```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
import pandas as pd
from collections import Counter
! pip install networkx
! pip install plotly
! pip install colorlover
```

```
Requirement already satisfied: networkx in /usr/local/lib/python3.6/dist-packages (2.4)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx) (4.4.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (4.1.1)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly) (1.3.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from plotly) (1.12.0)
Requirement already satisfied: colorlover in /usr/local/lib/python3.6/dist-packages (0.3.0)
```

```
import networkx as nx
```

```
link ='https://drive.google.com/open?id=1p7wjO3x6TItoZbN8avEtYoIg1MMVy8T2'
fluff, id = link.split('=')
print (id)
```

```
1p7wjO3x6TItoZbN8avEtYoIg1MMVy8T2
```

```
downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('tweets2009-06-0115.zip')
df = pd.read_csv('tweets2009-06-0115.zip', sep='\t', compression='zip')
```

```
df.head()
```

	date	user	tweet
0	2009-06-01 21:43:59	burtonator	No Post Title
1	2009-06-01 21:47:23	burtonator	No Post Title
2	2009-06-02 01:15:44	burtonator	No Post Title
3	2009-06-02 05:17:52	burtonator	No Post Title
4	2009-06-02 23:58:25	burtonator	No Post Title

```
allTweets = df["tweet"].str.cat(sep=' ')
tweetWords = [word.strip(""" ,.:'\";""").lower() for word in allTweets.split()]
```

```
hashTags = [word for word in tweetWords if word.startswith("#")]
hashTagsCounter = Counter(hashTags)
```

```
hashTagsCounter.most_common(100)
```

```
[('#iranelection', 26853),
 ('#followfriday', 16400),
 ('#jobs', 13322),
 ('#iremember', 11057),
 ('#spymaster', 10587),
 ('#ff', 10446),
 ('#squarespace', 9198),
 ('#tcot', 7691),
 ('#fb', 6107),
 ('#cnnfail', 4451),
 ('#11thcommandment', 3429),
 ('#jtv', 3317),
 ('#140mafia', 3144),
 ('#iran', 2935),
 ('#', 2895),
 ('#news', 2837),
 ('#quote', 2750),
 ('#vampirebite', 2634),
 ('#1', 2587),
 ('#bsb', 2433),
 ('#tweetmyjobs', 2086),
 ('#iphone', 1697),
 ('#lastfm', 1599),
 ('#mp2', 1589),
 ('#niley', 1528),
 ('#music', 1489),
 ('#p2', 1439),
 ('#follow', 1390),
 ('#pawpawty', 1305),
 ('#hhrs', 1256),
 ('#fail', 1246),
 ('#twitter', 1216),
 ('#tlot', 1214),
 ('#facebook', 1177),
 ('#sgp', 1151),
 ('#mashchat', 1143),
 ('#tinychat', 1111),
 ('#2', 1107),
 ('#digg', 1102),
 ('#gop', 1009),
 ('#phish', 1001),
 ('#mlb', 962),
 ('#travel', 932),
 ('#bonnaroo', 887),
 ('#twitpocalypse', 879),
 ('#iranelections', 857),
 ('#rt', 856),
 ('#zensursula', 811),
 ('#jamlegend', 790),
 ('#quotes', 756),
 ('#tehran', 749),
 ('#tech', 737),
 ('#love', 709),
 ('#pens', 708),
 ('#jobfeedr', 701),
 ('#teaparty', 677),
 ('#wordpress', 669),
 ('#obama', 665),
 ('#lol', 659),
 ('#redwings', 655),
 ('#socialmedia', 648),
 ('#photography', 645),
 ('#lofnotc', 634),
 ('#3', 632),
 ('#beatlesporn', 624),
 ('#mousavi', 619),
 ('#nascar', 613),
 ('#job', 602),
 ('#fashion', 586),
 ('#wine', 579),
 ('#mileybrazil', 578),
 ('#redsox', 577),
 ('#video', 574),
 ('#peterfacinelli', 567),
 ('#design', 566),
 ('#spon', 548),
 ('#newiran', 547),
 ('#bpt09', 526),
 ('#nhl', 525),
 ('#business', 522),
 ('#thedowntownfiction', 518),
 ('#tl', 517),
 ('#blogpotomac', 516),
 ('#north', 514),
 ('#green', 497),
 ('#marketing', 493),
 ('#youtube', 487),
 ('#photo', 483),
 ('#art', 479),
 ('#free', 468),
 ('#politics', 466),
 ('#4', 457),
 ('#haiku', 456),
 ('#blogger', 456),
 ('#sinanews', 453),
 ('#frenchmcflyteam', 449),
 ('#lakers', 447),
 ('#weather', 446),
 ('#lemans', 435),
 ('#seo', 434)]
```

Q1

(a)

```
youtubeTag = df[df["tweet"].str.lower().str.contains("#youtube", na=False)].copy()


def addMentionedColumn(df):

    def mentionsList(txt):
        allWords = [word.strip(""" ,.:'\";""").lower() for word in txt.split()]
        allNames = [word.strip("@") for word in allWords if word.startswith("@")]
        uniqueNames = list(set(allNames))
        return allNames

    df["mentioned"] = df["tweet"].apply(mentionsList)
```

```
addMentionedColumn(youtubeTag)
```

```
youtubeTag.shape
```

```
(617, 4)
```

```
def mentionGraph(df):
    g = nx.Graph()
    count=0
    for (index, date, user, tweet, mentionedUsers) in df.itertuples():
      for mentionedUser in mentionedUsers:
        count+=1
        if (user in g) and (mentionedUser in g[user]):
          g[user][mentionedUser]["numberMentions"] += 1

        else:
          g.add_edge(user, mentionedUser, numberMentions=1)

    print(count)
    return g
```

```
youtubeGraph = mentionGraph(youtubeTag)
```

```
246
```

```
print("# nodes:", len(youtubeGraph.nodes()))
print("# edges:", len(youtubeGraph.edges()))
```

```
# nodes: 271
# edges: 233
```

```
print(len(youtubeTag['user'].unique()))#user that does not mention anyone and not mentioned by anybody will not show
#at the same time the use who is mentioned but did not tweet anything will appear in graph as well
```

```
288
```

There are 271 nodes and 233 edges in the mention graph.

