Summary:

I completed this program for one more week. Due to the requirement, I implement all without using package such as read and write file, find shortest path. Though it exists some little bit problem because different algorithm, I think it is good enough.

Challenge and insights:

1. Choosing programming language?

   Due to the purpose of assignment is foundation, I choose Java because it is easy to write with the array collection and without network library.

2. How to save a network graph?

   I mean how to implement the data structure of a graph. I use a weight array to implement graph. There are two main ways to build graph. The first one is to build a graph class, which need to add edge and vertex, which is easy to read. However, you have to revise this data structure frequently in order to adjust different problem such as calculating degree and closeness centrality. The second way is to use an array, which is not easy to read, but it is easy to revise and you can use it in almost of situation. In my program, I use a two-dimensional array. The value of array is the weight or distance and the index

represent node. I use MAX_VALUE to represent no edge between two nodes. Though I use it in calculating degree, closeness, betweenness and clustering Coefficient, I build edge class and node class in visualizing the network. This is because I don't need to consider the weight of edges and need to revise its value frequently in visualizing.

3. Key to calculating properties of nodes

   With the help of graph array, it is easy to calculate degree and cluster coefficient. I only need to search the array and find whether two nodes are connected. It is much more complex for calculating closeness and betweenness. The key to solving it is find the shortest path. I choose Dijkstra algorithm. While searching the shortest path, my program can find the shortest path from one node to the other nodes showing step by step and the distance of the path. Then, I can calculate closeness by the distance and calculate betweenness by the step of path.

4. Implement Dijkstra

   I think implementing Dijkstra algorithm is the most difficult in this assignment. Although I spend a lot of time on it, I still complete it perfectly. My program takes only one shortest path between two nodes. In addition, it is hard to save all the paths and update the

shortest during traverse. Because a new shortest path is generated from an old shortest path, I use an array to save the previous node of last node of a path and track back to find the path and update the array after each step during traverse. The foundation of search shortest path is calculating distance and update distance. In this, I need to label the node which has been found and compare all valid path. At first, I don't know how to record the distance and new a lot of array to record it. Finally, I find going ahead to a new node only need compare at most N times, because we have recorded a previous distance array for shortest distance.

5. Visualizing network

   There are two difficulties in this section. The first is to find the forced-direct formula. There are some differences between the formulas on website and on our sides. I choose one formula on a IMB BBS website. The second is to show the graph in an appropriate place. The implement of the algorithm is much easier than showing the graph on screen. Most one third part of program is about how to show it in an appropriate scale. You have to try out parameters. Finally, the graph scale is based on the number of nodes and showing place and I think it is a correct graph.

6. Twitter API

   In this section, I think the hardest thing is to build an appropriate graph of 1000 people. What is an appropriate graph? We can't just get one person and get 1000 his followers. I think an optimal way is to get an organization twitter such as NYU twitter and get 1000 his followers, then get followers of 1000 people each. It is more possible for those people who are in the same organization to be connected. Unfortunately, I can't implement it because of the limitation of twitter API. So, I choose 31 twitters from NYU poly twitter's follower and they have 20-200 follower. I got 1923 edges and 1804 nodes.

7. Process data from twitter.

   I use array to represent graph and the graph must be a connected graph, so I have to translate discrete ID string to integer number. After doing this, I can use the function in processing Zachary data.

Interpretation of results:

Zachary:

Degree:

most of nodes have less than 5 degree and few of nodes have more than 10 degree. I consider the nodes with less degree are students and the nodes with more degree are coach.

Closeness:

In this data, the nodes of the higher degree have higher closeness, which means it is a center of the graph. However the difference between other node's closeness isn't too far. I think this is because the network is not spread a lot and the connection between these people is tight.

Betweenness:

Some of betweenness of node are much more than others. Node 1 is 518 times and node 20 is 255 times, however some of node is zero and the other is less than 100 times. I think the higher betweenness node is regarded a core people in this network relationship. Communication has to go though them.

