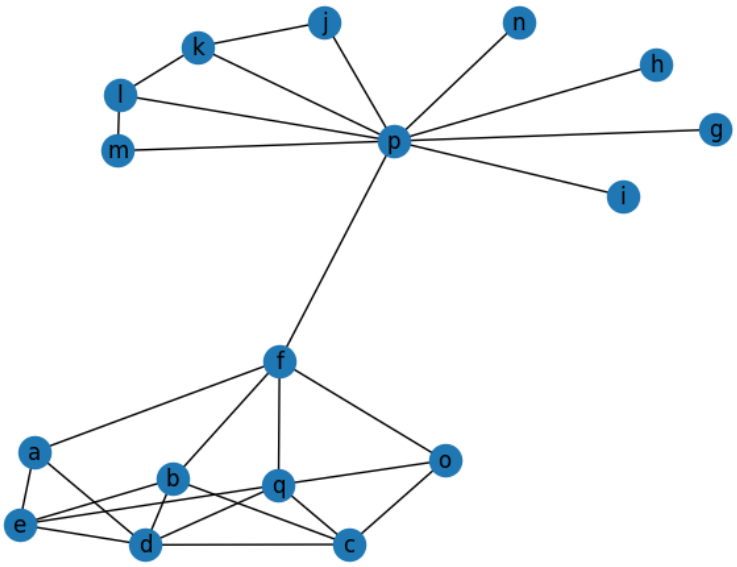
```
import networkx as nx
import matplotlib.pyplot as plt



# Create a graph
G = nx.Graph()

# Add nodes
G.add_nodes_from(["a","b","c","d","e","f","g","h","i","k","l","m","n","o","p","q"])

# Add edges
G.add_edges_from([("a","f"), ("a", "d"),("a","e"),("b","f"),("b","d"),("b","e"),("b","c"),("c","o"),("c","q"),("c","d"),("d"

# Visualize the graph
nx.draw(G, with_labels=True)
plt.show()
```




```
nx.clustering(G)
```

```
{'a': 0.3333333333333333,
 'b': 0.3333333333333333,
 'c': 0.5,
 'd': 0.5,
 'e': 0.5,
 'f': 0.1,
 'g': 0,
 'h': 0,
 'i': 0,
 'k': 0.6666666666666666,
 'l': 0.6666666666666666,
 'm': 1.0,
 'n': 0,
 'o': 0.6666666666666666,
 'p': 0.08333333333333333,
 'q': 0.4,
 'j': 1.0}
```

```
nx.degree(G)
```

```
DegreeView({'a': 3, 'b': 4, 'c': 4, 'd': 5, 'e': 4, 'f': 5, 'g': 1, 'h': 1, 'i': 1, 'k': 3, 'l': 3, 'm': 2, 'n': 1,
'o': 3, 'p': 9, 'q': 5, 'j': 2})
```

```
sorted(nx.degree_centrality(G).values())
m_influential=nx.degree_centrality(G)
for w in sorted(m_influential,key=m_influential.get,reverse=True):
  print(w,m_influential[w])
```

```
p 0.5625
d 0.3125
f 0.3125
q 0.3125
b 0.25
c 0.25
```

```
e 0.25
a 0.1875
k 0.1875
l 0.1875