(b)

```
import matplotlib.pyplot as plt
```

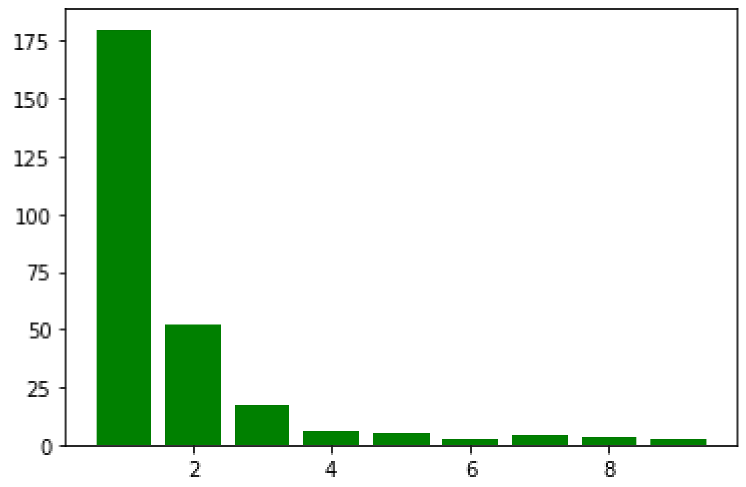
```
degrees=youtubeGraph.degree()
degreeList=list(degrees)
```

```
degreeList[0]
```

```
('speedy3702', 1)
```

```
degreeCount={}
for item in degreeList:
  if item[1] in degreeCount:
    degreeCount[item[1]]+=1
  else:
    degreeCount[item[1]]=1
```

```
plt.bar(degreeCount.keys(), degreeCount.values(), color='g')
plt.show()
```



The majority of the nodes only has one degree and the node degree represents the connections among users, thus the nodes that have high degree tends to be the central of the social network.

(c)

```
d=('david',{'2':4,'van':5})
print(d[1]['2'])
```

```
4
```

```
mentionList=[]
#need add data=True otherwise there are only two values(u,v)
for u,v,weight in youtubeGraph.edges(data=True):
  mentionList.append((u,v,weight))
mentionList=sorted(mentionList, key=lambda x: x[2]['numberMentions'])
#sort the list of tuples by the value of third dict value
```

```
mentionList[-5:]
```

```
[('dabloguiman', 'avidya', {'numberMentions': 2}),
 ('dabloguiman', 'caferozella', {'numberMentions': 2}),
 ('growline', 'justkarl', {'numberMentions': 2}),
 ('kevinsoberg', 'keystroke', {'numberMentions': 2}),
 ('ossguy', "cdibona's", {'numberMentions': 3})]
```

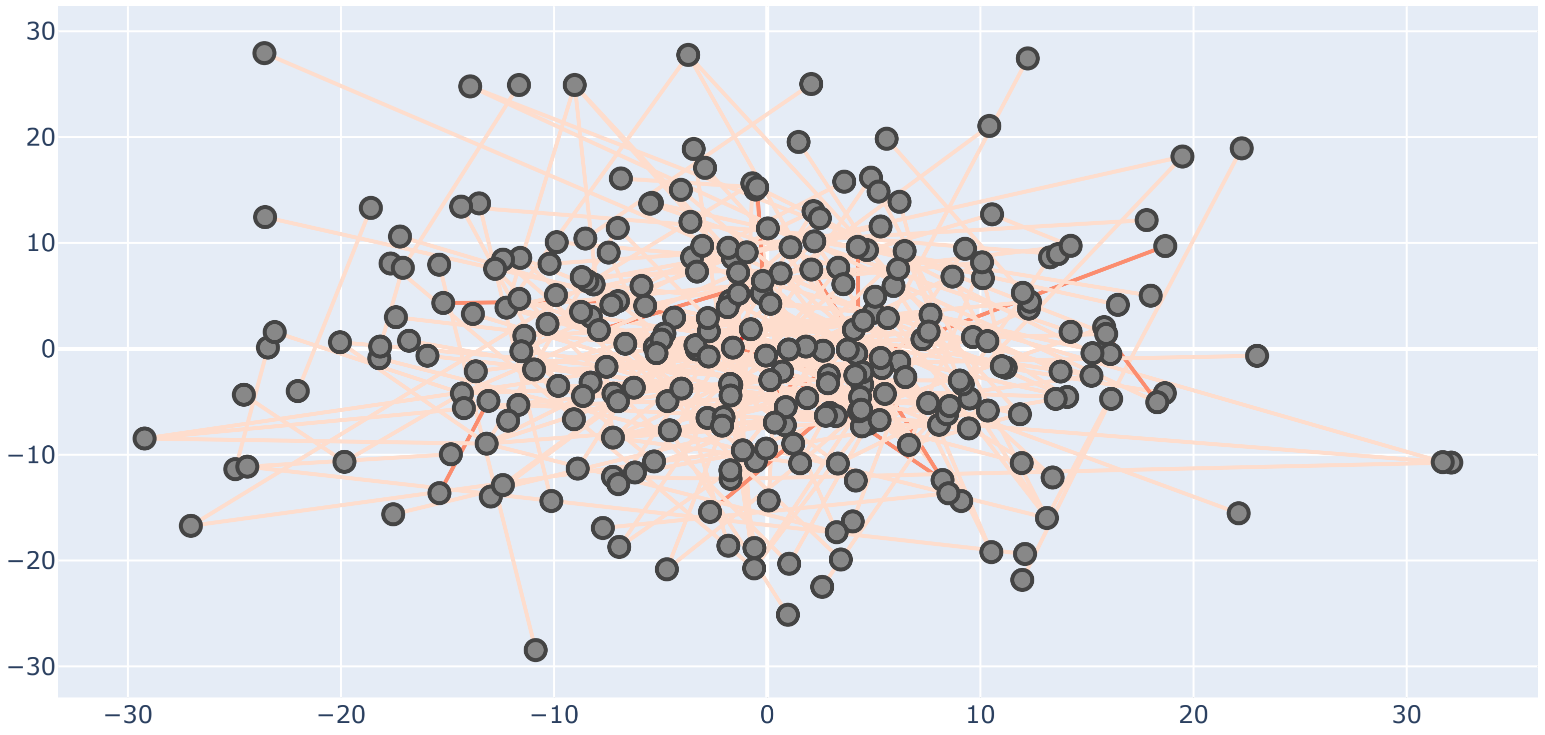
(d)

```
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly.graph_objs import *
import plotly.graph_objects as go
init_notebook_mode(connected=True)
```

```
import colorlover as cl
from IPython.display import HTML


HTML(cl.to_html( cl.flipper()['seq']['3'] ))
```

```
Blues      BuGn      BuPu      GnBu      Greens     Greys     OrRd

Oranges    PuBu      PuBuGn    PuRd      Purples    RdPu      Reds
```

```
red=cl.scales['3']['seq']['Reds']
red3 = cl.interp(red, 3)


def configure_plotly_browser_state():
  import IPython
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
        <script>
          requirejs.config({
            paths: {
              base: '/static/base',
              plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',
            },
          });
        </script>
        '''))


import random
def addRandomPositions(graph):
    posDict = dict((node,(random.gauss(0,10),random.gauss(0,10))) for node in graph.nodes())
    nx.set_node_attributes(graph, name="pos", values=posDict)
    #generate random position for nodes


addRandomPositions(youtubeGraph)


def plotNetwork(graph):
    scatters=[]

    for (node1, node2) in graph.edges():
        x0, y0 = graph.nodes[node1]['pos']
        x1, y1 = graph.nodes[node2]['pos']
        edgeWidth = graph[node1][node2]['numberMentions']
        s = go.Scatter(
                x=[x0, x1],
                y=[y0, y1],
                hoverinfo='none',
                mode='lines',
                line=scatter.Line(width=2 ,color=red3[edgeWidth-1]))
        scatters.append(s)
    for node in graph.nodes():
        xPos, yPos = graph.nodes[node]['pos']
        s = go.Scatter(
                x=[xPos],
                y=[yPos],
                hoverinfo='none',
                mode='markers',
                marker=dict(
                    color="#888",
                    size=10,
                    line=dict(width=2)))
        scatters.append(s)

    layout = Layout(showlegend=False)
    fig = Figure(data=scatters, layout=layout)
    iplot(fig, show_link=False)


configure_plotly_browser_state()
plotNetwork(youtubeGraph)
```



