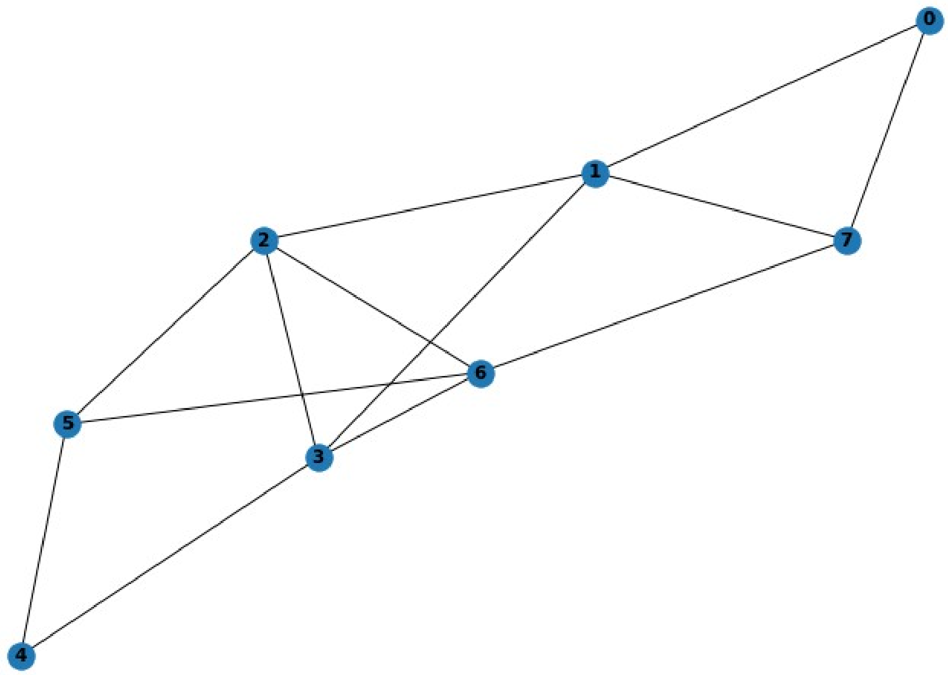
```
import networkx as nx
```

Készítsünk egy számítógépes szimulációt az alábbiak szerint!

1. feladat

Induljunk ki egy tetszőleges, néhány csomópontból álló összekötött hálózatból.

```
def nen(k):
    k1 = int(rand()*k)
    k2 = k1
    while (k1==k2):
        k1 = int(rand()*k)
    return k1,k2

g = nx.Graph()
for i in range(7):
    g.add_node(i) # node-ok
    g.add_edge(i,i+1) # kezdő edge-ek
g.add_edge(0, 7)
# generáljunk ki pár edge-et random módon
for i in range(6):
    n1, n2 = nen(8)
    g.add_edge(n1,n2)

#g.add_edges_from([(1, 2), (1, 3)])

figsize(10,7)
nx.draw(g, with_labels=True, font_weight='bold')
```

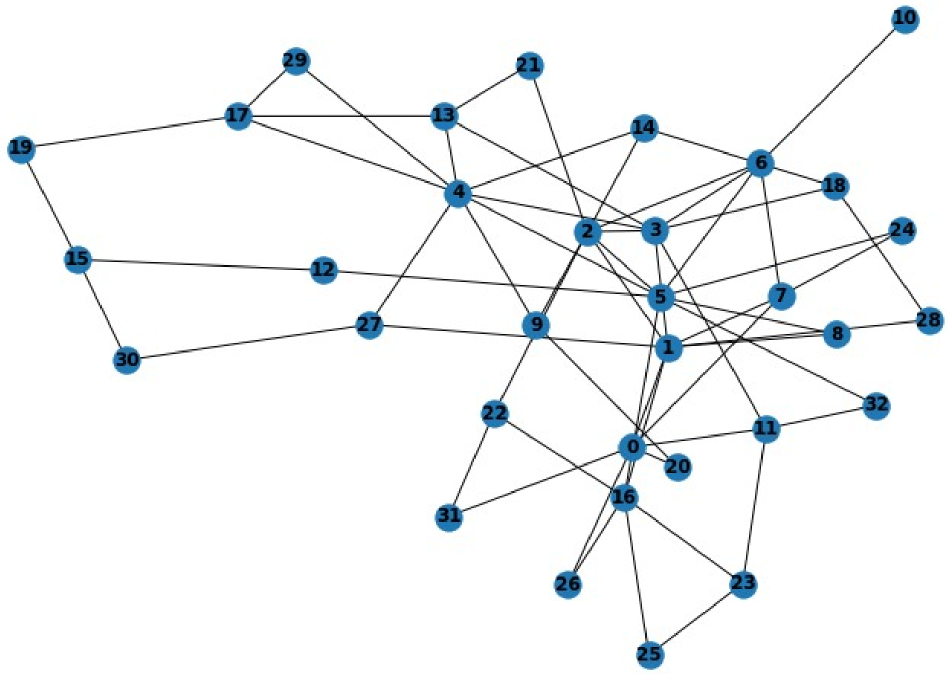


2. feladat

Minden egyes lépésben adjunk hozzá egy újabb csomópontot mely egy-egy éllel kapcsolódik *m* véletlenszerűen választott régi csúcshoz.

```
g2 = nx.Graph()
g2.add_nodes_from(g.nodes)
g2.add_edges_from(g.edges)

n = 25
m = 2

nodes = 0
g2.clear()
g2.add_nodes_from(g.nodes)
g2.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g2.nodes)[-1]+1 # az utolsó node sorszáma
    g2.add_node(nodes)
    #print(nodes)
    for j in range(m):
        mibe = nodes
        honnan = int(rand()*(nodes-1)) # random kiválasszuk hova
menjen az él, nem beleértve az újabb node-ot
        g2.add_edge(mibe,honnan)
```

```
figsize(10,7)
nx.draw(g2, with_labels=True, font_weight='bold')
```