o 0.1875
m 0.125
j 0.125
g 0.0625
h 0.0625
i 0.0625
n 0.0625
```

```
nx.eigenvector_centrality(G)
```

```
{'a': 0.260052126118304,
 'b': 0.34033521169485115,
 'c': 0.33573537673005904,
 'd': 0.4003877244350164,
 'e': 0.335597633354153,
 'f': 0.3515239993767802,
 'g': 0.049346465136389774,
 'h': 0.049346465136389774,
 'i': 0.049346465136389774,
 'k': 0.08689532548662873,
 'l': 0.08689532548662872,
 'm': 0.07012595369545559,
 'n': 0.049346465136389774,
 'o': 0.2606280728331461,
 'p': 0.20635775352120558,
 'q': 0.4026582489478831,
 'j': 0.07012595369545559}
```

```
nx.betweenness_centrality(G)
```

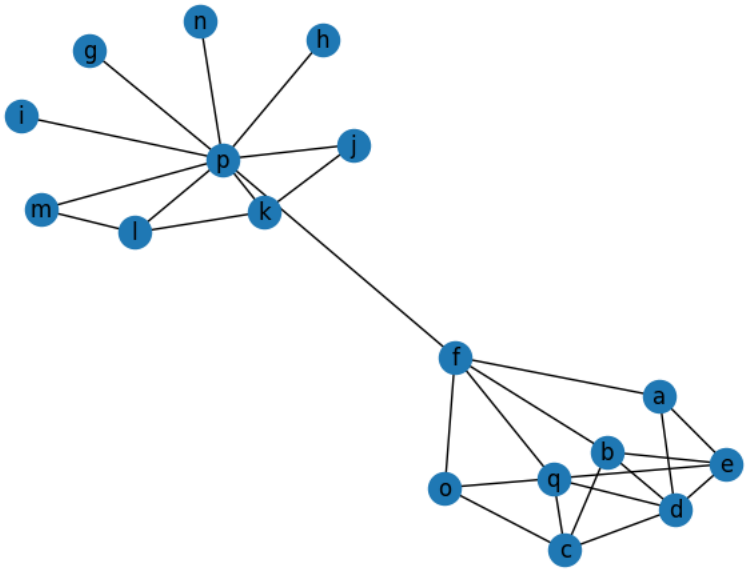
```
{'a': 0.05555555555555553,
 'b': 0.08611111111111111,
 'c': 0.010416666666666666,
 'd': 0.01875,
 'e': 0.007638888888888889,
 'f': 0.5451388888888888,
 'g': 0.0,
 'h': 0.0,
 'i': 0.0,
 'k': 0.004166666666666667,
 'l': 0.004166666666666667,
 'm': 0.0,
 'n': 0.0,
 'o': 0.027777777777777766,
 'p': 0.7333333333333333,
 'q': 0.09861111111111112,
 'j': 0.0}
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#G = nx.Graph()
#G = nx.path_graph(4, create_using=nx.DiGraph)
G1= nx.Graph(G)
```

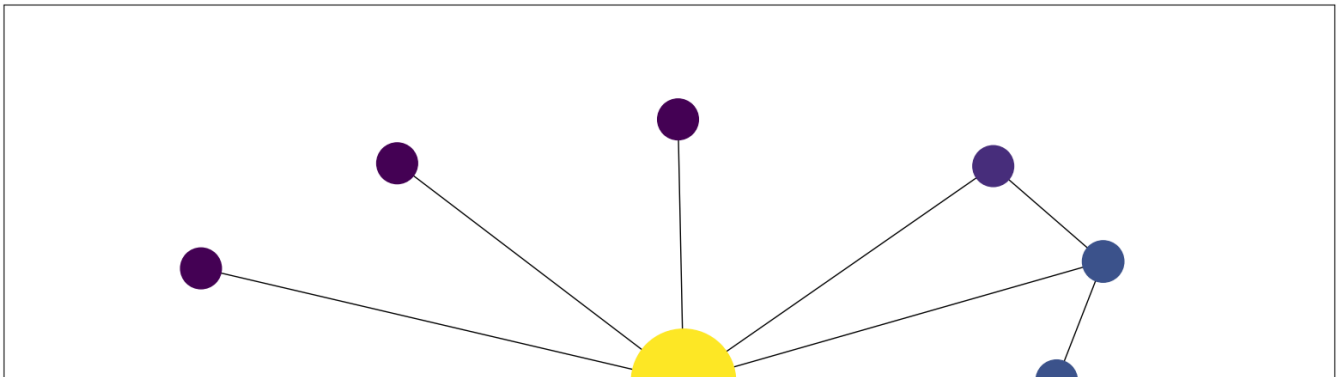
```
#nx.graph
```