Q2

(a)

```
import nltk
from nltk.tokenize import RegexpTokenizer
import re
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
commonWord=[]
userList=youtubeTag['user'].unique()
userTweetDict={}
for user in userList:
  userTweets=youtubeTag.loc[youtubeTag['user']==user,'tweet']
  totalTweet=''
  for tweet in userTweets:
    totalTweet=totalTweet+' '+tweet
  userTweetDict[user]=totalTweet
utSeries = pd.Series(userTweetDict)


regex_link = re.compile(r'\s*(?:https?:\/\/)?[\w.-]+(?:\.[\w.-]+)+[\w\-._~:/?#[\]@!\$&\'\(\)\*\+,;=.]+', flags=re.IG
stop_words = set(stopwords.words('english'))
tokenizer=RegexpTokenizer(r'\w+')
youtubeText=utSeries.str.lower().replace(regex_link,'').apply(lambda x: tokenizer.tokenize(x))
#youtubeText=youtubeText.apply(lambda x: [item for item in x if item not in stop_words])
youtubeText=youtubeText.apply(lambda x: x.remove('rt') if 'rt' in x else x)
#lambda if clause need a complete form 'else' is necessary



textDF=pd.DataFrame(youtubeText,columns=['text'])
for item in youtubeText:
  top3=Counter(item).most_common(3)
  '''if top3==None:
    commonWord.append(["NONE"])
    continue'''
  commonWord.append(top3)
textDF['top3']=commonWord

textDF['top3']=textDF['top3'].apply(lambda x: [item[0] for item in x ])

textDF.head()
```

	text	top3
unborn	[want, to, make, a, time, link, in, youtube]	[want, to, make]
animamundicm	[claudia, muzio, addio, del, passato, youtube,...	[youtube, claudia, muzio]
youtubehd	[top, youtube, hd, zach, s, schlieffen, plan, ...	[hd, youtube, top]
bondijunction	[what, happens, if, you, have, way, too, much,...	[what, happens, if]
speedy3702	None	[]

```
tweetText=textDF['text']
textCounter=Counter()
for item in tweetText:
  textCounter.update(item)

top20=textCounter.most_common(20)
print(top20)
```

```
[('youtube', 521), ('hd', 143), ('video', 97), ('s', 86), ('the', 83), ('new', 74), ('top', 72), ('today', 64), ('hot', 55),
```

From the top 20 most common words we can tell that the main theme of the tweets about youtube focus on hd videos and daily hot events or movie trailer.

(b)

