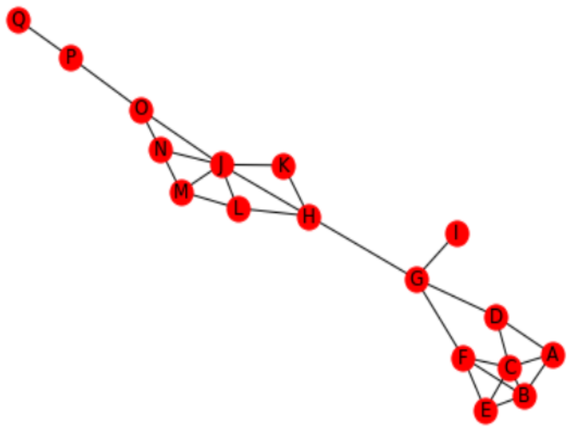
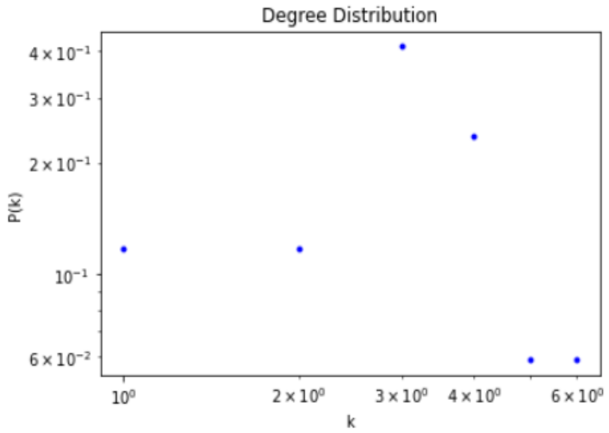
Computer Science and Engineering Department, SVNIT, Surat
Social Network Analysis
Assignment 4

U20CS005
BANSI MARAKANA

Generate the graph shown in figure 1 using Networkx and display the following network measures:
1) Degree Distribution
2) Degree centrality
3) Closeness centrality
4) Betweenness centrality
5) Clustering coefficient

```
import networkx as nx
import matplotlib.pyplot as plt
g = nx.Graph()
g.add_nodes_from(['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q'])
nx.add_cycle(g, ['G','F','E','B','A','D'])
nx.add_cycle(g, ['J','K','H','L','M','N','O'])
g.add_edges_from([("E","C"),("F","B"),("F","C"),("B","C"),("A","C"),("C","D"),("G","I"),("G","H"),("J","H"),("J","L"),("J","M"),("J","N"),("O","P"),("P","Q")])
nx.draw(g, with_labels = True, node_color ='red',node_size = 230)
import numpy as np
degree_hist=nx.degree_histogram(g)
degree_hist=np.array(degree_hist,dtype=float)
degree_prob=degree_hist/g.number_of_nodes()
plt.loglog(np.arange(degree_prob.shape[0]),degree_prob,'b.')
