# HW4 CS432

Ondra Torkilson

November 8, 2020

## Step 1

Figure 1 below shows the original karate club graph with the Mr. Hi faction colored red and the John A. faction colored blue.

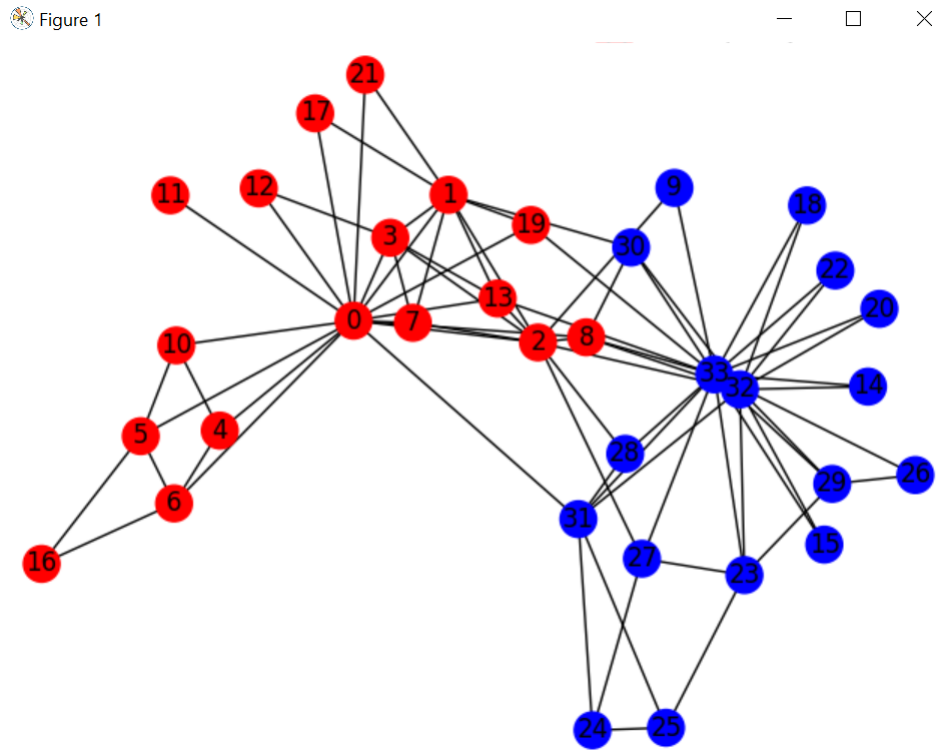


Figure 1: Original Karate Club Graph

## Step 2

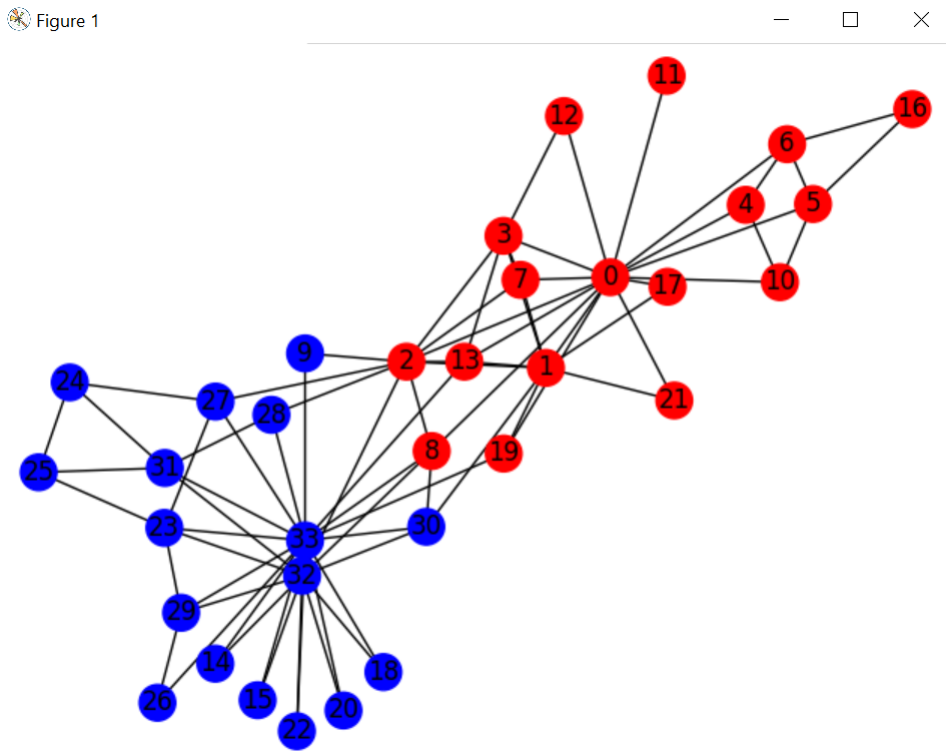


