Bader Alhodithi

DS210

Professor Leonidas Kontothanassis

Dataset: [Google Web Graph | Kaggle](https://www.kaggle.com/datasets/pappukrjha/google-web-graph)

https://www.kaggle.com/datasets/pappukrjha/google-web-graph

For my project, I decided to use the google dataset, which has nodes and edges, with the nodes representing web pages, along with directed edges showing the hyperlinks between these nodes. In this project, I used Rust to implement a visual graph, along with an analysis of such visuals. The code first reads the dataset, constructs a directed graph using a hashmap, and then finally performs different analysis methods on the graph.

To begin with, the code starts by reading about 25% of the dataset, which is still a lot, at about 218928 nodes, I did this as the program runtime was extremely long, the 25% may be adjusted of course depending on the computer power, but I found it optimal for my use, and the vertices still satisfy the minimum requirements. Moving on, the hashmap generates a graph with nodes as the keys and a list of neighbors as its values. I used a parser to extract two nodes, the source along with its destination, and added them in the graph. The graph is generated using plotters, then created as an SVG file in the directory, which can be opened in the browser, although due to the sheer amount of data, the graph itself will look cluttered, hence the need for analysis. (The folder should have the graph on it, running the code also gives the graph anyways)

In the analysis, I first made it such that it generates and outlines the number of vertices and edges, the average degree (number of connections), and degree distribution. Moreover, I also implemented a degree centrality measure, showing the “importance” of each node.

I implemented some helper functions, such as the fn(dfs), which is for the depth-first search, and another function to find the degree of the node. These functions are used as an analysis tool for the most part.

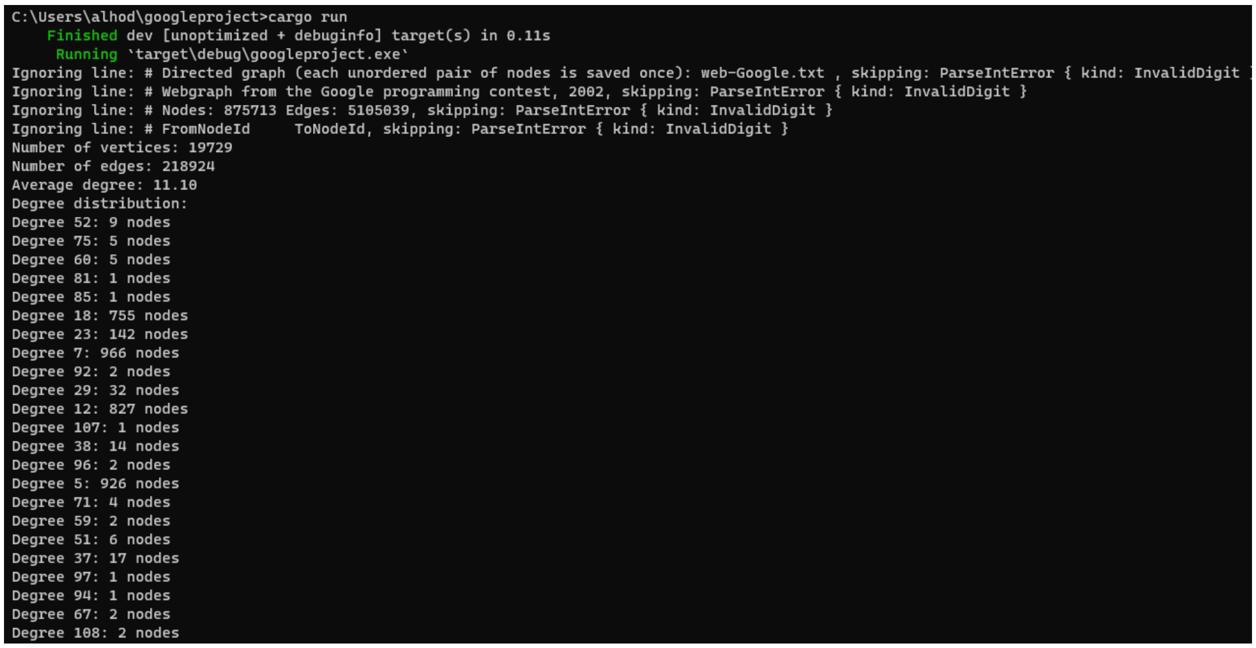
The project also has a separate test.rs file that uses a few functions to test whether our functions in the main.rs work. I will show an image of passed tests below with outputs.

Some drawbacks I had was that, other than the constant errors, I had a few ideas that could not be implemented properly, firstly, I tried to have the centrality measures as a separate table using the prettytable library, but for some reason the version i was using clashed with plotters i think and kept giving me an access violation error, hence why in my code I only display the first 50 values for centrality measure.

