CS532S19: Assignment #5

Due on Sunday, March 24, 2019

Nwala, Alexander C

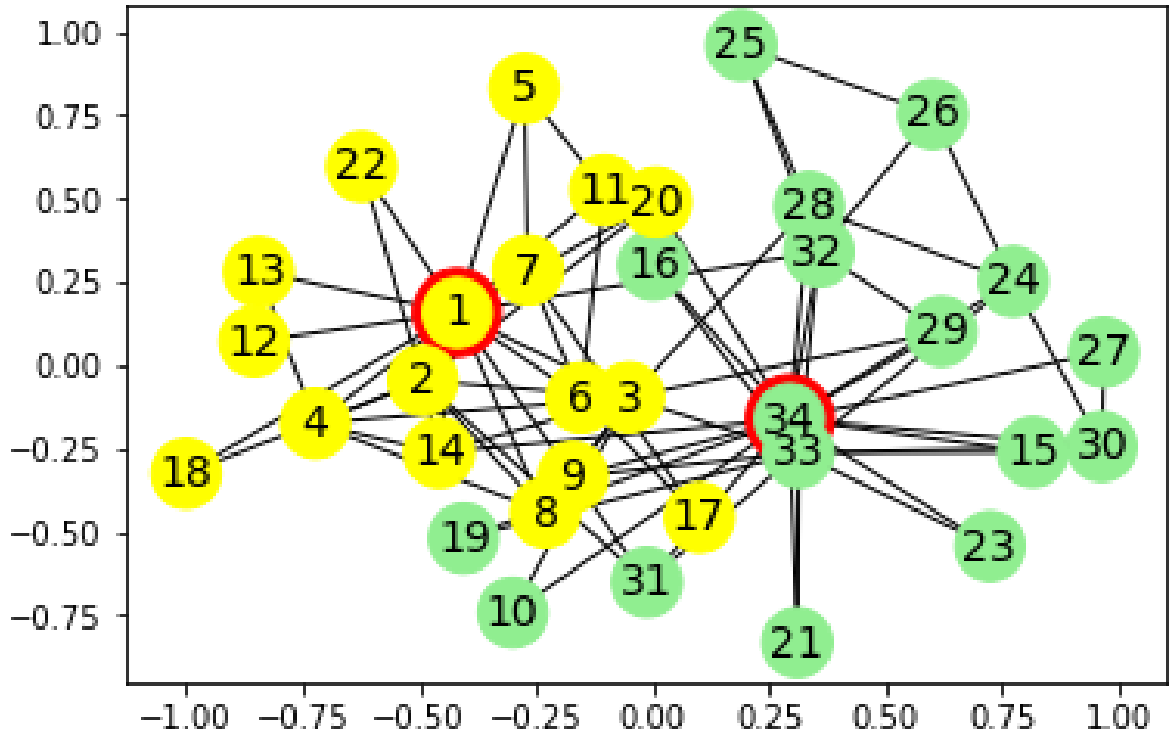
Giridharan Ganeshkumar

Question 1

We know the result of the Karate Club Zachary, 1977 split. Prove or disprove that the result of split could have been predicted by the weighted graph of social interactions. How well does the mathematical model represent reality?

We will start by drawing the original Zachary karate club network graph. In order to this we follow the below steps

1. Use of the Network x library to plot the graph
2. Get the graph from data from karate club graph method.
3. Make use of the spring layout to draw the graph using the data received in the previous step
4. Network x gives the option to draw nodes, edges and styles it accordingly
5. We also highlight Mr. Hi and Officer in the graph and color their network in the same colors.



Listing 1: Python Script

```
from matplotlib.pyplot import *
import networkx as nx
import math

G=nx.karate_club_graph()
(faction1_col, faction2_col) = ('yellow','lightgreen')
node_color = [faction1_col] * len(G.nodes())
node_dict = dict(G.nodes(data=True))
for n in G.nodes():
    if node_dict[n]['club'] == 'Officer':
```

Question 1 continued on next page...

```
        node_color[n] = faction2_col

pos = nx.spring_layout(G,scale=0.2)
new_labels = dict((x,x + 1) for x in G.nodes())
nx.draw_networkx_labels(G,pos,new_labels,font_size=14,font_color='black')
nx.draw_networkx_nodes(G,pos,{0:0,33:33},node_color=['red','red'],node_size
    =800)
nx.draw_networkx_nodes(G,pos,new_labels,node_color=node_color,node_size
    =500)
nx.draw_networkx_edges(G,pos)
```

Analysis on if we can make predictions of the karate club split that happened using the weighted graph of social interactions

1. The analysis starts by identifying the algorithm to be used to split the graph into two.
2. There are lot of algorithms that splits the graph based on the number of edges, the weight of the edges and other considerations based on the scenario.
3. For the given problem statement the algorithm we choose to split the graph into two communities is the girvan newman algorithm.
4. Network x provides a method that implements the girvan newman algorithm and splits the graph into to two communities.
5. The goal of this problem is to however go through the actual implementation of the algorithm which helps in explaining the split calculation and the theory behind the split of each iteration.
6. To do this there needs to be modification to the girvan newman provided by network x. We take the method implementation and changed it to fit for the the requirement. Below is the code that is modified.

