HW 5 - Graph Partitioning

Swathi Venkatesh

11/14/2021

Q1

Draw the original Karate club graph (before the split) and color the nodes according to the factions they belong to (John A or Mr. Hi). This should look similar to the graph on slide 92 - all edges should be present, just indicate the nodes in the eventual split by color.

Answer

```
import re
import numpy as np
import networkx as net
import matplotlib.pyplot as plt

def color(p,f,lst):
    col = []
    lent = len(p)
    if f == "" and len(lst) ==0:
        col = ['yellow'] * 34
        col[0] = "red"
        col[33] = "green"
    elif(f=="final" and len(lst) == 2):

        col = ['blue'] * 34
        for n in lst[0]:
            col[n] = "red"
        for t in lst[1]:
            col[t] ="green"
    return col

def findedge(G0):
    """
    The edge_betweenness in a networkx returns dictionary.
    Which is then made to a list; sorted and it returns edge with
   highest betweenness
    """
    betweenness = net.edge_betweenness_centrality(G0)
    betweenness_li = list(betweenness.items())
    betweenness_li.sort(key=lambda z: z[1], reverse=True)
    return betweenness_li[0][0]
```

```
31
32 def component(G):
33     if len(G.nodes()) == 1:
34         return [G.nodes()]
35     comp = (G.subgraph(c) for c in net.connected_components(G))
36     comp = list(comp)
37     cnt = 0
38     while len(comp) == 1:
39         cnt +=1
40         G.remove_edge(*findedge(G))
41
42         comp =(G.subgraph(c) for c in net.connected_components(G))
43         comp = list(comp)
44     return comp
45
46 def graph(G, color, path,empty,edgecl_map,wei_map):
47     string = re.search(r'((q[0-9]\/)([0-9a-zA-z]*\.png))',path)
48     name = string.group(3)
49     plt.figure(figsize=(15,8.8))
50     if empty == "":
51         net.draw_kamada_kawai(G,with_labels=True, node_color = color)
52     else:
53
54         pos = net.spring_layout(G, k=0.3*1/np.sqrt(len(G.nodes()))
    +0.1, iterations=20)
55         net.draw(G,with_labels=True ,node_color = color,pos=pos,
    edge_color=edgecl_map,width = list(wei_map))
56
57     plt.savefig(path, format="PNG")
58     plt.show()
59     plt.close()
60     return 0
61
62 def coloredges(G,tuplesEdgeToRemove,reset):
63     """
64     This builds the edge attributes color and weight
65     """
66     tot = G.number_of_edges()
67     coloredge_map = ['black'] * tot
68     wei_map = [1.5] * tot
69     if(reset =="n"):
70         tot = -1
71         for n in G.edges:
72             tot += 1
73             if tuplesEdgeToRemove == n:
74                 coloredge_map[tot] = 'blue'
75                 wei_map[tot] = 3.2
```

```
76      return coloredge_map,wei_map
77
78  def girvannewman_graph(G):
79      comp = (G.subgraph(c) for c in net.connected_components(G))
80      comp = list(comp)
81      c =0
82      while len(comp) == 1:
83          c +=1
84          path = "q2/" +str(c) +"x.png"
85          #find the best edge and return as a list
86          bestedge = findedge(G)
87          edgecolor,weightMap = coloredges(G, bestedge,"n")
88          graph(G,col,path,"spacing",edgecolor,weightMap)
89          #Remove the best edge
90          G.remove_edge(*bestedge)
91          #ReSet everything back to black and reset the weight too
92          edgecolor,weightMap = coloredges(G, bestedge,"")
93          path = "q2/" + str(c) +"y.png"
94
95          graph(G,col,path,"spacing",edgecolor, weightMap)
96
97      return 0
98  try:
99      k = net.karate_club_graph()
100     t = k
101     """
102     Got the data from networkx
103     parsed the data to assign colorcoding for the two main leaders
104     then plotted the graph
105     """
106     k = net.karate_club_graph()
107     col = color(k, "","")
108     graph(k,col, "q1/karate.png","","","")
109     """
110     passed the retrieved data to the Girvan newman algorithm
111     color coded the result of the splitted group
112     then plotted the graph
113     """
114     final = component(k)
115     #Set the colors based on the list received
116     col = color(k,"final",final)
117     graph(k,col,"q1/finalgroup.png","","","")
118     girvannewman_graph(t)
119 except Exception as e:
120     print(e)
```

Listing 1: mygraph.py

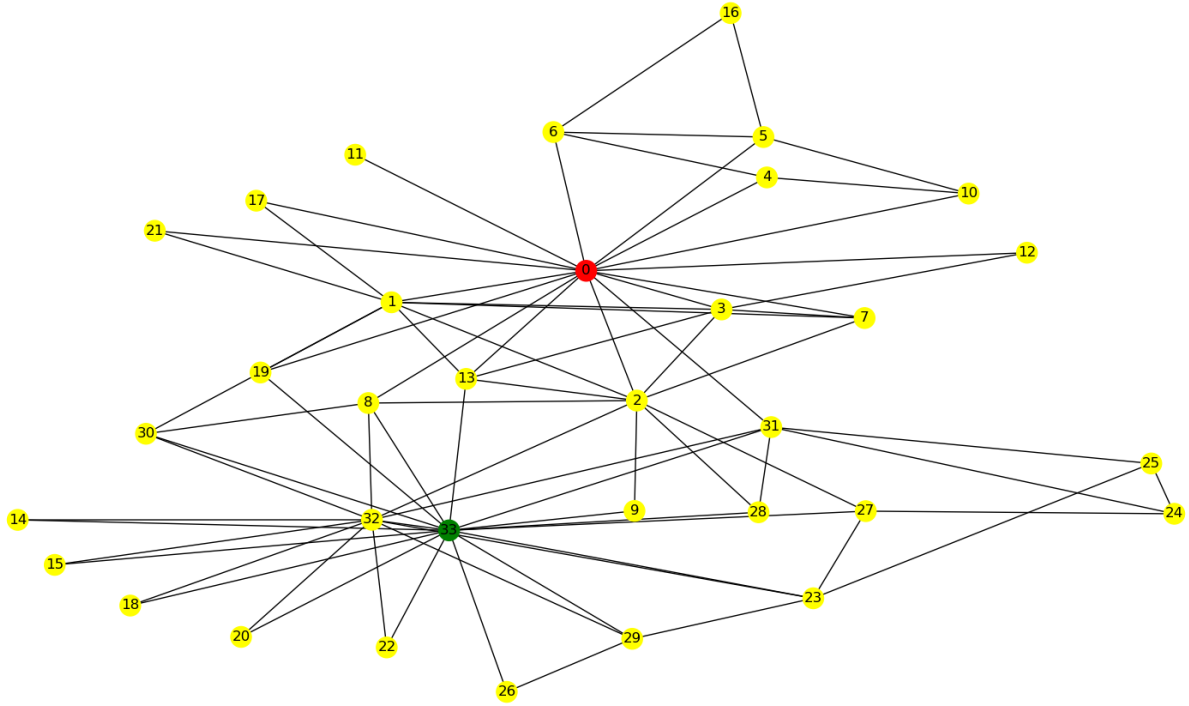

Figure 1: Dataset having two groups Red: John A and Green: Mr.Hi and yellow for others