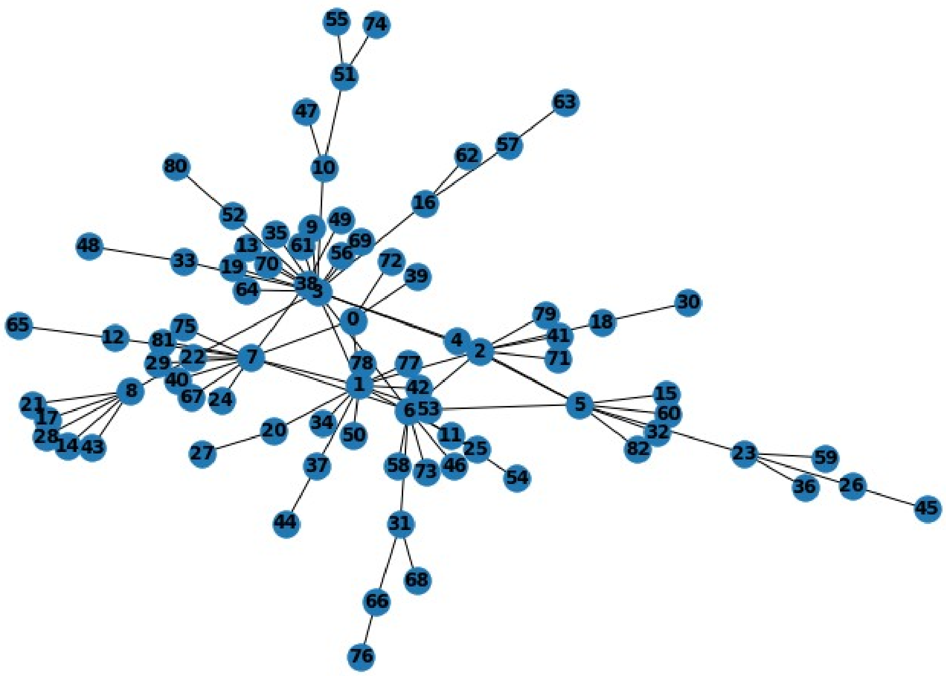


3. feladat

Ehhez úgy válasszunk véletlenszerűen csúcsokat, hogy a kiválasztás valószínűsége arányos legyen a régi csúcsok pillanatnyi fokszámával. (Azt, hogy a nagyobb fokszámú csúcs nagyobb eséllyel kap új élt, preferenciális kapcsolódásnak hívják.)

```
g3 = nx.Graph()
g3.add_nodes_from(g.nodes)
g3.add_edges_from(g.edges)

n = 75
m = 1

nodes = 0
g3.clear()
g3.add_nodes_from(g.nodes)
g3.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g3.nodes)[-1]+1
    g3.add_node(nodes)
    for j in range(m):
        mibe = nodes
```

```
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g3.nodes)[0:-2:1], weights =
ravel(list(g3.degree))[1:-3:2], k=1)[0]
        g3.add_edge(mibe,honnan)

figsize(10,7)
nx.draw(g3, with_labels=True, font_weight='bold')
```



```
random.choices(list(g2.nodes), weights = ravel(list(g2.degree))[0:-
1:2], k=1)[0]
```

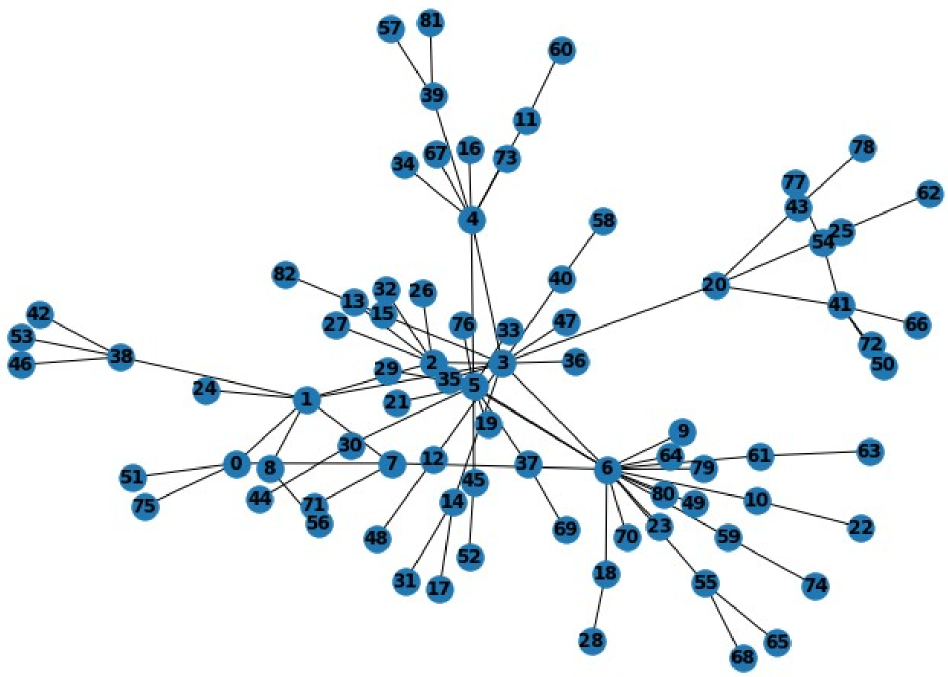
```
15
```

4. feladat

Próbálj ki különböző m-eket (pl. $m=1$, $m=2$, ...)