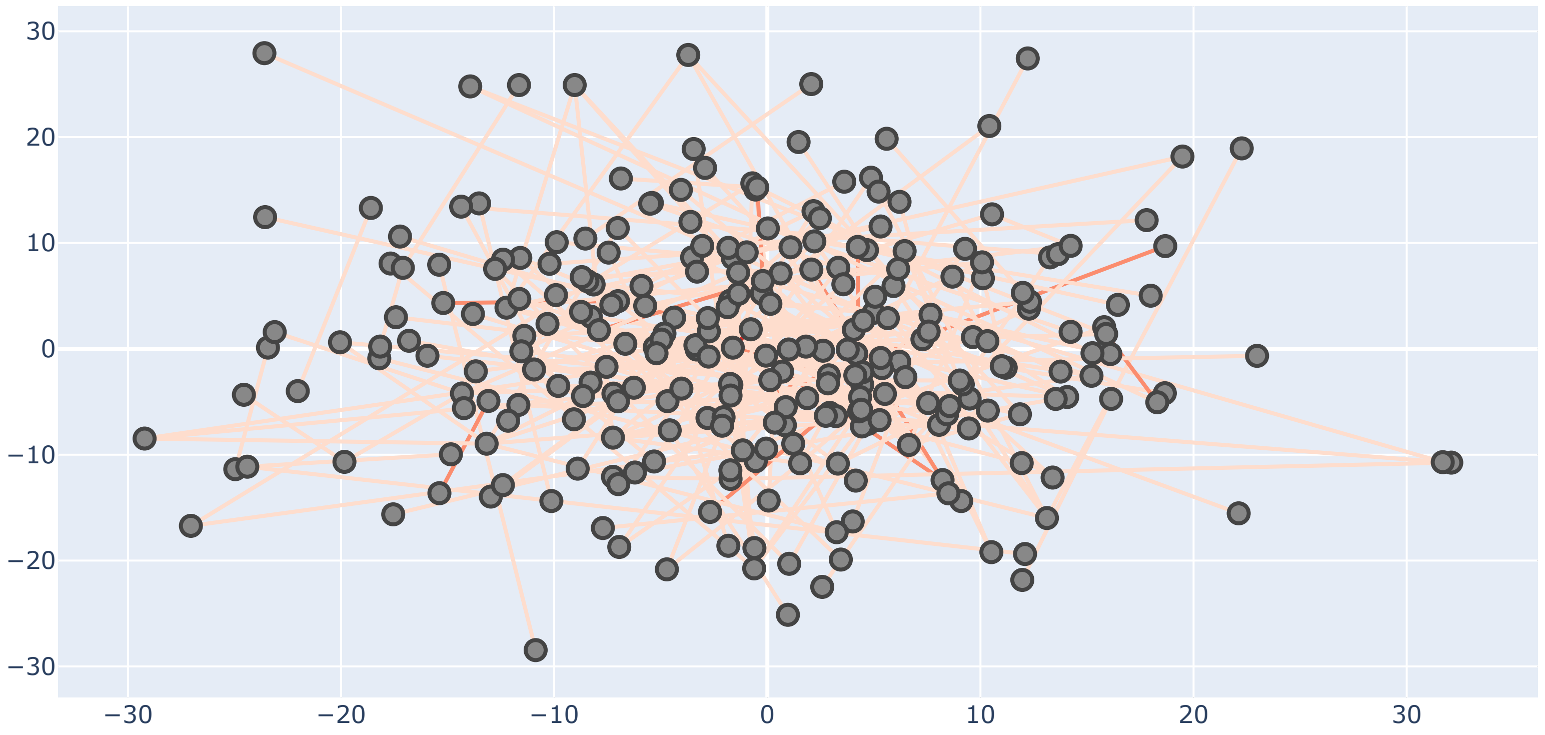
```
def plotHoverNetwork(graph,df):
    scatters=[]
    degrees=graph.degree()
    degreeList=list(degrees)
    for (node1, node2) in graph.edges():
        x0, y0 = graph.nodes[node1]['pos']
        x1, y1 = graph.nodes[node2]['pos']
        edgeWidth = graph[node1][node2]['numberMentions']
        s = go.Scatter(
                x=[x0, x1],
                y=[y0, y1],
                hoverinfo='none',
                mode='lines',
                line=scatter.Line(width=2 ,color=red3[edgeWidth-1]))
        scatters.append(s)


    count=0
    for node in graph.nodes():
        xPos, yPos = graph.nodes[node]['pos']
        topText=''
        nodeDegree=0
        if node in df.index:
          count+=1
          topText=df.loc[node,'top3']
          topText=' '.join(topText)

          for item in degreeList:
            if item[0]==node:
              nodeDegree=item[1]
              break
        else:
          topText='No record'
          nodeDegree=1
        s = go.Scatter(
                x=[xPos],
                y=[yPos],
                text = 'Top3:'+topText+'Degree:'+str(nodeDegree),

                hoverinfo="text",
                mode='markers',
                marker=dict(
                    color="#888",
                    size=10,
                    line=dict(width=2)))
        scatters.append(s)
    print(count)
    layout = Layout(showlegend=False)
    fig = Figure(data=scatters, layout=layout)
    iplot(fig, show_link=False)
```

```
'cwmacdon)' in youtubeGraph.nodes()
```

```
True
```

```
configure_plotly_browser_state()
plotHoverNetwork(youtubeGraph,textDF)
```

```
155
```



In the graph above, the hover information includes the top3 most common words in the tweet and also contains the node degree. Since some nodes in the graph are simply mentioned and they do not have any tweet or some tweets only contains some url links and those links are removed during the text processing steps, under these two situations the node has no hover text.

Q3

(a)

Closeness Centrality&Betweeness Centrality

```
from networkx.algorithms import closeness_centrality,betweenness_centrality
```

```
close_Cebtrality=closeness_centrality(youtubeGraph, u=None, distance=None, wf_improved=True)
```

```
close_Cebtrality=dict(close_Cebtrality)
```

```
between_centrality=betweenness_centrality(youtubeGraph)
```

```
type(between_centrality)
```

```
dict
```

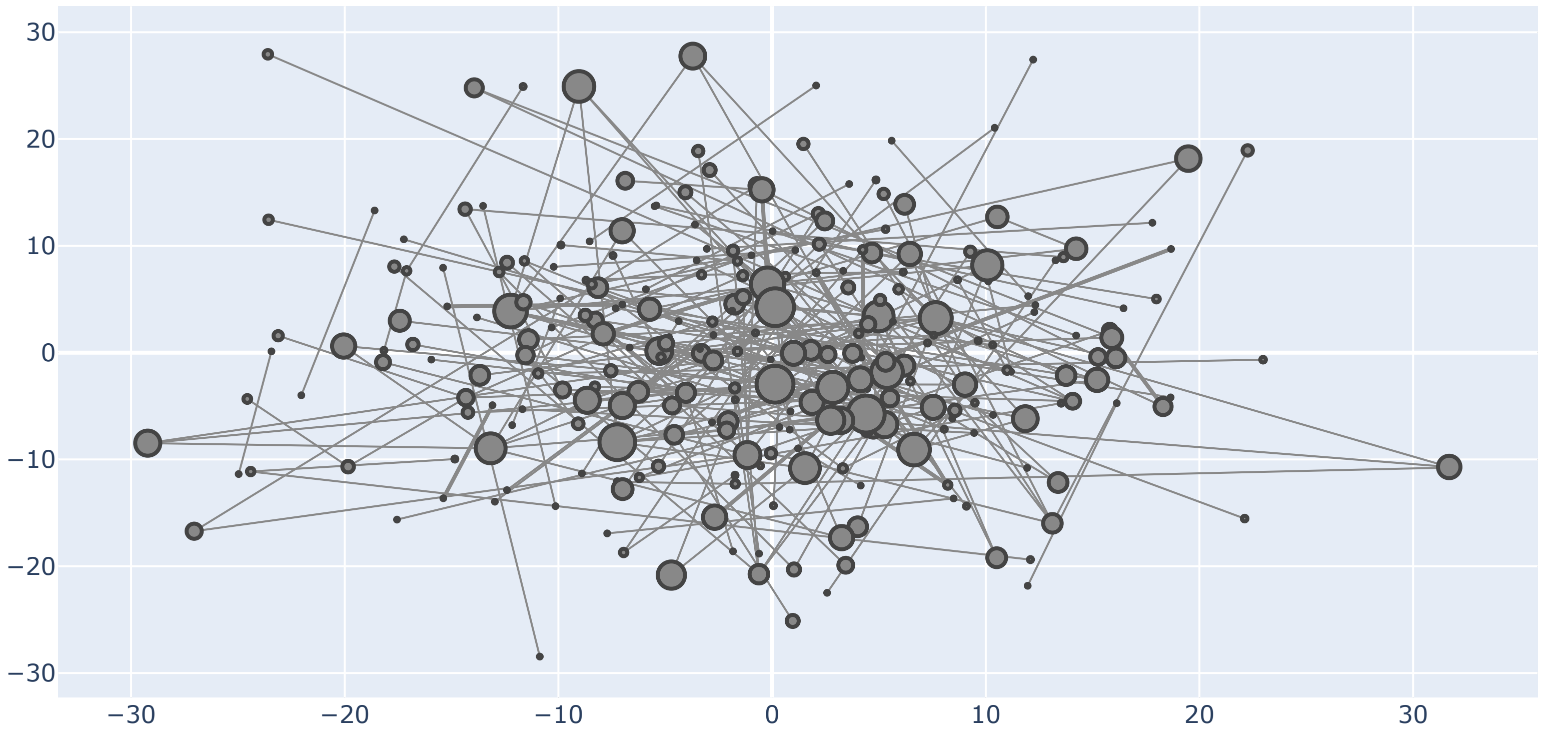
(b)

```
def plotNetworkSize(graph,centrality):
    scatters=[]

    for (node1, node2) in graph.edges():
        x0, y0 = graph.nodes[node1]['pos']
        x1, y1 = graph.nodes[node2]['pos']
        edgeWidth = graph[node1][node2]['numberMentions']
        s = Scatter(
                x=[x0, x1],
                y=[y0, y1],
                hoverinfo='none',
                mode='lines',
                line=scatter.Line(width=edgeWidth ,color='#888'))
        scatters.append(s)



    for node in graph.nodes():
        xPos, yPos = graph.nodes[node]['pos']
        s = Scatter(
                x=[xPos],
                y=[yPos],
                hoverinfo='none',
                mode='markers',
                marker=dict(
                    color="#888",
                    size=centrality[node]*500,
                    line=dict(width=2)))
        scatters.append(s)

    layout = Layout(showlegend=False)
    fig = Figure(data=scatters, layout=layout)
    iplot(fig, show_link=False)
```

```
type(close_Cebtrality)
```

```
list
```

```
configure_plotly_browser_state()
plotNetworkSize(youtubeGraph,close_Cebtrality)
```

