DS210 Final Project Report

The data set I used is Social Circles: Facebook, which is collected by Stanford Network Analysis Project. I used the combined data, which includes 88234 edges connected by 4039 nodes. I plan to use this dataset to perform a degree distribution first and then test the six degrees of separation. Here are the steps and related code I implemented.

Read the file into a graph

I created a module called txt_to_graph and a Pub struct called Graph, which includes the edges, nodes and number of nodes, and a function that can turn the file into the struct Graph. Reading edges was straightforward to finish, while reading nodes took me a while. I first read nodes from edges and push them, but the problem is all nodes will be read twice. Therefore, I found two useful function in Rust to help me: sort() help me sort the nodes from an ascending order, so that the repeated value are next to each other, and dedup() helps remove consecutive repeated elements in the vector, which leave one unique value for each node. The number of nodes was the length of the vector stored node.

Find Neighbors

In either degree distribution or six degrees of separation, finding the neighbors of each node is necessary. Therefore, I wrote a function called find_neighbors in main.rs. This function will read edges and output a Hashmap, whose key is the node and value is an adjacency list of its neighbors. The output of this function is:

{2828: [1684, 2661, 2674, 2708, 2716, 2719, 2742, 2765, 2778, 2793, 2796, 2823, 2833, 2849, 2869, 2875, 2901, 2936, 2937, 2939, 2970, 2973, 2989, 2991, 2992, 3027, 3035, 3038, 3049, 3051, 3060, 3076, 3082, 3101, 3102, 3106, 3113, 3115, 3116, 3120, 3132, 3146, 3148, 3152, 3169, 3170, 3172, 3179, 3185, 3201, 3209, 3247, 3252, 3261, 3265, 3272, 3277, 3280, 3291, 3298, 3320, 3330, 3332, 3335, 3344, 3347, 3350, 3351, 3377, 3378, 3381, 3391, 3396, 3397, 3406, 3411, 3416, 3434],.....2083: [1912, 1917, 1918, 1925, 1929, 1938, 1943, 1946, 1962, 1966, 1983, 1984, 1985, 1986, 1989, 2005, 2020, 2030, 2033, 2037, 2045, 2046, 2055, 2059, 2060, 2064, 2069, 2073, 2074, 2078, 2084, 2088, 2090, 2093, 2095, 2103, 2104, 2108, 2112, 2115, 2118, 2122, 2123, 2124, 2131, 2142, 2147, 2150, 2164, 2165, 2172, 2184, 2188, 2190, 2200, 2201, 2206, 2212, 2218, 2220, 2229, 2233, 2240, 2244, 2257, 2261, 2266, 2271, 2275, 2276, 2278, 2290, 2299, 2300, 2307, 2323, 2324, 2331, 2340, 2347, 2348, 2352, 2353, 2354, 2356, 2359, 2363, 2370, 2374, 2381, 2386, 2395, 2404, 2408, 2409, 2410, 2414, 2418, 2428, 2433, 2446, 2464, 2467, 2469, 2477, 2478, 2492, 2495, 2500, 2504, 2506, 2507, 2520, 2539, 2542, 2543, 2549, 2550, 2551, 2553, 2556, 2559, 2560, 2561, 2564, 2573, 2574, 2575, 2579, 2586, 2590, 2593, 2600, 2601, 2602, 2604, 2607, 2611, 2613, 2615, 2619, 2624, 2630, 2631, 2638, 2654, 2655]}

This hashmap will substantially increase the efficiency of the graph analysis.

Degree Distribution

Then I created another module called degree_distribution. Using the HashMap created, I am able to count the numbers of neighbors each node has. I used a vector to output this where the first element is the number of neighbors and the second is its counts. I used a vector instead of a HashMap because HashMap cannot be sorted, which is not interpretable. Then we divided the counts by the number of nodes to showcase the distribution of degrees. We printed in main and the output is:

```
Degree Distribution:
Degree 1: 0.018568952711067097
Degree 2: 0.024263431542461005
Degree 3: 0.0230255013617232
Degree 4: 0.024511017578608567
.
.
.
Degree 235: 0.0002475860361475613
Degree 245: 0.0002475860361475613
Degree 254: 0.0002475860361475613
Degree 291: 0.0002475860361475613
Degree 294: 0.0002475860361475613
Degree 347: 0.0002475860361475613
Degree 547: 0.0002475860361475613
Degree 755: 0.0002475860361475613
Degree 792: 0.0002475860361475613
Degree 1045: 0.0002475860361475613
```