Listing 2: Python Script

```
import networkx as nx
import operator
import matplotlib.pyplot as plt
import pylab as p

def girvan_newman (G):

    if len(G.nodes()) == 1:
        return [G.nodes()]

    def find_best_edge(G0):
        eb = nx.edge_betweenness_centrality(G0)
        return sorted(eb.items(), key = lambda x: float(x[1]), reverse =
            True)[0][0]

    components = nx.connected_component_subgraphs(G)
    clusterCount = 0
    while sum(1 for x in components) == 1:
        G.remove_edge(*find_best_edge(G))
        components = nx.connected_component_subgraphs(G)
        pos = nx.spring_layout(G,k=0.25,iterations=20)
        new_labels = dict((x,x + 1) for x in G.nodes())
        nx.draw_networkx_labels(G,pos,new_labels,font_size=14,font_color='
```

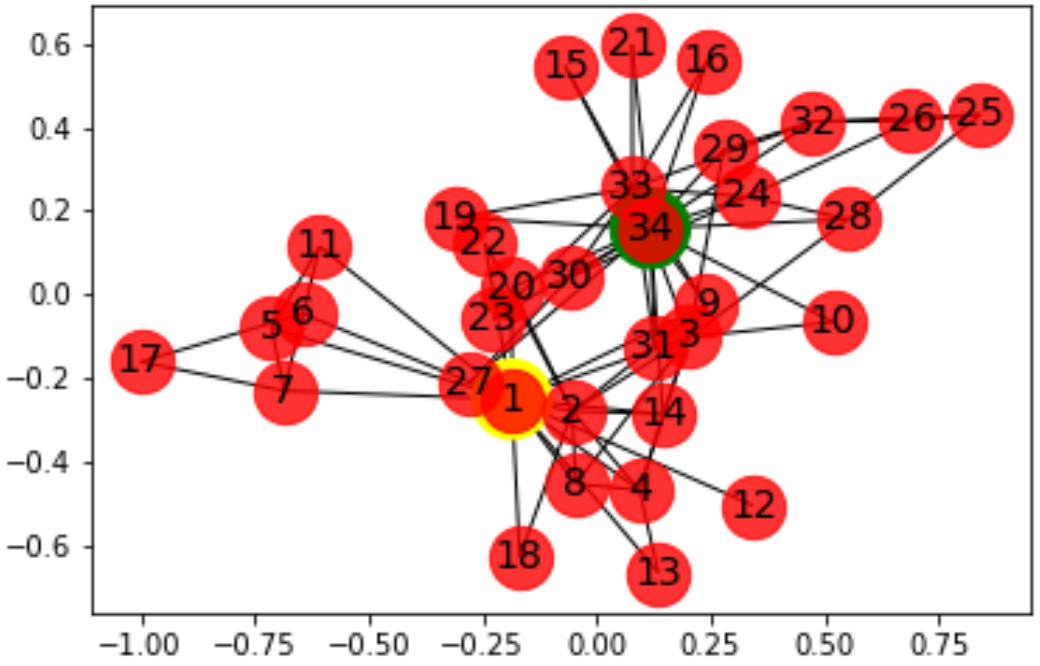
Question 1 continued on next page...

```
            black')
        nx.draw_networkx_nodes(G,pos,{0:0,33:33},node_color=['yellow','
            green'],node_size=800)
        # Now draw all the nodes, including leaders, using faction color
            scheme.
        nx.draw_networkx_nodes(G,pos,new_labels,node_color='r',node_size
            =500,alpha=0.8)
        nx.draw_networkx_edges(G,pos)
        clusterCount = clusterCount +1
        output = "C:\\Karate\\Iteration#%(id)02d.png" % {"id": clusterCount
            }
        p.savefig(output)
        p.close()
    result = [c.nodes() for c in components]

    for c in components:
        result.extend(girvan_newman(c))

    return result

G = nx.karate_club_graph()
kn = girvan_newman(G)
```

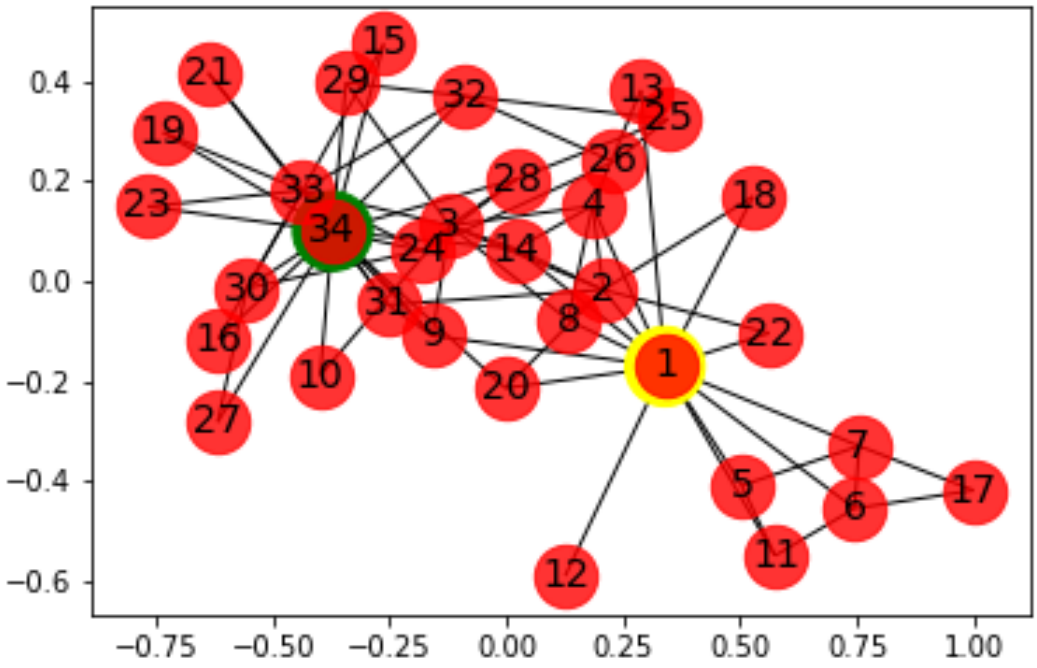
1. The above code is a copy of girvan newman method but modified, below we detail the modification to the code and also brief the advantages and disadvantage of the model and how close is to the the actual split that happened.
2. Get the graph from data from karate club graph method and store it in a variable G and pass it the girvan newman method.
3. Important modification to the method include to first fix the changes due to deprecated functions
4. Inside the method the important function is find best edge method that returns the edge with maximum edge betweenness in the current data set.
5. The method starts with calling the networkx method that computes the edge betweennness for all the data in the set.
6. The next step is to sort the dictionary values in descending order using the sorted function in operator library.
7. The function returns the first dictionary pair which is in turn the maximum in the list.
8. The next step is to remove the edge, in order to do this the remove edge method provided by network x is used.
9. The removal is done in a loop and the loop is kept active until the removal results in two separate communities.
10. As soon as the edge is removed, a graph is plotted with the current data set and links.
11. This is a recursive function with the check in the while loop.
12. The karate club goes through eleven iterations until the graph splits into two.
13. As seen the results of the mathematical model can be close to the actual result. Although there are lot of pros and cons to the model.

