1. Create a network

- **Generate network objects for the companies organizational structure (reports to), friendship, advice**
- **This networks are generated from the corresponding edgelists**
- **Also attach node characteristics from the corresponding nodelist**

In [71]:

```
# Importing the libraries i use
import pandas as pd
import networkx as nx
from community import community_louvain
import csv
import matplotlib.pyplot as plt
import seaborn as sns
```

In [10]:

```
# Import, read files and take a look at it
data = pd.read_csv('https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackhard/Krack-High-Tec-Attributes.csv')
edgeadvice = pd.read_csv('https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackhard/Krack-High-Tec-edgelist-Advice.txt',sep=' ')
edgefriend = pd.read_csv('https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackhard/Krack-High-Tec-edgelist-Friendship.txt',sep=' ')
edgereport= pd.read_csv('https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackhard/Krack-High-Tec-edgelist-ReportsTo.txt', sep=' ')
print(data.head())
print(edgeadvice.head())
print(edgefriend.head())
print(edgereport.head())
```

```
   ID  AGE  TENURE  LEVEL  DEPT
0   1   33   9.333      3     4
1   2   42  19.583      2     4
2   3   40  12.750      3     2
3   4   33   7.500      3     4
4   5   32   3.333      3     2
   Unnamed: 0  1  1.1  0
0         NaN  1    2  1
1         NaN  1    3  0
2         NaN  1    4  1
3         NaN  1    5  0
4         NaN  1    6  0
   Unnamed: 0  1  1.1  0
0         NaN  1    2  1
1         NaN  1    3  0
2         NaN  1    4  1
3         NaN  1    5  0
4         NaN  1    6  0
   Unnamed: 0  1  1.1  0
0         NaN  1    2  1
1         NaN  1    3  0
2         NaN  1    4  0
3         NaN  1    5  0
4         NaN  1    6  0
```

In [11]:

```
#clean the columns in the advice edgelist
edgeadvice.columns = ['NAN', 'IDfrom','IDto', 'weight']
edgeadvice=edgeadvice.drop('NAN',axis=1)
edgeadvice = edgeadvice[edgeadvice.weight != 0]
print(edgeadvice.head())

#clean the columns in the Friend edgelist
```

```
edgefriend.columns = ['NAN', 'IDfrom','IDto', 'weight']
edgefriend=edgefriend.drop('NAN',axis=1)
edgefriend = edgefriend[edgefriend.weight != 0]
print(edgefriend.head())

#clean the columns in the report to edgelist
edgereport.columns = ['NAN', 'IDfrom','IDto', 'weight']
edgereport=edgereport.drop('NAN',axis=1)
edgereport = edgereport[edgereport.weight != 0]
print(edgereport.head())
```

```
    IDfrom  IDto  weight
0        1     2       1
2        1     4       1
6        1     8       1
14       1    16       1
16       1    18       1
    IDfrom  IDto  weight
0        1     2       1
2        1     4       1
6        1     8       1
10       1    12       1
14       1    16       1
    IDfrom  IDto  weight
0        1     2       1
26       2     7       1
54       3    14       1
63       4     2       1
96       5    14       1
```

In [12]:

```
#generate the networks from the edgelist
A = nx.from_pandas_edgelist(edgeadvice, source='IDfrom', target='IDto', edge_attr='weigh
t', create_using=nx.DiGraph())
F = nx.from_pandas_edgelist(edgefriend, source='IDfrom', target='IDto', edge_attr='weigh
t', create_using=nx.DiGraph())
R = nx.from_pandas_edgelist(edgereport, source='IDfrom', target='IDto', edge_attr='weigh
t', create_using=nx.DiGraph())
```

In [13]:

```
#check the networks have been generoated correctly
print(nx.info(A))
print(nx.info(F))
print(nx.info(R))
```

```
DiGraph with 21 nodes and 190 edges
DiGraph with 21 nodes and 102 edges
DiGraph with 21 nodes and 20 edges
```