Except for some extremes, we can observe that the proportion of nodes is decreasing with the increase of degrees. Since Rust lacks a tool to visualize it, we cannot determine if this power–law distribution fits it well even though it did look like it. I did find some crates such as statrs::distribution that can perform hypothesis tests on different distributions but unfortunately, I didn't find the power-law one. But still, since we have the degree and frequency distribution, we can use the stats and apply them to other languages like R to evaluate them.

Six degrees of separation

Next, I created another model called six_degrees_of_separation.Using the adjacency list from find_neignbor function, I used Breadth First Search to find the distances between starting nodes and other nodes and output it as a HeshMap. This hashmap can help me count the number of nodes by distance. In main.rs, I can set the nodes I want to check in a for loop. For example, below is the output of nodes 1 to 10:

Numbers of nodes by distances from node 1
{4: 519, 5: 117, 3: 1742, 2: 1171, 1: 347, 6: 142, 0: 1}
Numbers of nodes by distances from node 2
{7: 142, 4: 1742, 6: 117, 2: 330, 5: 519, 0: 1, 3: 1171, 1: 17}
Numbers of nodes by distances from node 3
{2: 337, 0: 1, 5: 519, 7: 142, 3: 1171, 1: 10, 4: 1742, 6: 117}
Numbers of nodes by distances from node 4
{2: 330, 4: 1742, 6: 117, 1: 17, 0: 1, 5: 519, 7: 142, 3: 1171}
Numbers of nodes by distances from node 5
{4: 1742, 7: 142, 3: 1171, 5: 519, 1: 10, 6: 117, 0: 1, 2: 337}
Numbers of nodes by distances from node 6
{6: 117, 7: 142, 3: 1171, 1: 13, 5: 519, 4: 1742, 2: 334, 0: 1}
Numbers of nodes by distances from node 7
{5: 519, 6: 117, 0: 1, 3: 1171, 2: 341, 7: 142, 1: 6, 4: 1742}
Numbers of nodes by distances from node 8
{1: 20, 3: 1699, 5: 519, 4: 1103, 2: 438, 6: 117, 0: 1, 7: 142}
Numbers of nodes by distances from node 9
{3: 1171, 2: 339, 4: 1742, 7: 142, 6: 117, 1: 8, 5: 519, 0: 1}
Numbers of nodes by distances from node 10
{7: 142, 2: 290, 0: 1, 4: 1742, 6: 117, 3: 1171, 5: 519, 1: 57}

As the output shows, we can see that most of the nodes are 2 to 4 distances away from other nodes. Although the six degrees of separation states that all people have six or fewer social connections with others, the output shows that there's some nodes that are 7 distances away from each other. Furthermore, when I tried to run every node, some of them even have nodes 8 distances away. For example:

Numbers of nodes by distances from node 4036
{2: 58, 8: 142, 4: 263, 5: 1853, 7: 64, 3: 4, 6: 1653, 1: 1, 0: 1}

Therefore, we cannot be convinced that this graph follows six degrees of separation, but it did make sense. This dataset was recorded by Stanford University in 2011, when users were only around 1.23 Billion. I am pretty sure the concept of six degrees of separation will work much better on Today's social media because people are fully connected to each other, like in the real world.

Finally, I added some tests at the end of main.rs to test if every function works well(except the one reading the file). To conclude, I did face some challenges when running the code because it is really difficult to notice every reference/deference, type and more stuff on the first try, so you need to run and revise them over and over again. Fortunately, both degree distribution and six degree of separation analysis both work well on this dataset ultimately.

