**4. Betweenness Centrality**

characters = np.loadtxt('characters.txt',dtype=int)

comics = np.loadtxt('comics.txt',dtype=int)

B = nx.Graph()

B.add\_nodes\_from(characters,bipartite=0)

B.add\_nodes\_from(comics,bipartite=1)

h = nx.read\_edgelist('out.txt',create\_using=nx.Graph(),nodetype=int)

B.add\_edges\_from(h.edges)

#creating unipartite network out of bipartite network

G = bipartite.weighted\_projected\_graph(B,characters)

#counting betweenness centrality

betCent = nx.betweenness\_centrality(G)

#sorting betweenness centrality in descending order

betCent\_sorted=dict(sorted(betCent.items(), key=lambda item: item[1],reverse=True))

#sorting in a text file

with open('betCent\_sorted.txt','w') as f:

print(betCent\_sorted,file=f)

#getting top 100 nodes

N\_top=100

keys\_bet\_top=list(betCent\_sorted) [0:N\_top]

d = {}

with open("dict.txt",encoding="utf8") as file:

for line in file:

line = line.split(':')

if not line:

continue

d[line[0]] = line[1:]

top\_nodes = keys\_bet\_top

with open('top\_nodes.txt','w') as g:

for i in range(len(top\_nodes)):

print(top\_nodes[i],"\t:\t",d[str(top\_nodes[i])][0],file=g)