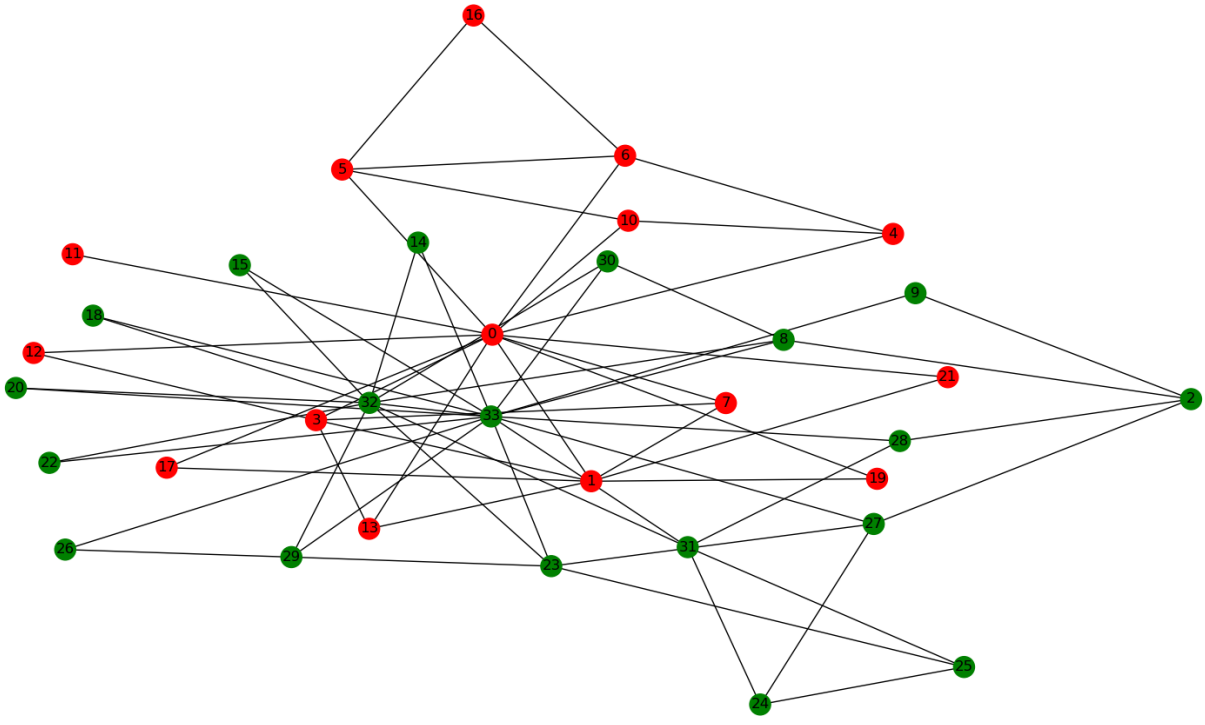

Figure 2: Appropriately divided node colors but edges are same

Discussion

I used networkx library. For this graph, I highlighed the two major parts of the Zachary karate club, John A which represents the Red color and node number 0, while Mr Hi represent the color green and node number 33.

- The driver for the question

```
106     k = net.karate_club_graph()
107     col = color(k, "","")
108     graph(k,col, "q1/karate.png","","","")
```

Listing 2: mygraph.py

- The color function builds the list of the color for each nodes

```
6  def color(p,f,lst):
7      col = []
8      lent = len(p)
9      if f == "" and len(lst) ==0:
10         col = ['yellow'] * 34
11         col[0] = "red"
12         col[33] = "green"
13     elif(f=="final" and len(lst) == 2):
14
15         col = ['blue'] * 34
16         for n in lst[0]:
17             col[n] = "red"
18         for t in lst[1]:
19             col[t] ="green"
20     return col
```

Listing 3: Building list for the color node in the graph

- A graphing function is used for graphing the data.

```
46 def graph(G, color, path,empty,edgecl_map,wei_map):
47     string = re.search(r'((q[0-9]\/)([0-9a-zA-z]*\.png))',path)
48     name = string.group(3)
49     plt.figure(figsize=(15,8.8))
50     if empty == "":
51         net.draw_kamada_kawai(G,with_labels=True, node_color =
       color)
52     else:
53
54         pos = net.spring_layout(G, k=0.3*1/np.sqrt(len(G.nodes()))
       +0.1, iterations=20)
55         net.draw(G,with_labels=True ,node_color = color,pos=pos,
       edge_color=edgecl_map,width = list(wei_map))
```

```
56
57      plt.savefig(path, format="PNG")
58      plt.show()
59      plt.close()
60      return 0
```

Listing 4: Making the plot for all the graphs (snapshot in mygraph.py)

Q: How many nodes eventually go with John and how many with Mr. Hi?

Out of 34 nodes 17 nodes belong to John A and other 17 belong to Mr. Hi

Q2

Keeping the node colors the same as they were in Q1, run multiple iterations of the Girvan-Newman graph partioning algorithm (see Module-07 Social Networks, slides 90-99) on the Karate Club graph until the graph splits into two connected components. Include an image of the graph after each iteration in your report.

Note that you will have to implement the Girvan-Newman algorithm rather than relying on a built-in function, as a built-in function will automatically run the whole algorithm and you will not be able to view the intermediate graphs. Make sure that you explain in your report what the Girvan-Newman algorithm is doing.

Answer

```
78 def girvannewman_graph(G):
79     comp = (G.subgraph(c) for c in net.connected_components(G))
80     comp = list(comp)
81     c =0
82     while len(comp) == 1:
83         c +=1
84         path = "q2/" +str(c) +"x.png"
85         #find the best edge and return as a list
86         bestedge = findedge(G)
87         edgecolor,weightMap = coloredges(G, bestedge,"n")
88         graph(G,col,path,"spacing",edgecolor,weightMap)
89         #Remove the best edge
90         G.remove_edge(*bestedge)
91         #ReSet everything back to black and reset the weight too
92         edgecolor,weightMap = coloredges(G, bestedge,"")
93         path = "q2/" + str(c) +"y.png"
94
```

```
95          graph(G,col,path,"spacing",edgecolor, weightMap)
96
97      return 0
```

Listing 5: Girvan-newman algorithm (snapshot from mygraph.py)