In [14]:

```
#replace coded value with text string for manager level
cleanup_nums = {"LEVEL":     {1: "CEO", 2: "Vice President", 3: "Manager"}
                }
data.replace(cleanup_nums, inplace=True)
data.head(21)
```

Out[14]:

	ID	AGE	TENURE	LEVEL	DEPT
0	1	33	9.333	Manager	4
1	2	42	19.583	Vice President	4
2	3	40	12.750	Manager	2
3	4	33	7.500	Manager	4
4	5	32	3.333	Manager	2
5	6	59	28.000	Manager	1

	ID	AGE	TENURE	LEVEL	DEPT
6	7	55	30.000	CEO	0
7	8	34	11.333	Manager	1
8	9	62	5.417	Manager	2
9	10	37	9.250	Manager	3
10	11	46	27.000	Manager	3
11	12	34	8.917	Manager	1
12	13	48	0.250	Manager	2
13	14	43	10.417	Vice President	2
14	15	40	8.417	Manager	2
15	16	27	4.667	Manager	4
16	17	30	12.417	Manager	1
17	18	33	9.083	Vice President	3
18	19	32	4.833	Manager	2
19	20	38	11.667	Manager	2
20	21	36	12.500	Vice President	1

In [15]:

```
#set the dataframe called "data" to be a diction list but first transposed it so it work with the edge list
data_dict=data.T.to_dict()
```

In [17]:

```
#sets the atrributes of the networks with the information stored in the data_dict dataframe
nx.set_node_attributes(A, data_dict)
nx.set_node_attributes(F, data_dict)
nx.set_node_attributes(R, data_dict)
```

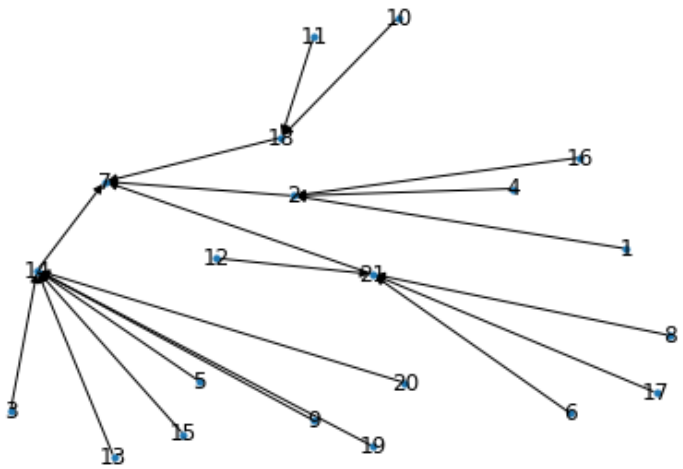
In [18]:

```
#check that the labels have being assigned correctly.
print(nx.get_node_attributes(A, 'LEVEL'))
```

```
{1: 'Vice President', 2: 'Manager', 4: 'Manager', 8: 'Manager', 16: 'Manager', 18: 'Manager', 6: 'CEO', 7: 'Manager', 3: 'Manager', 9: 'Manager', 10: 'Manager', 11: 'Manager', 12: 'Manager', 14: 'Manager', 17: 'Vice President', 20: 'Vice President', 5: 'Manager', 13: 'Vice President', 19: 'Manager', 15: 'Manager'}
```

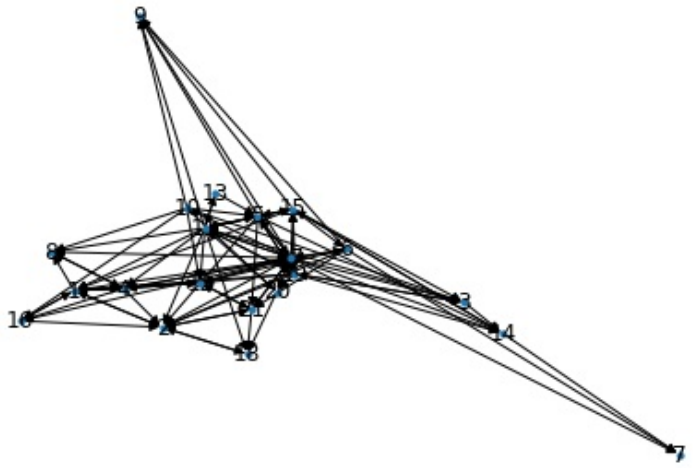
In [24]:

```
nx.draw(R, with_labels = True, node_size=10)
```