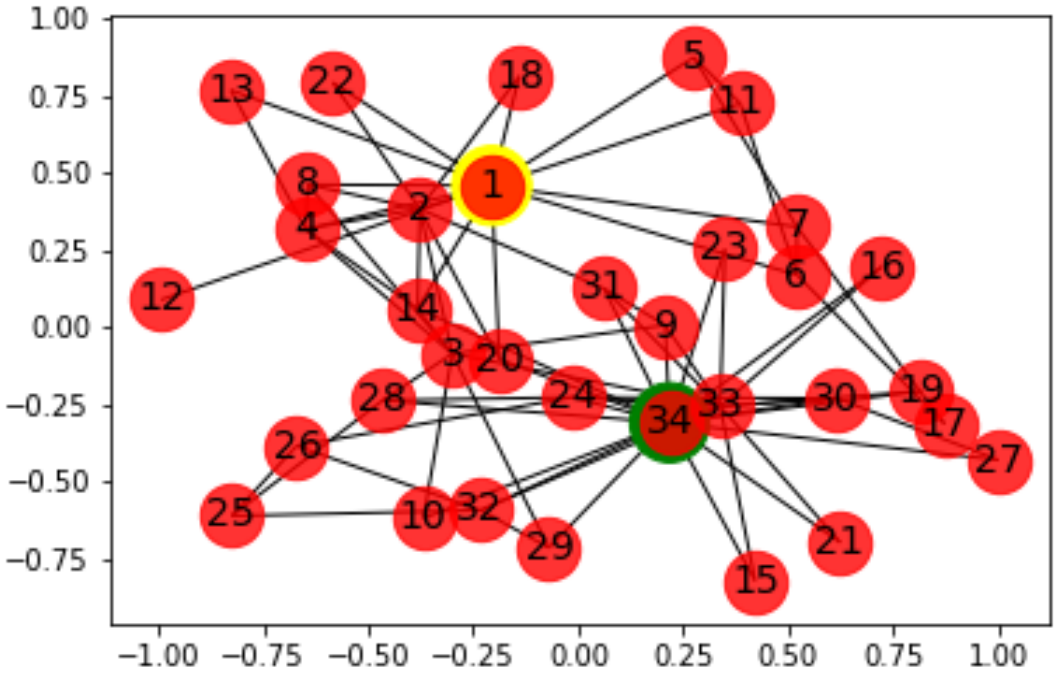
Question 1 continued on next page...

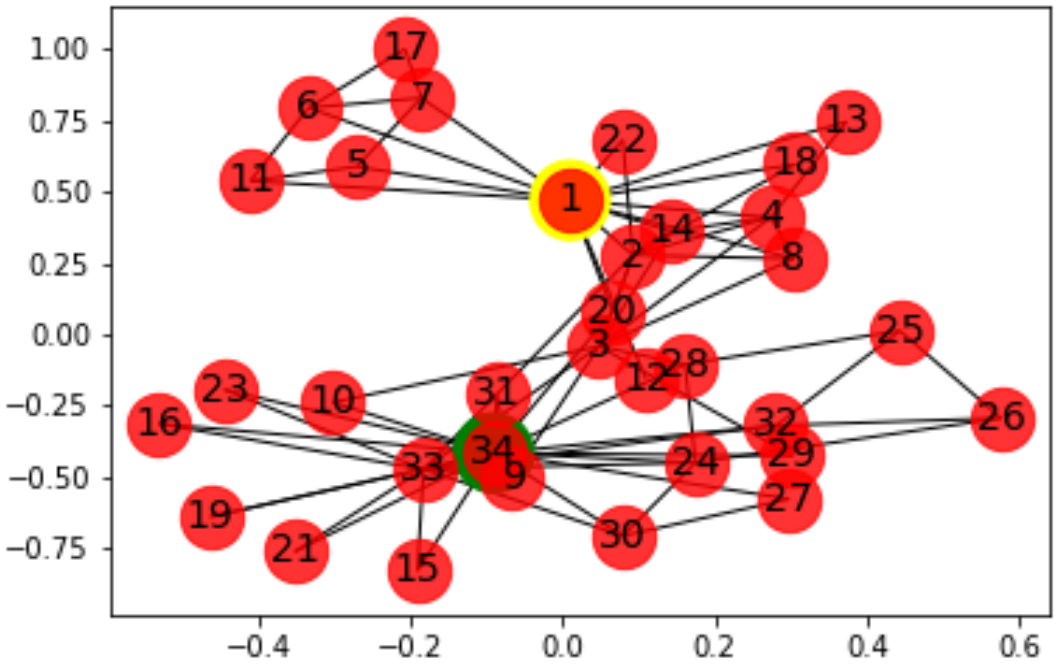
14. The main advantages is the ability to predict system behavior, the predictions are useful.
15. The predictions can be accurate based on the assumption that are made.
16. The appropriate use of models and their output can contribute to effective decision makings
17. Models gives an abstraction over data with which inferences can be made
18. There are limitation though, the model assumes the future will be like the past
19. The sampling of data also contributes to errors
20. The usefulness of a model may be limited by its original purpose.
21. Its hard to know the accuracy of the models, the specific number with several digits does not mean its accurate. Precision sometime is not equivalent to accuracy of predictions.
22. Also with social network the behavior is volatile leads to additional approximation that needs to be made in the models.
23. Although these limitations are mostly challenges that the model needs to address but still servers to be a good way to forecast and decision making.

