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

The purpose of the project is to analyze the clustering properties of the dataset using triangle counts to measure the tendency of nodes to form tightly-knit groups/ the local clustering coefficient. I'm interested in seeing how closely connected groups of people (triangles) form in different social networks, hence analyzing the Facebook data set.

This data set is formatted specifically for graph analysis, with nodes representing users and edges representing relationships (friendships), making them directly applicable to triangle counting and clustering coefficient computation in Rust. The dataset is also large enough to work for a meaningful analysis of real-world social networks. I'm particularly interested in Facebook because I read about small-world phenomena on large platforms (people are only a few links away from each other despite social and geographic backgrounds).

Graph.rs

I used HashMap and HashSet to store the graph and perform efficient lookups for neighbors and edges. For reading the file to load graph data I used BufReader and File.
The struct graph is the main data structure representing the graph, and adjacency_list is a hashmap where the key is a node, and the value is a vector of the node's neighbor.

For graph implementation, I created the function 'new' to initialize an empty graph with an empty HashMap. The function add_edge adds an undirected edge between nodes u and v. I used entry to ensure a vector exists for each node (or_insert_with(Vec::new) inserts a new empty vector if the node is not already in the HashMap.) '.push' appends v to the neighbors of u and appends u to the neighbors of v. The function load_graph_from_file reads a graph description from a file. It splits the line into u and v (edges) and calls add_edge to store the edge in the graph.

The function for degree calculation (degree) returns the number of neighbors (degree) of a given node using get, and if the node is not in the graph, it returns 0 (.map_or(0, |neighbors| neighbors.len()))
The function neighbors returns a reference to the vector of neighbors for a given node or returns None if the node does not exist. The fmt function iterates through the HashMap and prints each node and its neighbors from adjacency list.

Using HashMap and Vec ensures efficient lookups and traversal.

Triangle_count.rs

The function count_triangles_for_node counts the number of triangles a specific node is part of. It retrieves the node's neighbors from the graph's adjacency list (using match self.neighbors(node)) and then

checks every pair of these neighbors to determine if they are also connected to each other using graph.is_connected. The triangle count increments if they are connected (share an edge). The triangle count is divided by 3 to give the correct total number of distinct triangles, correcting the triple counting.

Then, the count_all_triangles function iterates over every node in the graph, summing up their individual triangle counts.

Clustering.rs

The clustering module calculates how tightly each node is connected to its neighbors (local clustering coefficient) and then averages the measure for the entire graph (Average clustering coefficient). The function compute_local_clustering_coefficient function retrieves all of the neighbors of the given node using the neighbors() method of the Graph struct. If the node has fewer than two neighbors, it returns 0.0 because no triangles can form. If not, it counts how many of the node's neighbors are also connected to each other, indicating the presence of triangles. Since each triangle is counted once per node involved, the total is divided by three. This number is then compared to the maximum possible triangles that could form between the node's neighbors, which gives the node's local clustering coefficient (a value between 0 and 1)

Then, the compute_average_clustering_coefficient function computes this local measure for every node by iterating over all nodes in the graph and calling the function compute_local_clustering_coefficient for each node. It initializes a running total and counts how many nodes have been examined/processed. For each node retrieved from self.nodes(), the function adds the node's local clustering coefficient (compute_local_clustering_coefficient(node)) to the running total, and the node count is increment by 1. After each node is visited, the function divides the total local clustering coefficient by the total number of nodes, which gives the mean of the results. The average indicates how clustered the network is overall.

Test.rs

The tests verify that the clustering coefficient values computed by the program are valid.

How the tests work: A new graph is created. The test selects a particular node (node 1) and computes the local clustering coefficient graph.compute_local_clustering_coefficient(node).
The test then uses an assertion assert!(coefficient >= 0.0 && coefficient <= 1.0, "Coefficient out of range") to check that the coefficient is between 0 and 1 since as we know a clustering coefficient represents a probability-esque measure of how connected a node's neighbors are. If the value lies outside this range, it means that there is a logical error in the code.

The other test calls graph.compute_average_clustering_coefficient(), and an assertion again checks whether the average coefficient falls within the range of 0 to 1, if it doesn't, the test fails, as something is incorrect in either the averaging logic or the calculations of individual node coefficients.