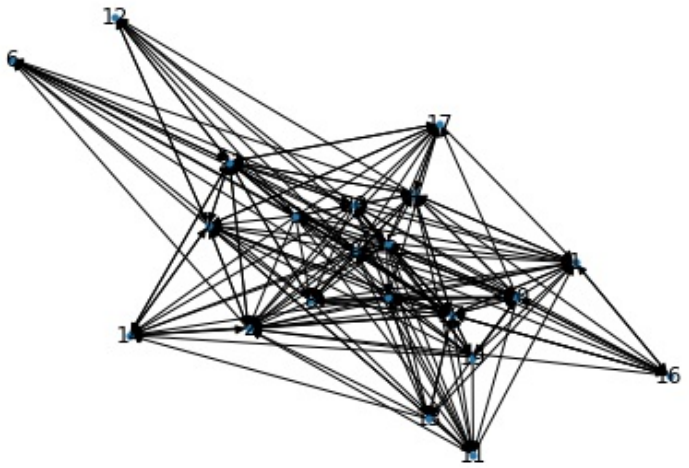
In [41]:

```
nx.draw(F, with_labels = True, node_size=10)
```



In [26]:

```
nx.draw(A, with_labels = True, node_size=10)
```



In [ ]:

```

```

2. Analysis

A: Network Network level characteristics. Find the overal network level of:

- **Density**
- **Transistivity (Clustering Coefficient)**
- **Reciprocity**

In [44]:

```
#density
print(nx.density(A))
print(nx.density(F))
print(nx.density(R))
```

```
0.4523809523809524
0.24285714285714285
0.047619047619047616
```

In [45]:

```
#transitvity (Clustering Coefficient)
print(nx.transitivity(A))
print(nx.transitivity(F))
print(nx.transitivity(R))
```

```
0.4651600753295669
0.27581863979848864
0
```

In [46]:

```
#reciprocity
print(nx.reciprocity(A))
print(nx.reciprocity(F))
print(nx.reciprocity(R))
```

```
0.47368421052631576
0.45098039215686275
0.0
```

for the different networks. Describe and interpret the results. Answer the following questions:

- **Are relationships like friendship and advice giving usually reciprocal?**

reciprocity in this csae is almost identical, in some cases an a advise could a frinedsship, but in the other 50% of the cases will friendship and profesional business differ

- **Are friends of your friends also your friends?**

the clustering coefficent show you have alomst a 30% chance for friend of friend are your friends

- **Are the employees generally more likely to be in a friendship or advice-seeking relationship?**

the density coefficent s of the networks show People are more advice seeking than making friendship

B: Node level characteristics: Likewise, find out:

- **Who is most popular in the networks. Who is the most wanted friend, and advice giver?**

In [64]:

```
edgeadvice.groupby('IDfrom').weight.sum().sort_values(ascending=False).head()
```

Out[64]:

```
IDfrom
15    20
18    17
3     15
5     15
10    14
Name: weight, dtype: int64
```

In [66]:

```
#calculate the three centrality measurments
centrality_dgr = nx.degree_centrality(A)
centrality_eigen = nx.eigenvector_centrality_numpy(A, weight='weight')
centrality_between = nx.betweenness_centrality(A, weight='weight')
```

In [67]:

```
#assign the cenrality measurements as a node attribute
nx.set_node_attributes(A, centrality_dgr, 'centrality_dgr')
nx.set_node_attributes(A, centrality_eigen, 'centrality_eigen')
nx.set_node_attributes(A, centrality_between, 'centrality_between')
```

