NAME: R.RAHINI ROLL NO: 215229128

SMA LAB 4: CALCULATING CENTRALITY MEASURES AND CREATING AN INTREST GRAPH FOR A GITHUB USER

In [2]:

```
import networkx as nx
```

1: CREATE AN INTEREST GRAPH OF A GITHUB USER BY ADDING 'FOLLOWS' AS EDGES

In [4]:

```
from github import Github # pip install pygithub

# XXX: Specify your own access token here

ACCESS_TOKEN = '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'

# Specify a username and repository of interest for that user.

USER = 'ptwobrussell'
REPO = 'Mining-the-Social-Web'
#REPO = 'Mining-the-Social-Web-2nd-Edition'

client = Github(ACCESS_TOKEN, per_page=100)
user = client.get_user(USER)
repo = user.get_repo(REPO)

# Get a list of people who have bookmarked the repo.
# Since you'll get a lazy iterator back, you have to traverse
# it if you want to get the total number of stargazers.

stargazers = [ s for s in repo.get_stargazers() ]
print("Number of stargazers", len(stargazers))
```

```
Number of stargazers 1209
```

2.CALCULATING THE DEGREE, BETWEENNESS , AND CLOSENESS CENTRALITY MEASURES OF A KRACKHARDT KITE GRAPH

In [17]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [18]:

```
from operator import itemgetter
from IPython.display import HTML
from IPython.core.display import display

display(HTML('<img  src="resources/ch08-github/kite-graph.png"  width="400px">'))

# The classic Krackhardt kite graph
kkg  =  nx.generators.small.krackhardt_kite_graph()

print("Degree Centrality")
print(sorted(nx.degree_centrality(kkg).items(),
             key=itemgetter(1),  reverse=True))
print()

print("Betweenness Centrality")
print(sorted(nx.betweenness_centrality(kkg).items(),
             key=itemgetter(1),  reverse=True))
print()

print("Closeness Centrality")
print(sorted(nx.closeness_centrality(kkg).items(),
             key=itemgetter(1),  reverse=True))
```

```
Degree Centrality
[(3, 0.6666666666666666), (5, 0.5555555555555556), (6, 0.555555555555555
6), (0, 0.4444444444444444), (1, 0.4444444444444444), (2, 0.33333333333333
33), (4, 0.3333333333333333), (7, 0.3333333333333333), (8, 0.2222222222222
222), (9, 0.1111111111111111)]

Betweenness Centrality
[(7, 0.38888888888888884), (5, 0.23148148148148148), (6, 0.231481481481481
48), (8, 0.2222222222222222), (3, 0.10185185185185183), (0, 0.023148148148
148143), (1, 0.023148148148148143), (2, 0.0), (4, 0.0), (9, 0.0)]

Closeness Centrality
[(5, 0.6428571428571429), (6, 0.6428571428571429), (3, 0.6), (7, 0.6), (0,
0.5294117647058824), (1, 0.5294117647058824), (2, 0.5), (4, 0.5), (8, 0.42
857142857142855), (9, 0.3103448275862069)]
```

STEP 1

In [6]:

```
g = nx.DiGraph()
g.add_node(repo.name  +  '(repo)',type='repo',lang=repo.language,owner=user.login)
for sg in stargazers:
    g.add_node(sg.login + '(user)', type='user')
    g.add_edge(sg.login + '(user)', repo.name + '(repo)',type='gazes')
```

