**Social Network Analysis of Our Class**

18S151514孟繁续

# INTRODUCTION

In recent years, Social Network Analysis (SNA) is very popular. A social network is a set of people (or organizations or other social entities) connected by a set of social relationships. Social network analysis provides both a visual and a mathematical analysis of human relationships. In this paper, We have collected three kinds of social network data in our class, including knowing someone’s name, knowing someone’s hometown, and knowing someone’s dialect. And then we will do some computer simulations to these data and analyze the networks to make some conclusion about the relationship in our class.

# METHODS AND TOOLS

The project were programmed with python language and running on the anaconda and Pycharm, downloaded at :

1) https://www.anaconda.com/download/#windows

2) <http://www.jetbrains.com/pycharm/download/#section=windows>

Besides, to complete the simulation work, some open-source Python libraries are needed:

* 1. NumPy: [http://numpy.scipy.org/.](http://numpy.scipy.org/)
  2. MatplotLib: [http://matplotlib.sourceforge.net](http://matplotlib.sourceforge.net/).
  3. xlrd: https://pypi.python.org/pypi/xlrd.
  4. xlwt: <https://pypi.python.org/pypi/xlwt/>.
  5. NetworkX: [http://networkx.lanl.gov/.](http://networkx.lanl.gov/)
  6. Tkinter: <https://wiki.python.org/moin/TkInter>

There are three way to run this program:

1. The simplest is run GUI.py with python platform.Chose which data (name, hometown, dialect) you want to read. Then click the button you interest.
2. The second way is run readme.ipynb, use shift+enter to run every cell
3. The third way is run \*.py file on pycharm or input command in termal.

# MODELING AND ANALYSIS

## GUI

The GUI is exploit by python ‘s tkinder model, and it is shown as Figure 1:

## 

Figure1: GUI

## load data

Choose name and press LoadData button in GUI;

Or, open the anaconda notebook platform, run the cell[1] in readme.ipynb file;

Or open the pycharm platform, run: >>>python loadData.py name

to read the first sheet of original name.xlsx, which describes the relationship of knowing someone’s name.

And the result is show as figure 2.

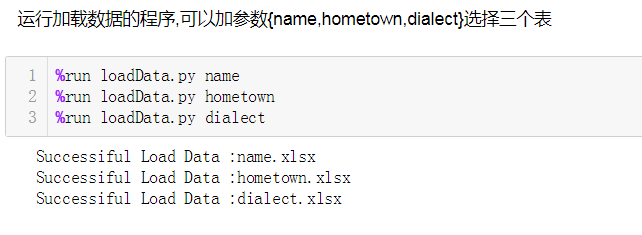


Figure2: Load Data

## Knowing someone’s name Modeling the network

Then, build the graph base on file network.xslx.

Choose name and press DrawGraph button in GUI;

Or run the cell[2] in readme.ipynb ;

Or open the pycharm platform and run : >>>python drawGraph.py name

the network is shown as figure 3.

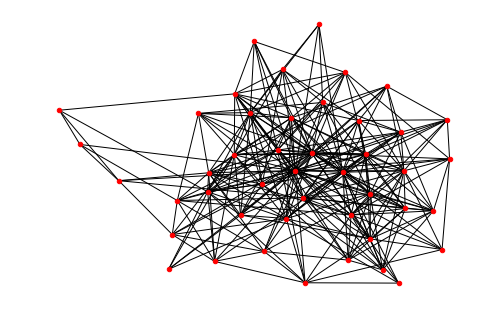


Figure3: DrawGraph

## Calculate Degree of the node

Choose name and press DrawGraph button in GUI;

Run the cell[3] in readme.ipynb

or open the pycharm platform and run: >>>python Degree.py name

To show the degree of these nodes as Figure4

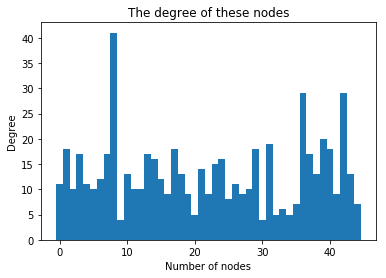


Figure4: The degree of these nodes

Calculate some basic information and analyze the network properties of the graph showed in table1:

TABLE 1: Network properties of figure1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | type | total nodes | total edges | average degree |
| Knowing someone’s name | undirected | 45 | 297 | 13.2 |

## The distribution of the node

The nodes of larger degree is more important in the network. The degree metric can help us find the pivotal roles in our class, who contact with more people.