```
nx.draw(G,with_labels=True)
```




```
pos=nx.spring_layout(G)
betCent=nx.betweenness_centrality(G,normalized=True,endpoints=True)
node_color=[20000.0*G.degree(v)for v in G]
node_size=[v*10000 for v in betCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G,pos=pos,with_labels=False,node_color=node_color,node_size=node_size)
sorted(betCent,key=betCent.get,reverse=True)[:5]
```

```
['p', 'f', 'q', 'b', 'a']
```

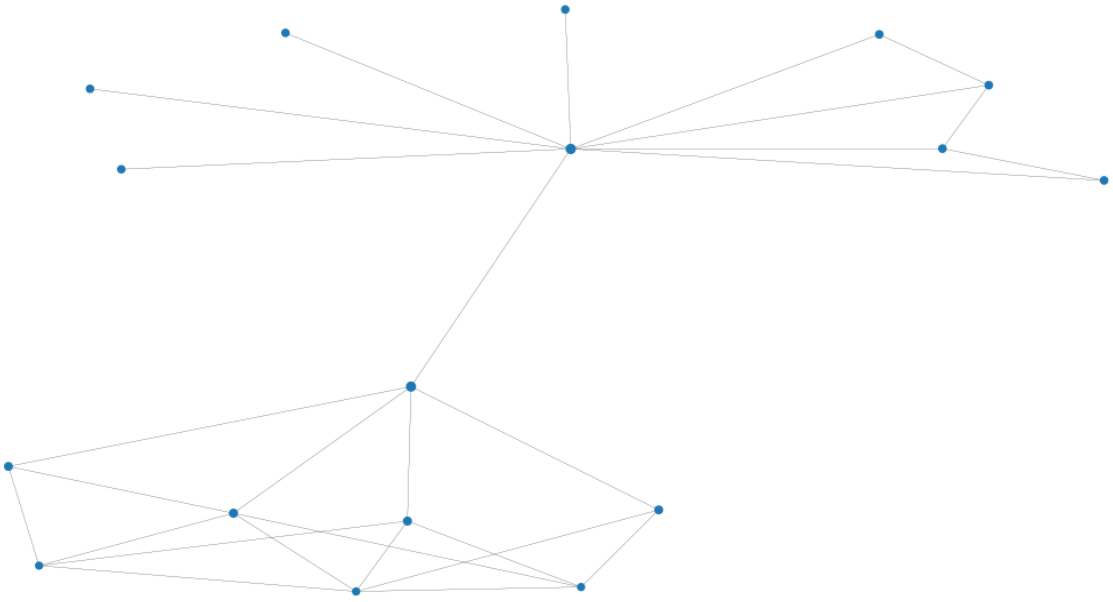


```
#closeness centrality
closeness_centrality=nx.centrality.closeness_centrality(G)
(sorted(closeness_centrality.items(),key=lambda item:item[1],reverse=True))[:8]
```

```
[('p', 0.6153846153846154),
 ('f', 0.5925925925925926),
 ('q', 0.45714285714285713),
 ('b', 0.4444444444444444),
 ('a', 0.43243243243243246),
 ('o', 0.43243243243243246),
 ('k', 0.41025641025641024),
 ('l', 0.41025641025641024)]
```

```
node_size=[v*50 for v in closeness_centrality.values()]
plt.figure(figsize=(15,8))
nx.draw_networkx(G,pos=pos,node_size=node_size,with_labels=False,width=0.15)
plt.axis("off")
```

```
(-0.4923445676018209,
 0.6112157147670971,
 -1.2071103477898757,
 1.1795898505505982)
```



```
#bridges
nx.has_bridges(G)
```

```
True
```