In [7]:

```
import sys

for i, sg in enumerate(stargazers):

    # Add "follows" edges between stargazers in the graph if any relationships exist
    try:
        for follower in sg.get_followers():
            if follower.login + '(user)' in g:
                g.add_edge(follower.login + '(user)', sg.login + '(user)',
                           type='follows')
    except Exception as e: #ssl.SSLError
        print("Encountered an error fetching followers for", sg.login, \
              "Skipping.", file=sys.stderr)
        print(e, file=sys.stderr)

    print("Processed", i+1, " stargazers. Num nodes/edges in graph", \
          g.number_of_nodes(), "/", g.number_of_edges())
    print("Rate limit remaining", client.rate_limiting)
```

```
Processed 1  stargazers. Num nodes/edges in graph 1210 / 1212
Rate limit remaining (4984, 5000)
Processed 2  stargazers. Num nodes/edges in graph 1210 / 1214
Rate limit remaining (4983, 5000)
Processed 3  stargazers. Num nodes/edges in graph 1210 / 1220
Rate limit remaining (4981, 5000)
Processed 4  stargazers. Num nodes/edges in graph 1210 / 1222
Rate limit remaining (4980, 5000)
Processed 5  stargazers. Num nodes/edges in graph 1210 / 1223
Rate limit remaining (4979, 5000)
Processed 6  stargazers. Num nodes/edges in graph 1210 / 1227
Rate limit remaining (4975, 5000)
Processed 7  stargazers. Num nodes/edges in graph 1210 / 1237
Rate limit remaining (4970, 5000)
Processed 8  stargazers. Num nodes/edges in graph 1210 / 1237
Rate limit remaining (4969, 5000)
Processed 9  stargazers. Num nodes/edges in graph 1210 / 1239
Rate limit remaining (4967, 5000)
Processed 10  stargazers. Num nodes/edges in graph 1210 / 1242
```

In [8]:

```
nx.write_gpickle(g,"github.gpickle.1")
```

STEP 2:EXPLORE THE GRAPH WITH THE UPDATES "FOLLOWS" EDGES

In [19]:

```
# a.Get the information about the updated graph

print(nx.info(g))
```

```
DiGraph with 1210 nodes and 2868 edges
```

In [40]:

```
# b. Find the number of 'follows' edges

print(len([e for e in g.edges(data=True) if e [2]['type'] =='follows']))
```

```
1659
```

In [11]:

```
from collections import Counter
```

In [12]:

```
# c.Find the number of popular users and the top 10 users

c = Counter ([e[1] for e in g.edges(data=True) if e[2]['type'] == 'follows'])
popular_users = [ (u,f) for (u, f) in c.most_common() if f > 1]

print("Number of popular users", len(popular_users))
print("Top 10 popular users:",popular_users[:10])
```

```
Number of popular users 263
Top 10 popular users: [('kennethreitz(user)', 171), ('ptwobrussell(user)',
134), ('daimajia(user)', 39), ('angusshire(user)', 23), ('hammer(user)', 2
2), ('jakubroztocil(user)', 22), ('dgryski(user)', 19), ('isnowfy(user)',
18), ('japerk(user)', 17), ('timelyportfolio(user)', 14)]
```

In [13]:

```
# d.Remove the super node from the graph and calculate the centrality measures

h = g.copy()
h.remove_node('Mining-the-Social-Web(repo)')
```

STEP 3:VISUALIZE THE CREATED INTEREST GRAPH

In [20]:

```
# create a subgraph from the original interest graph-select the user nodes and get the

mtsw_users = [n for n in g if g.nodes[n]['type'] == 'user']
h = g.subgraph(mtsw_users)

print("Stats on the extracted subgraph")
print(nx.info(h))
```

```
Stats on the  extracted  subgraph
DiGraph with 1209 nodes and 1659 edges
```

In [51]:

```
print("Stats on the full graph")
print(nx.info(g))
print()

# Create a subgraph from a collection of nodes. In this case, the
# collection is all of the users in the original interest graph

mtsw_users = [n for n in g if g.nodes [n]['type'] == 'user']
h = g.subgraph(mtsw_users)

print("Stats on the extracted subgraph")
print(nx.info(h))
```

```
Stats on the full graph
DiGraph with 1210 nodes and 2868 edges

Stats on the  extracted  subgraph
DiGraph with 1209 nodes and 1659 edges
```

In [16]:

```
# visualize the extracted graph using matplotlib and networkx

import matplotlib.pyplot as plt
%matplotlib inline


fig = plt.figure(figsize=(15,15))
ax = fig.add_subplot(111)
labels = dict([(n, n.split('(user)')[0]) for n in h.nodes()])

nx.draw(h, pos=nx.spring_layout(h),
        arrows=False, ax=ax, node_size=50,
        edge_color='#aaaaaa',
        alpha=0.8, labels=labels, font_size=8)
```