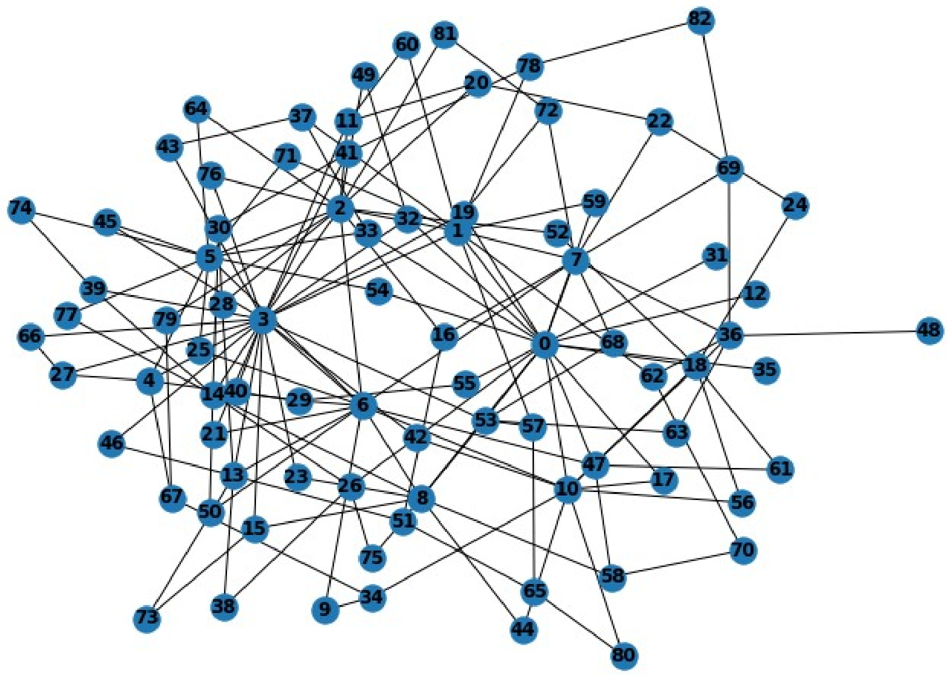
```
g4 = nx.Graph()
g4.add_nodes_from(g.nodes)
g4.add_edges_from(g.edges)

n = 75
m = 1

nodes = 0
g4.clear()
g4.add_nodes_from(g.nodes)
```

```
g4.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g4.nodes)[-1]+1
    g4.add_node(nodes)
    for j in range(m):
        mibe = nodes
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g4.nodes)[0:-2:1], weights =
ravel(list(g4.degree))[1:-3:2], k=1)[0]
        g4.add_edge(mibe,honnan)

figsize(10,7)
nx.draw(g4, with_labels=True, font_weight='bold')
```



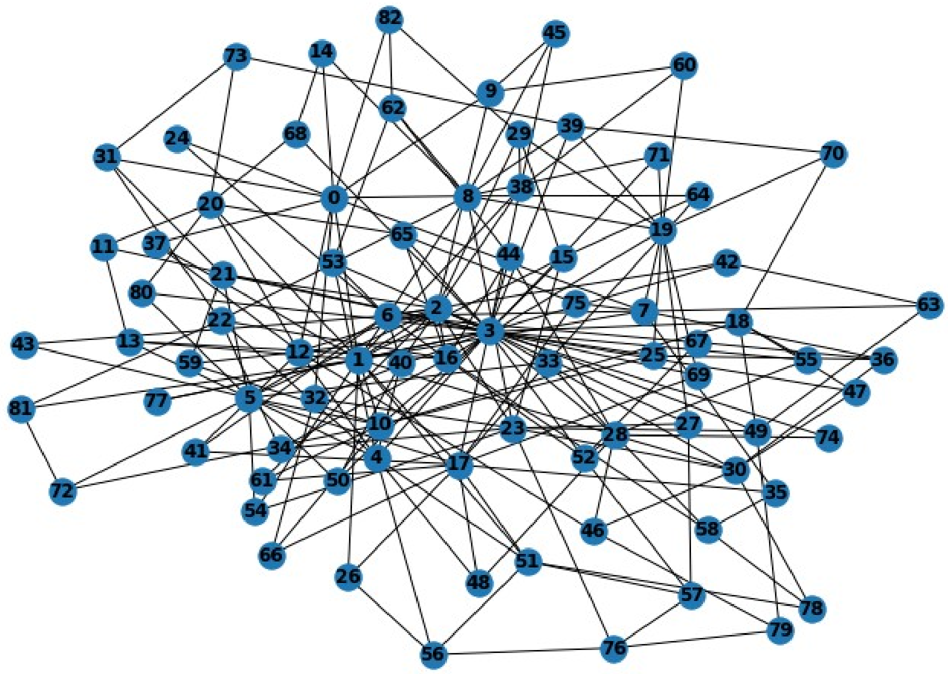

```
n = 75
m = 2

nodes = 0
g4.clear()
g4.add_nodes_from(g.nodes)
g4.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g4.nodes)[-1]+1
    g4.add_node(nodes)
    for j in range(m):
```

```
        mibe = nodes
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g4.nodes)[0:-2:1], weights =
ravel(list(g4.degree))[1:-3:2], k=1)[0]
        g4.add_edge(mibe,honnan)

figsize(10,7)
nx.draw(g4, with_labels=True, font_weight='bold')
```