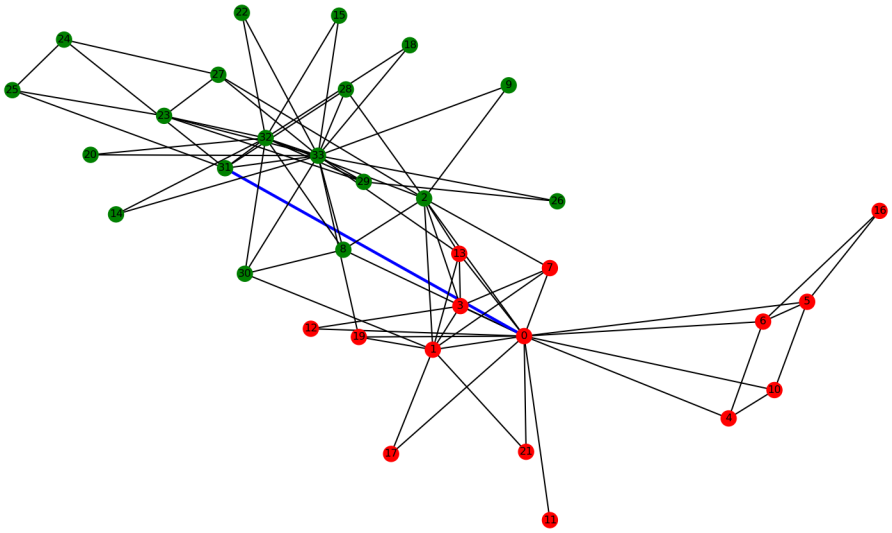

Figure 3: Iteration 1 highlighted to remove nodeedge in blue

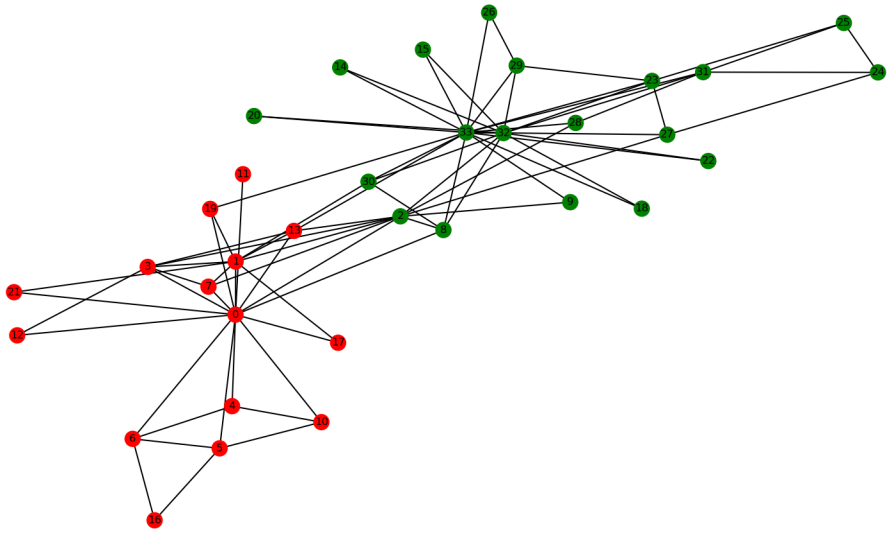

Figure 4: Iteration 1 showing nodeedge being removed

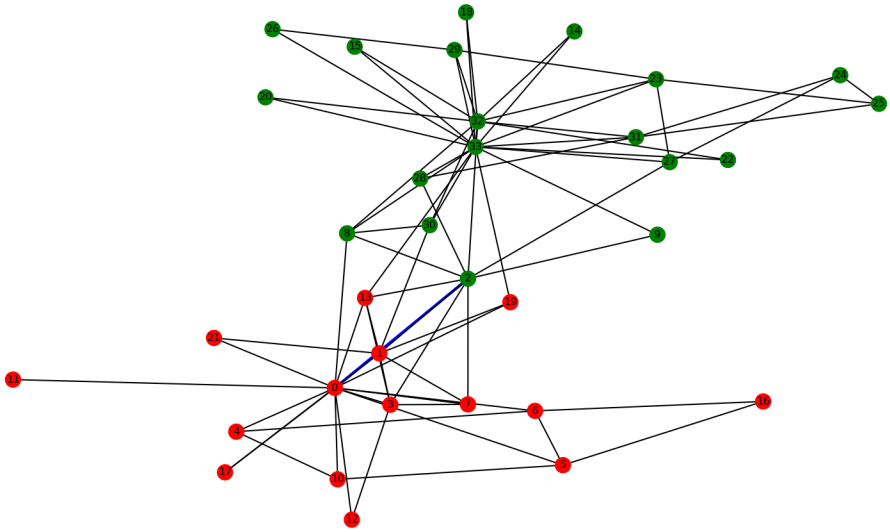

Figure 5: Iteration 2 highlighted to remove nodeedge in blue

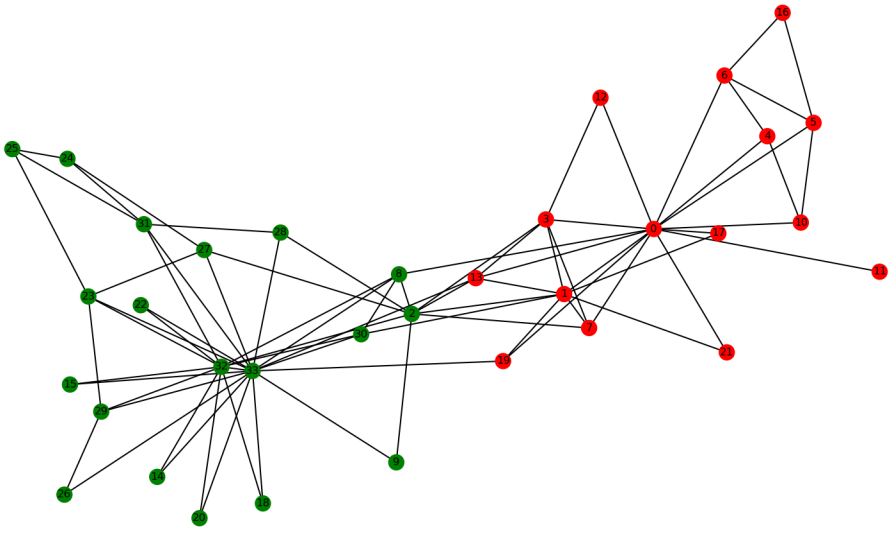

Figure 6: Iteration 2 showing nodeedge being removed

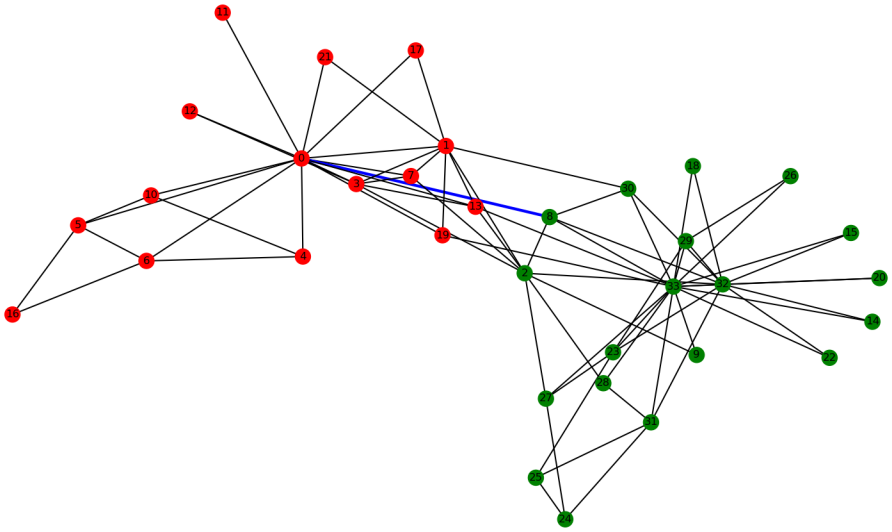

Figure 7: Iteration 3 highlighted to remove nodeedge in blue

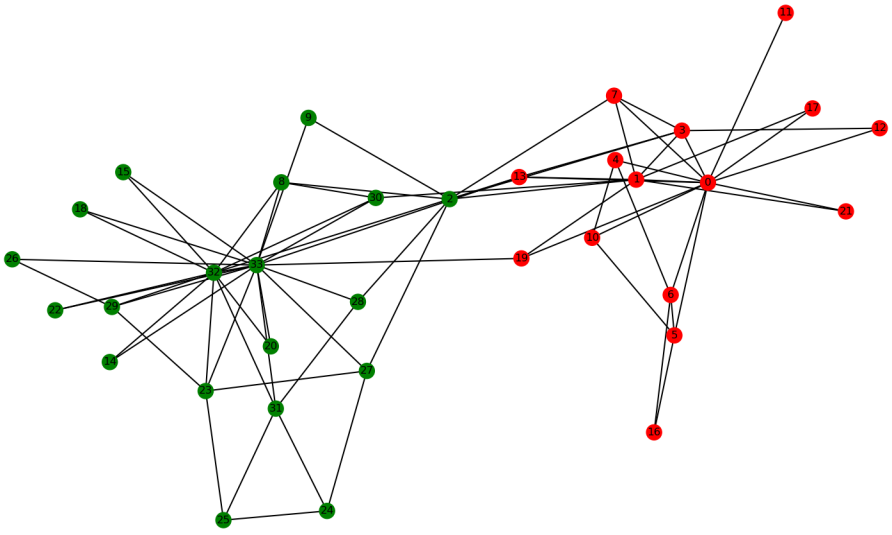

Figure 8: Iteration 3 showing nodeedge being removed

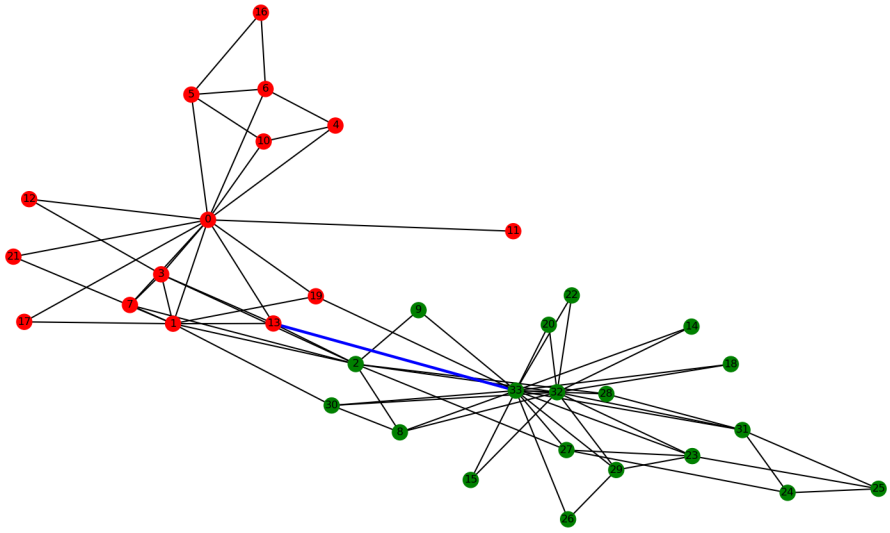

Figure 9: Iteration 4 highlighted to remove nodeedge in blue

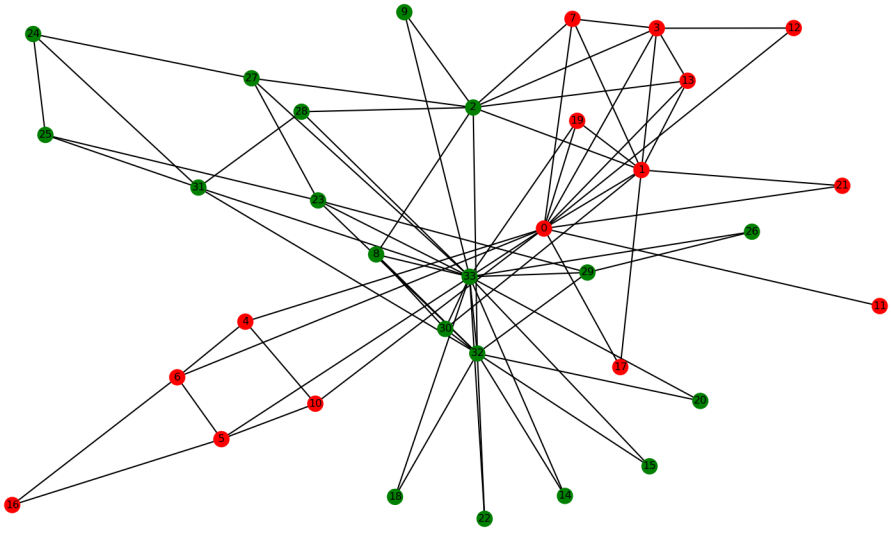

Figure 10: Iteration 4 showing nodeedge being removed