Choose name and press Distribution button in GUI;

Or run the cell[4] in readme.ipynb

Or open the pycharm platform and run:

>>>python Distribution.py name

The degree distribution graph is showed in Figure5:

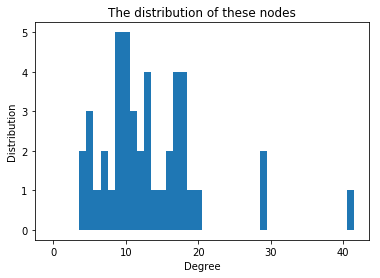


Figure5: The distribution of these nodes

From the degree distribution, we want to find the top ten students who has largest degree. We execute the command on the platform, and get the ten students:

[No.8, No.42, No.36, No.39, No.31, No.29, No.1, No.40, No.17, No.13]

These ten people know more students’ name in our class.

## Average shortest path length

Since the graph is unconnected, remove the single points and compute average path length of connected subgraph.

Choose name and press AverageShortestPathLength button in GUI;

Or run the cell[5] in readme.ipynb

or open the pycharm platform and run:

>>>python ASPL.py name

We get average path equals to 1.72424. In other words, the graph has strong ability to transmit information. We can see from figure2, the graph is very dense. For any nodes A and B, if A want to pass a message to B, the average path between them is 1.72424

## Clustering coefficient

Choose name and press Clustering button in GUI;

Or run the cell[6] in readme.ipynb

or open the pycharm platform and run:

>>>python Clustering.py name

We get clustering coefficient equals to 0.58666. In other words, the ratios our friends know each other is bigger than a half.

## Coreness

A higher-coreness node is more important than a lower-coreness node in a network. And if a network with a higher coreness will have better robustness against intentional attacks.

Choose name and press Coreness button in GUI;

Or run the cell[7] in readme.ipynb

or open the pycharm platform and run:

>>>python Coreness.py name

We compute the coreness of every nodes:

[9, 10, 8, 11, 8, 10, 10, 11, 11, 4, 10, 9, 8, 10, 10, 10, 8, 11, 9, 8, 5, 9, 8, 11, 10, 7, 10, 8, 8, 11, 4, 11, 5, 4, 5, 7, 11, 10, 9, 11, 10, 8, 11, 10, 7]

The Coreness of the network is 11.

We remove the node whose degree less than flag, and output the left graph step by step as Figure6.