In [69]:

```
A.nodes(data=True)
```

Out[69]:


```
NodeDataView({1: {'ID': 2, 'AGE': 42, 'TENURE': 19.583, 'LEVEL': 'Vice President', 'DEPT'
: 4, 'centrality_dgr': 0.9500000000000001, 'centrality_eigen': 0.23975517131939636, 'cent
rality_between': 0.03617585630743525}, 2: {'ID': 3, 'AGE': 40, 'TENURE': 12.75, 'LEVEL':
'Manager', 'DEPT': 2, 'centrality_dgr': 1.05, 'centrality_eigen': 0.4034046996761808, 'ce
ntrality_between': 0.015620300751879698}, 4: {'ID': 5, 'AGE': 32, 'TENURE': 3.333, 'LEVEL
': 'Manager', 'DEPT': 2, 'centrality_dgr': 1.0, 'centrality_eigen': 0.20856935901019266,
'centrality_between': 0.03607560568086884}, 8: {'ID': 9, 'AGE': 62, 'TENURE': 5.417000000
000001, 'LEVEL': 'Manager', 'DEPT': 2, 'centrality_dgr': 0.9, 'centrality_eigen': 0.22390
78112793787, 'centrality_between': 0.010459482038429406}, 16: {'ID': 17, 'AGE': 30, 'TENU
RE': 12.417, 'LEVEL': 'Manager', 'DEPT': 1, 'centrality_dgr': 0.6000000000000001, 'centra
lity_eigen': 0.1639787619287734, 'centrality_between': 0.0018421052631578945}, 18: {'ID':
19, 'AGE': 32, 'TENURE': 4.833, 'LEVEL': 'Manager', 'DEPT': 2, 'centrality_dgr': 1.6, 'ce
ntrality_eigen': 0.31459136879213195, 'centrality_between': 0.233991228070175 44}, 21: {'c
entrality_dgr': 1.3, 'centrality_eigen': 0.37109164350481727, 'centrality_between': 0.158
22890559732666}, 6: {'ID': 7, 'AGE': 55, 'TENURE': 30.0, 'LEVEL': 'CEO', 'DEPT': 0, 'cent
rality_dgr': 0.55, 'centrality_eigen': 0.2502211732537505, 'centrality_between': 0.0}, 7:
{'ID': 8, 'AGE': 34, 'TENURE': 11.333, 'LEVEL': 'Manager', 'DEPT': 1, 'centrality_dgr': 1
.05, 'centrality_eigen': 0.3095789367063831, 'centrality_between': 0.07269632414369255},
3: {'ID': 4, 'AGE': 33, 'TENURE': 7.5, 'LEVEL': 'Manager', 'DEPT': 4, 'centrality_dgr': 1
.0, 'centrality_eigen': 0.12353506753878116, 'centrality_between': 0.01738095238095238},
9: {'ID': 10, 'AGE': 37, 'TENURE': 9.25, 'LEVEL': 'Manager', 'DEPT': 3, 'centrality_dgr':
0.8500000000000001, 'centrality_eigen': 0.0735052356288695, 'centrality_between': 0.01040
5179615705931}, 10: {'ID': 11, 'AGE': 46, 'TENURE': 27.0, 'LEVEL': 'Manager', 'DEPT': 3,
'centrality_dgr': 1.1500000000000001, 'centrality_eigen': 0.1653665636434499, 'centrality
_between': 0.048149540517961574}, 11: {'ID': 12, 'AGE': 34, 'TENURE': 8.917, 'LEVEL': 'Ma
nager', 'DEPT': 1, 'centrality_dgr': 0.7000000000000001, 'centrality_eigen': 0.2241811391
7637583, 'centrality_between': 0.003153717627401838}, 12: {'ID': 13, 'AGE': 48, 'TENURE':
0.25, 'LEVEL': 'Manager', 'DEPT': 2, 'centrality_dgr': 0.45, 'centrality_eigen': 0.163527
16315090263, 'centrality_between': 0.0006683375104427735}, 14: {'ID': 15, 'AGE': 40, 'TEN
URE': 8.417, 'LEVEL': 'Manager', 'DEPT': 2, 'centrality_dgr': 0.7000000000000001, 'centra
lity_eigen': 0.20635410622668995, 'centrality_between': 0.0015497076023391813}, 17: {'ID'
: 18, 'AGE': 33, 'TENURE': 9.083, 'LEVEL': 'Vice President', 'DEPT': 3, 'centrality_dgr':
0.7000000000000001, 'centrality_eigen': 0.19425173032704338, 'centrality_between': 0.0066
62489557226399}, 20: {'ID': 21, 'AGE': 36, 'TENURE': 12.5, 'LEVEL': 'Vice President', 'DE
PT': 1, 'centrality_dgr': 1.0, 'centrality_eigen': 0.17446162015480435, 'centrality_betwe
en': 0.02099832915622389}, 5: {'ID': 6, 'AGE': 59, 'TENURE': 28.0, 'LEVEL': 'Manager', 'D
EPT': 1, 'centrality_dgr': 1.0, 'centrality_eigen': 0.08821983185673203, 'centrality_betw
een': 0.013364661654135338}, 13: {'ID': 14, 'AGE': 43, 'TENURE': 10.417, 'LEVEL': 'Vice P
resident', 'DEPT': 2, 'centrality_dgr': 0.5, 'centrality_eigen': 0.07961598913645403, 'ce
ntrality_between': 0.00234962406015037 6}, 19: {'ID': 20, 'AGE': 38, 'TENURE': 11.667, 'LE
VEL': 'Manager', 'DEPT': 2, 'centrality_dgr': 0.75, 'centrality_eigen': 0.079615989136454
14, 'centrality_between': 0.001984126984126984}, 15: {'ID': 16, 'AGE': 27, 'TENURE': 4.66
7, 'LEVEL': 'Manager', 'DEPT': 4, 'centrality_dgr': 1.2000000000000002, 'centrality_eigen
': 0.0889358605652996, 'centrality_between': 0.016138262322472847}})
```