Figure 2: 1st Iteration

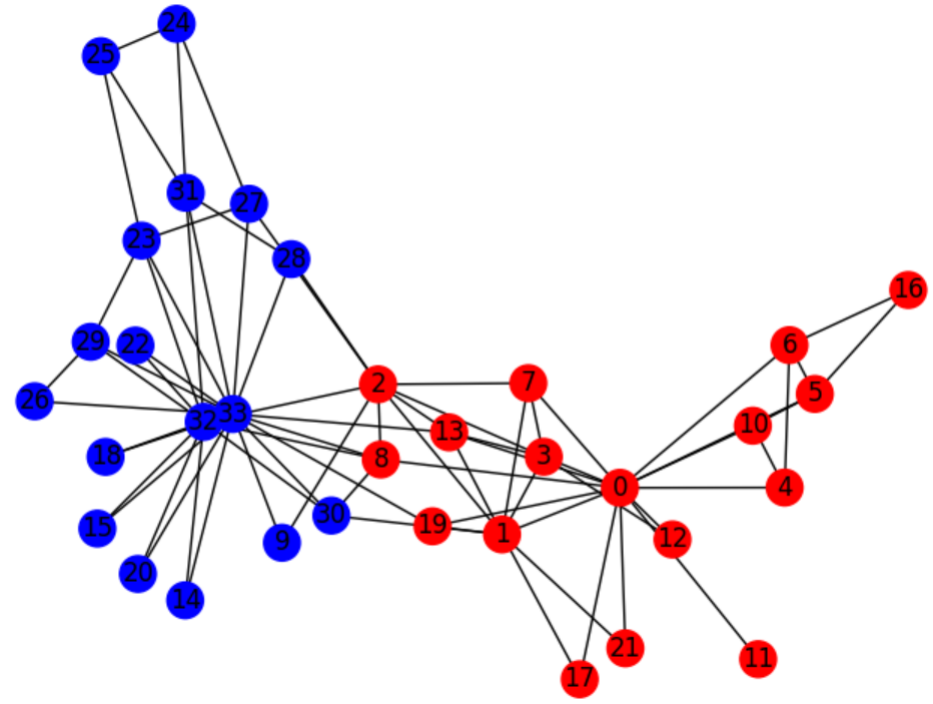


Figure 3: 2nd Iteration

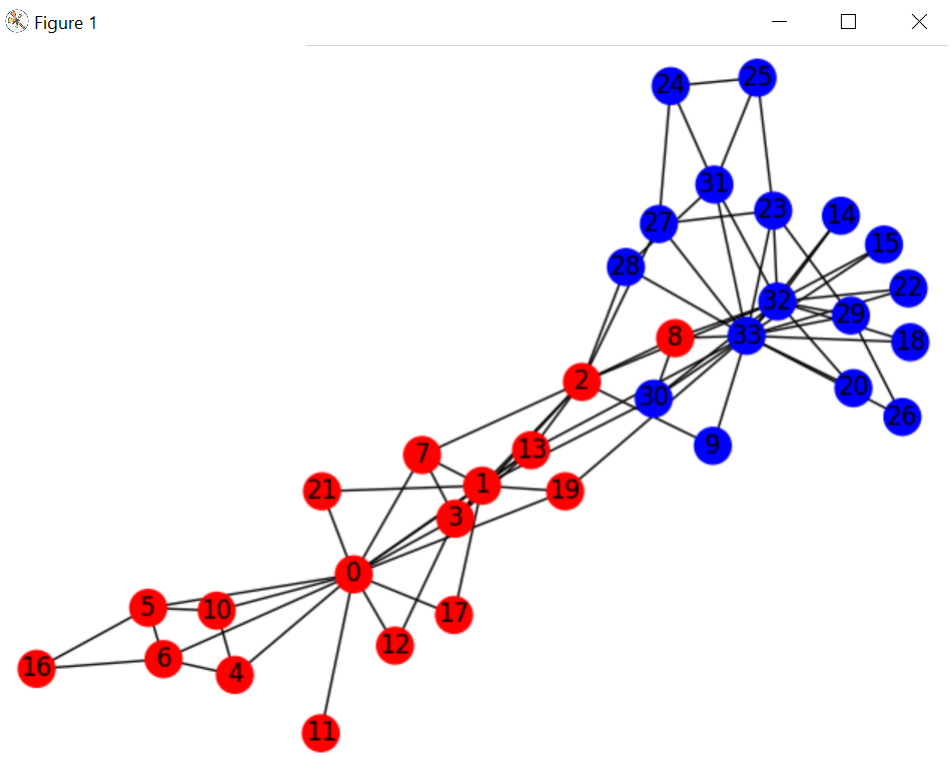


Figure 4: 3rd Iteration

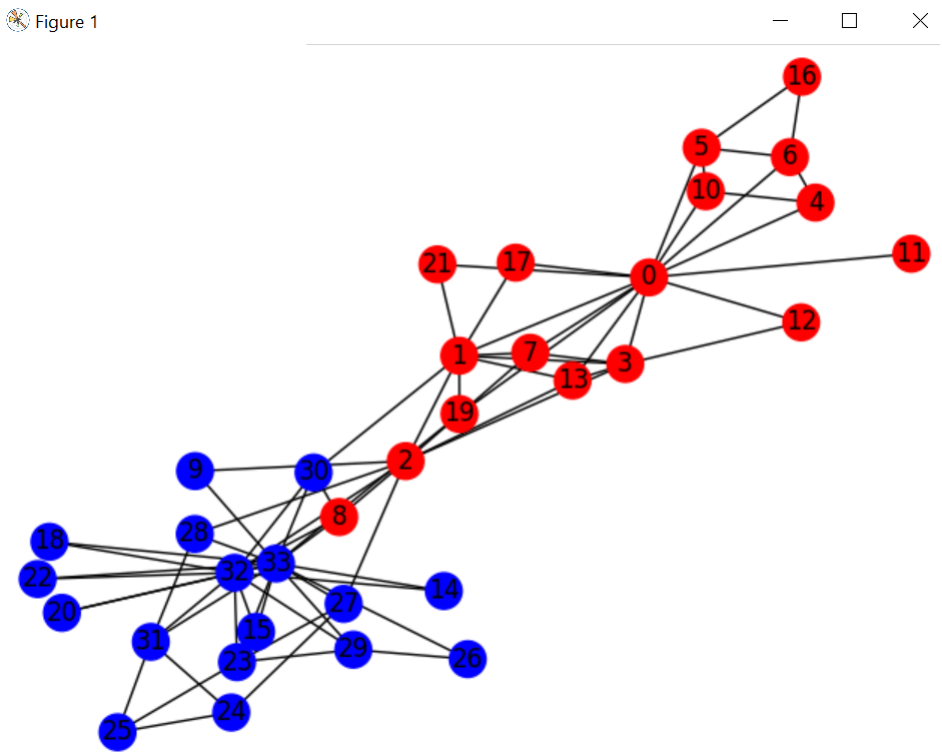


Figure 5: 4th Iteration

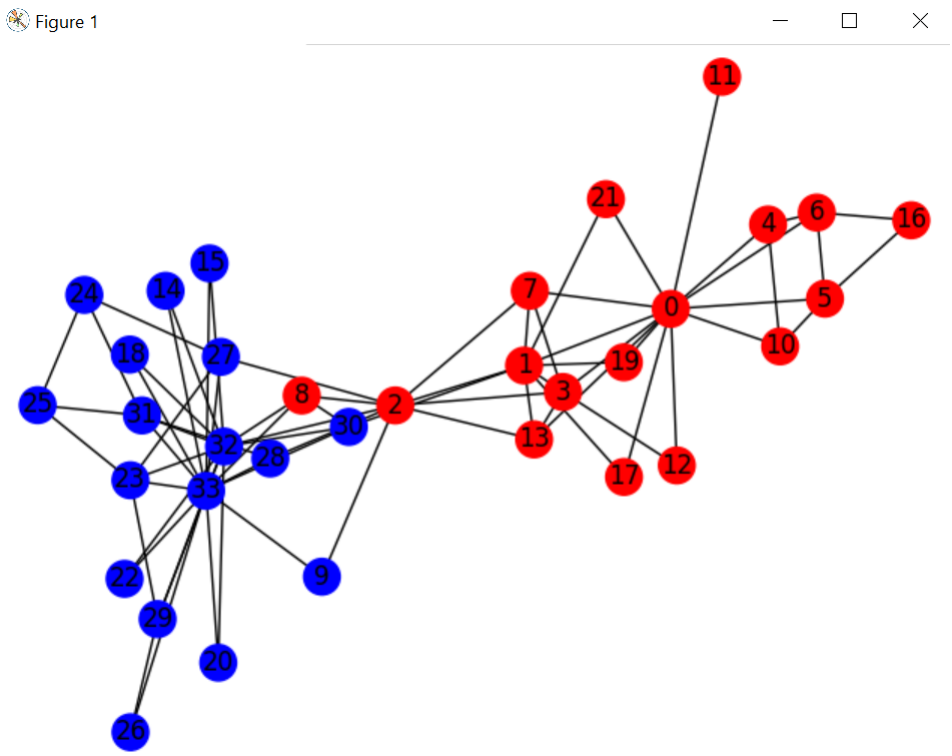


Figure 6: 5th Iteration