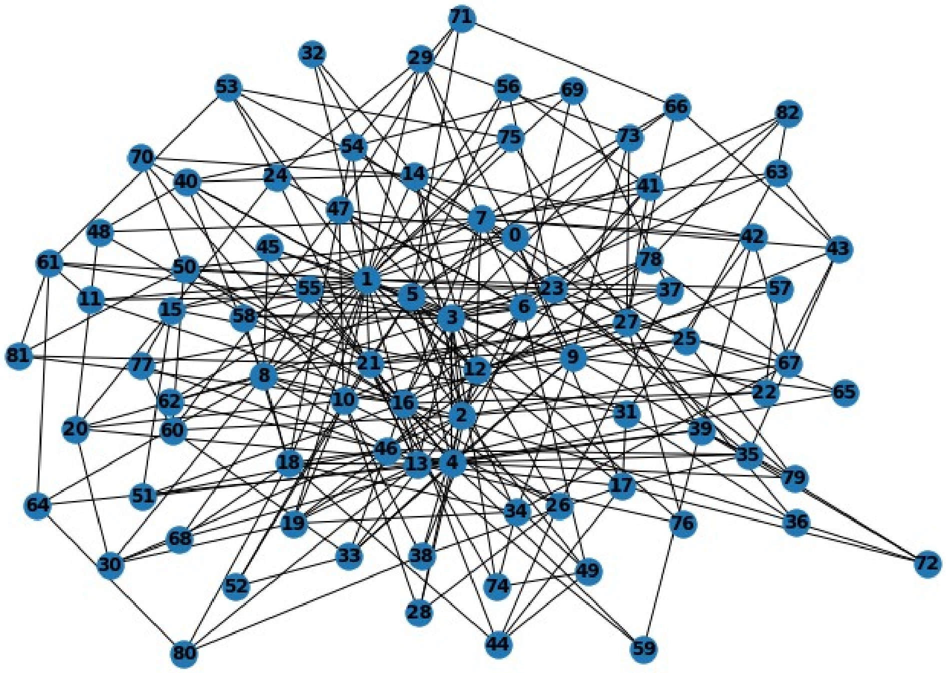


```
n = 75
m = 3

nodes = 0
g4.clear()
g4.add_nodes_from(g.nodes)
g4.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g4.nodes)[-1]+1
    g4.add_node(nodes)
    for j in range(m):
        mibe = nodes
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g4.nodes)[0:-2:1], weights =
ravel(list(g4.degree))[1:-3:2], k=1)[0]
        g4.add_edge(mibe,honnan)
```

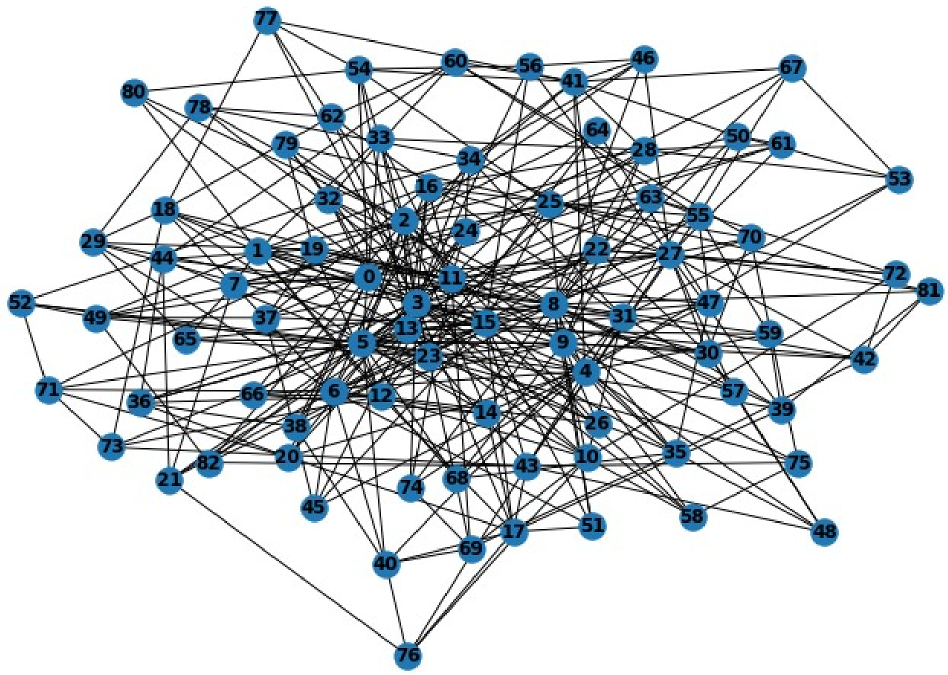
```
figsize(10,7)
nx.draw(g4, with_labels=True, font_weight='bold')
```