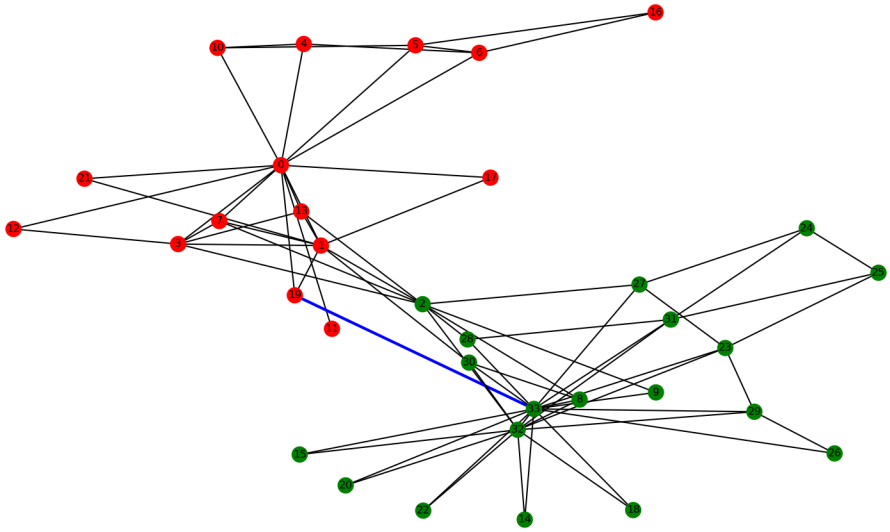

Figure 11: Iteration 5 highlighted to remove nodeedge in blue

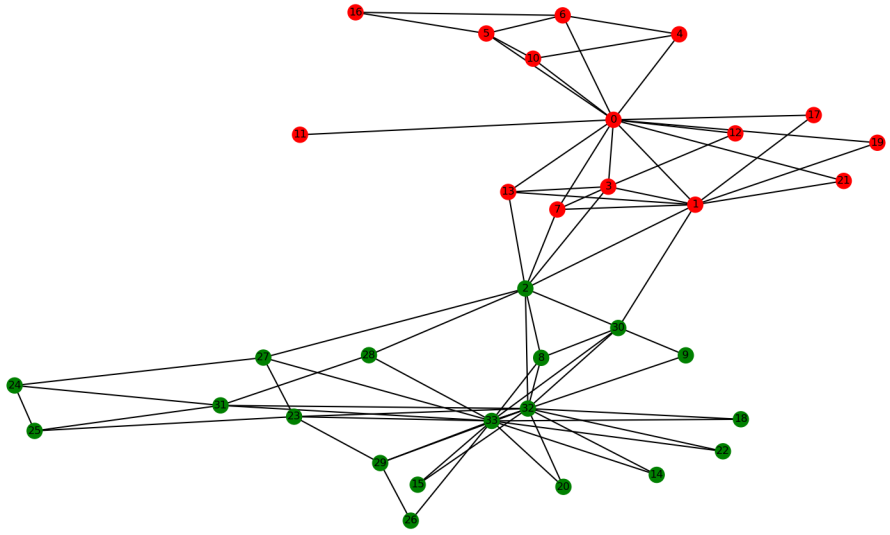

Figure 12: Iteration 5 showing nodeedge being removed

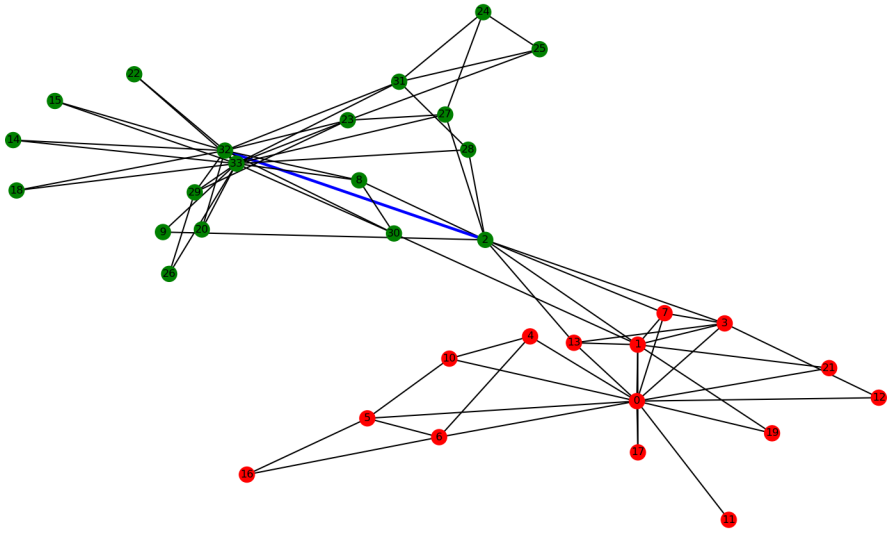

Figure 13: Iteration 6 highlighted to remove nodeedge in blue

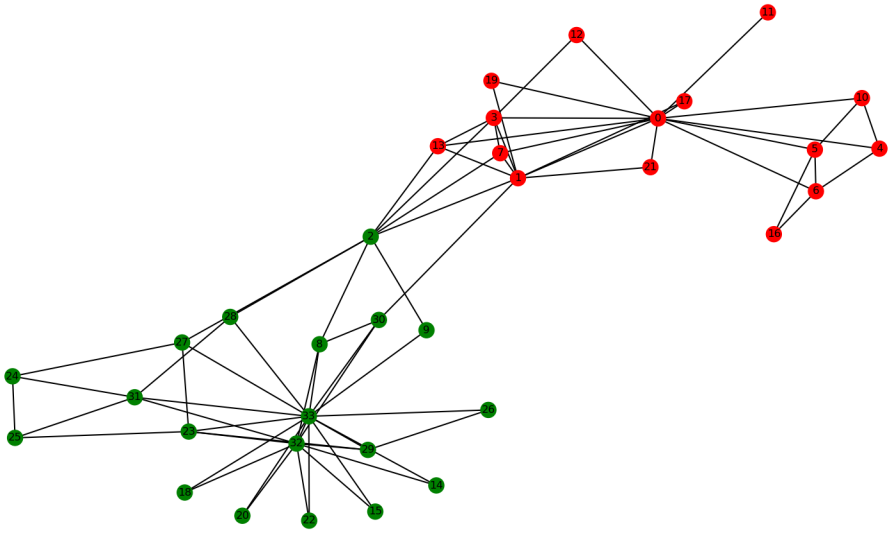

Figure 14: Iteration 6 showing nodeedge being removed

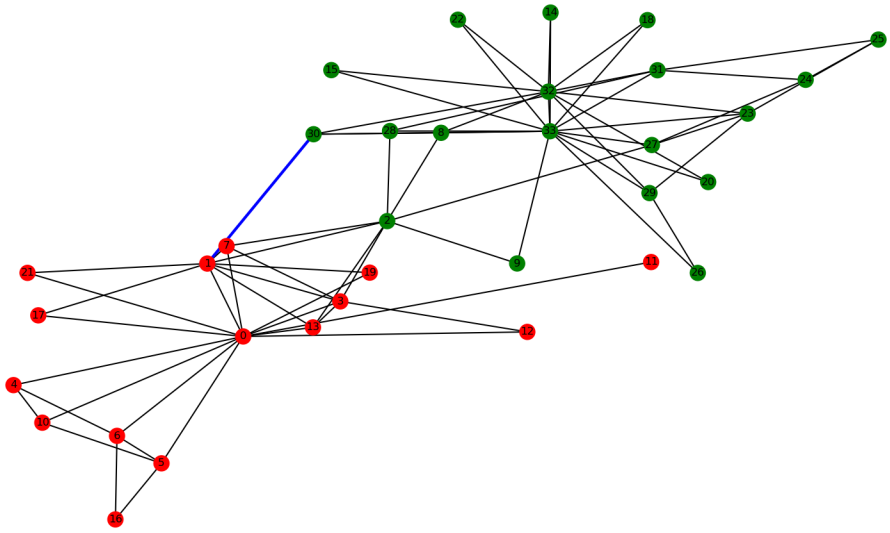

Figure 15: Iteration 7 highlighted to remove nodeedge in blue

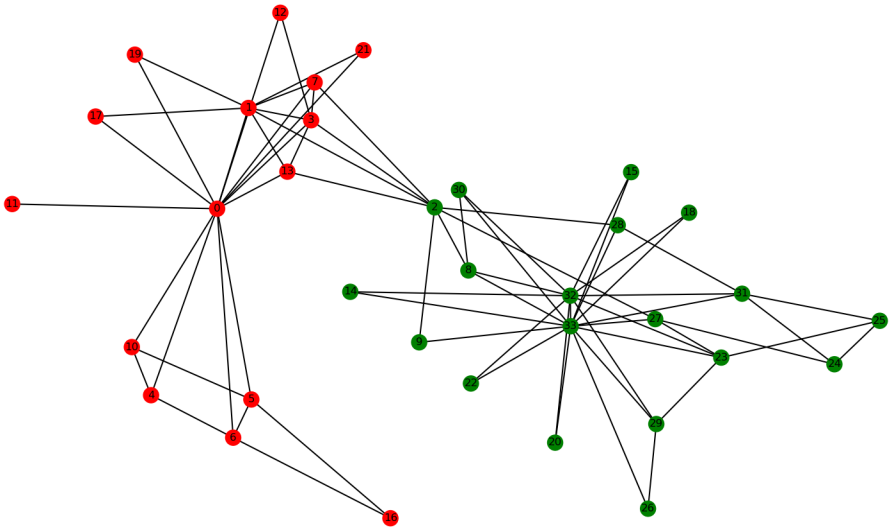

Figure 16: Iteration 7 showing nodeedge being removed

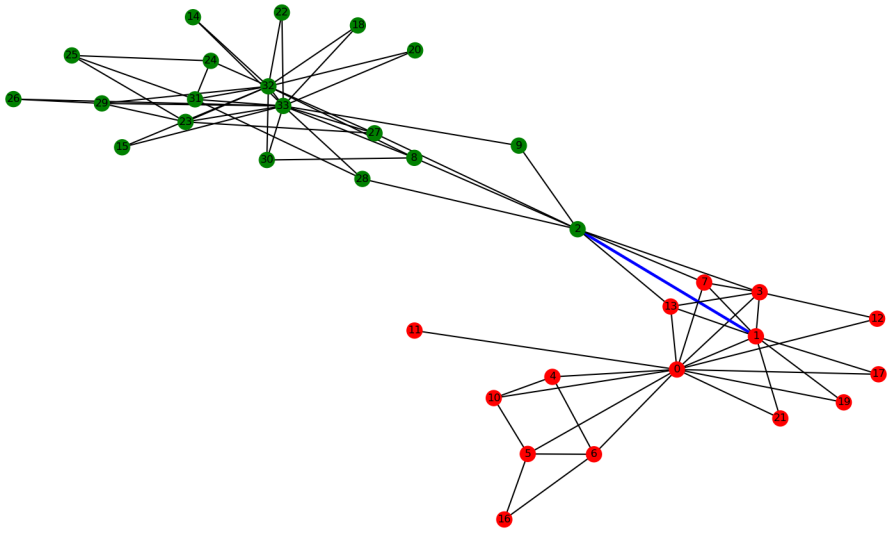

Figure 17: Iteration 8 highlighted to remove nodeedge in blue

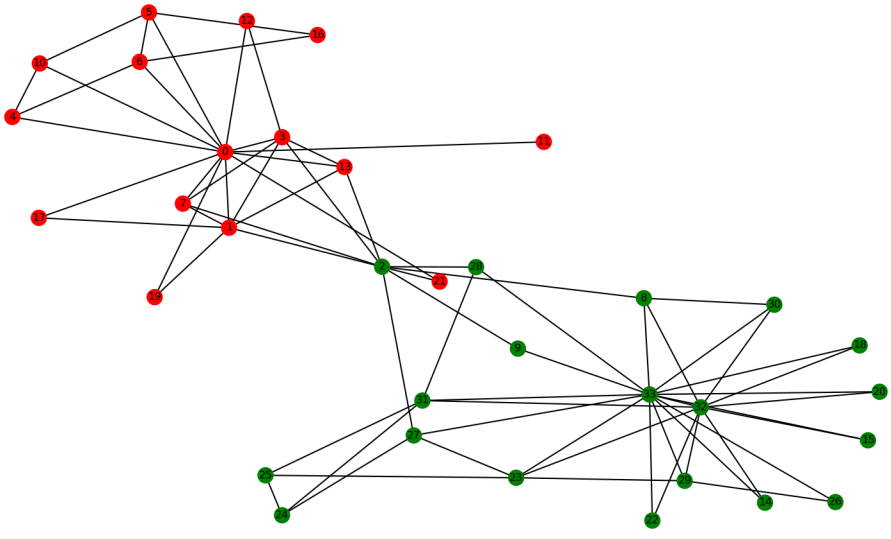