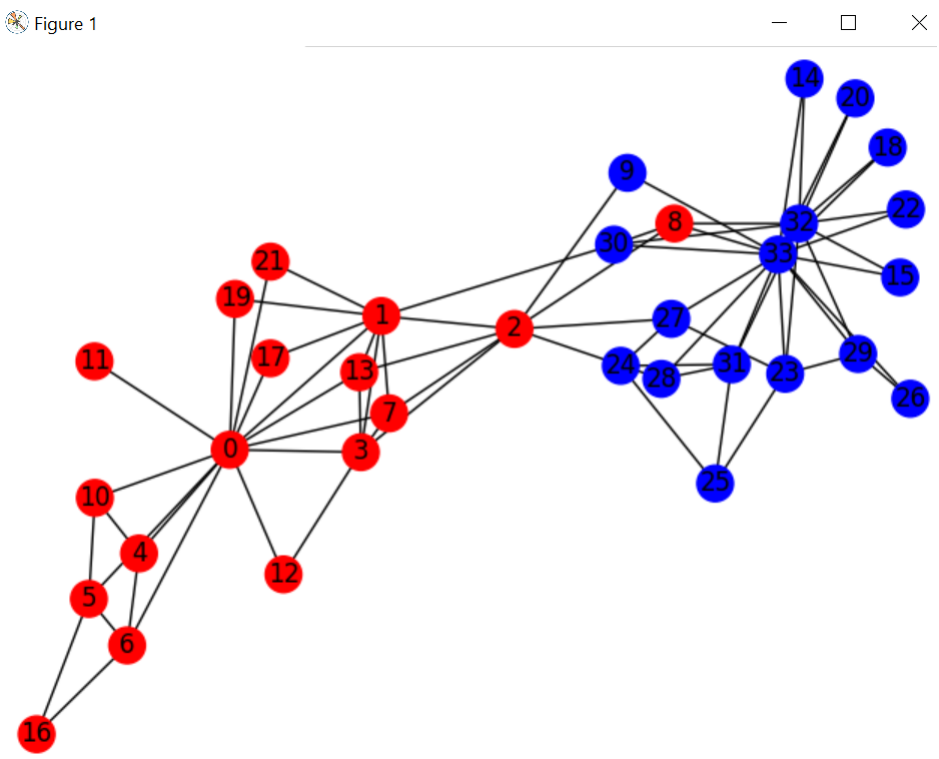


Figure 7: 6th Iteration

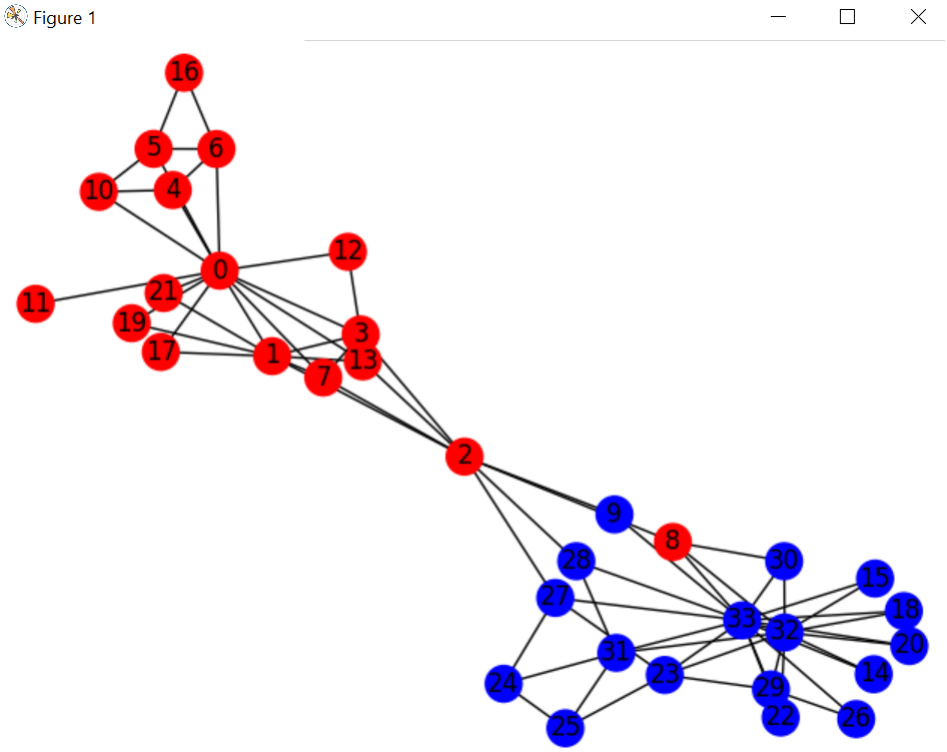


Figure 8: 7th Iteration

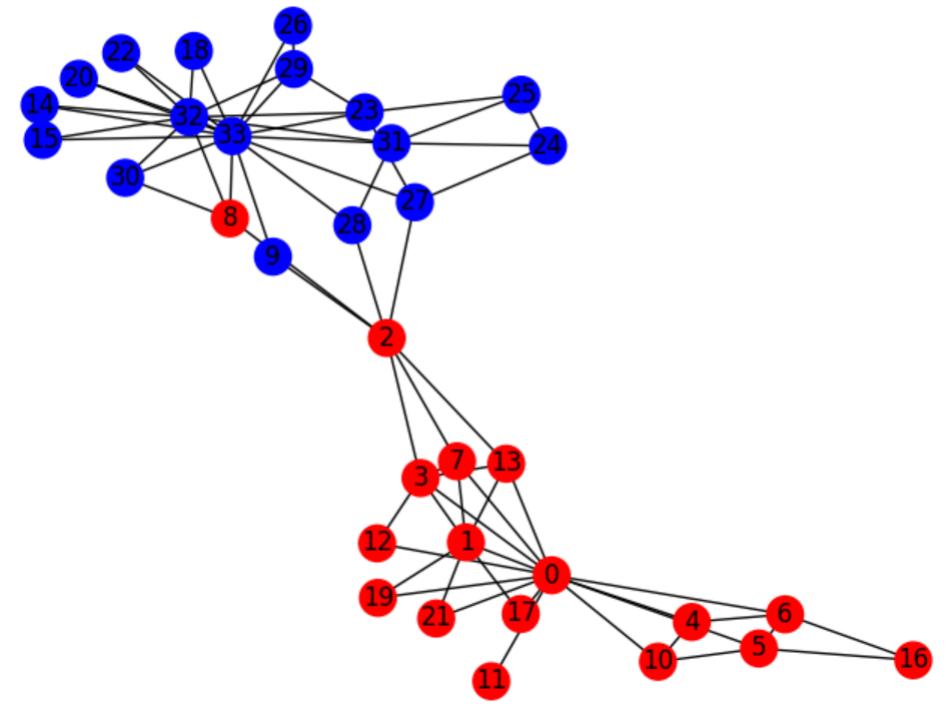


Figure 9: 8th Iteration

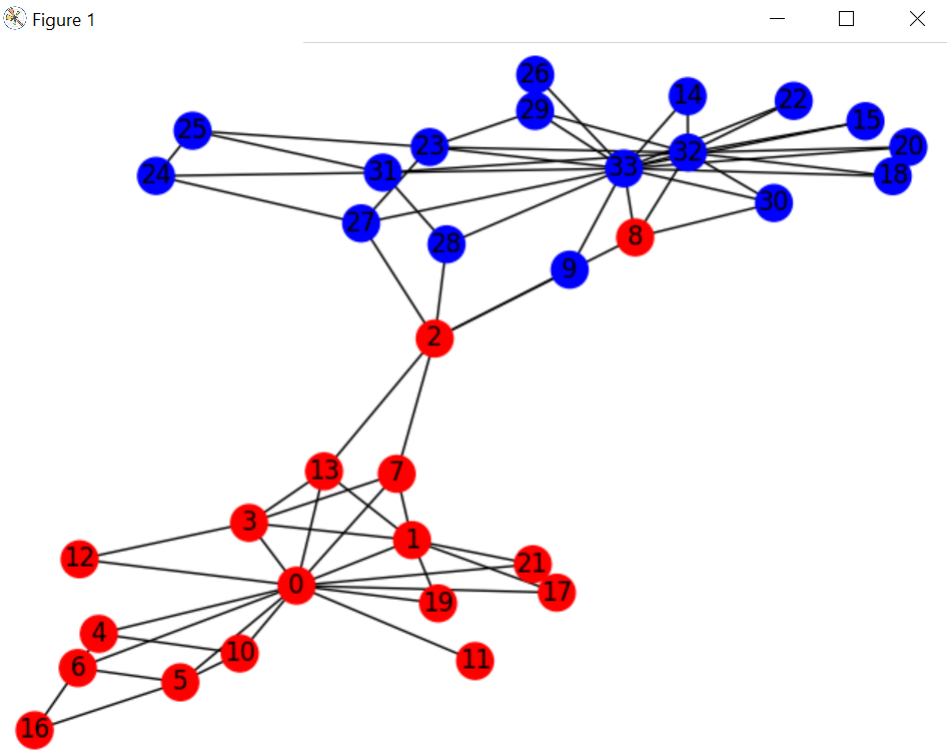


Figure 10: 9th Iteration

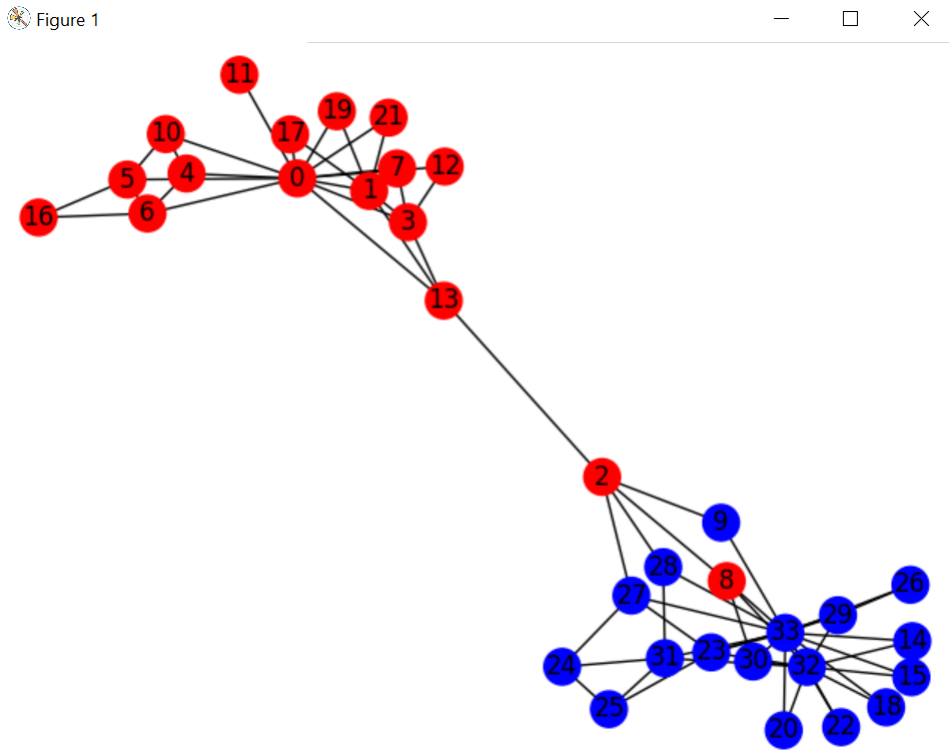