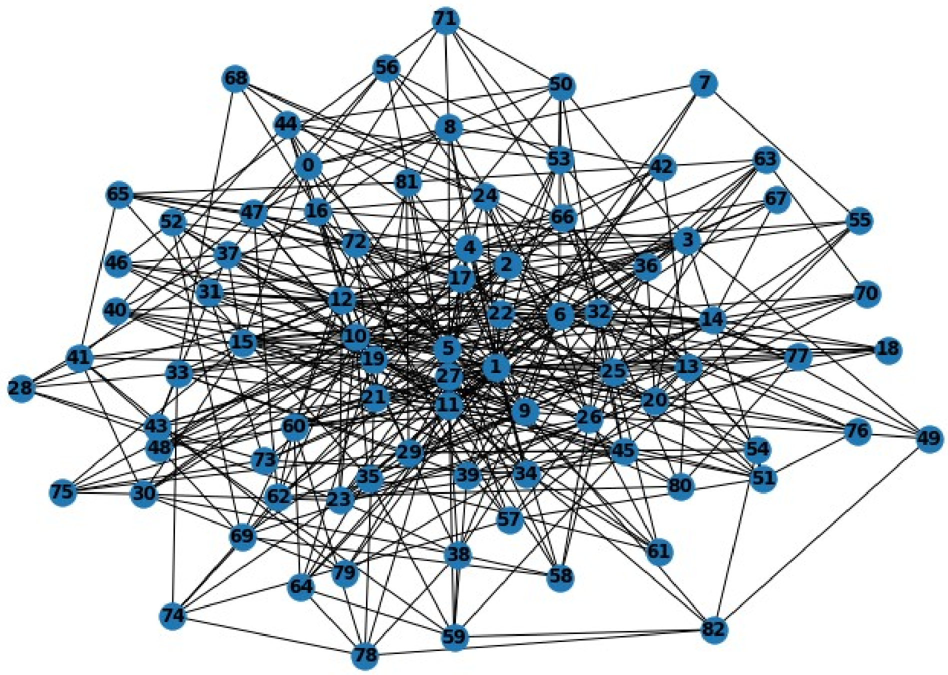

```
n = 75
m = 4

nodes = 0
g4.clear()
g4.add_nodes_from(g.nodes)
g4.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g4.nodes)[-1]+1
    g4.add_node(nodes)
    for j in range(m):
        mibe = nodes
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g4.nodes)[0:-2:1], weights =
ravel(list(g4.degree))[1:-3:2], k=1)[0]
        g4.add_edge(mibe,honnan)

figsize(10,7)
nx.draw(g4, with_labels=True, font_weight='bold')
```




```
n = 75
m = 5

nodes = 0
g4.clear()
g4.add_nodes_from(g.nodes)
g4.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g4.nodes)[-1]+1
    g4.add_node(nodes)
    for j in range(m):
        mibe = nodes
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g4.nodes)[0:-2:1], weights =
ravel(list(g4.degree))[1:-3:2], k=1)[0]
        g4.add_edge(mibe,honnan)

figsize(10,7)
nx.draw(g4, with_labels=True, font_weight='bold')
```



```
n = 75
m = 6

nodes = 0
g4.clear()
g4.add_nodes_from(g.nodes)
g4.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g4.nodes)[-1]+1
    g4.add_node(nodes)
    for j in range(m):
        mibe = nodes
#         honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g4.nodes)[0:-2:1], weights =
ravel(list(g4.degree))[1:-3:2], k=1)[0]
        g4.add_edge(mibe,honnan)

figsize(10,7)
nx.draw(g4, with_labels=True, font_weight='bold')
```



Megjegyzés:

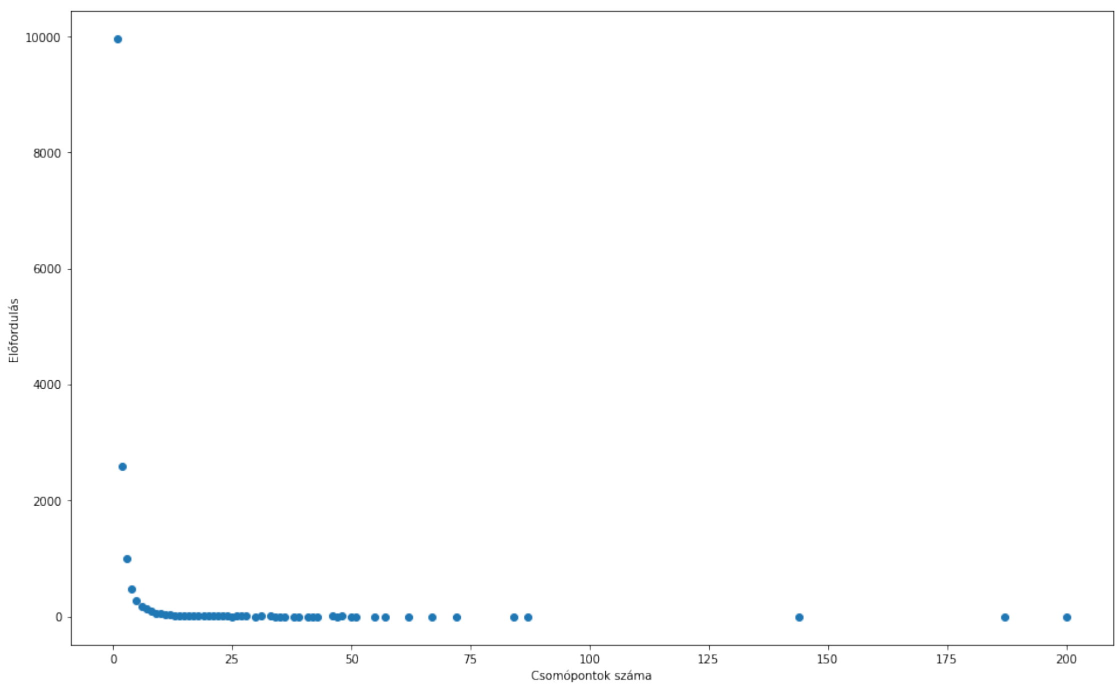
Azt várjuk, hogy $m=1$ esetén, minél régebb óta van bent egy csomópont, annál több él fog kapcsolódni hozzá - és persze az új élek erősen háttérbe szorulnak. Ám a kiinduló csomópontok $m>1$ esetén hátrányba is kerülhetnek, ha m-nél kevesebb volt a fokszámuk kiinduló helyzetben.