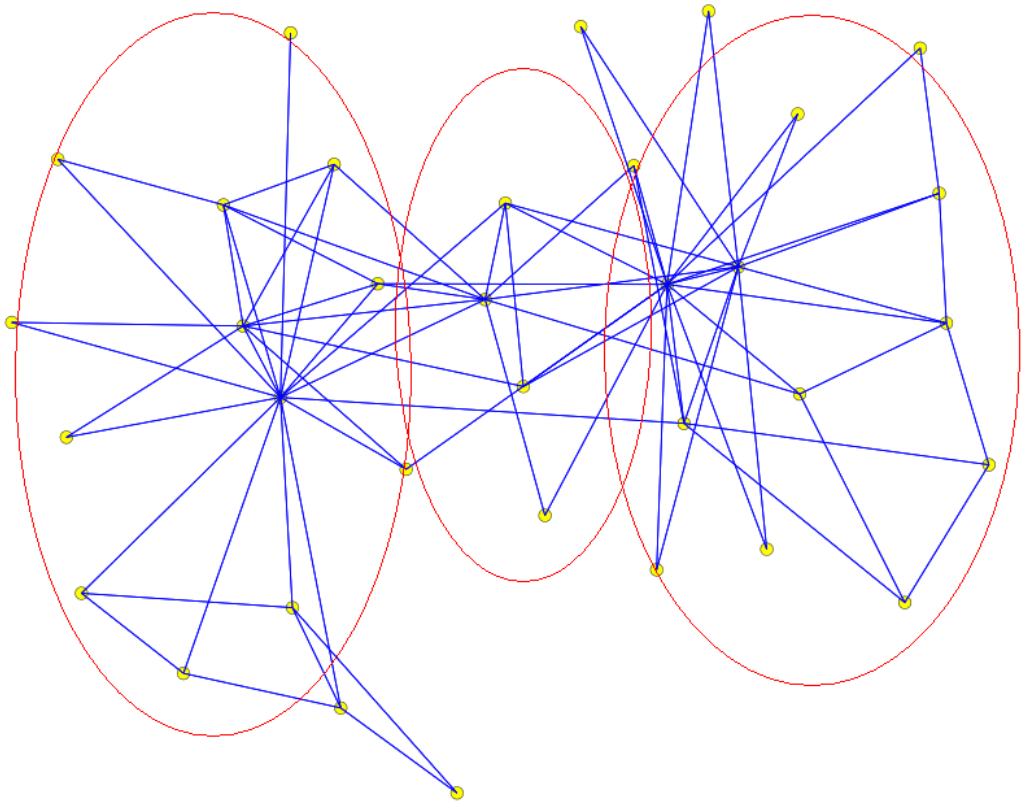
Cluster Coefficient:

People who has a higher degree has a lower clustering coefficient. It is because the people has lower degree know little people and is in a small network circle. The denominator and the numerator are small. However, the people who have the higher degree and lower betweenness are connected with a lot of people, but these people don't connect with each other. Although the numerator is big, the denominator is much bigger. Node 12 is clustering coefficient is NaN, which means there is

only one edge to node 12.

Network Visualization:

From the picture, it is easy to see there are three groups in this network relationship.



In addition, most of nodes are connected with few other nodes. In other hand, few nodes are connected to most of other node. This is very reasonable. In actual life, most of people always are connected with some central people and ignore others.

Twitter Data:

I choose data from a center of a group. I take 31 twitters from NYU-poly's

twitter’s followers, I think people following same organization are more likely to be connected. However, the followers of these 31 twitters have little intersection. For example, A has 100 followers, B has 200 followers but none of 200 people follow A. The graph I build has 31 centers, so the betweenness of other nodes is zero. The effect of center is very obvious in this network, the closeness of 31 nodes are much larger than others, which means they are easy to connect with others. Most of node’s clustering coefficient is zero or NaN because the data is not big enough to get enough information.