Test output:

```
[1001910000@working-section-0 DS210final]$ cargo test
   Compiling DS210final v0.1.0 (/opt/app-root/src/DS210final)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 3.24s
     Running unittests src/main.rs (target/debug/deps/DS210final-a64287395a722fec)

running 2 tests
test test::tests::test_average_clustering_coefficient ... ok
test test::tests::test_local_clustering_coefficient ... ok

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

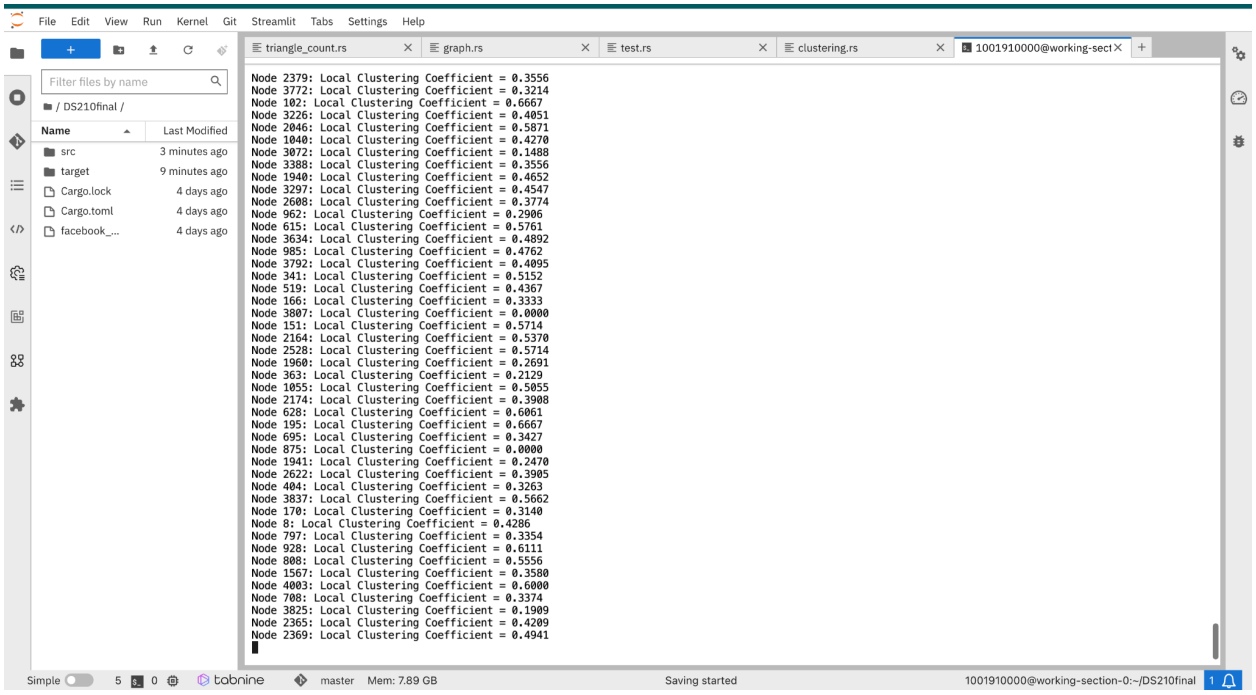
Cargo run output

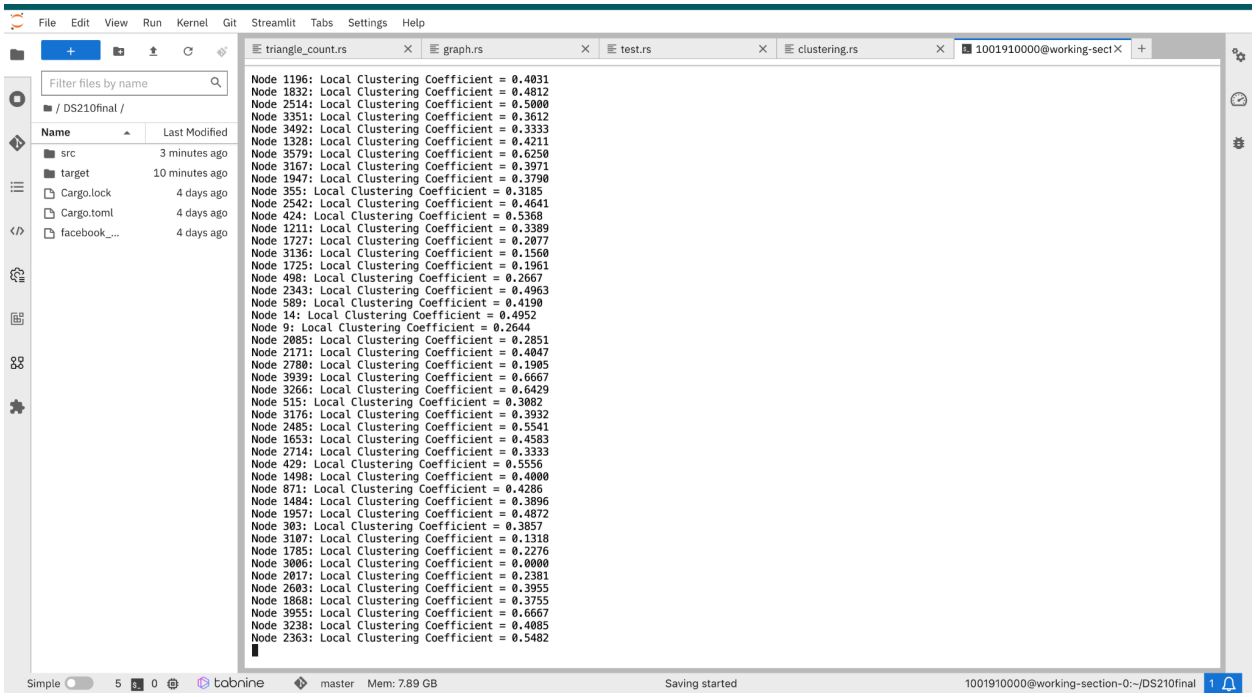
Total triangles in the graph: 3222693
Average (Global) Clustering Coefficient: 0.3825
(Local clustering screenshots at the end of the document)

The total triangle count is high, given the network's size (4039 nodes and 88234). This suggests that many triples of people form fully connected triads, as in if you pick a random user and look at their friends, there is a high chance that those friends are also connected to each other. I can infer from this result that Facebook has closely knit groups rather than random links.

The average clustering coefficient of 0.3825 quantifies this conclusion as a value of 1.0 would mean that, on average, every pair of a user's friends are also friends with each other (a perfect cluster.) Hence, 0.38 indicates that while not all the friends of a friend are connected, the network is significantly more clustered than a random graph. This goes with the fact that friends tend to share mutual acquaintances on media, creating small-world effects. The metrics I found from my project reinforce the idea that social networks,, like Facebook's, evolve into groups of closely connected users rather than loosely or randomly connected users. In the future, I would love to see how the clustering and triangles in Facebook compare to Instagram and Twitter, where you can follow someone who doesn't have to follow you back.

Some screenshots of the local clustering coefficient list (not all results):


File Edit View Run Kernel Git Streamlit Tabs Settings Help
triangle_count.rs
graph.rs
test.rs
clustering.rs
1001910000@working-sect
Filter files by name
/ DS210final /
Name
Last Modified
src 3 minutes ago
target 9 minutes ago
Cargo.lock 4 days ago
Cargo.toml 4 days ago
facebook_... 4 days ago
Node 2379: Local Clustering Coefficient = 0.3556
Node 3772: Local Clustering Coefficient = 0.3214