5. feladat

Növesszünk néhány nagy (10.000 vagy 100.000 csomópontból álló) hálózatot és készítsd el a csomópontok fokszámainak eloszlását.

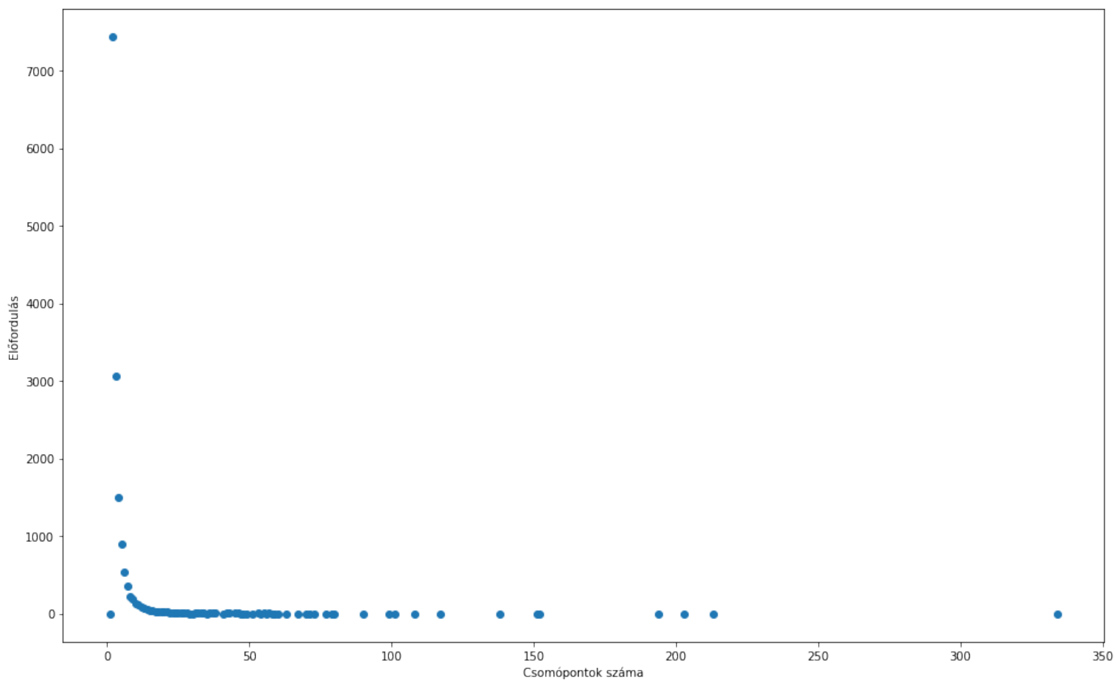
```
g5 = nx.Graph()
g5.add_nodes_from(g.nodes)
g5.add_edges_from(g.edges)

n = 15000
m = 1

nodes = 0
g5.clear()
g5.add_nodes_from(g.nodes)
g5.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g5.nodes)[-1]+1
```

```
    g5.add_node(nodes)
    for j in range(m):
        mibe = nodes
#        honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g5.nodes)[0:-2:1], weights =
ravel(list(g5.degree))[1:-3:2], k=1)[0]
        g5.add_edge(mibe,honnan)

val1 = list(ravel(list(g5.degree))[1:-3:2])
values1 = list(dict.fromkeys(val1))
valno1 = []
for i in values1:
    valno1.append(val1.count(i))
figsize(16,10)
plt.plot(values1, valno1,"o")
plt.xlabel("Csomópontok száma")
plt.ylabel("Előfordulás")
plt.show()
```




```
n = 15000
m = 2

nodes = 0
g5.clear()
g5.add_nodes_from(g.nodes)
g5.add_edges_from(g.edges)

for i in range(n):
    nodes = list(g5.nodes)[-1]+1
    g5.add_node(nodes)
```

```
    for j in range(m):
        mibe = nodes
#         honnan = int(rand()*(nodes-1))
        honnan = random.choices(list(g5.nodes)[0:-2:1], weights = 
ravel(list(g5.degree))[1:-3:2], k=1)[0]
        g5.add_edge(mibe,honnan)

val2 = list(ravel(list(g5.degree))[1:-3:2])
values2 = list(dict.fromkeys(val2))
valno2 = []
for i in values2:
    valno2.append(val2.count(i))
figsize(16,10)
plt.plot(values2, valno2,"o")
plt.xlabel("Csomópontok száma")
plt.ylabel("Előfordulás")
plt.show()
```