Figure 18: Iteration 8 showing nodeedge being removed

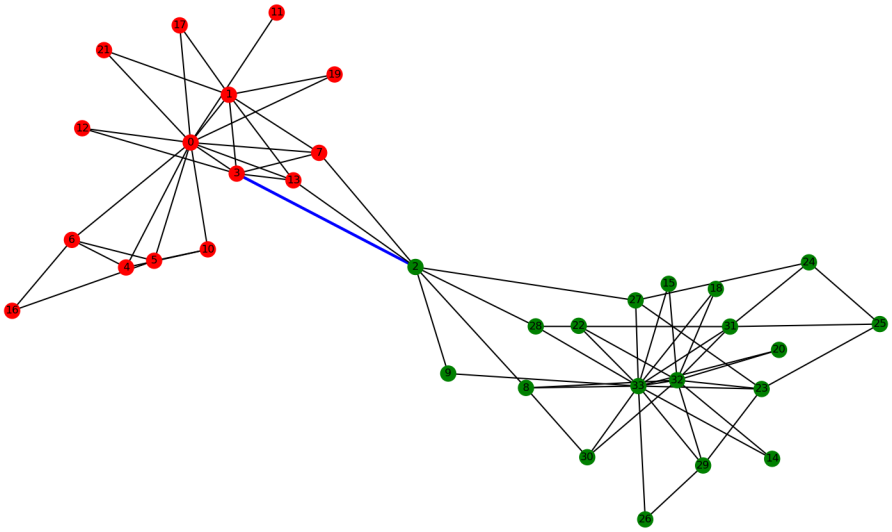

Figure 19: Iteration 9 highlighted to remove nodeedge in blue

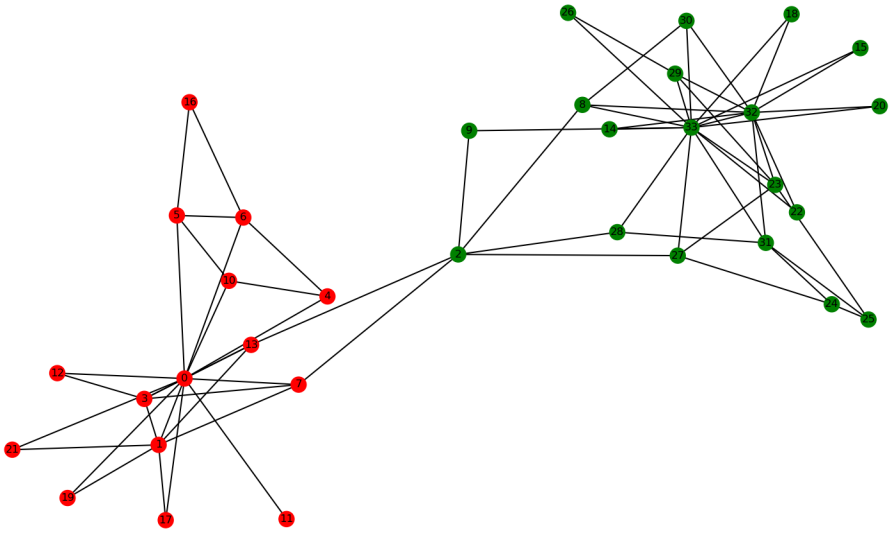

Figure 20: Iteration 9 showing nodeedge being removed

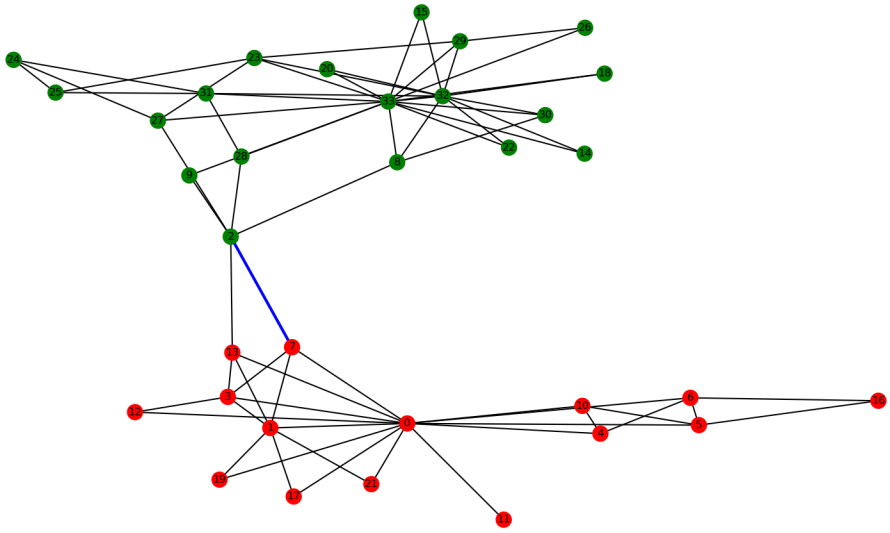

Figure 21: Iteration 10 highlighted to remove nodeedge in blue

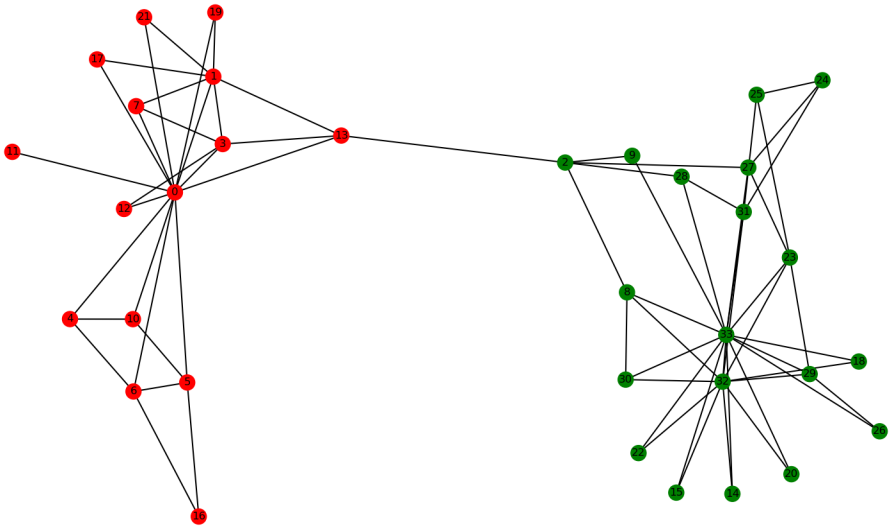

Figure 22: Iteration 10 showing nodeedge being removed

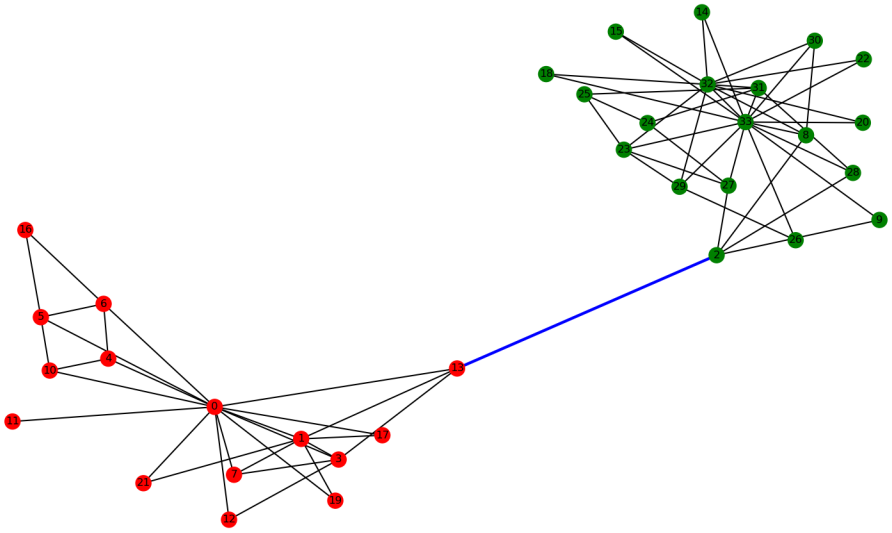

Figure 23: Iteration 11 highlighted to remove nodeedge in blue

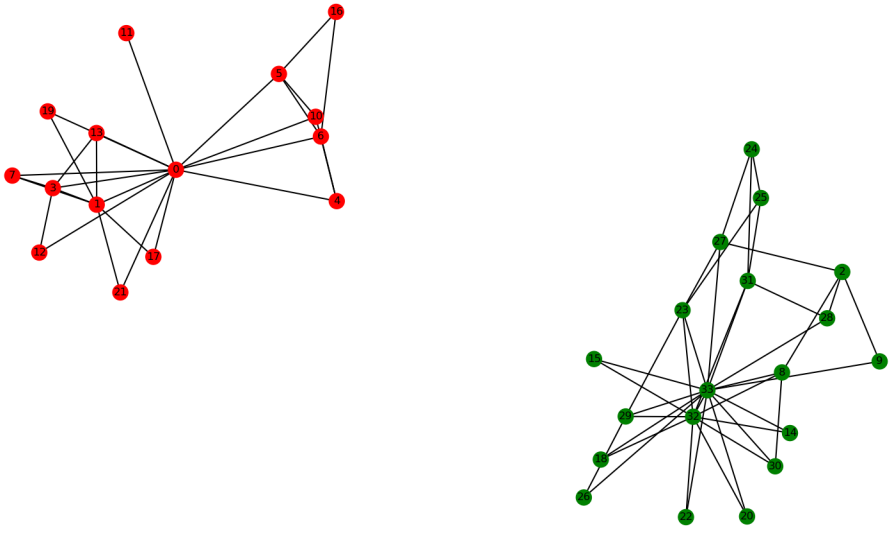

Figure 24: Iteration 11 showing nodeedge being removed

Discussion

In the girvannewman_graph() the edge betweenness is calculated for each edge in the graph i.e., it finds the best edge and it is returned as a list is stored in bestedge. Then, it removes the edge with highest betweenness. The betweenness is calculated for all the other nodes and they are reset back with the weights. The edge betweenness returns a dictionary, which is then made to a list, sorted and returns the edge with highest betweenness. The coloredges() builds the edge color list and width of the edge list, to set the color coding and the width variable.

Q: How many iterations did it take to split the graph?

Iterations of the Girvan-Newman graph partioning algorithm is run and it took 11 iteration to complete break the graph into two.

Q3

Compare the connected components of the Girvan-Newman split graph (Q2) with the connected components of the actual split Karate club graph (Q1).

