DS210 Final Project

My project aims to identify the users with the greatest influence on the Facebook network. I implemented 3 calculations to determine the influence of each node in this data set. The first is the **degree of a node**, namely the number of nodes that are directly connected to the target node. The second is the **number of neighbors at distance 2**, which counts the number of nodes the target nodes' direct neighbors are connected to. The last one is **betweenness centrality**. This calculates the percentage of times a node has appeared in the shortest paths of any other pairs of nodes in the network. This can tell if each node is located towards the center of the network or not.

All three of these functions are located in the graph.rs module. The logic behind them is explained in detail in the comments. There's also a module named algorithm.rs where I implemented the calculation for the Dijkstra shortest paths in order to obtain the shortest paths of all possible pairs of nodes. This way, I can calculate betweenness centrality. In my main function that lives in the main.rs module, I called the functions in the other 2 modules to calculate the degree of each node, the number of neighbors at distance 2, and betweenness centrality. I then printed out the top 10 nodes in each of the 3 categories. My logic for splitting my code into these 3 modules is that the construction of the graph and calculations are all located in the graph module; the main function is in its own module; and an algorithm that one of the calculations needs is in another module so that the graph module is tidier.

The code usually takes around a minute and 15 seconds to run with cargo run. This is mainly caused by the computation of Dijkstra shortest paths as it keeps iterating through different combinations of nodes in order to return every shortest path in the data set. However, it only takes 17 seconds to run with cargo run --release.

The following are the results of the code.

```
Top 10 nodes with highest betweenness centrality:
1. Node 107: 0.2462
2. Node 1684: 0.1784
3. Node 1912: 0.1492
4. Node 3437: 0.1283
5. Node 0: 0.0831
6. Node 348: 0.0433
7. Node 686: 0.0357
8. Node 414: 0.0215
9. Node 1465: 0.0172
10. Node 698: 0.0163
Top 10 nodes highest degrees:
1. Node 107: 1045
2. Node 1684: 792
3. Node 1912: 755
4. Node 3437: 547
5. Node 0: 347
6. Node 2543: 294
7. Node 2347: 291
```

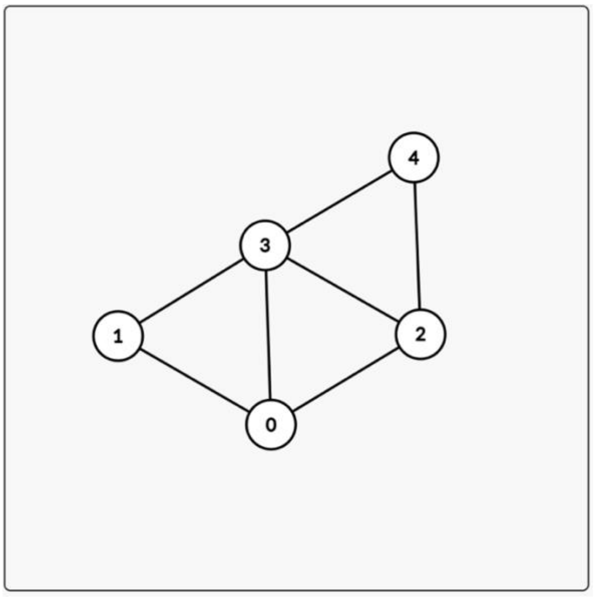
```
8. Node 1888: 254
9. Node 1800: 245
10. Node 1663: 235
Top 10 nodes with most neighbors at distance 2:
1. Node 58: 2903
2. Node 171: 2145
3. Node 563: 2029
4. Node 428: 2005
5. Node 1534: 1947
6. Node 1666: 1944
7. Node 1171: 1899
8. Node 1726: 1801
9. Node 1450: 1799
10. Node 1419: 1798
```

A lot of insights can be drawn from these results. For example, the 5 nodes with the highest betweenness centrality are the same as the 5 with the highest degrees. The explanation could be that since these nodes are connected to so many other nodes, there's a high likelihood that the shortest paths between any other pairs of nodes have to go through them.

It's also interesting that the nodes with the greatest number of neighbors at distance 2 are quite different from the nodes with the highest degrees. An explanation of this could be that even though a node may not have a lot of direct neighbors, most of its neighbors are very well connected. As a result, a node with a high number of neighbors at distance 2 doesn't necessarily mean that it has a high degree.

Based on the output of the code, we can tell that user 107, 1684, and 1912 are likely to be the top three users with the greatest influence on Facebook. This is because they not only are connected directly with the greatest number of people but are also located towards the center of the vast network of users due to their high betweenness centrality value.

In my test code in the main.rs module, I made the following small undirected graph:

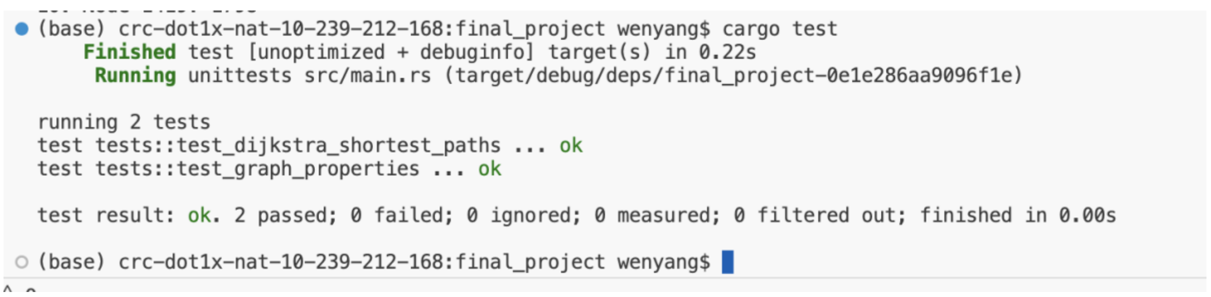


```
0 1
0 2
0 3
1 3
2 3
2 4
3 4
```

As you can see from the graph:
Node 0 has a degree of 3 and 1 neighbor at distance two
Node 1 has a degree of 2 and 2 neighbors at distance two
Node 2 has a degree of 3 and 1 neighbor at distance two
Node 3 has a degree of 4 and 0 neighbor at distance two
Node 4 has a degree of 2 and 2 neighbors at distance two

For node 0, the shortest path to each node is the following:
Shortest path to node 0: []
Shortest path to node 1: [0, 1]
Shortest path to node 2: [0, 2]
Shortest path to node 3: [0, 3]
Shortest path to node 4: [0, 2, 4]

I tested these results against the results obtained by calling my node_degree, neighbors_at_dist_2, and dijkstra_shortest_paths functions using the assert_eq! function. This way, I'm able to determine if my calculations for node degree, neighbors at distance 2, and Dijkstra shortest paths behaves correctly. The results of my test code demonstrate that these functions behave as expected.



```
● (base) crc-dot1x-nat-10-239-212-168:final_project wenyang$ cargo test
      Finished test [unoptimized + debuginfo] target(s) in 0.22s
       Running unittests src/main.rs (target/debug/deps/final_project-0e1e286aa9096f1e)

  running 2 tests
  test tests::test_dijkstra_shortest_paths ... ok
  test tests::test_graph_properties ... ok

  test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

○ (base) crc-dot1x-nat-10-239-212-168:final_project wenyang$ 
```