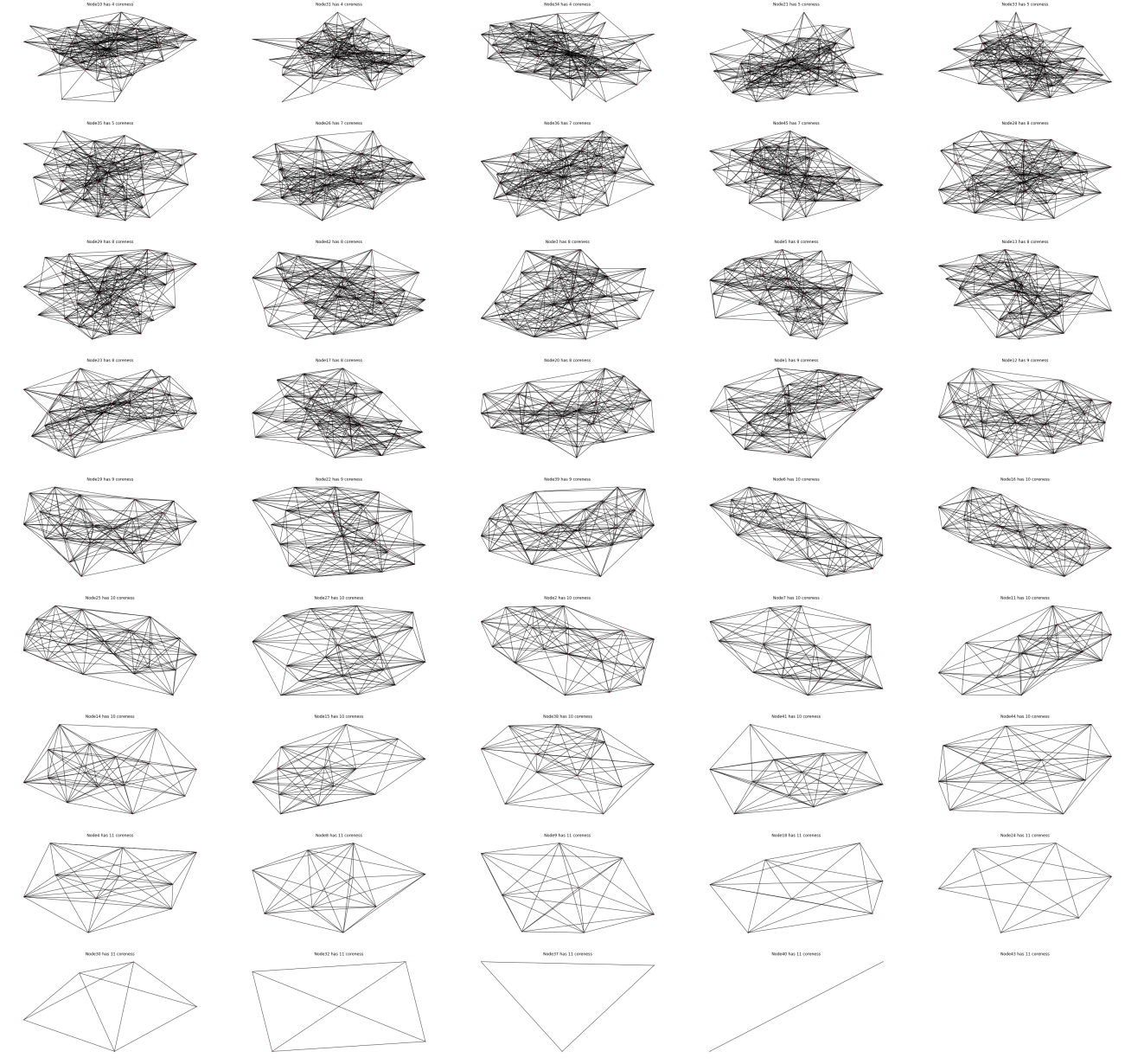


Figure6: coreness calculating

## Dynamical behaviors

We test the robustness and fragility of figure 3 with some attacks. The attacks mean remove some nodes in the graph. There are two kinds of attacks: one is removing the nodes with larger degree, which is an intentional attack. And the other is randomly removing the nodes, which is a random attack.

Choose name and press Attack button in GUI;

Or run the cell[8] in readme.ipynb

or open the pycharm platform and run:

>>>python Attack.py name

We can get the stimulation with attacks which shown as figure 7.

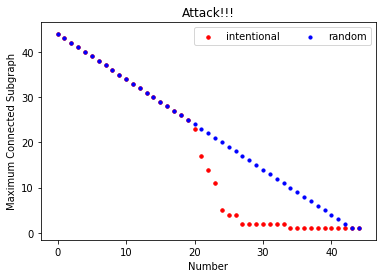
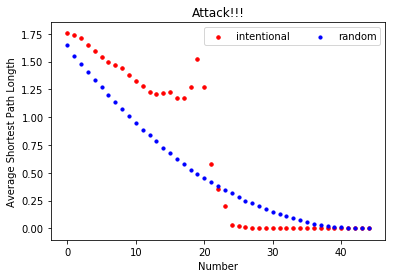


Figure 7. Robustness and fragility

From figure 7, we can find that for intentional attack, the graph quickly divided into some small sub graphs, and the average length became longer. Comparing with intentional attack, the random attack’s speed of dividing into sub graphs seems to be a liner function. It doesn’t decline as fast as a intentional attack. So under a random attack, the average path length of the graph maintains for a long time while under an intentional attack, it changes quickly.