Answer

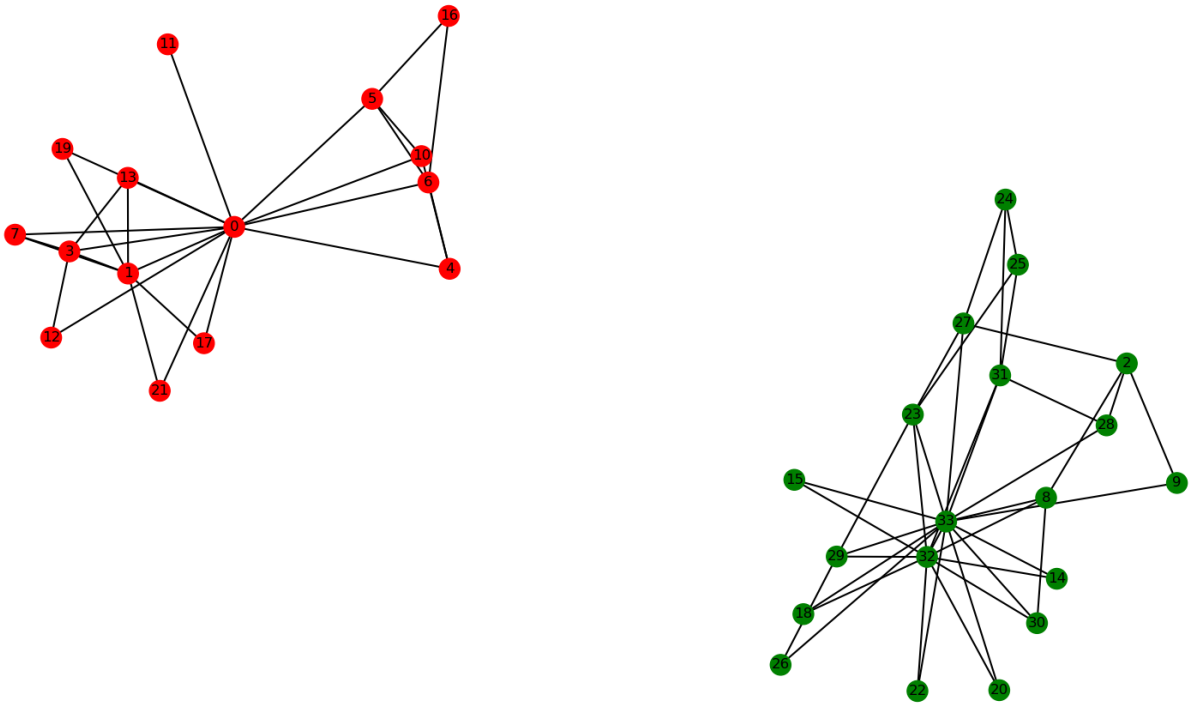

Figure 25: Final split by Girvan-Newman algorithm

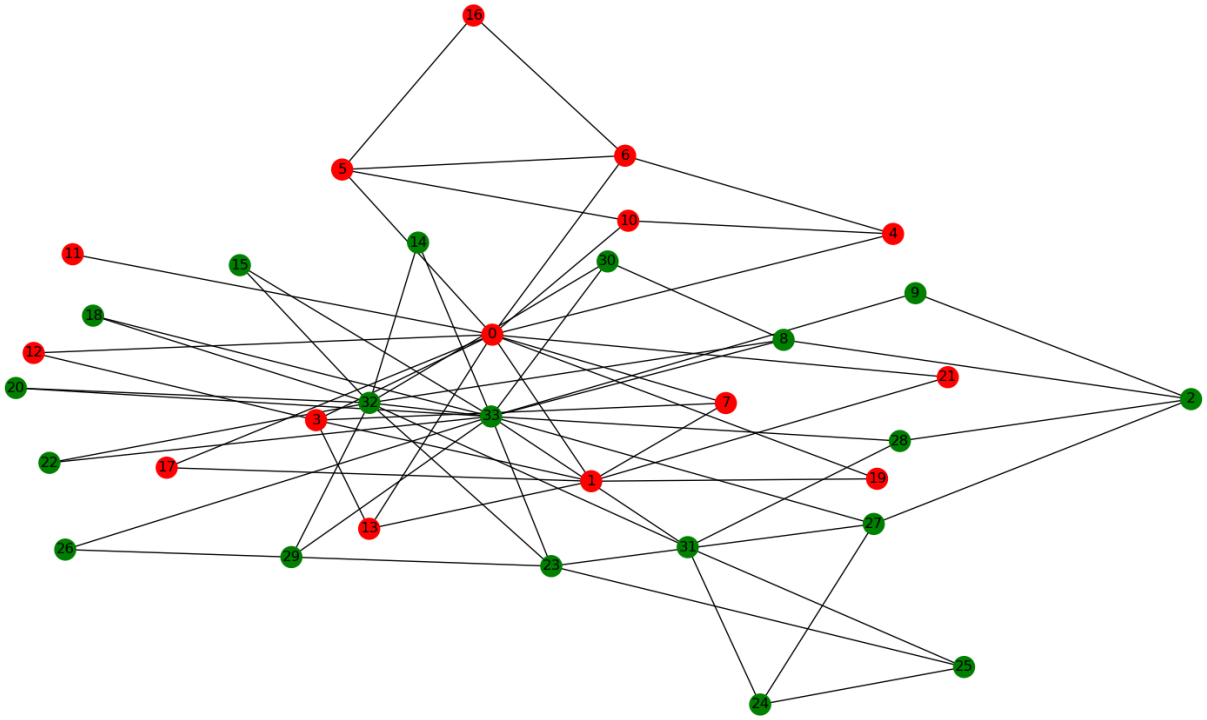

Figure 26: Appropriately divided node colors but edges are same

Discussion

Q: Did all of the same colored nodes end up in the same group? If not, what is different?

Yes all the colored node ended up in the same group. The driver code for this is:

```
114     final = component(k)
115     #Set the colors based on the list received
116     col = color(k,"final",final)
117     graph(k,col,"q1/finalgroup.png","","","")
118     girvannewman_graph(t)
119 except Exception as e:
120     print(e)
```

Listing 6: mygraph.py

The component function is used to breakdown the nodes into various node colors of green and red

```
32 def component(G):
33     if len(G.nodes()) == 1:
34         return [G.nodes()]
35     comp = (G.subgraph(c) for c in net.connected_components(G))
36     comp = list(comp)
```

```
37      cnt = 0
38      while len(comp) == 1:
39          cnt +=1
40          G.remove_edge(*findedge(G))
41
42          comp =(G.subgraph(c) for c in net.connected_components(G))
43          comp = list(comp)
44      return comp
```

Listing 7: Get the NodeView object separating into red and green categories

Then launched the driver for Girvan-newman algorithm on line 118

```
118     girvannewman_graph(t)
```

Listing 8: Girvan-newman algorithm

Then found the maximum edge called it the findedge

```
22  def findedge(G0):
23      """
24      The edge_betweenness in a networkx returns dictionary.
25      Which is then made to a list; sorted and it returns edge with
        highest betweenness
26      """
27      betweenness = net.edge_betweenness_centrality(G0)
28      betweenness_li = list(betweenness.items())
29      betweenness_li.sort(key=lambda z: z[1], reverse=True)
30      return betweenness_li[0][0]
```

Listing 9: findedge

Then built the edge color list and width of the edge list, to set the color coding and width variable

```
62  def coloredges(G,tuplesEdgeToRemove,reset):
63      """
64      This builds the edge attributes color and weight
65      """
66      tot = G.number_of_edges()
67      coloredge_map = ['black'] * tot
68      wei_map = [1.5] * tot
69      if(reset =="n"):
70          tot = -1
71          for n in G.edges:
72              tot += 1
73              if tuplesEdgeToRemove == n:
74                  coloredge_map[tot] = 'blue'
75                  wei_map[tot] = 3.2
76      return coloredge_map,wei_map
```

Listing 10: Setting and reset edges color and width values

Then plotted using the plot function and built the path for after the removal of edges

```
84          path = "q2/" +str(c) +"x.png"
85          #find the best edge and return as a list
86          bestedge = findedge(G)
87          edgecolor,weightMap = coloredges(G, bestedge,"n")
88          graph(G,col,path,"spacing",edgecolor,weightMap)
89          #Remove the best edge
90          G.remove_edge(*bestedge)
91          #ReSet everything back to black and reset the weight too
92          edgecolor,weightMap = coloredges(G, bestedge,"")
93          path = "q2/" + str(c) +"y.png"
94
95          graph(G,col,path,"spacing",edgecolor, weightMap)
```

Listing 11: mygraph.py

Q4

We know the group split in two different groups. Suppose the disagreements in the group were more nuanced. What would the clubs look like if they split into 3, 4, and 5 groups? A single node can be considered as a ”group”.