In [78]:

```
A.edges(data=True)
```

Out[78]:


```
OutEdgeDataView([(1, 2, {'weight': 1}), (1, 4, {'weight': 1}), (1, 8, {'weight': 1}), (1,
16, {'weight': 1}), (1, 18, {'weight': 1}), (1, 21, {'weight': 1}), (2, 6, {'weight': 1})
, (2, 7, {'weight': 1}), (2, 21, {'weight': 1}), (4, 1, {'weight': 1}), (4, 2, {'weight':
1}), (4, 6, {'weight': 1}), (4, 8, {'weight': 1}), (4, 10, {'weight': 1}), (4, 11, {'weig
ht': 1}), (4, 12, {'weight': 1}), (4, 16, {'weight': 1}), (4, 17, {'weight': 1}), (4, 18,
{'weight': 1}), (4, 20, {'weight': 1}), (4, 21, {'weight': 1}), (8, 2, {'weight': 1}), (8
, 4, {'weight': 1}), (8, 6, {'weight': 1}), (8, 7, {'weight': 1}), (8, 10, {'weight': 1})
, (8, 11, {'weight': 1}), (8, 18, {'weight': 1}), (8, 21, {'weight': 1}), (16, 1, {'weigh
t': 1}), (16, 2, {'weight': 1}), (16, 10, {'weight': 1}), (16, 18, {'weight': 1}), (18, 1
, {'weight': 1}), (18, 2, {'weight': 1}), (18, 3, {'weight': 1}), (18, 4, {'weight': 1}),
(18, 5, {'weight': 1}), (18, 7, {'weight': 1}), (18, 8, {'weight': 1}), (18, 9, {'weight'
: 1}), (18, 10, {'weight': 1}), (18, 11, {'weight': 1}), (18, 13, {'weight': 1}), (18, 14
, {'weight': 1}), (18, 15, {'weight': 1}), (18, 16, {'weight': 1}), (18, 19, {'weight': 1
}), (18, 20, {'weight': 1}), (18, 21, {'weight': 1}), (21, 2, {'weight': 1}), (21, 3, {'w
eight': 1}), (21, 4, {'weight': 1}), (21, 6, {'weight': 1}), (21, 7, {'weight': 1}), (21,
8, {'weight': 1}), (21, 12, {'weight': 1}), (21, 14, {'weight': 1}), (21, 17, {'weight':
```