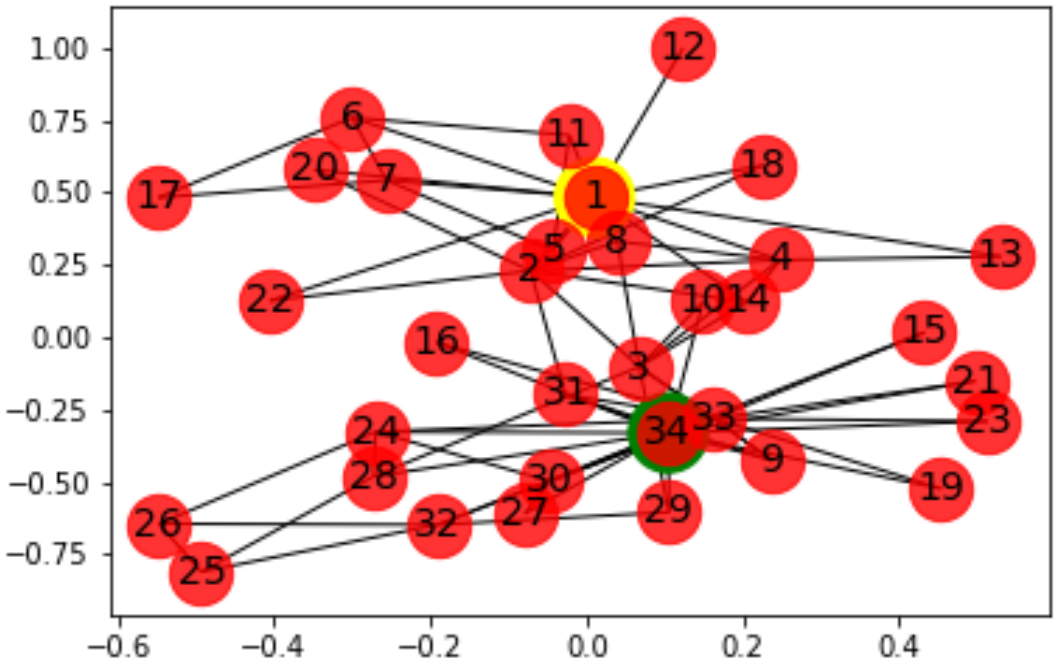


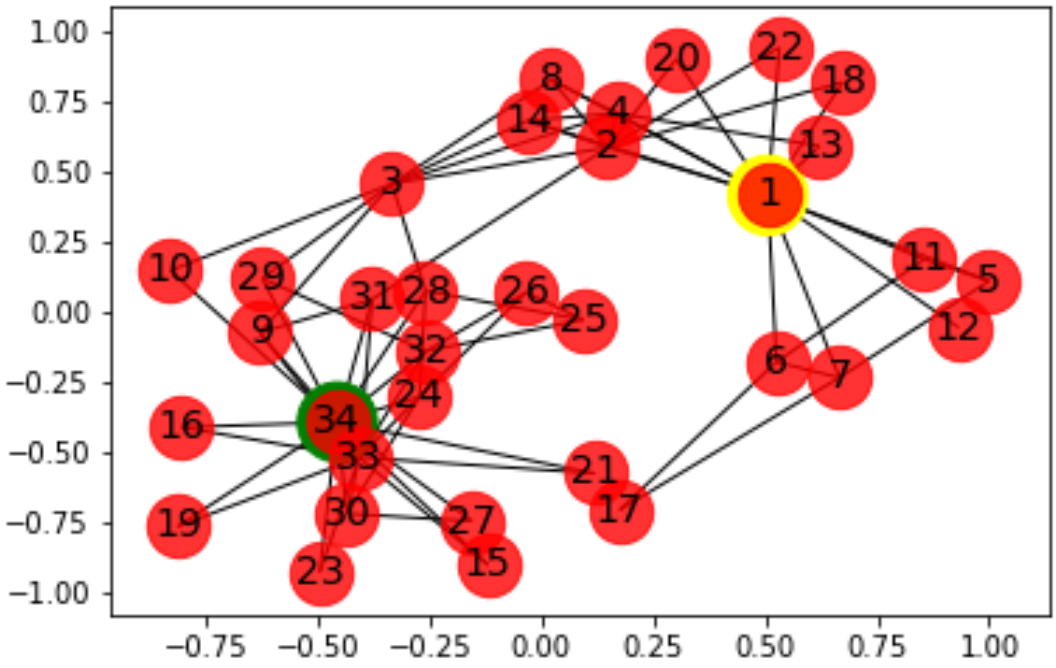
Question 1 continued on next page...