plt.xlabel('k')
plt.ylabel("P(k)")
plt.title('Degree Distribution')
plt.show()

deg_centrality = nx.degree_centrality(g)
print("Degree Centrality: ")
print(deg_centrality)
close_centrality = nx.closeness_centrality(g)
print("\nCloseness Centrality: ")
print(close_centrality)
bet_centrality = nx.betweenness_centrality(g, normalized = True, endpoints = False)
print("\nBetweenness Centrality: ")
print(bet_centrality)

print("\nClustering Coefficient:")
print(nx.clustering(g))
```

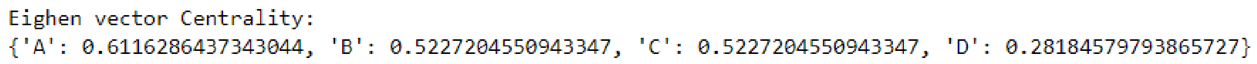




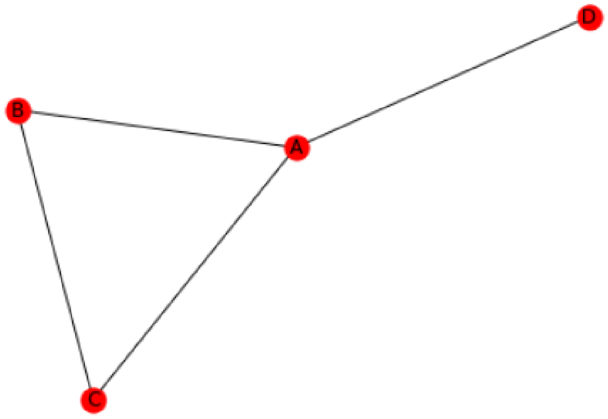
```
Degree Centrality:
{'A': 0.1875, 'B': 0.25, 'C': 0.3125, 'D': 0.1875, 'E': 0.1875, 'F': 0.25, 'G': 0.25, 'H': 0.25, 'I': 0.0625, 'J': 0.375, 'K':
0.125, 'L': 0.1875, 'M': 0.1875, 'N': 0.1875, 'O': 0.1875, 'P': 0.125, 'Q': 0.0625}

Closeness Centrality:
{'A': 0.2909090909090909, 'B': 0.2962962962962963, 'C': 0.3018867924528302, 'D': 0.35555555555555557, 'E': 0.2909090909090909,
'F': 0.36363636363636365, 'G': 0.4444444444444444, 'H': 0.45714285714285713, 'I': 0.3137254901960784, 'J': 0.42105263157894735,
'K': 0.36363636363636365, 'L': 0.37209302325581395, 'M': 0.3137254901960784, 'N': 0.32653061224489793, 'O': 0.3333333333333333,
'P': 0.26229508196721313, 'Q': 0.21052631578947367}

Betweenness Centrality:
{'A': 0.004166666666666667, 'B': 0.008333333333333333, 'C': 0.025, 'D': 0.1375, 'E': 0.0, 'F': 0.22916666666666666, 'G': 0.5791
666666666666, 'H': 0.5375, 'I': 0.0, 'J': 0.425, 'K': 0.0, 'L': 0.0375, 'M': 0.004166666666666667, 'N': 0.0125, 'O': 0.23333333
333333334, 'P': 0.125, 'Q': 0.0}

Clustering Coefficient:
{'A': 0.6666666666666666, 'B': 0.6666666666666666, 'C': 0.5, 'D': 0.3333333333333333, 'E': 1.0, 'F': 0.5, 'G': 0, 'H': 0.333333
3333333333, 'I': 0, 'J': 0.3333333333333333, 'K': 1.0, 'L': 0.6666666666666666, 'M': 0.6666666666666666, 'N': 0.666666666666666
6, 'O': 0.3333333333333333, 'P': 0, 'Q': 0}
```

6) Generate the graph shown in figure 2 using Networkx and display its eigenvector centrality.