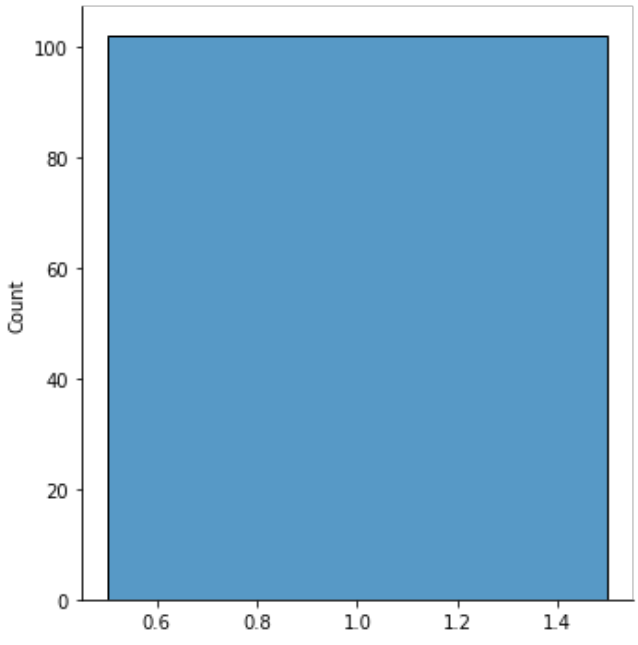
1}), (21, 18, {'weight': 1}), (21, 20, {'weight': 1}), (6, 21, {'weight': 1}), (7, 2, {'weight': 1}), (7, 6, {'weight': 1}), (7, 11, {'weight': 1}), (7, 12, {'weight': 1}), (7, 14, {'weight': 1}), (7, 17, {'weight': 1}), (7, 18, {'weight': 1}), (7, 21, {'weight': 1}), (3, 1, {'weight': 1}), (3, 2, {'weight': 1}), (3, 4, {'weight': 1}), (3, 6, {'weight': 1}), (3, 7, {'weight': 1}), (3, 8, {'weight': 1}), (3, 9, {'weight': 1}), (3, 10, {'weight': 1}), (3, 11, {'weight': 1}), (3, 12, {'weight': 1}), (3, 14, {'weight': 1}), (3, 17, {'weight': 1}), (3, 18, {'weight': 1}), (3, 20, {'weight': 1}), (3, 21, {'weight': 1}), (9, 1, {'weight': 1}), (9, 2, {'weight': 1}), (9, 6, {'weight': 1}), (9, 7, {'weight': 1}), (9, 8, {'weight': 1}), (9, 10, {'weight': 1}), (9, 11, {'weight': 1}), (9, 12, {'weight': 1}), (9, 14, {'weight': 1}), (9, 16, {'weight': 1}), (9, 17, {'weight': 1}), (9, 18, {'weight': 1}), (9, 21, {'weight': 1}), (10, 1, {'weight': 1}), (10, 2, {'weight': 1}), (10, 3, {'weight': 1}), (10, 4, {'weight': 1}), (10, 5, {'weight': 1}), (10, 8, {'weight': 1}), (10, 11, {'weight': 1}), (10, 13, {'weight': 1}), (10, 15, {'weight': 1}), (10, 16, {'weight': 1}), (10, 17, {'weight': 1}), (10, 18, {'weight': 1}), (10, 19, {'weight': 1}), (10, 20, {'weight': 1}), (11, 1, {'weight': 1}), (11, 2, {'weight': 1}), (11, 7, {'weight': 1}), (12, 7, {'weight': 1}), (12, 21, {'weight': 1}), (14, 2, {'weight': 1}), (14, 7, {'weight': 1}), (14, 18, {'weight': 1}), (14, 21, {'weight': 1}), (17, 1, {'weight': 1}), (17, 2, {'weight': 1}), (17, 4, {'weight': 1}), (17, 7, {'weight': 1}), (17, 21, {'weight': 1}), (20, 1, {'weight': 1}), (20, 2, {'weight': 1}), (20, 6, {'weight': 1}), (20, 8, {'weight': 1}), (20, 11, {'weight': 1}), (20, 12, {'weight': 1}), (20, 14, {'weight': 1}), (20, 15, {'weight': 1}), (20, 16, {'weight': 1}), (20, 17, {'weight': 1}), (20, 18, {'weight': 1}), (20, 21, {'weight': 1}), (5, 1, {'weight': 1}), (5, 2, {'weight': 1}), (5, 6, {'weight': 1}), (5, 7, {'weight': 1}), (5, 8, {'weight': 1}), (5, 10, {'weight': 1}), (5, 