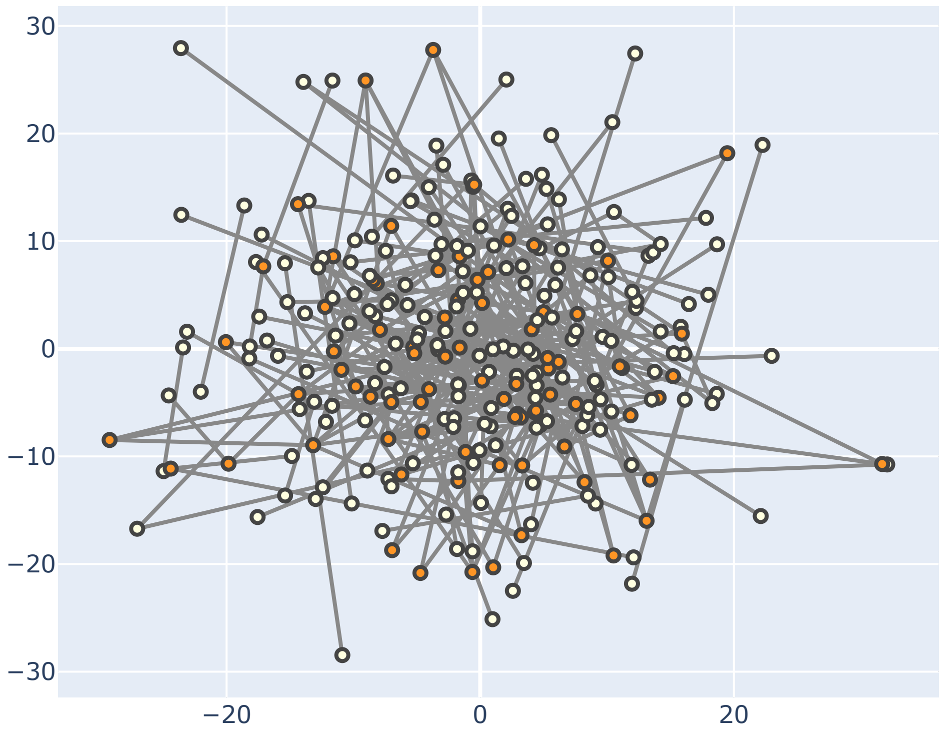


```
HTML(cl.to_html( cl.flipper()['seq']['9'] ))
```

```
Blues          BuGn           BuPu           GnBu

Greens         Greys          OrRd           Oranges

PuBu           PuBuGn         PuRd           Purples

RdPu           Reds           YlGn           YlGnBu

YlOrBr         YlOrRd
```

```
ybr=cl.scales['9']['seq']['YlOrBr']
ybr3 = cl.interp(ybr, 9)
HTML(cl.to_html(ybr3))
```

```
def plotNetworkColor(graph,centrality):
    scatters=[]

    for (node1, node2) in graph.edges():
        x0, y0 = graph.nodes[node1]['pos']
        x1, y1 = graph.nodes[node2]['pos']

        s = Scatter(
                x=[x0, x1],
                y=[y0, y1],
                hoverinfo='none',
                mode='lines',
                line=scatter.Line(width=2 ,color='#888'))
        scatters.append(s)



    for node in graph.nodes():
        xPos, yPos = graph.nodes[node]['pos']
        c=0
        if centrality[node]==0:
          c=0

        if centrality[node]>0 and centrality[node]<0.5:
          c=4
        if centrality[node]>0.5 and centrality[node]<1:
          c=6
        if centrality[node]>1:
          c=9


        s = Scatter(
                x=[xPos],
                y=[yPos],
                hoverinfo='none',
                mode='markers',
                marker=dict(
                    color=ybr3[c],
                    line=dict(width=2)))
        scatters.append(s)

    layout = Layout(showlegend=False)
    fig = Figure(data=scatters, layout=layout)
    iplot(fig, show_link=False)

configure_plotly_browser_state()
plotNetworkColor(youtubeGraph,between_centrality)
```



(c)

```
import operator

close_Cebtrality=sorted(close_Cebtrality.items(),key=operator.itemgetter(1))
```

```
between_centrality=sorted(between_centrality.items(),key=operator.itemgetter(1))

top_Close=close_Cebtrality[-10:]
top_Between=between_centrality[-10:]

print(top_Close)
print(top_Between)
```

```
[('jsbond', 0.03109935332157554), ('growline', 0.03160095579450418), ('zaibatsu', 0.03160095579450418), ('smoshian', 0.032010
[('keystroke', 0.0011565468814539446), ('boraz', 0.0011657258249575474), ('smoshian', 0.0014319151865620266), ('brian8907', 0
```