Figure 11: 10th Iteration

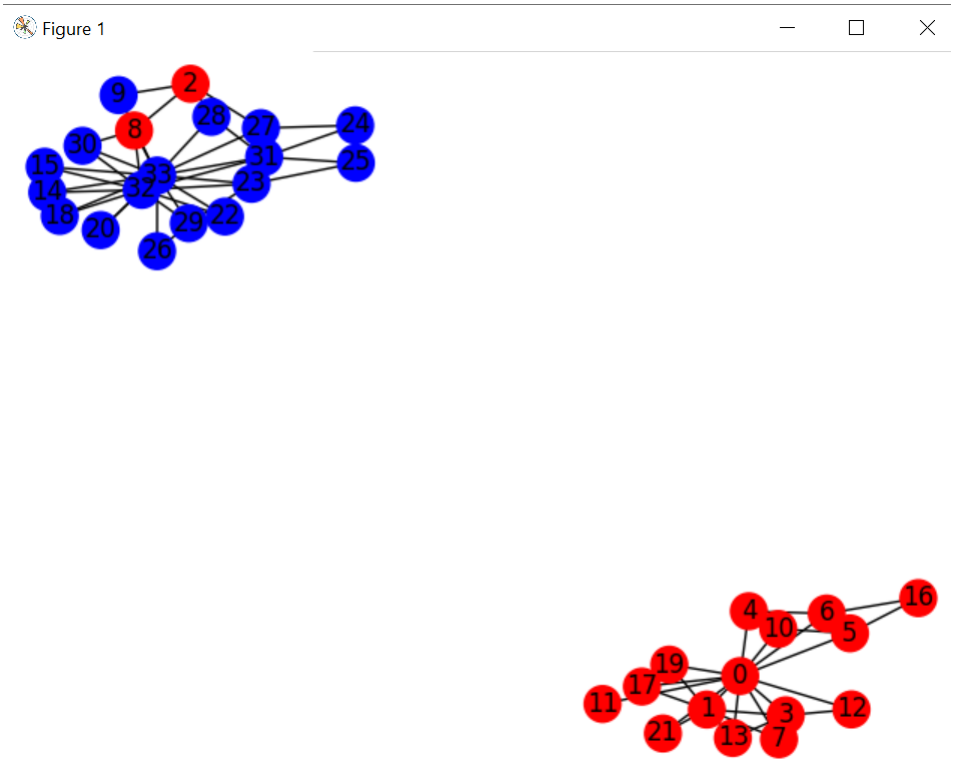


Figure 12: 11th Iteration with Two Components

## Step 3

From the steps of the split, we see that there are two members from the Mr. Hi faction (2 and 8) that end up on the John A. side. Aside from that, the groups are nearly homogeneous, with all of the original Mr. Hi nodes in one connected component, and all of the John A. nodes in the other component. Accordingly, the weighted graph of social interactions has done a decent job at predicting the split of the club into its two respective factions. Of course, it is not perfect, as stated above regarding numbers 2 and 8. The split took eleven iterations of the Girvan-Newman algorithm in which my python script found the edge with the highest edge betweenness connectivity, and then removed that edge. Please see the script and it's explanation on the last page of this report.

