Data Set Link:https://snap.stanford.edu/data/github-social.html

Resources/Collaboration: Rust documentation, Lecture notes, Stackoverflow, VSCode Rust extension

This dataset is an undirected dataset that shows GitHub developers and their GitHub followers. This dataset includes Github users with 10 or more repositories which is 37,700 users. Each user is a node, and the edges for this dataset are if a user (node) is a follower of another user. There are 289,000 edges in this dataset which is fairly large. However, compared to other social networking platforms it is relatively small. Because this dataset is fairly large, running a BFS on the graph for every node takes around 10 minutes.

I thought this would be an interesting dataset because I just started using GitHub for some personal programming projects. However, unlike many social networking sites, I do not have many followers on GitHub. Because of this, I would think that the average degrees of separation would be pretty high for Github and that the maximum distance between two nodes would be pretty large as well. Previous to starting this project, I made the prediction that the maximum distance between two nodes would be greater than 6.

To complete this project I made functions to complete a BFS using an adjacency list. I then also created functions that analyzed the information from the BFS. These functions were separated into two modules, with one module having the search algorithms being done, and the other module having the analysis functions. I also had a third module which made an adjacency list.

For my BFS functions, first I made a function that does a BFS for every node. This returns a vector of vectors which has each node and then the distance between that node and every other node. Since my dataset is very large this function takes around 8 minutes to complete. Because of this, I added a print statement in the function that prints the node that the function is on, so the user can tell how far along the algorithm is. I also have a function that prints the BFS for every node. However, in my main file, I have this function commented out because it takes an extremely long time to print. Finally, I have a function called OneBFS which does a BFS for one selected node and prints it. I thought that this was a good function to have so a user could see the distance between a selected node and the other nodes in the dataset without having to do the BFS for all of the nodes. This function takes a very short time to run as well.

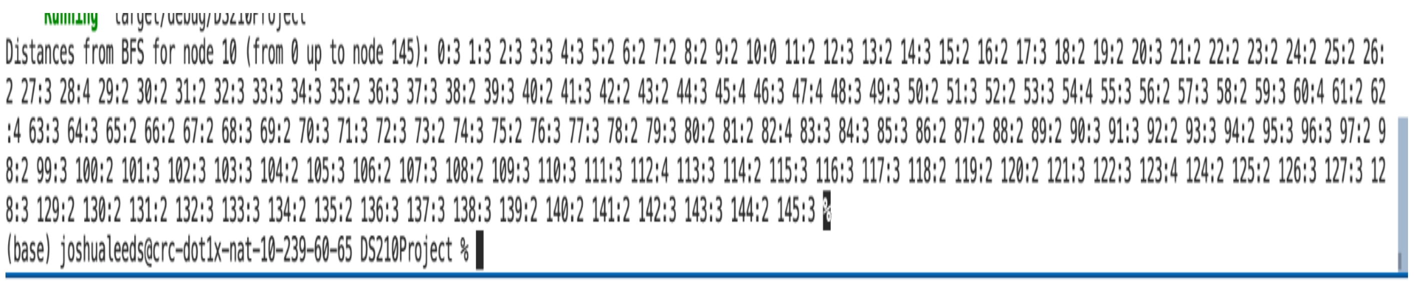
For the next part of the project, I made three functions that analyze the BFSs. First I have a function called average_distance. This function calculates the average distance between nodes in the graph. From this function, I got that the average degree of separation between nodes was 3.246. This was surprising to me because I thought that the average distance between nodes would be slightly higher. However, after running OneBFS for a couple of different nodes, I realized that this average made sense for my dataset.

Next, I had a function called furthest which calculated which nodes were the furthest degrees of separation away from each other and how many degrees of separation away from each other they are. From this function, the result was that 3 sets of nodes were 11 degrees of separation away from each other. I was extremely surprised by this output because although I thought the furthest distance would be greater than 6 I thought it still would be close to 6, 11 is very large. However, once looking into the nodes with 11 degrees of separation away from each other, I found that all of them only had one follower (edge) and that their follower only followed one other person. Because of this, it makes sense that they are 11 degrees away from each other.

Finally, I have a function called degree_distribution which gets the percentage of connections that occur in x degrees, with x being a selected number by the user. From this function, I ran a for loop in the main function with x being 1 through 6 and found that 99.7% of the connections are between 1 and 6, then when x was between 1 through 7 I found that almost 100% of the connection were between 1 and 7 degrees away. The most common degree of separation was 3 degrees with 53.25% of the connections.