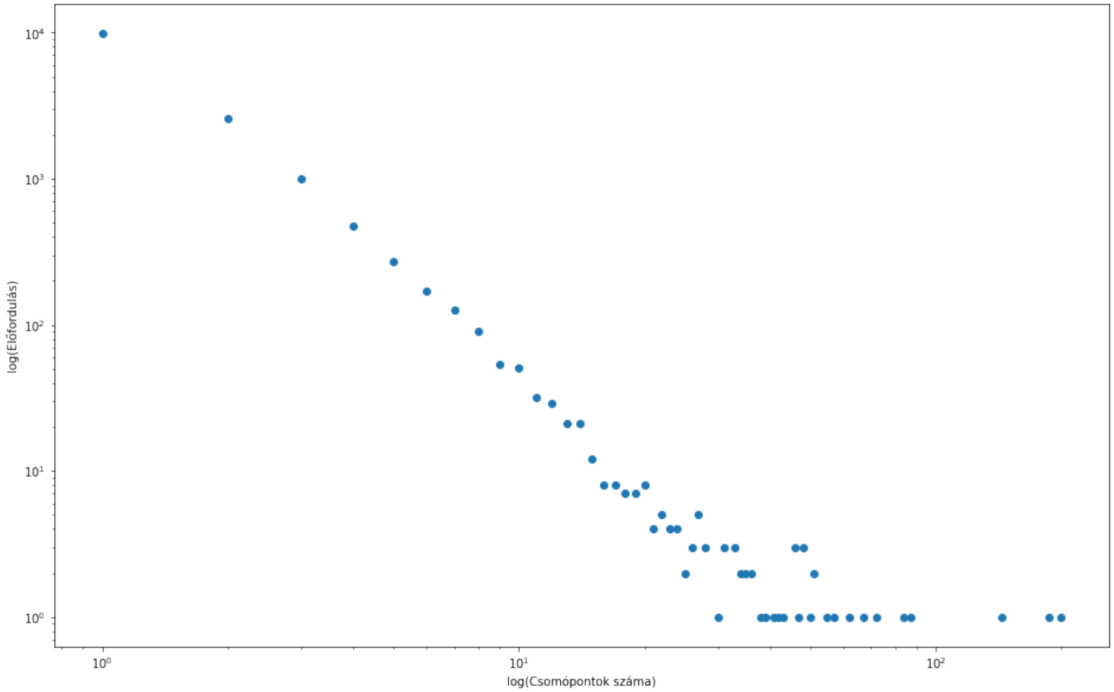


6. feladat

Ábrázold log-log ploton és állapítsd meg a kapott eloszlás hatványkitevőjét.

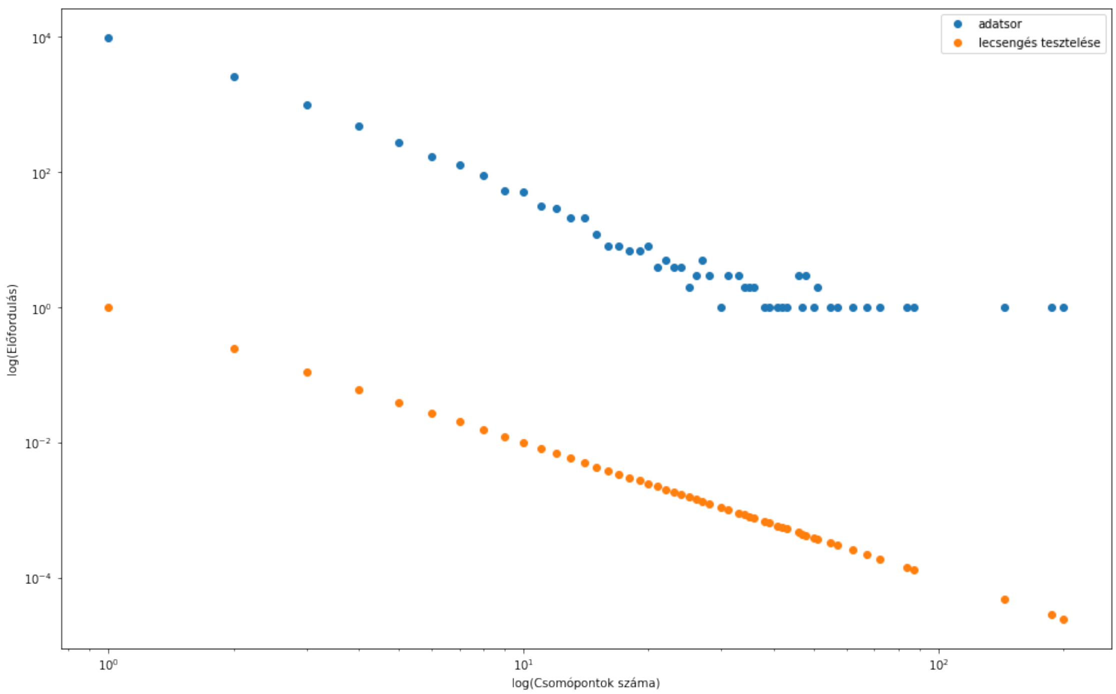
```
# m = 1

figsize(16,10)
plt.yscale("log")
plt.xscale("log")
plt.plot(values1, valno1,"o")
```

```
plt.xlabel("log(Csomópontok száma)")
plt.ylabel("log(Előfordulás)")
plt.show()
```