Node 102: Local Clustering Coefficient = 0.6667
Node 3226: Local Clustering Coefficient = 0.4051
Node 2046: Local Clustering Coefficient = 0.5871
Node 1040: Local Clustering Coefficient = 0.4270
Node 3072: Local Clustering Coefficient = 0.1488
Node 3388: Local Clustering Coefficient = 0.3556
Node 1940: Local Clustering Coefficient = 0.4652
Node 3297: Local Clustering Coefficient = 0.4547
Node 2608: Local Clustering Coefficient = 0.3774
Node 962: Local Clustering Coefficient = 0.2906
Node 615: Local Clustering Coefficient = 0.5761
Node 3634: Local Clustering Coefficient = 0.4892
Node 985: Local Clustering Coefficient = 0.4762
Node 3792: Local Clustering Coefficient = 0.4095
Node 341: Local Clustering Coefficient = 0.5152
Node 519: Local Clustering Coefficient = 0.4367
Node 166: Local Clustering Coefficient = 0.3333
Node 3807: Local Clustering Coefficient = 0.0000
Node 151: Local Clustering Coefficient = 0.5714
Node 2164: Local Clustering Coefficient = 0.5370
Node 2528: Local Clustering Coefficient = 0.5714
Node 1960: Local Clustering Coefficient = 0.2691
Node 363: Local Clustering Coefficient = 0.2129
Node 1055: Local Clustering Coefficient = 0.5055
Node 2174: Local Clustering Coefficient = 0.3908
Node 628: Local Clustering Coefficient = 0.6061
Node 195: Local Clustering Coefficient = 0.6667
Node 695: Local Clustering Coefficient = 0.3427
Node 875: Local Clustering Coefficient = 0.0000
Node 1941: Local Clustering Coefficient = 0.2470
Node 2622: Local Clustering Coefficient = 0.3905
Node 404: Local Clustering Coefficient = 0.3263
Node 3837: Local Clustering Coefficient = 0.5662
Node 170: Local Clustering Coefficient = 0.3140
Node 8: Local Clustering Coefficient = 0.4286
Node 797: Local Clustering Coefficient = 0.3354
Node 928: Local Clustering Coefficient = 0.6111
Node 808: Local Clustering Coefficient = 0.5556
Node 1567: Local Clustering Coefficient = 0.3580
Node 4003: Local Clustering Coefficient = 0.6000
Node 708: Local Clustering Coefficient = 0.3374
Node 3825: Local Clustering Coefficient = 0.1909
Node 2365: Local Clustering Coefficient = 0.4209
Node 2369: Local Clustering Coefficient = 0.4941
Simple 5 0 tabnine master Mem: 7.89 GB Saving started 1001910000@working-section-0:~/DS210final 1


File Edit View Run Kernel Git Streamlit Tabs Settings Help
triangle_count.rs
graph.rs
test.rs
clustering.rs
1001910000@working-sect
Filter files by name
/ DS210final /
Name
Last Modified
src 3 minutes ago
