VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Social Network Analytics Lab

Digital Assignment -2

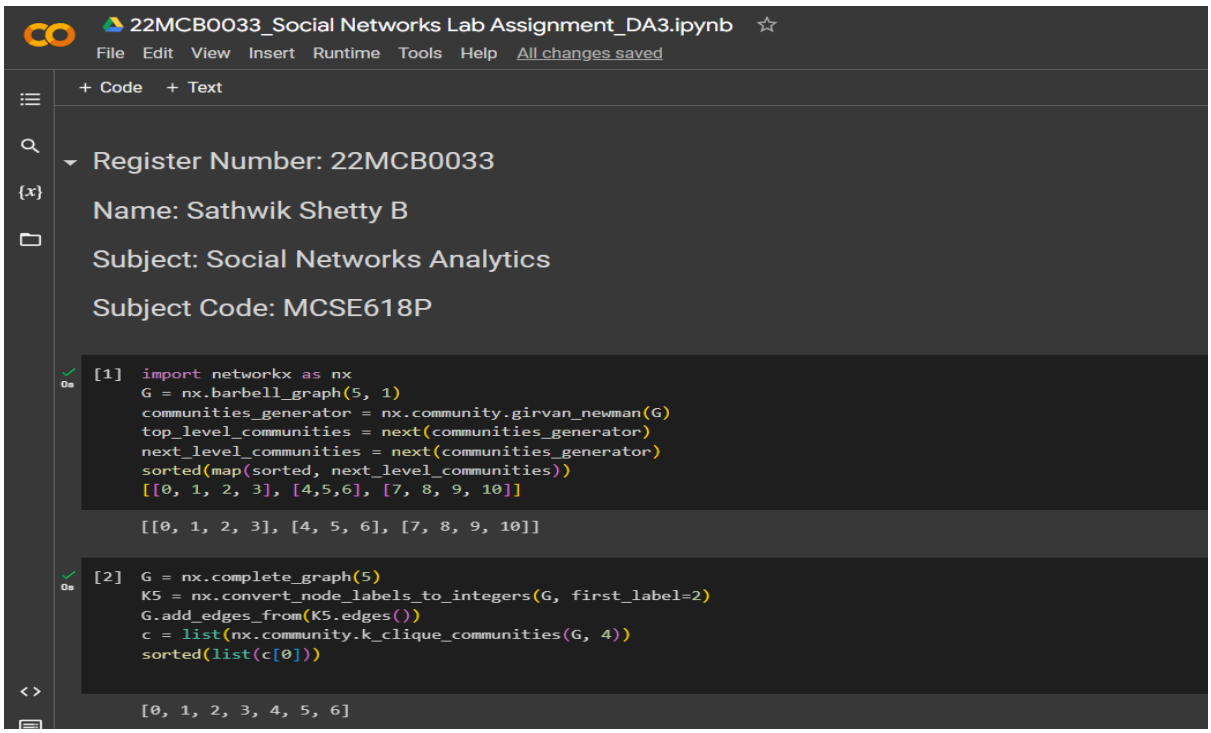
Register Number: 22MCB0033

Name: Sathwik Shetty B

Course Name: Social Network Analytics Lab

Course code: MCSE618P

Code and Output:



```
[3]  list(nx.community.k_clique_communities(G, 6))
```

```
[]
```

greedy_modularity_communities

```
[4]  G = nx.karate_club_graph()
     c = nx.community.greedy_modularity_communities(G)
     sorted(c[0])
```

```
[8, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]
```

naive_greedy_modularity_communities

```
[5]  G = nx.karate_club_graph()
     c = nx.community.naive_greedy_modularity_communities(G)
     sorted(c[0])
```

```
[8, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]
```

louvain_communities

```
[6]  import networkx as nx
     G = nx.petersen_graph()
     nx.community.louvain_communities(G, seed=123)
```

```
[{0, 4, 5, 7, 9}, {1, 2, 3, 6, 8}]
```