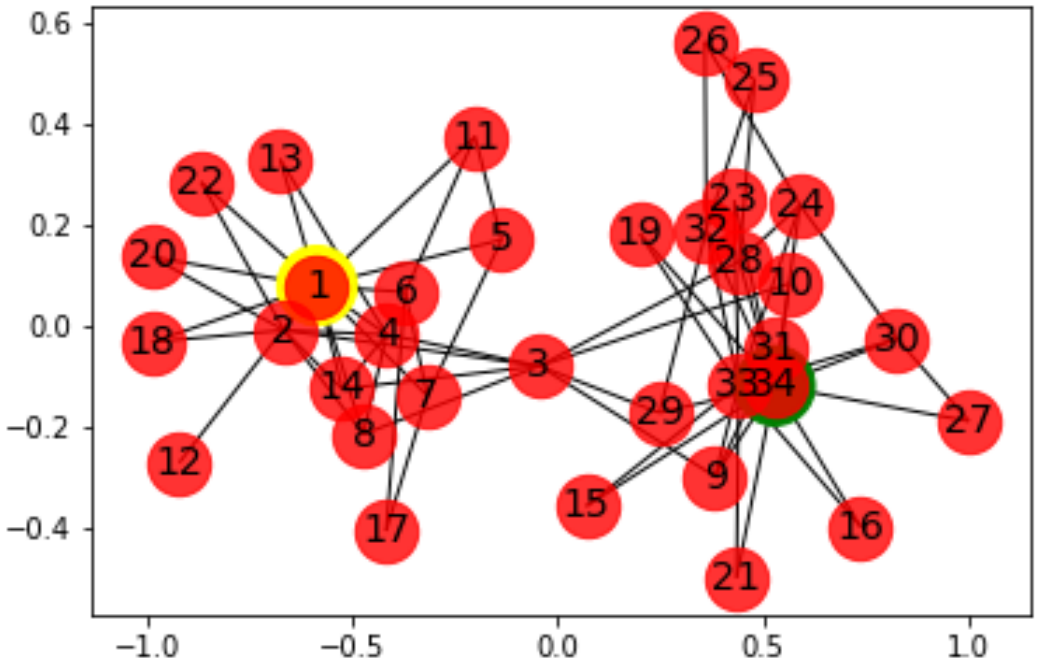

15
21
29
32
13
25
19
26
28
18
23
33
4
34
3
24
14
30
2
31
8
22
16
9
1
10
20
27
7
5
17
6
11
12
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
13
22
18
5
11
8
1
2
4
7
23
6
16
12
31
14
9
3
20
28
24
33
30
19
17
34
26
27
25
10
32
29
21
15

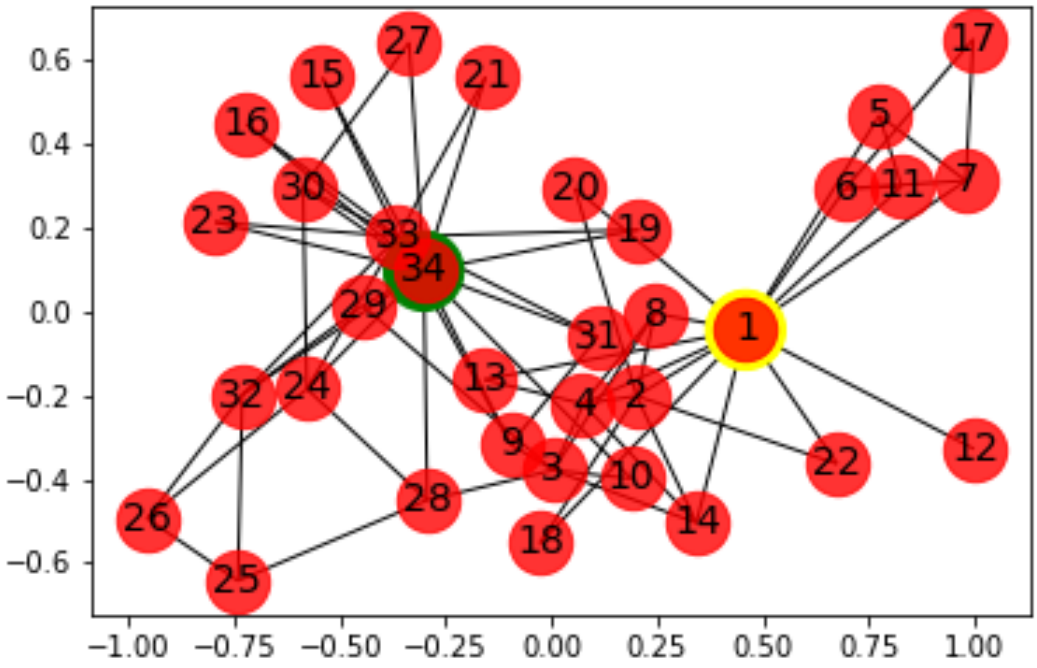

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.4
−0.2
0.0
0.2
0.4
0.6
17
6
7
11
5
22
13
18
1
4
14
2
8
20
3
25
28
12
23
10
31
32
26
16
34
33
9
24
29
27
19
21
15
30