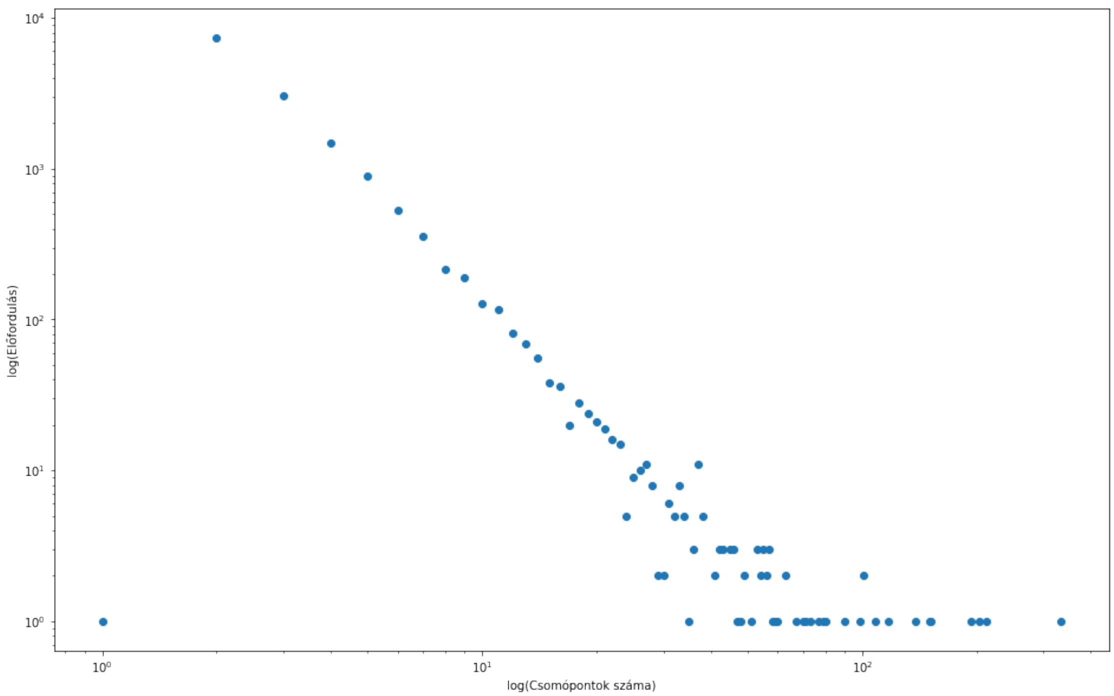


$m=1$ esetén $k=-2$-t kapunk a meredekségre. Ez azt jelenti, hogy az adatsor nagyjából x^{-2}-es viselkedést mutat.

```
figsize(16,10)
plt.yscale("log")
plt.xscale("log")
plt.plot(values1, valno1,"o", label = "adatsor")
plt.plot(values1, 1/(np.array(values1)**(2)),"o", label = "lecsengés
tesztelése")
plt.xlabel("log(Csomópontok száma)")
plt.ylabel("log(Előfordulás)")
plt.legend()
plt.show()
```

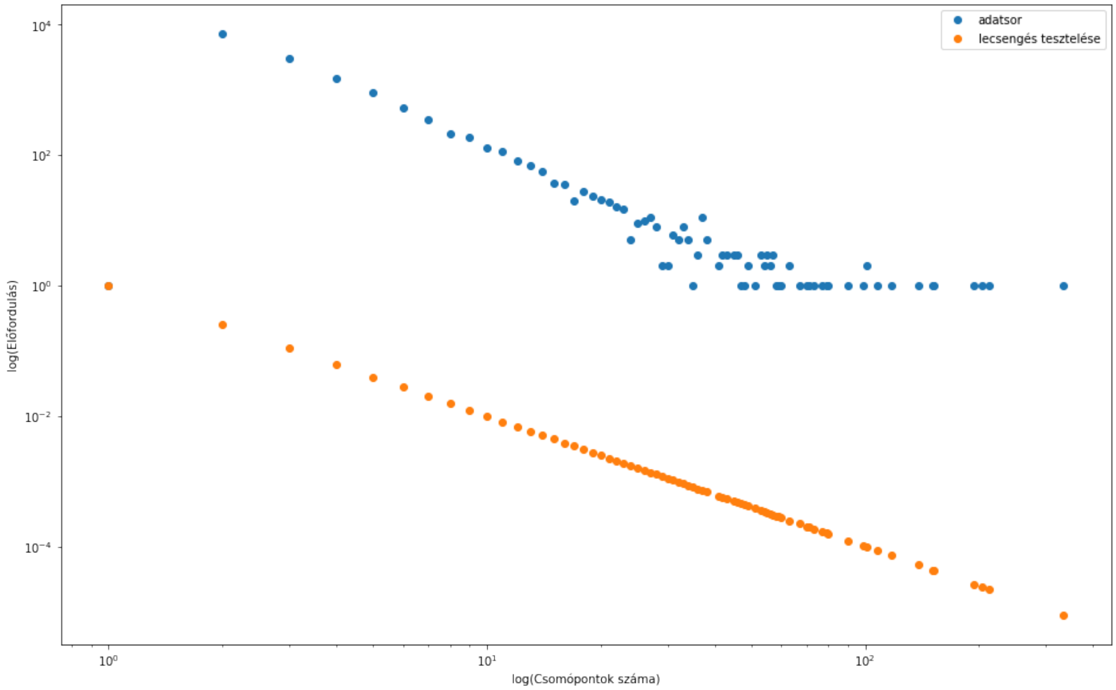



```
# m = 2

figsize(16,10)
plt.yscale("log")
plt.xscale("log")
plt.plot(values2, valno2,"o")
plt.xlabel("log(Csomópontok száma)")
plt.ylabel("log(Előfordulás)")
plt.show()
```



$m=2$ esetén is nagyjából $k=-2$-t kapunk a meredekségre. Ez azt jelenti, hogy az adatsor nagyjából x^{-2}-es viselkedést mutat.

```
figsize(16,10)
plt.yscale("log")
plt.xscale("log")
plt.plot(values2, valno2,"o", label = "adatsor")
plt.plot(values2, 1/(np.array(values2)**(2)),"o", label = "lecsengés
tesztelése")
plt.xlabel("log(Csomópontok száma)")
plt.ylabel("log(Előfordulás)")
plt.legend()
plt.show()
```


adatsor
lecsengés tesztelése
log(Előfordulás)
log(Csomópontok száma)
10^4
10^2
10^0
10^{-2}
10^{-4}
10^0
10^1
10^2