In conclusion, the figure 3 seems to have good robustness, but have better robustness under a random attack. It indicates the importance of the nodes with large degree.

## Knowing someone’s hometown

## Modeling the network

Build the graph base on file DrawGraph.xslx.

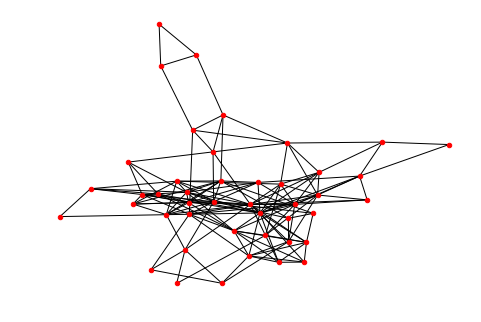
Choose hometown and press Coreness button in GUI;

Or run the cell[9] in readme.ipynb

or open the pycharm platform and run

>>>python drawGraph.py hometown

The network is shown as figure 8.



From the figure we can find it’s more thin than the first figure, because it’s difficult for us to know each other’s hometown.

Figure8: Knowing someone’s name

## Calculate Degree of the node

Choose hometown and press Degree button in GUI;

Or run the cell[10] in readme.ipynb

Or open the pycharm platform and run:

>>>python Degree.py hometown

To show the degree of these nodes as Figure9

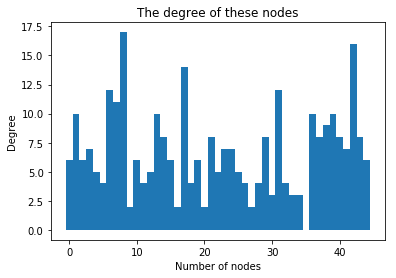


Figure9: The degree of these nodes

Calculate some basic information and analyze the network properties of the graph showed in table2:

TABLE 2: Network properties of figure1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | type | total nodes | total edges | average degree |
| Knowing someone’s name | undirected | 45 | 152 | 6.7555 |

## The distribution of the node

Choose hometown and press Distribution button in GUI;

Or run the cell[11] in readme.ipynb

Or open the pycharm platform and run:

>>>python Distribution.py hometown

The degree distribution graph is showed in Figure10:

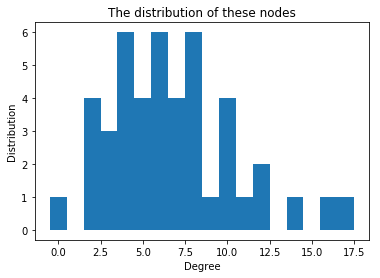


Figure10: The distribution of these nodes

From the degree distribution, we find the top ten students who have largest degree:

[No.8, No.42, No.17, No.6, No.31, No.7, No.13, No.39, No.36, No.1]

These ten people know more students’ hometown in our class.

## Average shortest path length

Choose hometown and press AverageShortestPathLength button in GUI;

Or run the cell[12] in readme.ipynb

Or open the pycharm platform and run:

>>>python ASPL.py hometown

We get average path equals to 2.28889. In other words, the graph still has strong ability to transmit information. For any nodes A and B, if A want to pass a message to B, the average path between them is 2.28889.

## Clustering coefficient

Choose hometown and press Clustering button in GUI;

Run the cell[13] in readme.ipynb

or open the pycharm platform and run:

>>>python Clustering.py hometown

We get clustering coefficient equals to 0.43268. In other words, the ratios our friends know each other is smaller than a half.