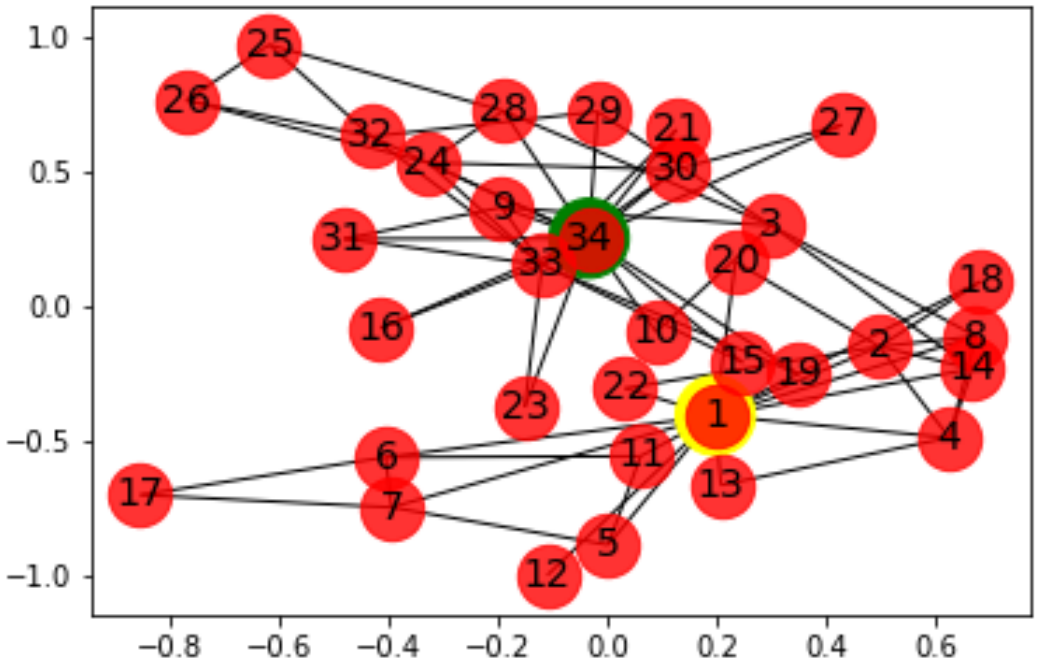

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.6
−0.4
−0.2
0.0
0.2
0.4
12
6
11
20
7
18
17
1
8
5
4
13
2
22
10
14
16
15
3
21
31
24
34
33
23
28
9
30
19
26
32
27
29
25



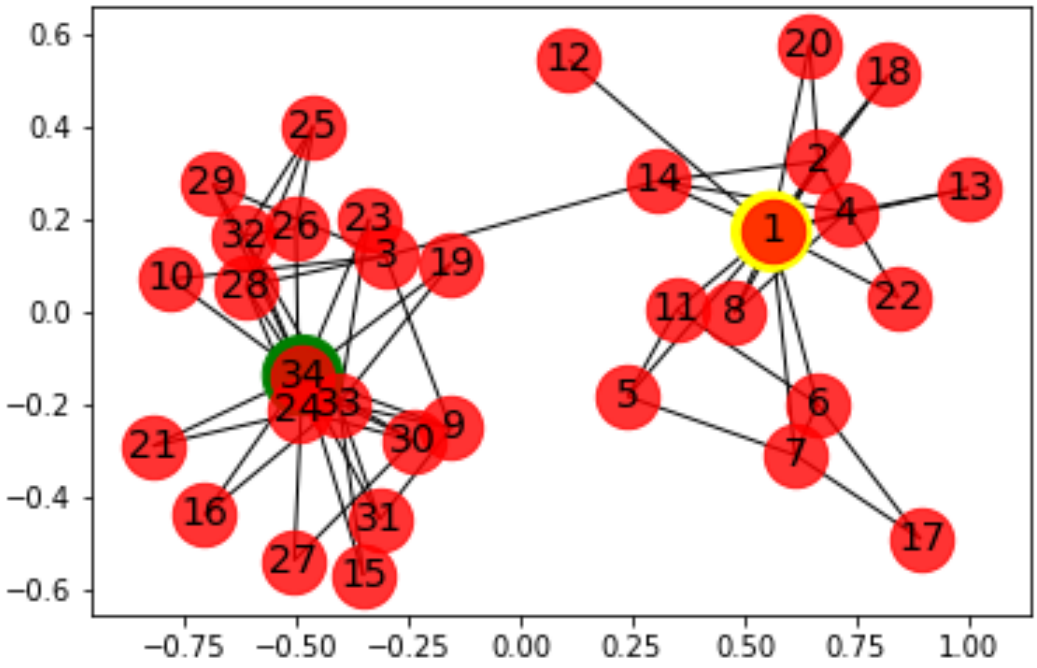


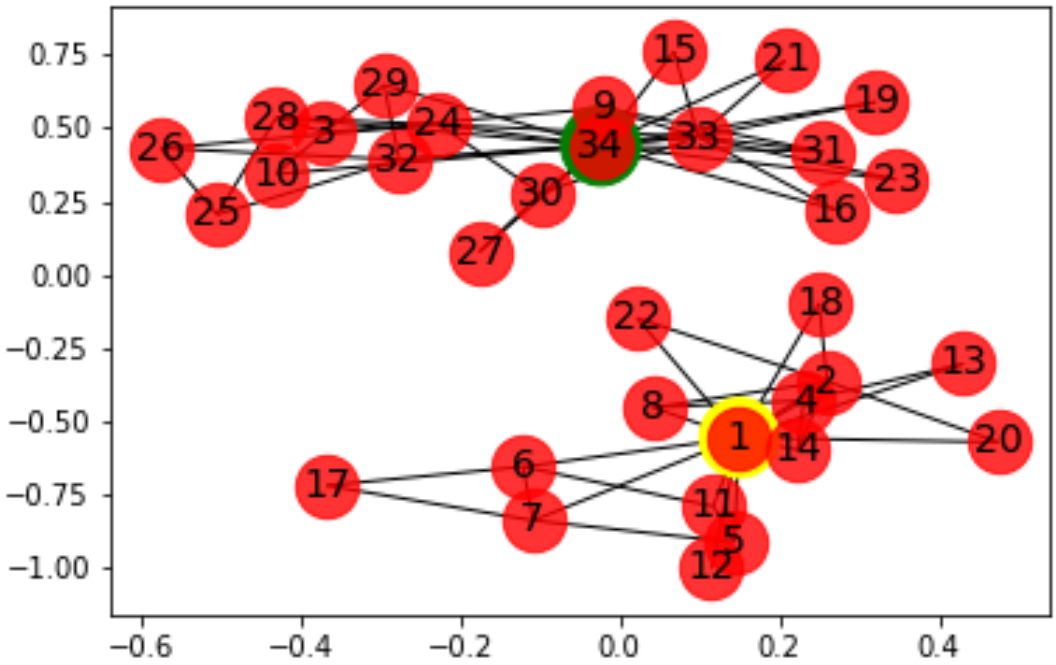
Question 1 continued on next page...