```
import networkx as nx
import matplotlib.pyplot as plt
g = nx.Graph()
g.add_nodes_from(['A','B','C','D'])
nx.add_cycle(g, ['C','B','A'])
g.add_edge("A","D")
nx.draw(g, with_labels = True, node_color ='red',node_size = 230)
eighen_centrality = nx.eigenvector_centrality(g,max_iter=50)
print("\nEighen vector Centrality: ")
print(eighen_centrality)
```

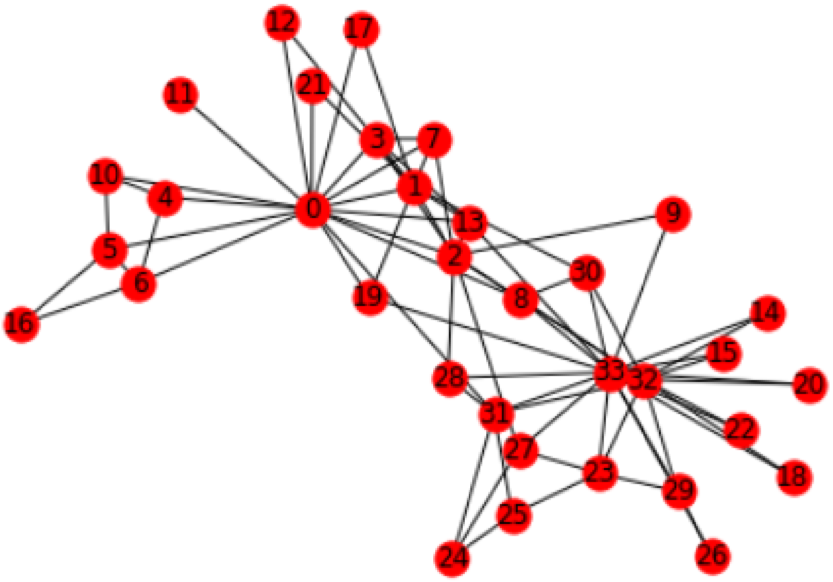


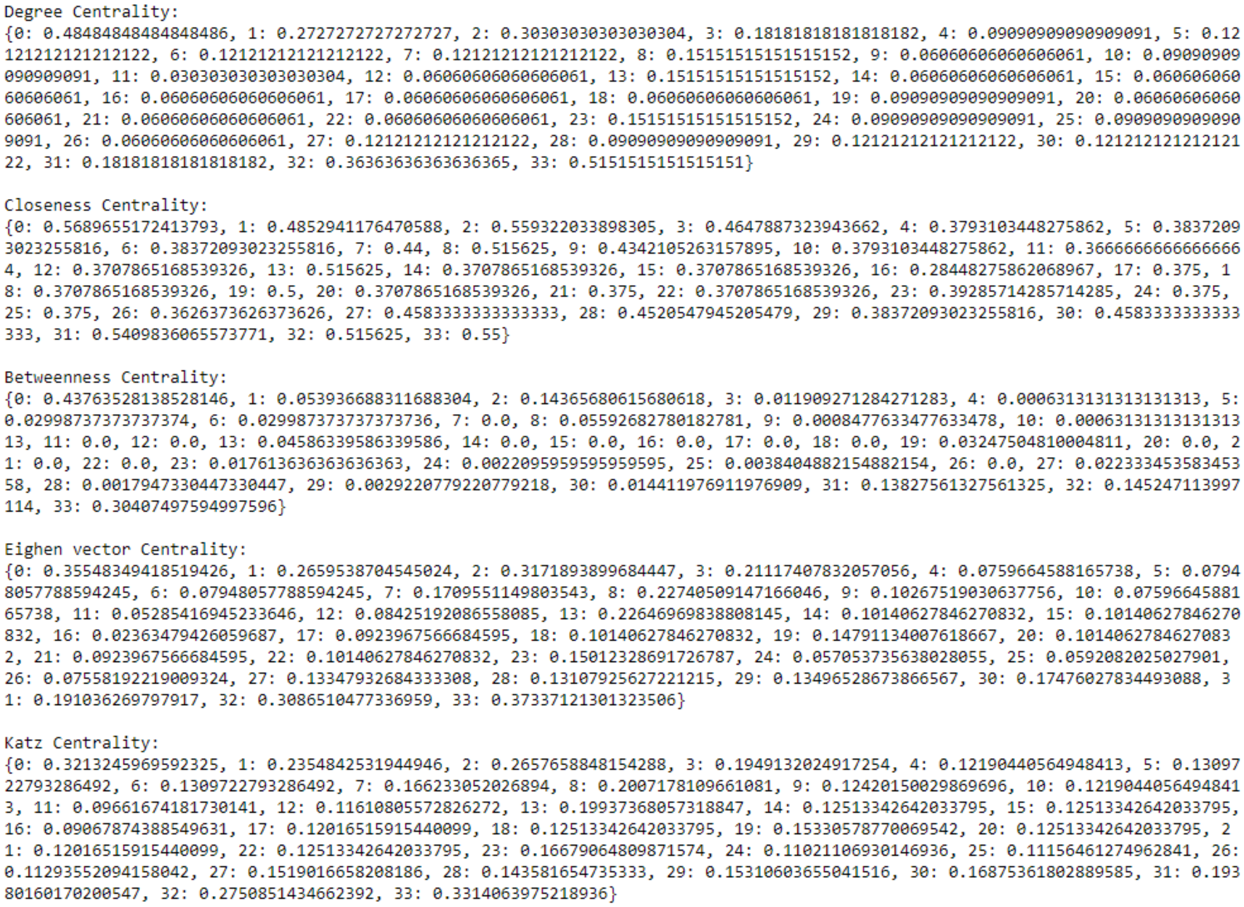
```
Eighen vector Centrality:
{'A': 0.6116286437343044, 'B': 0.5227204550943347, 'C': 0.5227204550943347, 'D': 0.28184579793865727}
```



7) Consider the famous social graph published in 1977 called Zachary's Karate Club graph. It is an in-built Graph in Networkx. Demonstrate all the centrality measures for this Graph in NetworkX.

```
import matplotlib.pyplot as plt
import networkx as nx
import math
g=nx.karate_club_graph()
nx.draw(g, with_labels = True, node_color ='red',node_size = 230)

deg_centrality = nx.degree_centrality(g)
print("Degree Centrality: ")
print(deg_centrality)
close_centrality = nx.closeness_centrality(g)
print("\nCloseness Centrality: ")
print(close_centrality)
bet_centrality = nx.betweenness_centrality(g, normalized = True, endpoints = False)
print("\nBetweenness Centrality: ")
print(bet_centrality)
eighen_centrality = nx.eigenvector_centrality(g,max_iter=50)
print("\nEighen vector Centrality: ")
print(eighen_centrality)
katz_centrality = nx.katz_centrality(g,alpha=0.1,beta=1.0,max_iter=50)
print("\nKatz Centrality: ")
print(katz_centrality)
```





```
Degree Centrality:
{0: 0.48484848484848486, 1: 0.2727272727272727, 2: 0.30303030303030304, 3: 0.18181818181818182, 4: 0.09090909090909091, 5: 0.12
121212121212122, 6: 0.12121212121212122, 7: 0.12121212121212122, 8: 0.15151515151515152, 9: 0.06060606060606061, 10: 0.09090909
