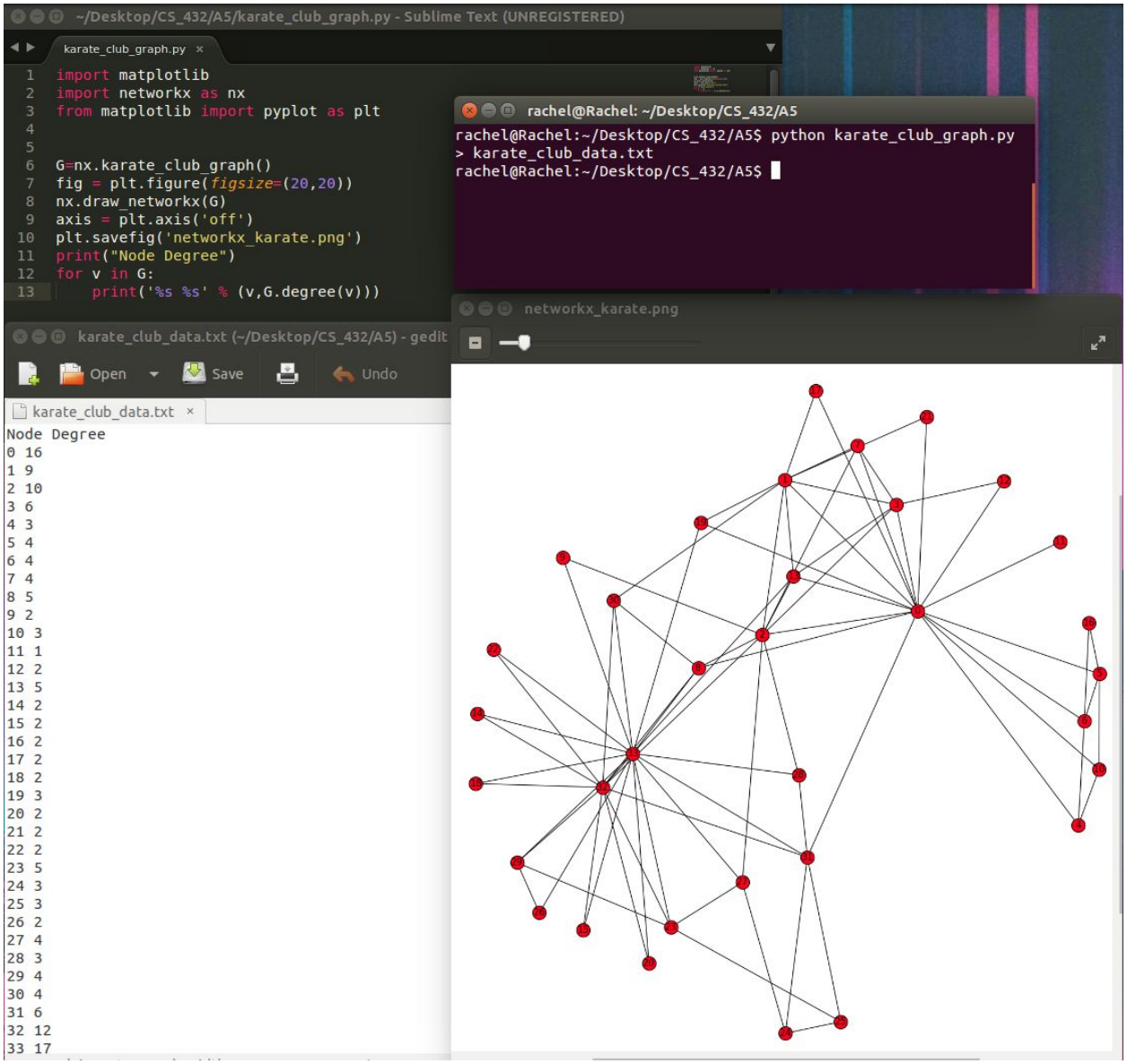
RACHEL MCCREARY
CS 432 ASSIGNMENT 5 REPORT