## Coreness

Choose hometown and press Coreness button in GUI;

Or run the cell[14] in readme.ipynb

Or open the pycharm platform and run:

>>>python Coreness.py hometown

We compute the coreness of every nodes:

[3, 5, 4, 4, 4, 4, 5, 5, 5, 2, 5, 3, 5, 6, 5, 4, 2, 6, 4, 4, 2, 5, 4, 5, 5, 4, 4, 2, 3, 5, 2, 5, 4, 2, 3, 0, 5, 5, 5, 5, 5, 4, 5, 6, 5]

The network’s coreness is 6

We remove the node whose degree less than flag, and output the left graph step by step as Figure11:

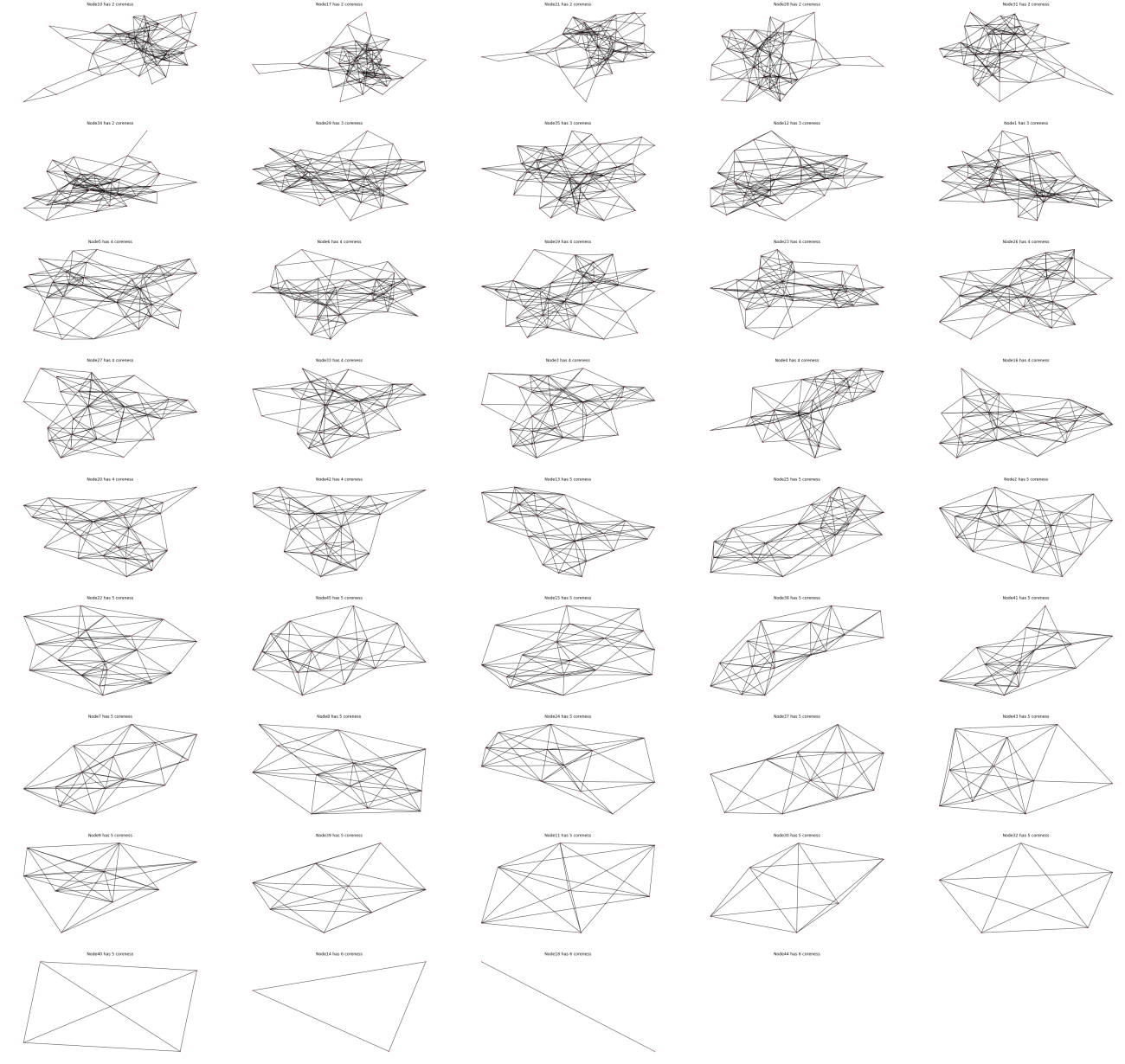


Figure11: coreness calculating

## Dynamical behaviors

We test the robustness and fragility of figure 7 with some attacks. There are two kinds of attacks: one is removing the nodes with larger degree, which is an intentional attack. And the other is randomly removing the nodes, which is a random attack.

Choose hometown and press Attack button in GUI;

Or run the cell[15] in readme.ipynb

Or open the pycharm platform and run:

>>>python Attack.py hometown

We can get the stimulation with attacks which shown as figure 12.

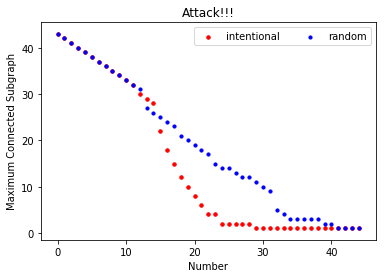
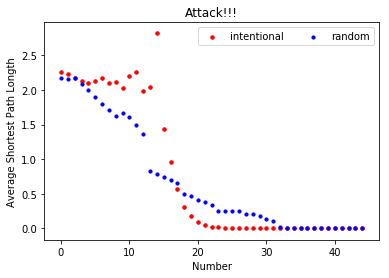


Figure 12. Robustness and fragility

From figure 12, we can find that for intentional attack, the graph quickly divided into some small sub graphs, and the average length became longer. Comparing with intentional attack, the random attack’s speed of dividing into sub graphs seems to be a liner function. It doesn’t decline as fast as an intentional attack. So under a random attack, the average path length of the graph maintains for a long time while under an intentional attack, it changes quickly.

Compare with figure 7, intentional attack incline earlier, we know that the first graph have better robustness.