# Question 1

Seen below are Figures 13, 14, and 15 which show the club continuing to split into three, four, and five components respectively. Interestingly, number 9 disconnects to make the third component of a single node. Then, Mr. Hi's faction splits into two which makes four components. Lastly, the John A. faction splits into two, which makes five components, with an original member from Mr. Hi's faction in either of those groups (2 and 8).

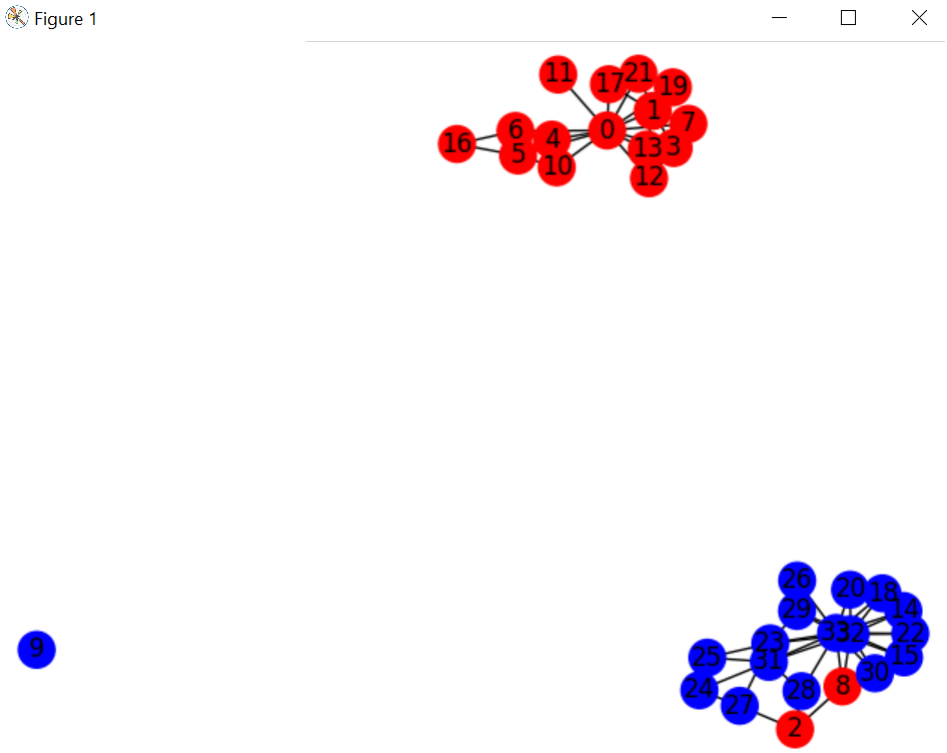


Figure 13: Three Components

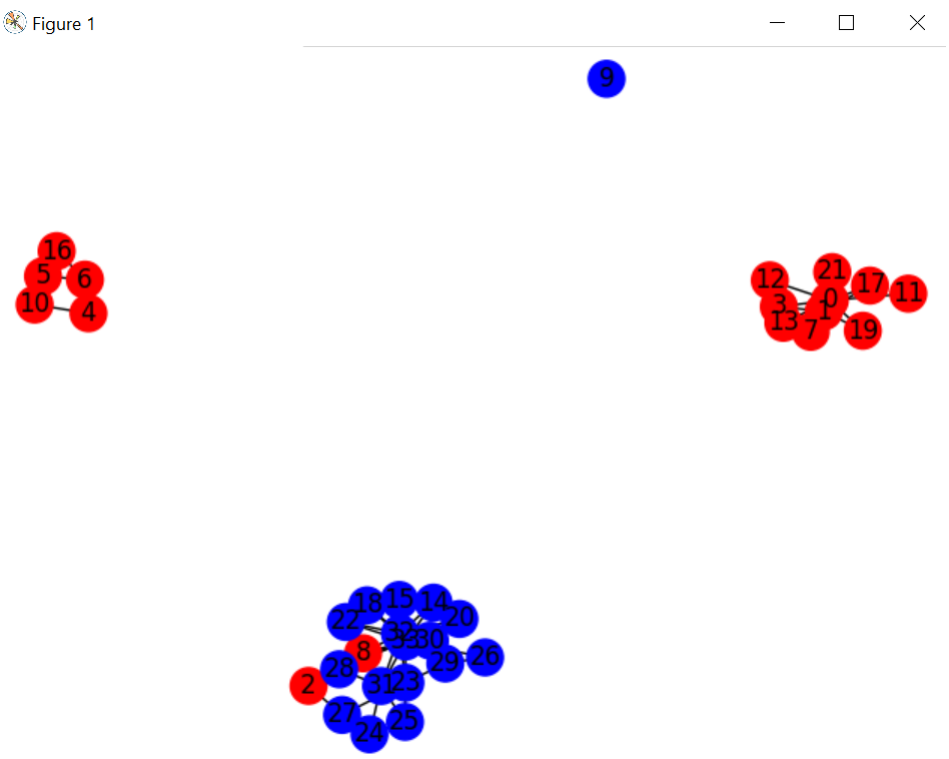


Figure 14: Four Components

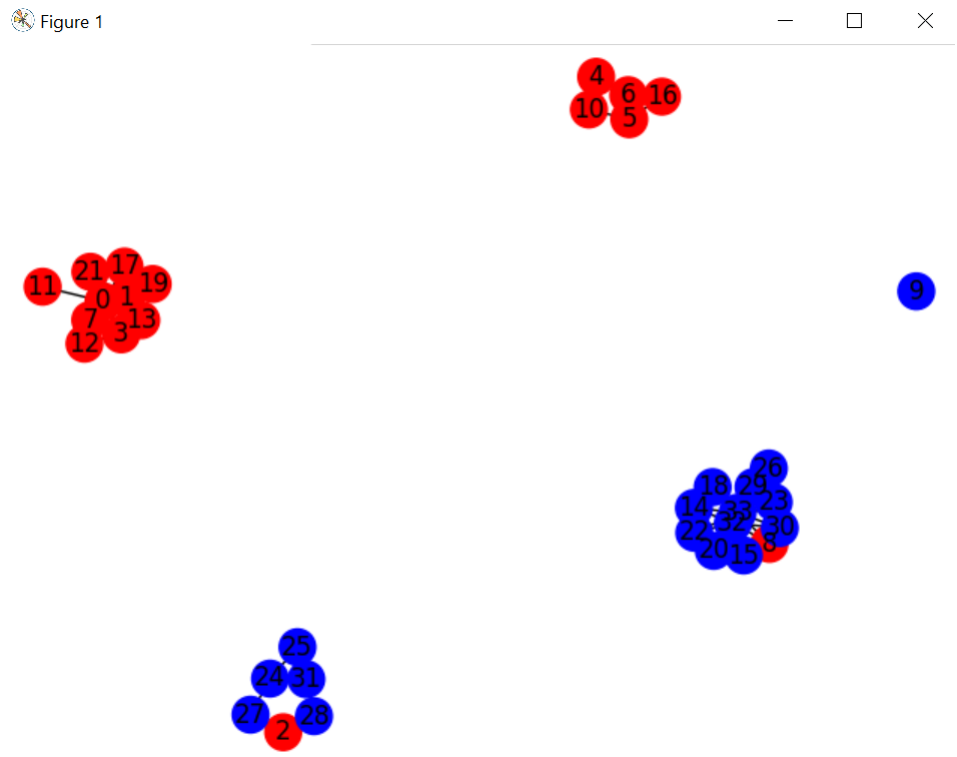


Figure 15: Five Components

## Python Script

The first block of code draws the karate club graph with each faction colored appropriately. The find_best_edge function creates a dictionary of the edge betweenness values of all the edges of the graph, then extracts the items of the graph (the edge betweenness value) into a list sorted in descending value so that the largest edge betweeness value is first. It returns the edge with the largest EBV. The third block of code is the implementation of the Girvan-Newman algorithm which continues to remove edges from the graph with the highest EBV values until the graph splits into two connected components. I used a spring layout, which naturally separated the factions for me according to the Fruchterman-Reingold force-directed algorithm. Once the graph split into two components, I break out of the while loop.

---

```
import matplotlib.pyplot as plt
import networkx as nx
import math

G = nx.karate_club_graph()
(faction1_col, faction2_col) = ('red','blue')
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':
        node_color[n] = faction2_col

nx.draw_spring(G, with_labels=True, node_color=node_color)
plt.show()

def find_best_edge(G):
    dictEdgeBetweenness = nx.edge_betweenness_centrality(G)
    #print(dictEdgeBetweenness)
    listEdgeBetweennessValues = sorted(dictEdgeBetweenness.items(),
                                       key=lambda x: x[1], reverse=True)
    #print(listEdgeBetweennessValues)
    return listEdgeBetweennessValues[0][0]

while True:
    G.remove_edge(*find_best_edge(G))
    nx.draw_spring(G, with_labels=True, node_color=node_color)
    plt.show()
    print(nx.number_connected_components(G))
    if(nx.number_connected_components(G) == 2):
        print('there are 2 components')
        nx.draw_spring(G, with_labels=True, node_color=node_color)
        plt.show()
        break
```