modularity

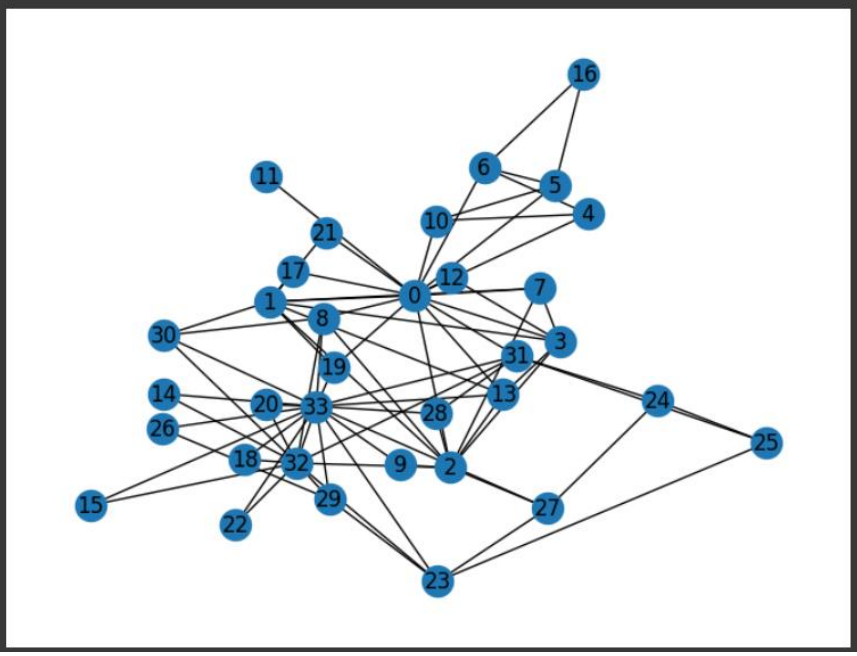
```
[7]  G = nx.barbell_graph(3, 0)
     nx.community.modularity(G, [{0, 1}, {2,3, 4, 5}])
```

```
0.12244897959183669
```

```
[8]  nx.community.modularity(G, nx.community.label_propagation_communities(G))
```

```
0.35714285714285715
```

```python
import matplotlib.pyplot as plt
%matplotlib inline
import networkx as nx
G = nx.karate_club_graph()
nx.draw_kamada_kawai(G, with_labels=True)
```




```python
# Calculating the betweenness centrality
btw_centrality = nx.algorithms.centrality.edge_betweenness_centrality(G)
# Sorting based on the betweenness centraliy and displaying the first 10 edges.
sorted(btw_centrality.items(), key = lambda item:item[1], reverse = True)[0:10]
```

```
[((0, 31), 0.1272599949070537),
 ((0, 6), 0.07813428401663695),
 ((0, 5), 0.07813428401663694),
 ((0, 2), 0.0777876807288572),
 ((0, 8), 0.07423959482783014),
 ((2, 32), 0.06898678663384543),
 ((13, 33), 0.06782389723566191),
 ((19, 33), 0.05938233879410351),
 ((0, 11), 0.058823529411764705),
 ((26, 33), 0.0542908072319837)]
```

```python
import networkx as nx

def girwan_newman(G, no_of_components_to_split):
    while no_of_components_to_split > nx.algorithms.components.number_connected_components(G):
        # Calculate the betweenness centrality
        btw_centrality = nx.algorithms.centrality.edge_betweenness_centrality(G)
        # Sort based on betweenness centrality
        sorted_edges = sorted(btw_centrality.items(), key=lambda item: item[1], reverse=True)
        print('Removing the edge', sorted_edges)
        # Remove edge with the highest centrality
        G.remove_edge(*sorted_edges[0][0])

    # Check if graph is split
    if no_of_components_to_split <= nx.algorithms.components.number_connected_components(G):
        # Plot the graph with nodes in different colors for each community
        nx.draw_spring(G, with_labels=True)

    # Return a list of nodes in each community
    list_of_nodes = [c for c in sorted(nx.connected_components(G), key=len, reverse=True)]
    return list_of_nodes
```

```python
[12] G = nx.karate_club_graph()
     communities = girwan_newman(G, 2)
     communities
```

```
Removing the edge [((0, 31), 0.1272599949070537), ((0, 6), 0.07813428401663695), ((0, 5), 0.07813428401663694), ((0, 2), 0.0777876807288572), ((0, 8), 0.07423959482783014), ((2, 32), 0.06898678663384543), (
Removing the edge [((0, 2), 0.11924273983097515), ((0, 8), 0.09923105217222859), ((2, 32), 0.08791752909399968), ((13, 33), 0.08660576895871015), ((0, 5), 0.07813428401663694), ((0, 6), 0.07813428401663694)
Removing the edge [((0, 8), 0.13782067605597018), ((13, 33), 0.10208533296768586), ((19, 33), 0.09142997525350469), ((0, 5), 0.07813428401663694), ((0, 6), 0.07813428401663694), ((0, 13), 0.0764549176313882
Removing the edge [((13, 33), 0.14617273782105492), ((19, 33), 0.13459061821754492), ((0, 13), 0.12320259255616288), ((0, 19), 0.1109476905953786), ((0, 5), 0.07813428401663695), ((0, 6), 0.07813428401663694
Removing the edge [((19, 33), 0.21966651886437982), ((0, 19), 0.19284112880583465), ((2, 32), 0.09492018676510654), ((0, 5), 0.07813428401663694), ((0, 6), 0.07813428401663694), ((2, 3), 0.06914311179017063
Removing the edge [((2, 32), 0.17861952861952871), ((1, 30), 0.15212489036018448), ((0, 1), 0.1298446651387828), ((1, 2), 0.10938304614775203), ((2, 3), 0.10332102537984895), ((30, 33), 0.09537107772401891),
Removing the edge [((1, 30), 0.25601957954899124), ((0, 1), 0.18108783917607446), ((30, 33), 0.13882381235322408), ((30, 32), 0.11202642673230905), ((2, 27), 0.10708485414367769), ((0, 5), 0.07813428401663636
Removing the edge [((1, 2), 0.19474153297682771), ((2, 8), 0.16948900772430184), ((2, 27), 0.16057635175282234), ((2, 3), 0.12700534759358287), ((2, 28), 0.11675579322638145), ((2, 7), 0.0935828877005347 6),
Removing the edge [((2, 3), 0.19191919191919182), ((2, 8), 0.16948900772430184), ((2, 27), 0.16057635175282234), ((2, 7), 0.1584967320261437), ((2, 13), 0.1584967320261437), ((2, 28), 0.11675579322638145),
Removing the edge [((2, 7), 0.25445632798573975), ((2, 13), 0.2544563279857 3975), ((2, 8), 0.16948900772430184), ((0, 7), 0.16889483065953653), ((0, 13), 0.16889483065953653), ((2, 27), 0.16057635175282234)
Removing the edge [((2, 13), 0.5080213903743315), ((0, 13), 0.33273915626856787), ((2, 8), 0.16948900772430184), ((2, 27), 0.16057635175282234), ((2, 28), 0.11675579322638145), ((1, 13), 0.10101010101010097
[{2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33},
 {0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21}]
```

```
[12] [{2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33},
      {0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21}]
```