## Knowing someone’s dialect

## Modeling the network

Build the graph base on file dialect.xslx.

Choose dialect and press DrawGraph button in GUI;

Or run the cell[16] in readme.ipynb

Or open the pycharm platform and run

>>>python drawGraph.py dialect

the network is shown as figure 13.

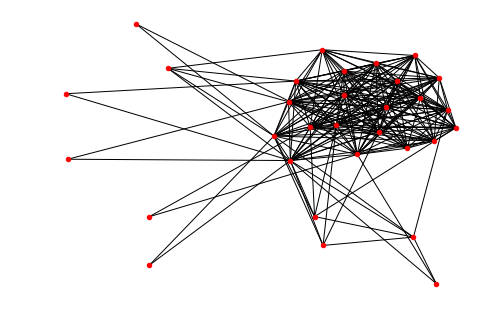


Figure13: Knowing someone’s dialect

From this figure, we can find it’s very dense, for most students can recognize other’s dialect even though they don’t know them. However, the degree of some node is very small, for it’s a dialect difficult to recognize.

## Calculate Degree of the node

Choose dialect and press Degree button in GUI;

Or run the cell[17] in readme.ipynb

Or open the pycharm platform and run:

>>>python Degree.py dialect

To show the degree of these nodes as Figure14



Figure14: The degree of these nodes

Calculate some basic information and analyze the network properties of the graph showed in table3:

TABLE 3: Network properties of figure1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | type | total nodes | total edges | average degree |
| Knowing someone’s name | undirected | 32 | 223 | 13.9375 |

## The distribution of the node

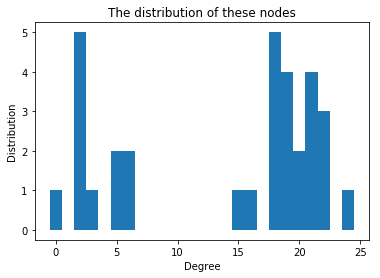
Choose dialect and press Distribution button in GUI;

Or run the cell[18] in readme.ipynb

Or open the pycharm platform and run:

>>>python Distribution.py dialect

The degree distribution graph is showed in Figure15:



Figur15: The distribution of these nodes

From the degree distribution, we want to find the top ten students who has largest degree. We execute the command on the platform, and get the ten students:

[No.11, No.4, No.24, No.29, No.12, No.20, No.21, No.17, No.2, No.26]

These ten people know more students’ dialect in our class.