Answer

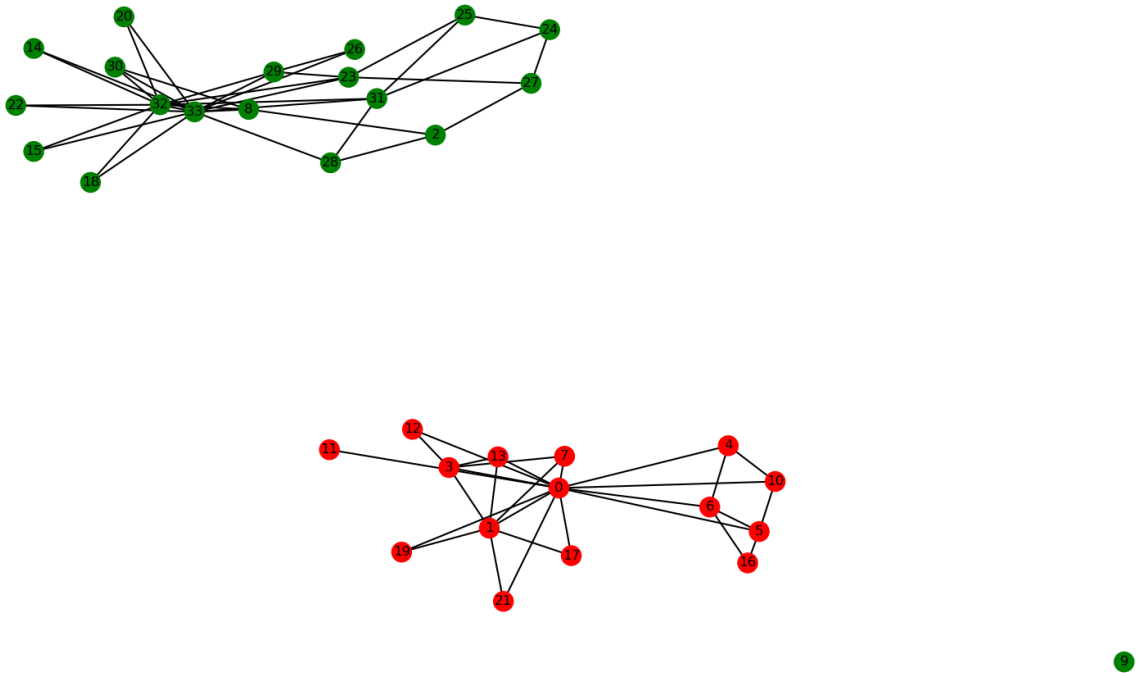

Figure 27: Group 3

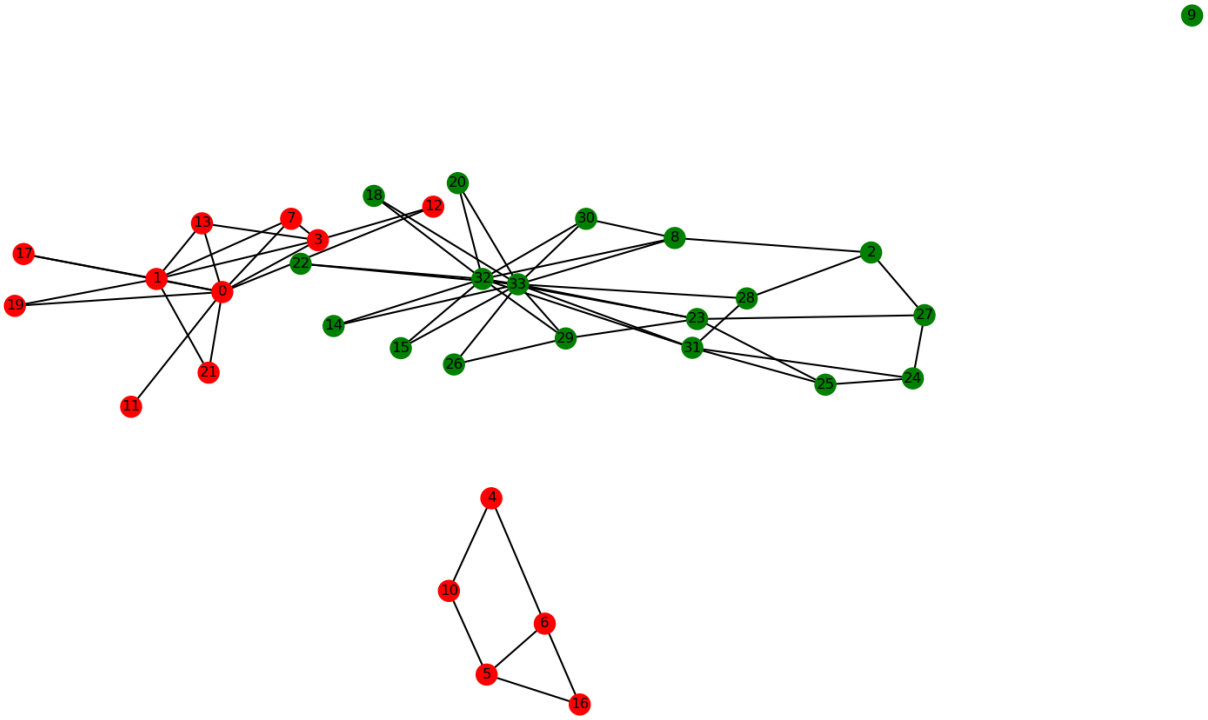

Figure 28: Group 4

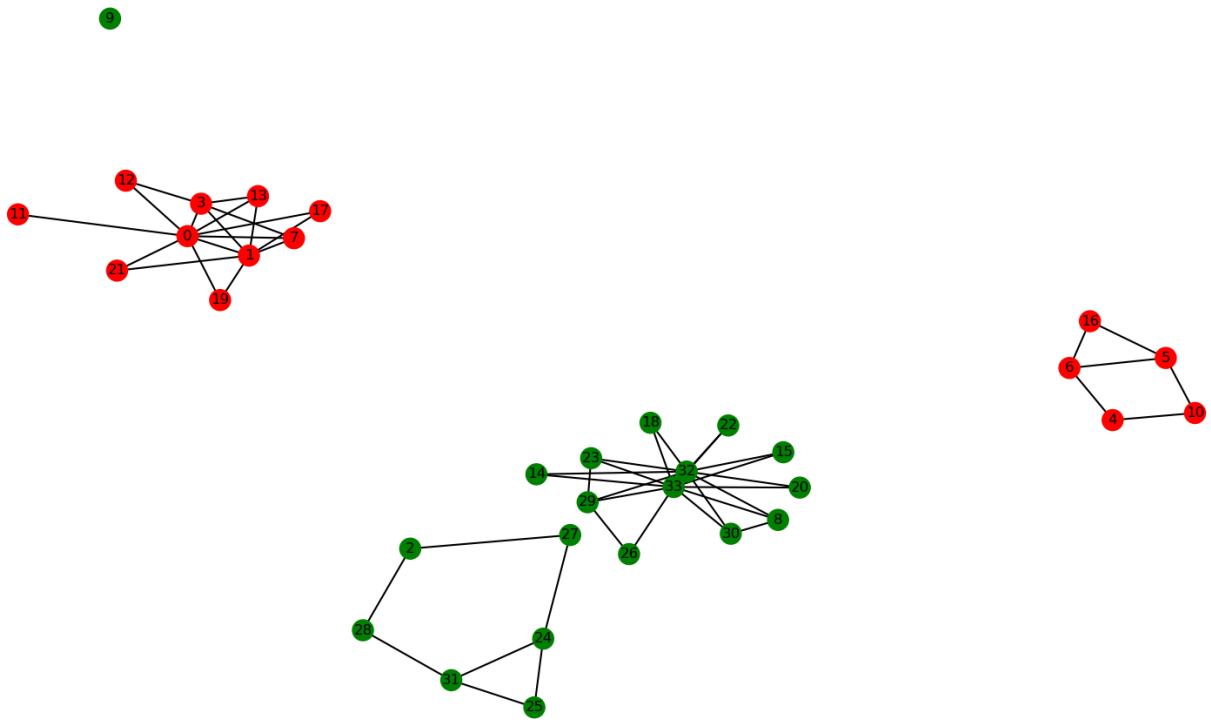

Figure 29: Group 5

Discussion

The same algorithm of previous question produced these outputs. This is how the club groups would look like if they are split into 3,4 and 5 groups.

References

- NetworkX, `https://networkx.org/documentation/stable/tutorial.html`
- NetworkX , `https://networkx.org/documentation/stable/reference/drawing.html#module-networkx.drawing.nx_pylab`
- NetworkX , `https://gawron.sdsu.edu/python_for_ss/course_core/book_draft/Social_Networks/Networkx.html`
- StackoverFlow, `https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib`
- StackoverFlow, `https://stackoverflow.com/questions/9012487/matplotlib-pyplot-savefig-outputs-blank-image`

- Girvan Newman, https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.centrality.girvan_newman.html