11, {'weight': 1}), (5, 13, {'weight': 1}), (5, 14, {'weight': 1}), (5, 16, {'weight': 1}), (5, 17, {'weight': 1}), (5, 18, {'weight': 1}), (5, 19, {'weight': 1}), (5, 20, {'weight': 1}), (5, 21, {'weight': 1}), (13, 1, {'weight': 1}), (13, 2, {'weight': 1}), (13, 5, {'weight': 1}), (13, 9, {'weight': 1}), (13, 14, {'weight': 1}), (13, 18, {'weight': 1}), (19, 1, {'weight': 1}), (19, 2, {'weight': 1}), (19, 3, {'weight': 1}), (19, 5, {'weight': 1}), (19, 7, {'weight': 1}), (19, 10, {'weight': 1}), (19, 11, {'weight': 1}), (19, 14, {'weight': 1}), (19, 15, {'weight': 1}), (19, 18, {'weight': 1}), (19, 20, {'weight': 1}), (15, 1, {'weight': 1}), (15, 2, {'weight': 1}), (15, 3, {'weight': 1}), (15, 4, {'weight': 1}), (15, 5, {'weight': 1}), (15, 6, {'weight': 1}), (15, 7, {'weight': 1}), (15, 8, {'weight': 1}), (15, 9, {'weight': 1}), (15, 10, {'weight': 1}), (15, 11, {'weight': 1}), (15, 12, {'weight': 1}), (15, 13, {'weight': 1}), (15, 14, {'weight': 1}), (15, 16, {'weight': 1}), (15, 17, {'weight': 1}), (15, 18, {'weight': 1}), (15, 19, {'weight': 1}), (15, 20, {'weight': 1}), (15, 21, {'weight': 1})])

In [80]:

```
weights = [z['weight'] for x,y,z in F.edges(data=True)]
sns.displot(weights)
```

Out[80]:

<seaborn.axisgrid.FacetGrid at 0x7f1fcba78d90>



C: Relational Characteristics: Answer the following questions:

- **Are managers from the same 1. department, or on the same 2. hirarchy, 3. age, or 4. tenuere more likely to become friends or give advice? (hint: assortiativity related)**

- **Are friends more likely to give each others advice?**

In [53]:

```
#avrage shortest path
print(nx.average_shortest_path_length(R))
print(nx.average_shortest_path_length(A))
print(nx.average_shortest_path_length(F))
```

```
0.12380952380952381
1.6404761904761904
1.9023809523809523
```

3. Visualizations

In [63]:

```
#here is a more clear of the graph that include the reports to and we get a clearer pictu
re the CEO, managerers adnvice president
plt.figure(figsize=(10, 10))
pos=nx.spring_layout(R, k=0.08)
nx.draw_networkx(R,pos,node_size=600, node_color='red',label='LEVEL')
```