## Average shortest path length

Since the graph is unconnected, remove the single points and compute average path length of connected subgraph.

Choose dialect and press AverageShortestPathLength button in GUI;

Or run the cell[19] in readme.ipynb

Or open the pycharm platform and run:

>>>python ASPL.py dialect

We get average path equals to 1.451613. In other words, the graph has strong ability to transmit information. We can see from figure12, the graph is very dense. For any nodes A and B, if A want to pass a message to B, the average path between them is 1.451613.

## Clustering coefficient

Choose dialect and press Clustering button in GUI;

Or run the cell[20] in readme.ipynb

Or open the pycharm platform and run:

>>>python Clustering.py dialect

We get clustering coefficient equals to 0.8035957. In other words, the ratios our friends know each other’s dialect is 80%.

## Coreness

A higher-coreness node is more important than a lower-coreness node in a network. And if a network with a higher coreness will have better robustness against intentional attacks.

Choose dialect and press Coreness button in GUI;

Or run the cell[21] in readme.ipynb

Or open the pycharm platform and run:

>>>python Coreness.py dailect

We compute the coreness of every nodes:

[2, 2, 16, 5, 16, 5, 15, 15, 0, 3, 2, 15, 16, 16, 16, 2, 5, 15, 15, 5, 16, 16, 15, 15, 16, 15, 16, 15, 15, 16, 16, 2]

The Coreness of the network is 16

We remove the node whose degree less than flag, and output the left graph step by step as Figure16