Final output(I skipped some parts)

Degree Distribution:
Degree 1: 0.018568952711067097
Degree 2: 0.024263431542461005
Degree 3: 0.0230255013617232
.
.
.
Degree 547: 0.0002475860361475613
Degree 755: 0.0002475860361475613
Degree 792: 0.0002475860361475613
Degree 1045: 0.0002475860361475613
==========
Six Degrees of Separation:
Numbers of nodes by distances from node 1
{2: 1171, 3: 1742, 4: 519, 1: 347, 0: 1, 5: 117, 6: 142}
Numbers of nodes by distances from node 2
{0: 1, 5: 519, 1: 17, 3: 1171, 4: 1742, 2: 330, 7: 142, 6: 117}
Numbers of nodes by distances from node 3
{4: 1742, 5: 519, 0: 1, 1: 10, 6: 117, 7: 142, 3: 1171, 2: 337}
Numbers of nodes by distances from node 4
{2: 330, 5: 519, 1: 17, 4: 1742, 6: 117, 3: 1171, 7: 142, 0: 1}
Numbers of nodes by distances from node 5
{6: 117, 2: 337, 3: 1171, 5: 519, 4: 1742, 1: 10, 0: 1, 7: 142}
Numbers of nodes by distances from node 6
{1: 13, 5: 519, 0: 1, 4: 1742, 6: 117, 2: 334, 3: 1171, 7: 142}
Numbers of nodes by distances from node 7
{5: 519, 6: 117, 3: 1171, 7: 142, 0: 1, 2: 341, 1: 6, 4: 1742}
Numbers of nodes by distances from node 8
{2: 438, 3: 1699, 1: 20, 6: 117, 4: 1103, 5: 519, 7: 142, 0: 1}
Numbers of nodes by distances from node 9
{2: 339, 3: 1171, 0: 1, 7: 142, 1: 8, 5: 519, 6: 117, 4: 1742}
Numbers of nodes by distances from node 10
{0: 1, 4: 1742, 6: 117, 2: 290, 3: 1171, 1: 57, 5: 519, 7: 142}
.
.
.
Numbers of nodes by distances from node 4029
{5: 1853, 8: 142, 6: 1653, 7: 64, 2: 57, 1: 2, 0: 1, 3: 4, 4: 263}
Numbers of nodes by distances from node 4030
{7: 64, 8: 142, 4: 263, 3: 4, 0: 1, 6: 1653, 2: 57, 5: 1853, 1: 2}
Numbers of nodes by distances from node 4031
{3: 4, 5: 1853, 7: 64, 6: 1653, 4: 263, 8: 142, 1: 19, 0: 1, 2: 40}
Numbers of nodes by distances from node 4032
{1: 11, 5: 1653, 0: 1, 3: 263, 6: 64, 2: 52, 4: 1853, 7: 142}
Numbers of nodes by distances from node 4033
{2: 57, 3: 4, 0: 1, 5: 1853, 1: 2, 4: 263, 7: 64, 6: 1653, 8: 142}
Numbers of nodes by distances from node 4034
{6: 1653, 2: 56, 0: 1, 8: 142, 7: 64, 5: 1853, 4: 263, 3: 4, 1: 3}
Numbers of nodes by distances from node 4035
{0: 1, 5: 1853, 8: 142, 2: 57, 1: 2, 6: 1653, 7: 64, 3: 4, 4: 263}
Numbers of nodes by distances from node 4036
{3: 4, 4: 263, 0: 1, 7: 64, 2: 58, 5: 1853, 8: 142, 1: 1, 6: 1653}
Numbers of nodes by distances from node 4037
{4: 263, 8: 142, 5: 1853, 7: 64, 3: 4, 2: 57, 6: 1653, 1: 2, 0: 1}
Numbers of nodes by distances from node 4038
{7: 64, 5: 1853, 3: 4, 0: 1, 6: 1653, 8: 142, 2: 55, 1: 4, 4: 263}
Numbers of nodes by distances from node 4039
{4: 263, 6: 1653, 5: 1853, 1: 9, 7: 64, 0: 1, 2: 50, 8: 142, 3: 4}

Original Dataset

https://snap.stanford.edu/data/ego-Facebook.html

Reference:

Crate Statrs: https://docs.rs/statrs/latest/statrs/distribution/struct.Normal.html

Six_degrees of separation: https://en.wikipedia.org/wiki/Six_degrees_of_separation

Facebook Users over Year:

https://www.oberlo.com/statistics/how-many-users-does-facebook-have#:~:text=From%202018%20to%202022%2C%20however,annual%20growth%20rate%20of%2010.9%25.