The results shared many common items, compare the two I suppose the betweeness centrality is better because it calculate the centrality based on paths that went through it, while closeness measure on shortest path that can only reveal the access efficiency

Q4

(a)

```
from networkx.algorithms.clique import find_cliques

numClique=list(find_cliques(youtubeGraph))

numClique
```

```
[['', 'wisdomismisery'],
 ['pepper_10', 'justmepammy'],
 ['carlos__', 'plazanetwork'],
 ['carlos__', 'laquesefue'],
 ['plazanetwork', 'lauradark', 'dark_warlike'],
 ['plazanetwork', 'franmx'],
 ['plazanetwork', 'javier_af'],
 ['plazanetwork', 'tmeister'],
 ['plazanetwork', 'machelino'],
 ['plazanetwork', 'edalgomezn'],
 ['bostonmarketer', 'barbarahauck'],
 ['meaghery)', 'khuda1'],
 ['justkarl', 'growline'],
 ['nickolaswriter', 'zaibatsu'],
 ['nickolaswriter', 'brian8907'],
 ['nickolaswriter', 'phabi'],
 ['ladynin', 'laimaz'],
 ['kaytlynbrianne', 'smoshian'],
 ['imaginedpm', 'oliksi'],
 ['keystroke', 'kevinsoberg', 'jasper_j'],
 ['keystroke', 'jdscolam'],
 ['dullym', 'susannaz'],
 ['dullym', 'charlieprofit'],
 ['robot117', 'cnnfail'],
 ['aznasser', 'steinboy'],
 ['naomihart', 'janlot56'],
 ['facso', 'georch'],
 ['nuztoradrt', 'buthaina'],
 ['perkinsb', 'smoshian'],
 ['mschieck', 'iranelectionthanks'],
 ['thegwenster', 'don_crowther'],
 ['mio', 'crashover'],
 ['chikatze', 'copakennet'],
 ['growline', 'ozuckan'],
 ['growline', 'jdscolam'],
 ['sogeshirts', 'deathwish808'],
 ['_vinyltap', 'justmepammy'],
 ['worldlistener', 'annschilling'],
 ['dfizzy', 'saltyproduction'],
 ['corinnamilborn', 'svejk'],
 ['javier_af', 'laquesefue'],
 ['famously', 'vtgt'],
 ['famouslv', "famouslv's"],
 ['lanicew88', 'makael86'],
 ['lanicew88', 'theskorpion'],
 ['lanicew88', 'misstp90'],
 ['lanicew88', 'brianbee'],
 ['lanicew88', 'sweetaddictions'],
 ['jkimisyellow', 'smoshian'],
 ['innovation', 'formerstortje'],
 ['vmlemon', 'dcfemella'],
 ['elocio', 'justmepammy'],
 ['jamesakersjr', 'fleetadmiralj'],
 ['stopthedictator', 'elliotbrk'],
 ['stopthedictator', 'zackstanton'],
 ['stopthedictator', 'amberrl'],
 ['stopthedictator', 'brneyesuss'],
 ['stopthedictator', 'jhwilensky'],
 ['betterbizideas', 'dpbkmb'],
 ['avmeyer', 'vanallen78'],
 ['avmeyer', '4dogz'],
 ['susannaz', 'twazzup)'],
 ['susannaz', 'janlot56'],
 ["pbanolka's", 'banolka'],
 ['colearchambault', 'cole2026'],
 ['ossguy', 'foolip'],
 ['ossguy', "cdibona's"],
 ['misssideways', 'doubledown_insl'],
 ['lmighton', 'caferozella'],
 ['lmangueart', 'filmclassics'],
 ['michael_sauer', 'freshzweinull'],
 ['cdibona', 'buckybit'],
 ['plazanetwork)', 'georch'],
 ['stanwong27', 'filmclassics'],
 ['ozuckan', 'hugobiwan', 'phabi'],
 ['ozuckan', 'freeman59'],
 ['ozuckan', 'olivierl'],
 ['vasalisa', 'dabloguiman'],
 ['justmepammy', 'ginthegin'],
 ['justmepammy', 'debra47'],
 ['yagglo', 'lvenselaar'],
 ['machelino', 'laquesefue'],
 ['cole2026', 'mohamadreza'],
 ['danineteen', 'smoshian'],
 ['paulodemoc', 'bubbaloox3'],
 ['fraeulein_m', 'grmon'],
 ['cafepressmemaws', 'dabloguiman', 'sblanco'],
 ['cafepressmemaws', 'mariabarrett'],
 ['buthaina', 'stopahmadi'],
 ['buthaina', 'harkatur'],
 ['buthaina', 'irannewsnow'],
 ['lauradark', 'laquesefue'],
 ['jakrose', 'swichi293'],
 ['flashability', 'marcned'],
 ['scottmonaco', 'mediapost)'],
 ['sandragaspar', 'ocean_raven'],
 ['sandragaspar', 'stopthedictato'],
 ['delsquacho', 'deathwish808'],
 ['twitt_consult', 'conceptionblog'],
 ['bennybtl', 'swichi293'],
 ['arturs', 'laimaz'],
 ['jaytheblogger', 'oscargodson'],
 ['jaytheblogger', 'yeco)'],
 ['twiterbh)', 'cleomorgause'],
 ['warholreject', 'jrea'],
 ['tole_cover', 'robertashley'],
 ['tole_cover', 'shawnelliott'],
 ['swichi293', 'culinaryculture'],
 ['akisamexamaya', 'svejk'],
 ['wisdomismisery', 'mmefaerie'],
 ['dpbkmb', 'mashable'],
 ['barbarahauck', 'sarahmerion'],
 ['magiccitymayhem', 'justmikeyhrc'],
 ['boraz', 'clung'],
 ['boraz', 'maureenogle'],
 ['boraz', 'tonycomstock'],
 ['boraz', 'drisis'],
 ['boraz', 'jeniburns'],
 ['boraz', 'edtroy'],
 ['boraz', 'carltonhoytphd'],
 ['boraz', 'leshemmings', 'ornette303'],
 ['lizmac57', 'dwahmadinejad', 'zaibatsu'],
 ['lizmac57', 'dwahmadinejad', 'brian8907'],
 ['copakennet', 'rosenkrieger'],
 ['mashable', 'mynumberone1988'],
 ['adam_walters', 'deathwish808', 'jenocide312'],