Question 1 continued on next page...



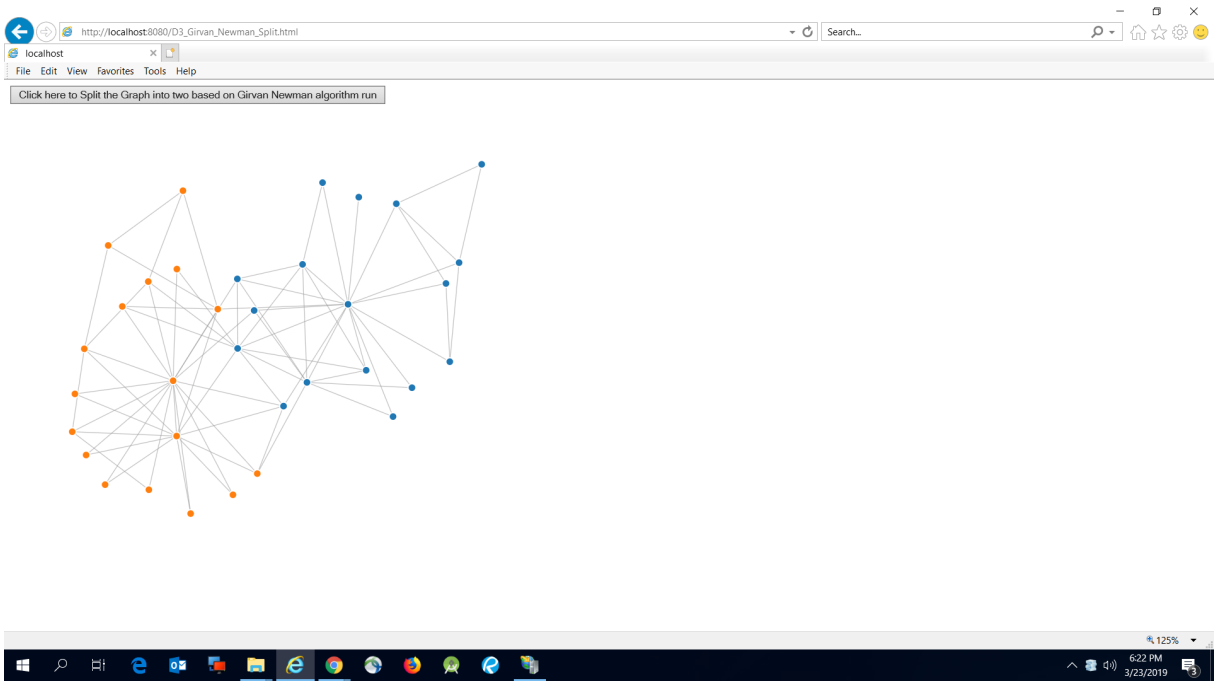


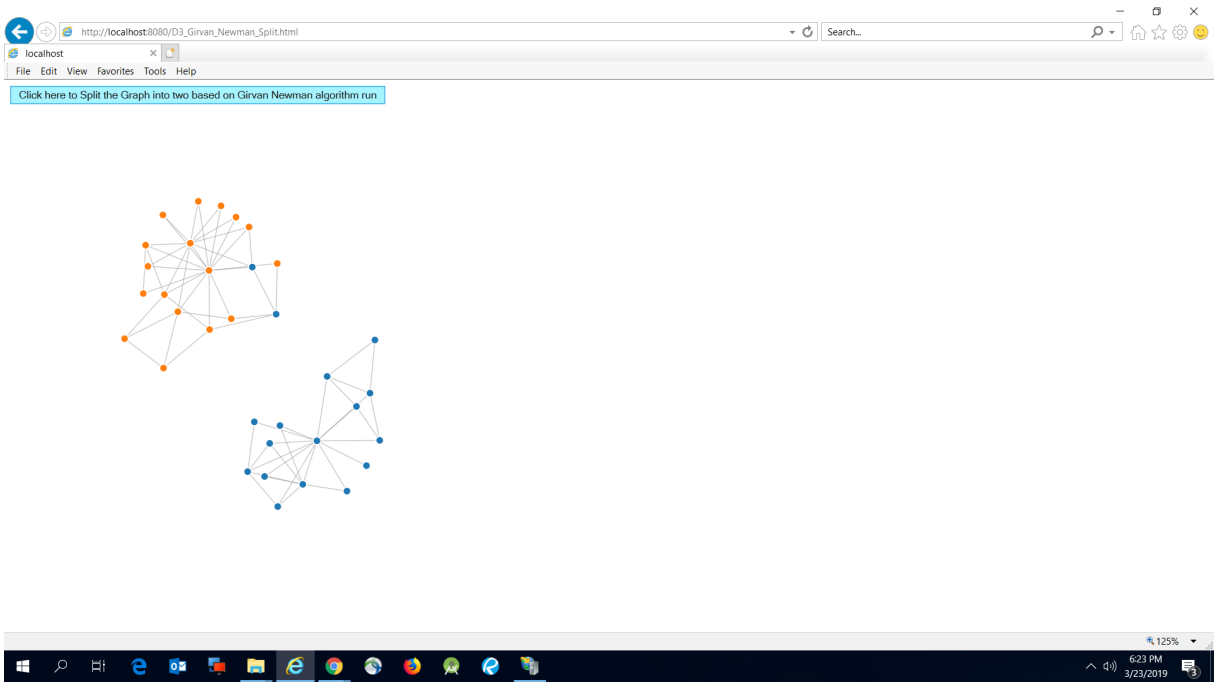
Question 2 continued on next page...