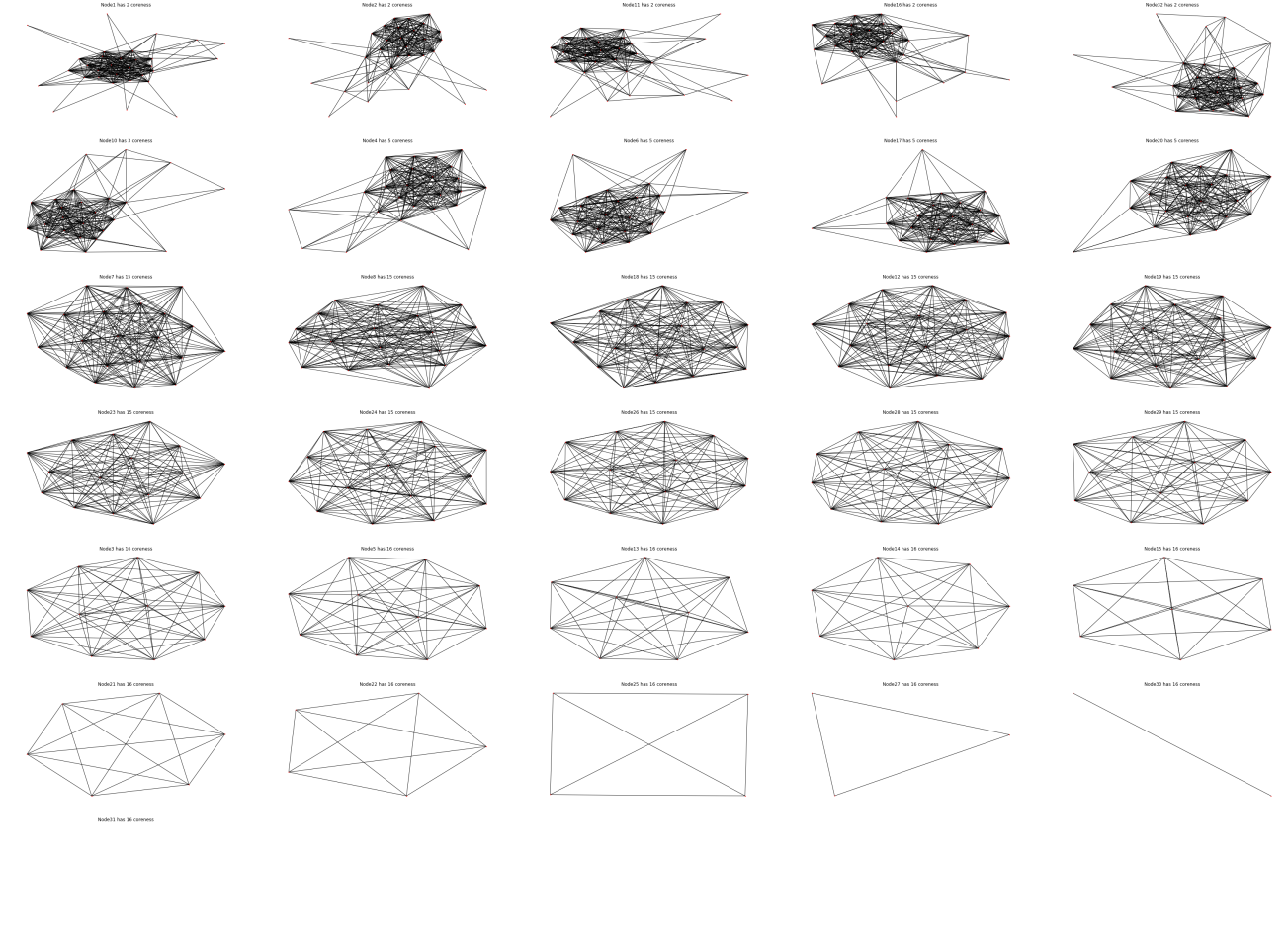


Figure16: coreness calculating

## Dynamical behaviors

We test the robustness and fragility of figure 12 with some attacks. The attacks mean remove some nodes in the graph. There are two kinds of attacks: one is removing the nodes with larger degree, which is an intentional attack. And the other is randomly removing the nodes, which is a random attack.

Choose dialect and press Attack button in GUI;

Or run the cell[22] in readme.ipynb

Or open the pycharm platform and run:

>>>python Attack.py dialect

We can get the stimulation with attacks which shown as figure 17.

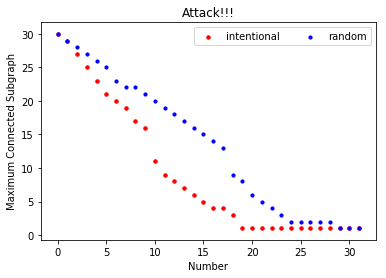
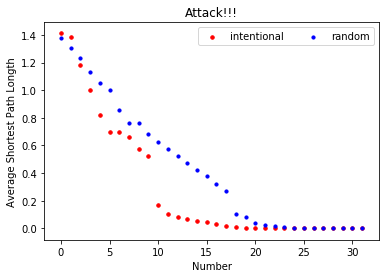


Figure 17. Robustness and fragility

From figure 17, we can find that for intentional attack, the graph quickly divided into some small sub graphs. Comparing with intentional attack, the random attack’s speed of dividing into sub graphs seems to be a liner function. It doesn’t decline as fast as an intentional attack.

Compare with Figure 7 and figure 12, it’s has better robustness.

# CONCLUSION

In this paper, we presented complex network theorem to analyze three relations of our class. The three relations included knowing someone’s name, knowing someone’s hometown and knowing someone’s dialect. Through simulating and analyzing the network properties and dynamical behaviors, we find that:1) A few people control the power in the first network. That is, they are very popular among the students and play an important role to transmit information.2) The second network seems to be sparser with fewer edges, that means people don’t know each other’s hometown very much unless friends.3) Third network shows plenty of dialects are easy to recognize, but some are difficult to recognize.