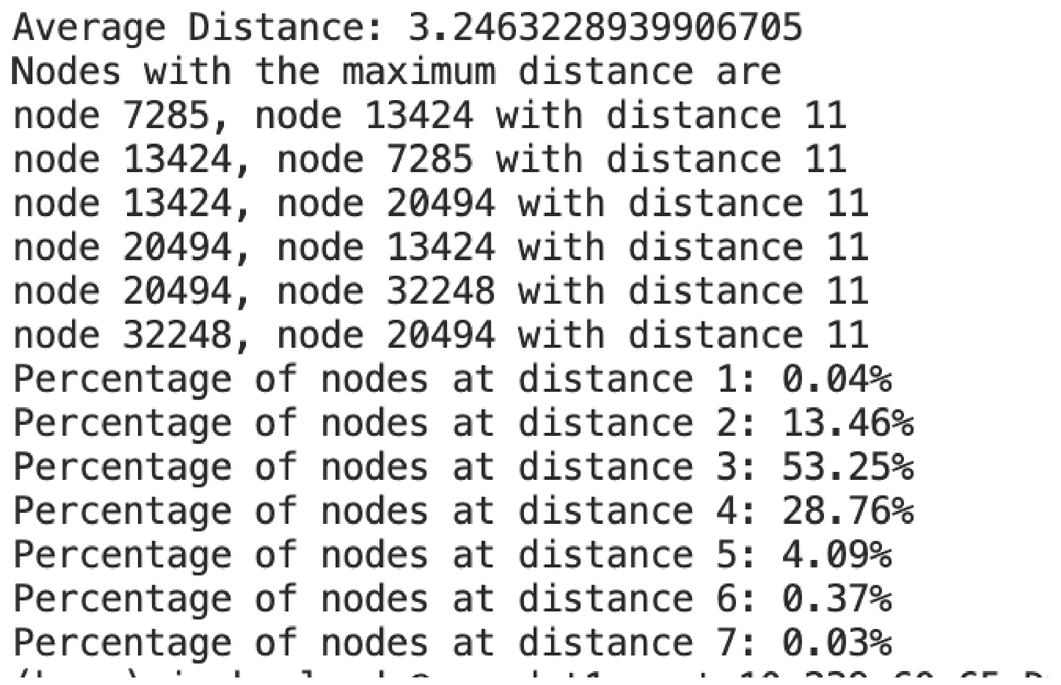
Code Output:

This is the output for OneBFS for node 10, and the final node is 145. This shows the degrees of separation for 10 to all numbers 0 - 145.



```
    Running target/debug/DS210Project
Distances from BFS for node 10 (from 0 up to node 145): 0:3 1:3 2:3 3:3 4:3 5:2 6:2 7:2 8:2 9:2 10:0 11:2 12:3 13:2 14:3 15:2 16:2 17:3 18:2 19:2 20:3 21:2 22:2 23:2 24:2 25:2 26:
2 27:3 28:4 29:2 30:2 31:2 32:3 33:3 34:3 35:2 36:3 37:3 38:2 39:3 40:2 41:3 42:2 43:2 44:3 45:4 46:3 47:4 48:3 49:3 50:2 51:3 52:2 53:3 54:4 55:3 56:2 57:3 58:2 59:3 60:4 61:2 62
:4 63:3 64:3 65:2 66:2 67:2 68:3 69:2 70:3 71:3 72:3 73:2 74:3 75:2 76:3 77:3 78:2 79:3 80:2 81:2 82:4 83:3 84:3 85:3 86:2 87:2 88:2 89:2 90:3 91:3 92:2 93:3 94:2 95:3 96:3 97:2 9
8:2 99:3 100:2 101:3 102:3 103:3 104:2 105:3 106:2 107:3 108:2 109:3 110:3 111:3 112:4 113:3 114:2 115:3 116:3 117:3 118:2 119:2 120:2 121:3 122:3 123:4 124:2 125:2 126:3 127:3 12
8:3 129:2 130:2 131:2 132:3 133:3 134:2 135:2 136:3 137:3 138:3 139:2 140:2 141:2 142:3 143:3 144:2 145:3 %
(base) joshualeeds@crc-dot1x-nat-10-239-60-65 DS210Project %
```

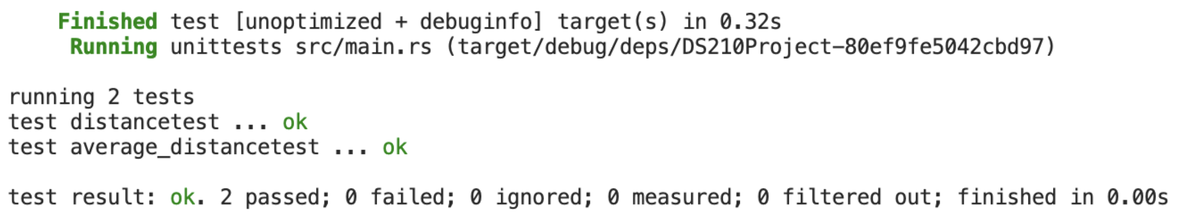
This is the output of the average distance, the maximum distance between nodes, and the percentage distribution of degree separation. This shows the average distance: 3.24, and the maximum distance between nodes of 11. This max distance occurred between node 7285 and node 13424, node 13424 and node 20494, and node 20494 and node 32248.



```
Average Distance: 3.2463228939906705
Nodes with the maximum distance are
node 7285, node 13424 with distance 11
node 13424, node 7285 with distance 11
node 13424, node 20494 with distance 11
node 20494, node 13424 with distance 11
node 20494, node 32248 with distance 11
node 32248, node 20494 with distance 11
Percentage of nodes at distance 1: 0.04%
Percentage of nodes at distance 2: 13.46%
Percentage of nodes at distance 3: 53.25%
Percentage of nodes at distance 4: 28.76%
Percentage of nodes at distance 5: 4.09%
Percentage of nodes at distance 6: 0.37%
Percentage of nodes at distance 7: 0.03%
```

Test Output:

I also made two tests that check the analysis functions average_distance and furthest. For these tests, I created a small graph data set (I got this test dataset from one of the lectures) and then calculated the average separation between nodes and the furthest separation between nodes by hand. Then I ran the functions on the test data set to see if my calculated number and the function output were the same. Both tests passed.



```
    Finished test [unoptimized + debuginfo] target(s) in 0.32s
     Running unittests src/main.rs (target/debug/deps/DS210Project-80ef9fe5042cbd97)

running 2 tests
test distancetest ... ok
test average_distancetest ... ok

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