 ['wendybama', 'phabi'],
 ['viper82', 'hugobiwan', 'phabi'],
 ['viper82', 'freeman59'],
 ['viper82', 'olivierl'],
 ['smoshian', 'adamtube'],
 ['smoshian', 'musicloverxd'],
 ['smoshian', 'bensstudio'],
 ['smoshian', 'erinisbeastly'],
 ['smoshian', 'katiedidnot'],
 ['hammarstrand', 'keyvan'],
 ['emccareers', 'smith_cameron'],
 ['marihani', 'serbuvlad', 'janlot56'],
 ['franosch', 'zackstanton'],
 ['jsbond', 'zaibatsu'],
 ['jsbond', 'ghbetbeze'],
 ['jsbond', 'brian8907'],
 ['jsbond', 'phabi'],
 ['guillaume_w', 'avidya'],
 ['smileofcrash', 'nomemoryjill'],
 ['regretful', 'deathwish808'],
 ['yokibics', 'zaibatsu'],
 ['yokibics', 'brian8907'],
 ['pollypearson', 'smith_cameron'],
 ['magixblog', 'derwebarchitekt'],
 ['xponent', 'brian8907'],
 ['thamesstreet', 'smosh'],
 ['jslefanu', 'brneyesuss'],
 ['tidewaters', 'mynumberone1988'],
 ['minsd', 'dabloguiman', 'romout'],
 ['prcizmadia', 'danawalker'],
 ['iforia', 'derwebarchitekt'],
 ['iforia', 'meringer'],
 ['danawalker', 'jacobdiggle'],
 ['mitchellmckenna', 'cwmacdon)'],
 ['tdaxp', 'latvianman101'],
 ['potent_one', 'mynumberone1988'],
 ['beaugiles)', 'duncn'],
 ['ikriz', 'newenglanddeb'],
 ['yelperalp', 'rohansingh'],
 ['lextar', 'ubahnverleih'],
 ['farshadns', 'dabloguiman'],
 ['tmeister', 'laquesefue'],
 ['mynumberone1988', 'quitzlipochtli'],
 ['mynumberone1988', 'msvfab'],
 ['zaibatsu', 'hugobiwan', 'toniodelaconcha'],
 ['bensstudio', 'smosh'],
 ['khuda1', 'flipbooks'],
 ['jdscolam', 'filthyhipster'],
 ['brneyesuss', 'rakidd'],
 ['caferozella', 'vasalisa:rt'],
 ['caferozella', 'dabloguiman'],
 ['m8b_', 'kopfkribbeln'],
 ['zackstanton', 'notcobmiscavige'],
 ['maverickny', 'carltonhoytphd'],
 ['maverickny', 'clung'],
 ['justinvincent', 'undisco'],
 ['hugobiwan', 'toniodelaconcha', 'brian8907'],
 ['hugobiwan', 'toniodelaconcha', 'phabi'],
 ['mnrmg', 'mikepacker'],
 ['youtubehd', 'speedy3702'],
 ['mtlns', 'georch'],
 ['nut_cookie', 'stopthedictato'],
 ['katgib', 'joemescher'],
 ['laquesefue', 'edalgomezn'],
 ['deathwish808', 'ethanjaynes', 'beshirthappy'],
 ['discokevin', 'wick0r'],
 ['egilfujikawanes', 'mynumberone1988:shame'],
 ['natachaqs', 'nathalto'],
 ['eljerrywhite', 'georch'],
 ['iranelectionthx', 'ktrader'],
 ['critter8875', 'edtroy'],
 ['manymanypeople', 'clung'],
 ['manymanypeople', 'maureenogle'],
 ['manymanypeople', 'drisis'],
 ['manymanypeople', 'jeniburns'],
 ['manymanypeople', 'edtroy'],
 ['manymanypeople', 'carltonhoytphd'],
 ['manymanypeople', 'tonycomstock'],
 ['m_n_silva', 'lpedromachado'],
 ['dabloguiman', 'avidya'],
 ['prezford', 'stopahmadi'],
 ['maureenoglert', 'tonycomstock']]
```

```
len(numClique)
```

```
208
```

```
maxClique=0
for item in numClique:
  cliqueLength=len(item)
  if cliqueLength>maxClique:
    maxClique=cliqueLength

print(maxClique)
```

```
3
```

```
nodeClique={}
for item in youtubeGraph.nodes:
  for clique in numClique:
    if item in clique:
      if item in nodeClique.keys():
        nodeClique[item]+=1
      else:
        nodeClique[item]=1


nodeClique
```

```
{'': 1,
 '4dogz': 1,
 '_vinyltap': 1,
 'adam_walters': 1,
 'adamtube': 1,
 'akisamexamaya': 1,
 'amberrl': 1,
 'annschilling': 1,
 'arturs': 1,
 'avidya': 2,
 'avmeyer': 2,
 'aznasser': 1,
 'banolka': 1,
 'barbarahauck': 2,
 'beaugiles)': 1,
 'bennybtl': 1,
 'bensstudio': 2,
 'beshirthappy': 1,
 'betterbizideas': 1,
 'boraz': 8,
 'bostonmarketer': 1,
 'brian8907': 6,
 'brianbee': 1,
 'brneyesuss': 3,
 'bubbaloox3': 1,
 'buckybit': 1,
 'buthaina': 4,
 'cafepressmemaws': 2,
 'caferozella': 3,
 'carlos__': 2,
 'carltonhoytphd': 3,
 'cdibona': 1,
 "cdibona's": 1,
 'charlieprofit': 1,
 'chikatze': 1,
 'cleomorgause': 1,
 'clung': 3,
 'cnnfail': 1,
 'cole2026': 2,
 'colearchambault': 1,
 'conceptionblog': 1,
 'copakennet': 2,
 'corinnamilborn': 1,
 'crashover': 1,
 'critter8875': 1,
 'culinaryculture': 1,
 'cwmacdon)': 1,
 'dabloguiman': 6,
 'danawalker': 2,
 'danineteen': 1,
 'dark_warlike': 1,
 'dcfemella': 1,
 'deathwish808': 5,
 'debra47': 1,
 'delsquacho': 1,
 'derwebarchitekt': 2,
 'dfizzy': 1,
 'discokevin': 1,
 'don_crowther': 1,
 'doubledown_insl': 1,
 'dpbkmb': 2,
 'drisis': 2,
 'dullym': 2,
 'duncn': 1,
 'dwahmadinejad': 2,
 'edalgomezn': 2,
 'edtroy': 3,
 'egilfujikawanes': 1,
 'eljerrywhite': 1,
 'elliotbrk': 1,
 'elocio': 1,
 'emccareers': 1,
 'erinisbeastly': 1,