target 10 minutes ago
Cargo.lock 4 days ago
Cargo.toml 4 days ago
facebook_... 4 days ago
Node 1196: Local Clustering Coefficient = 0.4031
Node 1832: Local Clustering Coefficient = 0.4812
Node 2514: Local Clustering Coefficient = 0.5000
Node 3351: Local Clustering Coefficient = 0.3612
Node 3492: Local Clustering Coefficient = 0.3333
Node 1328: Local Clustering Coefficient = 0.4211
Node 3579: Local Clustering Coefficient = 0.6250
Node 3167: Local Clustering Coefficient = 0.3971
Node 1947: Local Clustering Coefficient = 0.3790
Node 355: Local Clustering Coefficient = 0.3185
Node 2542: Local Clustering Coefficient = 0.4641
Node 424: Local Clustering Coefficient = 0.5368
Node 1211: Local Clustering Coefficient = 0.3389
Node 1727: Local Clustering Coefficient = 0.2077
Node 3136: Local Clustering Coefficient = 0.1560
Node 1725: Local Clustering Coefficient = 0.1961
Node 498: Local Clustering Coefficient = 0.2667
Node 2343: Local Clustering Coefficient = 0.4963
Node 589: Local Clustering Coefficient = 0.4190
Node 14: Local Clustering Coefficient = 0.4952
Node 9: Local Clustering Coefficient = 0.2644
Node 2085: Local Clustering Coefficient = 0.2851
Node 2171: Local Clustering Coefficient = 0.4047
Node 2780: Local Clustering Coefficient = 0.1905
Node 3939: Local Clustering Coefficient = 0.6667
Node 3266: Local Clustering Coefficient = 0.6429
Node 515: Local Clustering Coefficient = 0.3082
Node 3176: Local Clustering Coefficient = 0.3932
Node 2485: Local Clustering Coefficient = 0.5541
Node 1653: Local Clustering Coefficient = 0.4583
Node 2714: Local Clustering Coefficient = 0.3333
Node 429: Local Clustering Coefficient = 0.5556
Node 1498: Local Clustering Coefficient = 0.4000
Node 871: Local Clustering Coefficient = 0.4286
Node 1484: Local Clustering Coefficient = 0.3896
Node 1957: Local Clustering Coefficient = 0.4872
Node 303: Local Clustering Coefficient = 0.3857
Node 3107: Local Clustering Coefficient = 0.1318
Node 1785: Local Clustering Coefficient = 0.2276
Node 3006: Local Clustering Coefficient = 0.0000
Node 2017: Local Clustering Coefficient = 0.2381
Node 2603: Local Clustering Coefficient = 0.3955
Node 1868: Local Clustering Coefficient = 0.3755
Node 3955: Local Clustering Coefficient = 0.6667
Node 3238: Local Clustering Coefficient = 0.4085
Node 2363: Local Clustering Coefficient = 0.5482
Simple 5 0 tabnine master Mem: 7.89 GB Saving started 1001910000@working-section-0:~/DS210final 1