090909091, 11: 0.030303030303030304, 12: 0.06060606060606061, 13: 0.15151515151515152, 14: 0.06060606060606061, 15: 0.060606060
60606061, 16: 0.06060606060606061, 17: 0.06060606060606061, 18: 0.06060606060606061, 19: 0.09090909090909091, 20: 0.06060606060
606061, 21: 0.06060606060606061, 22: 0.06060606060606061, 23: 0.15151515151515152, 24: 0.09090909090909091, 25: 0.0909090909090
9091, 26: 0.06060606060606061, 27: 0.12121212121212122, 28: 0.09090909090909091, 29: 0.12121212121212122, 30: 0.12121212121212121
22, 31: 0.18181818181818182, 32: 0.36363636363636365, 33: 0.5151515151515151}

Closeness Centrality:
{0: 0.5689655172413793, 1: 0.4852941176470588, 2: 0.559322033898305, 3: 0.4647887323943662, 4: 0.3793103448275862, 5: 0.3837209
3023255816, 6: 0.38372093023255816, 7: 0.44, 8: 0.515625, 9: 0.4342105263157895, 10: 0.3793103448275862, 11: 0.3666666666666666
4, 12: 0.3707865168539326, 13: 0.515625, 14: 0.3707865168539326, 15: 0.3707865168539326, 16: 0.28448275862068967, 17: 0.375, 1
8: 0.3707865168539326, 19: 0.5, 20: 0.3707865168539326, 21: 0.375, 22: 0.3707865168539326, 23: 0.39285714285714285, 24: 0.375,
25: 0.375, 26: 0.3626373626373626, 27: 0.4583333333333333, 28: 0.4520547945205479, 29: 0.38372093023255816, 30: 0.4583333333333
333, 31: 0.5409836065573771, 32: 0.515625, 33: 0.55}

Betweenness Centrality:
{0: 0.43763528138528146, 1: 0.053936688311688304, 2: 0.14365680615680618, 3: 0.011909271284271283, 4: 0.0006313131313131313, 5:
0.02998737373737374, 6: 0.029987373737373736, 7: 0.0, 8: 0.05592682780182781, 9: 0.0008477633477633478, 10: 0.0006313131313131313
13, 11: 0.0, 12: 0.0, 13: 0.04586339586339586, 14: 0.0, 15: 0.0, 16: 0.0, 17: 0.0, 18: 0.0, 19: 0.03247504810004811, 20: 0.0, 2
1: 0.0, 22: 0.0, 23: 0.017613636363636363, 24: 0.0022095959595959595, 25: 0.0038404882154882154, 26: 0.0, 27: 0.02233345358345