 'ethanjaynes': 1,
 'facso': 1,
 'famouslv': 2,
 "famouslv's": 1,
 'farshadns': 1,
 'filmclassics': 2,
 'filthyhipster': 1,
 'flashability': 1,
 'fleetadmiralj': 1,
 'flipbooks': 1,
 'foolip': 1,
 'formerstortje': 1,
 'fraeulein_m': 1,
 'franmx': 1,
 'franosch': 1,
 'freeman59': 2,
 'freshzweinull': 1,
 'georch': 4,
 'ghbetbeze': 1,
 'ginthegin': 1,
 'grmon': 1,
 'growline': 3,
 'guillaume_w': 1,
 'hammarstrand': 1,
 'harkatur': 1,
 'hugobiwan': 5,
 'iforia': 2,
 'ikriz': 1,
 'imaginedpm': 1,
 'innovation': 1,
 'iranelectionthanks': 1,
 'iranelectionthx': 1,
 'irannewsnow': 1,
 'jacobdiggle': 1,
 'jakrose': 1,
 'jamesakersjr': 1,
 'janlot56': 3,
 'jasper_j': 1,
 'javier_af': 2,
 'jaytheblogger': 2,
 'jdscolam': 3,
 'jeniburns': 2,
 'jenocide312': 1,
 'jhwilensky': 1,
 'jkimisyellow': 1,
 'joemescher': 1,
 'jrea': 1,
 'jsbond': 4,
 'jslefanu': 1,
 'justinvincent': 1,
 'justkarl': 1,
 'justmepammy': 5,
 'justmikeyhrc': 1,
 'katgib': 1,
 'katiedidnot': 1,
 'kaytlynbrianne': 1,
 'kevinsoberg': 1,
 'keystroke': 2,
 'keyvan': 1,
 'khuda1': 2,
 'kopfkribbeln': 1,
 'ktrader': 1,
 'ladynin': 1,
 'laimaz': 2,
 'lanicew88': 5,
 'laquesefue': 6,
 'latvianman101': 1,
 'lauradark': 2,
 'leshemmings': 1,
 'lextar': 1,
 'lizmac57': 2,
 'lmangueart': 1,
 'lmighton': 1,
 'lpedromachado': 1,
 'lvenselaar': 1,
 'm8b_': 1,
 'm_n_silva': 1,
 'machelino': 2,
 'magiccitymayhem': 1,
 'magixblog': 1,
 'makael86': 1,
 'manymanypeople': 7,
 'marcned': 1,
 'mariabarrett': 1,
 'marihani': 1,
 'mashable': 2,
 'maureenogle': 2,
 'maureenoglert': 1,
 'maverickny': 2,
 'meaghery)': 1,
 'mediapost)': 1,
 'meringer': 1,
 'michael_sauer': 1,
 'mikepacker': 1,
 'minsd': 1,
 'mio': 1,
 'misssideways': 1,
 'misstp90': 1,
 'mitchellmckenna': 1,
 'mmefaerie': 1,
 'mnrmg': 1,
 'mohamadreza': 1,
 'mschieck': 1,
 'msvfab': 1,
 'mtlns': 1,
 'musicloverxd': 1,
 'mynumberone1988': 5,
 'mynumberone1988:shame': 1,
 'naomihart': 1,
 'natachaqs': 1,
 'nathalto': 1,
 'newenglanddeb': 1,
 'nickolaswriter': 3,
 'nomemoryjill': 1,
 'notcobmiscavige': 1,
 'nut_cookie': 1,
 'nuztoradrt': 1,
 'ocean_raven': 1,
 'oliksi': 1,
 'olivierl': 2,
 'ornette303': 1,
 'oscargodson': 1,
 'ossguy': 2,
 'ozuckan': 4,
 'paulodemoc': 1,
 "pbanolka's": 1,
 'pepper_10': 1,
 'perkinsb': 1,
 'phabi': 6,
 'plazanetwork': 7,
 'plazanetwork)': 1,
 'pollypearson': 1,
 'potent_one': 1,
 'prcizmadia': 1,
 'prezford': 1,
 'quitzlipochtli': 1,
 'rakidd': 1,
 'regretful': 1,
 'robertashley': 1,
 'robot117': 1,
 'rohansingh': 1,
 'romout': 1,
 'rosenkrieger': 1,
 'saltyproduction': 1,
 'sandragaspar': 2,
 'sarahmerion': 1,
 'sblanco': 1,
 'scottmonaco': 1,
 'serbuvlad': 1,
 'shawnelliott': 1,
 'smileofcrash': 1,
 'smith_cameron': 2,
 'smosh': 2,
 'smoshian': 9,
 'sogeshirts': 1,
 'speedy3702': 1,
 'stanwong27': 1,
 'steinboy': 1,
 'stopahmadi': 2,
 'stopthedictato': 2,
 'stopthedictator': 5,
 'susannaz': 3,
 'svejk': 2,
 'sweetaddictions': 1,
 'swichi293': 3,
 'tdaxp': 1,
 'thamesstreet': 1,
 'thegwenster': 1,
 'theskorpion': 1,
 'tidewaters': 1,
 'tmeister': 2,
 'tole_cover': 2,
 'toniodelaconcha': 3,
 'tonycomstock': 3,
 'twazzup)': 1,
 'twiterbh)': 1,
 'twitt_consult': 1,
 'ubahnverleih': 1,
 'undisco': 1,
 'vanallen78': 1,
 'vasalisa': 1,
 'vasalisa:rt': 1,
 'viper82': 3,
 'vmlemon': 1,
 'warholreject': 1,
 'wendybama': 1,
 'wick0r': 1,
 'wisdomismisery': 2,
 'worldlistener': 1,
 'xponent': 1,
 'yagglo': 1,
 'yeco)': 1,
 'yelperalp': 1,
 'yokibics': 2,
 'youtubehd': 1,
 'ytgt': 1,
 'zackstanton': 3,
 'zaibatsu': 5}
```

```
nodeLargest={}
for node in youtubeGraph.nodes:
  for clique in numClique:
    if node in clique:
      length=len(clique)
      if node in nodeLargest.keys():

        nodeLargest[node]=max(length,nodeLargest[node])
      else:
        nodeLargest[node]=length
print(nodeLargest)
```

```
{'speedy3702': 2, 'youtubehd': 2, 'mitchellmckenna': 2, 'cwmacdon)': 2, 'smith_cameron': 2, 'pollypearson': 2, 'emccareers': 2
```

(b)

The largest clique has the size of 3, and it stands for the size of a group people whom are connected from each other. From the size of the largest maximal clique containing each given node, we can see that the majority of the network nodes only contain a max clique of size two, which means people like to mention only one people in one tweet instead of many people.