Question 2

Use D3.js's force-directed graph layout to draw the Karate Club Graph before split. Color the nodes according to the factions they belong to (John A or Mr. Hi). After a button is clicked, split the graph based on the original graph split. Include a link to the HTML/JavaScript files in your report and all necessary screenshots.

Demo video of the graph split using d3.





Question 2 continued on next page...

1. We already have the code, we just extend it as shown below to dump the output of each iteration to json
2. json responses for each iteration is saved to the karate folder in the submission which we will use in the js to plot the graph

Listing 3: Python Script

```
import json
import numpy as np
import networkx as nx
import IPython
%matplotlib inline

g = nx.karate_club_graph()
nodes = [{'name': str(i), 'club': g.node[i]['club']}
         for i in g.nodes()]
links = [{'source': u[0], 'target': u[1]}
         for u in g.edges()]
with open('graph.json', 'w') as f:
    json.dump({'nodes': nodes, 'links': links},f, indent=4,)


def girvan_newman (G):

    if len(G.nodes()) == 1:
        return [G.nodes()]

    def find_best_edge(G0):
        eb = nx.edge_betweenness_centrality(G0)
        return sorted(eb.items(), key = lambda x: float(x[1]), reverse =
            True)[0][0]

    components = nx.connected_component_subgraphs(G)
    clusterCount = 0
    while sum(1 for x in components) == 1:
        G.remove_edge(*find_best_edge(G))
        components = nx.connected_component_subgraphs(G)

        clusterCount = clusterCount +1
        output = "C:\\Karate\\Iteration%(id)02d.json" % {"id": clusterCount
            }
        nodes = [{'name': str(i), 'club': G.node[i]['club']}
         for i in G.nodes()]
        links = [{'source': u[0], 'target': u[1]}
         for u in G.edges()]
        with open(output, 'w') as f:
            json.dump({'nodes': nodes, 'links': links},f, indent=4,)

    result = [c.nodes() for c in components]

    for c in components:
        result.extend(girvan_newman(c))

    return result
```

Question 2 continued on next page...

```
G1 = nx.karate_club_graph()
kn = girvan_newman(G1)
```

1. To implement a IIS express website is started at port 8080
2. The below is the html that is added to the website content folder, the folder also include the responses of json so that it can be passed to the d3.json method and have function that crease the nodes and the links as mentioned.
3. On button click we just split the graph, we take the final json response and pass it the d3/json and draw the graph on the svg.

Reference : Used reference in iPython - https://ipython-books.github.io

Listing 4: HTML and Script

```
<!DOCTYPE html>
<html>
<head>
        <script src="https://d3js.org/d3.v3.js"></script>
</head>
<input type="button" id="btnSplitGraph" onclick="splitGraph()" value="Click here to Split the Graph into two based on Girvan Newman algorithm run">
   </input>
<div id="d3beforeSplit"></div>
<div id="d3afterSplit"></div>
<style>
        .node {stroke: #fff; stroke-width: 1.5px;}
        .link {stroke: #999; stroke-opacity: .6;}
</style>
</html>
<script>



  var width = 600, height = 600;
  var color = d3.scale.category10();
  var force = d3.layout.force()
    .charge(-480)
    .linkDistance(120)
    .size([width, height]);
  var svg = d3.select("#d3beforeSplit").select("svg")
  if (svg.empty()) {
    svg = d3.select("#d3beforeSplit").append("svg")
          .attr("width", width)
          .attr("height", height);
  }

  d3.json("http://localhost:8080/graph.json", function(error, graph) {

    force.nodes(graph.nodes)
      .links(graph.links)
      .start();

    var link = svg.selectAll(".link")
```

Question 2 continued on next page...

```
      .data(graph.links)
      .enter().append("line")
      .attr("class", "link");


    var node = svg.selectAll(".node")
      .data(graph.nodes)
      .enter().append("circle")
      .attr("class", "node")
      .attr("r", 5)
      .style("fill", function(d) {

         return color(d.club);
      })
      .call(force.drag);


    node.append("title")
        .text(function(d) { return d.name; });


    force.on("tick", function() {
      link.attr("x1", function(d){return d.source.x})
          .attr("y1", function(d){return d.source.y})
          .attr("x2", function(d){return d.target.x})
          .attr("y2", function(d){return d.target.y});

      node.attr("cx", function(d){return d.x})
          .attr("cy", function(d){return d.y});
    });
  });

  function splitGraph() {
        d3.select("#d3beforeSplit").style("display", "none");
        var width = 600, height = 600;

        var color = d3.scale.category10();
        var force = d3.layout.force()
        .charge(-240)
        .linkDistance(60)
        .size([width, height]);
        var svg = d3.select("#d3afterSplit").select("svg")
        if (svg.empty()) {
        svg = d3.select("#d3afterSplit").append("svg")
                  .attr("width", width)
                  .attr("height", height);
        }

        d3.json("http://localhost:8080/Iteration11.json", function(error,
            graph) {

        force.nodes(graph.nodes)
        .links(graph.links)
        .start();

        var link = svg.selectAll(".link")
```

Question 2 continued on next page...

```
        .data(graph.links)
        .enter().append("line")
        .attr("class", "link");

        var node = svg.selectAll(".node")
        .data(graph.nodes)
        .enter().append("circle")
        .attr("class", "node")
        .attr("r", 5)
        .style("fill", function(d) {
         return color(d.club);
        })
        .call(force.drag);


        node.append("title")
        .text(function(d) { return d.name; });


                force.on("tick", function() {
                link.attr("x1", function(d){return d.source.x})
                  .attr("y1", function(d){return d.source.y})
                  .attr("x2", function(d){return d.target.x})
                  .attr("y2", function(d){return d.target.y});

                node.attr("cx", function(d){return d.x})
                  .attr("cy", function(d){return d.y});
                });

        });

  }
</script>
```