358, 28: 0.0017947330447330447, 29: 0.0029220779220779218, 30: 0.014411976911976909, 31: 0.13827561327561325, 32: 0.145247113997
114, 33: 0.30407497594997596}

Eighen vector Centrality:
{0: 0.35548349418519426, 1: 0.2659538704545024, 2: 0.3171893899684447, 3: 0.21117407832057056, 4: 0.0759664588165738, 5: 0.0794
8057788594245, 6: 0.07948057788594245, 7: 0.1709551149803543, 8: 0.22740509147166046, 9: 0.10267519030637756, 10: 0.07596645881
65738, 11: 0.05285416945233646, 12: 0.08425192086558085, 13: 0.22646969838808145, 14: 0.10140627846270832, 15: 0.10140627846270
832, 16: 0.02363479426059687, 17: 0.0923967566684595, 18: 0.10140627846270832, 19: 0.14791134007618667, 20: 0.1014062784627083
2, 21: 0.0923967566684595, 22: 0.10140627846270832, 23: 0.15012328691726787, 24: 0.057053735638028055, 25: 0.0592082025027901,
26: 0.07558192219009324, 27: 0.13347932684333308, 28: 0.13107925627221215, 29: 0.13496528673866567, 30: 0.17476027834493088, 3
1: 0.191036269797917, 32: 0.3086510477336959, 33: 0.37337121301323506}

Katz Centrality:
{0: 0.3213245969592325, 1: 0.2354842531944946, 2: 0.2657658848154288, 3: 0.1949132024917254, 4: 0.12190440564948413, 5: 0.13097
22793286492, 6: 0.1309722793286492, 7: 0.166233052026894, 8: 0.2007178109661081, 9: 0.12420150029869696, 10: 0.1219044056494841
3, 11: 0.09661674181730141, 12: 0.11610805572826272, 13: 0.19937368057318847, 14: 0.12513342642033795, 15: 0.12513342642033795,
16: 0.09067874388549631, 17: 0.12016515915440099, 18: 0.12513342642033795, 19: 0.15330578770069542, 20: 0.12513342642033795, 2
1: 0.12016515915440099, 22: 0.12513342642033795, 23: 0.16679064809871574, 24: 0.11021106930146936, 25: 0.11156461274962841, 26:
0.11293552094158042, 27: 0.1519016658208186, 28: 0.143581654735333, 29: 0.15310603655041516, 30: 0.16875361802889585, 31: 0.193
80160170200547, 32: 0.2750851434662392, 33: 0.3314063975218936}
```