PART 1

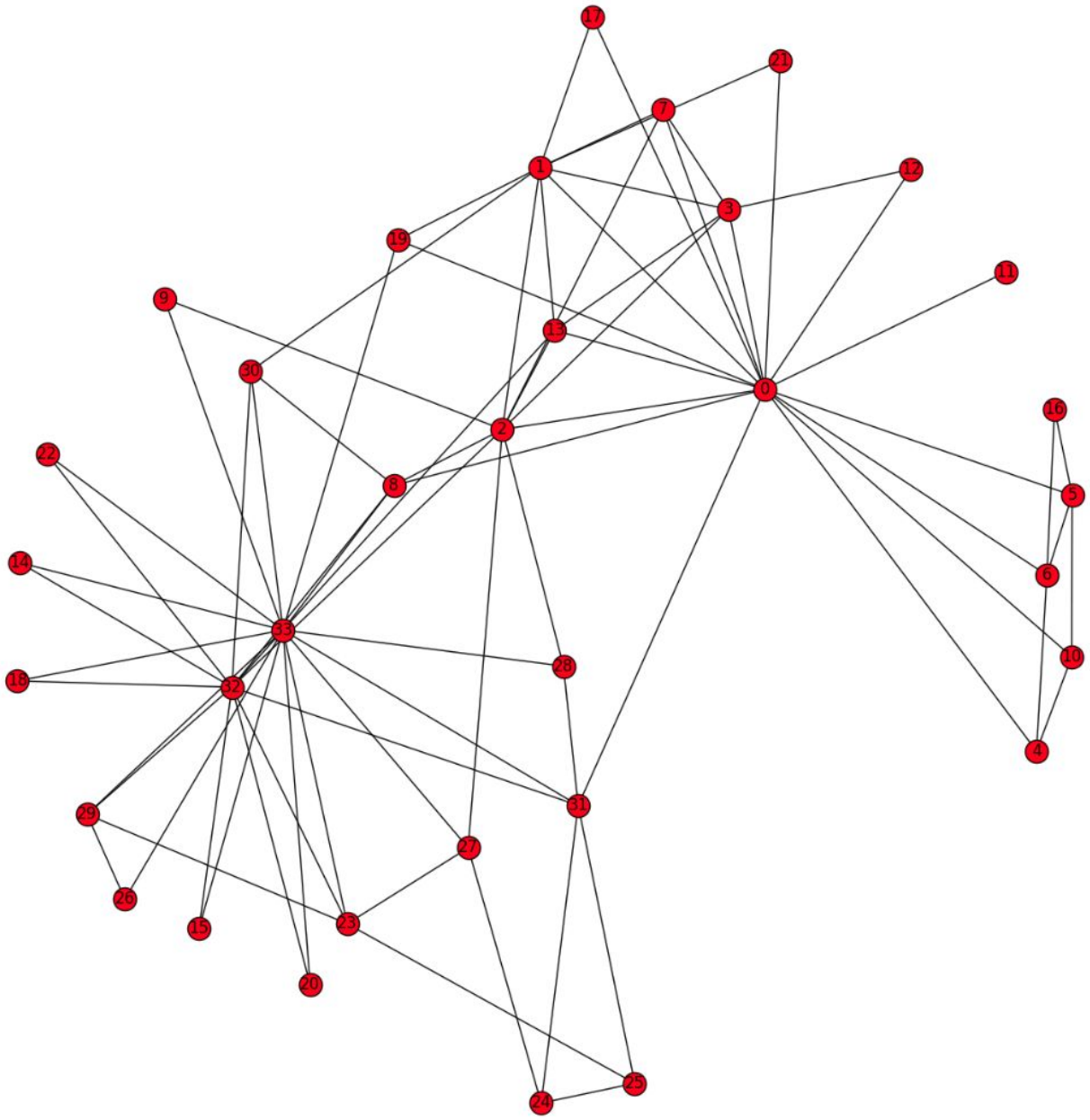
This report compares the actual Karate Club(Zachary, 1977) split with the graph I got after removing edges until the graph was disconnected (i.e., two clubs), always choosing the edge with the highest edge betweenness. This is to prove that the result of split could have been predicted by the weighted graph of social interactions.

First, using this and this for guidance, I printed out the initial Karate Club data and graph into files.

Demo:



Closer look at the graph:



Note: Unfortunately after two hours of not being able to figure out why I can't import igraph after a ton of googling and making sure everything was set up properly, I'm going to have to do without it. That's okay. Instead, I'll be using this for guidance.

So, I want to remove edges from this graph, always choosing the edge with the highest edge betweenness. Girvan & Newman algorithm to compute the betweenness of all edges:

```
while (betweenness of any edge < threshold):
        remove edge with lowest betweenness
        recalculate betweenness
```

The part of the code I'm referencing, boxed in red, does this algorithm in a while loop:

```
import networkx as nx

def girvan_newman (G):

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

    def find_best_edge(G0):
        """
        Networkx implementation of edge_betweenness
        returns a dictionary. Make this into a list,
        sort it and return the edge with highest betweenness.
        """
        eb = nx.edge_betweenness_centrality(G0)
        eb_il = eb.items()
        eb_il.sort(key=lambda x: x[1], reverse=True)
        return eb_il[0][0]

    components = nx.connected_component_subgraphs(G)

    while len(components) == 1:
        G.remove_edge(*find_best_edge(G))
        components = nx.connected_component_subgraphs(G)

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

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

    return result
```

When there are finally two components, the condition for re-entering that while loop fails, since it's succeeded in splitting the graph. If I get to the extra credit portion of this assignment, I will change this number of components to 3, then 4, then 5.

Note: Still having all sorts of problems installing what I need and tried to go through a lot of (unfortunately broken) code in blogs as a last resort, the latest shown above. A lot of tiny errors and problems popped up during this assignment while trying to recreate the split, and I don't have much to show for it. I won't worry so much next time about blowing up the group email with technical problems when Google isn't cutting it, and I'll take screenshots of everything that goes wrong if you want those in future reports.