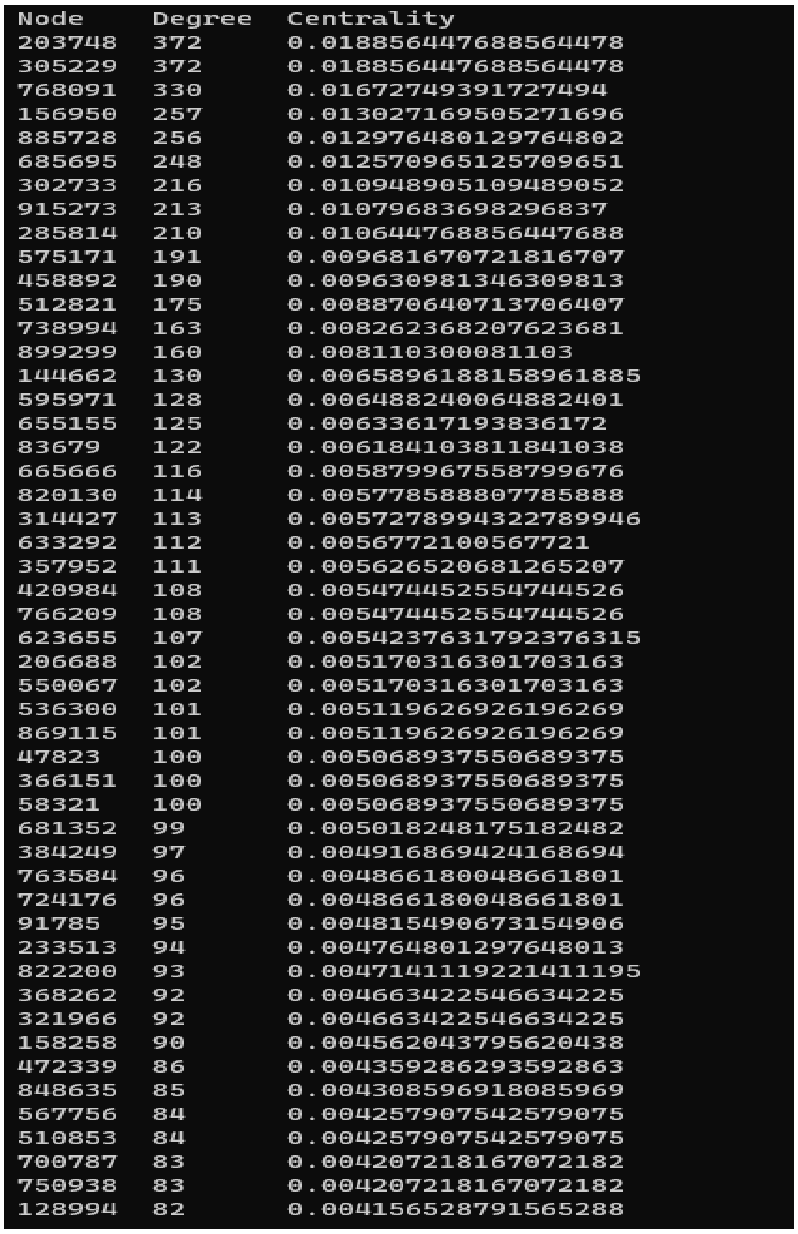
RUNNING CODE: I imported the dataset in my code with the assumption that it is in the same directory as the main.rs, meaning, dataset should be placed in the src directory.

Outputs:

Keep in mind, I am taking screenshots of the output, as it is long, I will only have screenshots of the beginning, where it shows a good amount of nodes and information, and the end with the centrality measure.

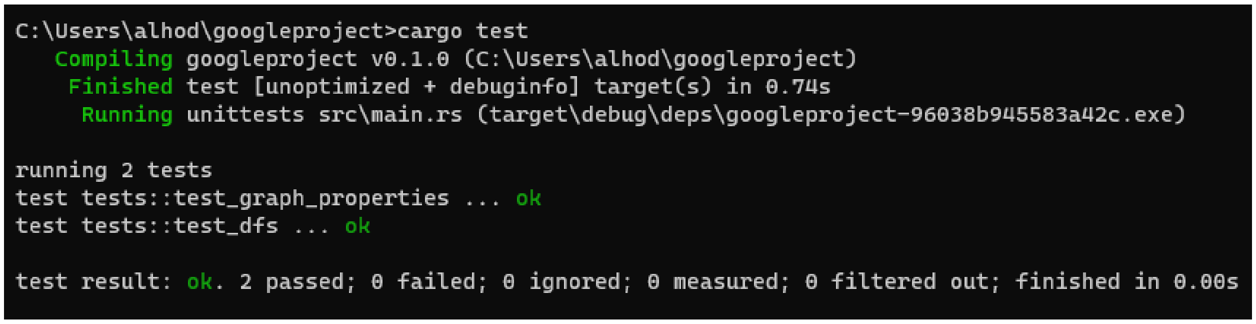


```
C:\Users\alhod\googleproject>cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.11s
     Running `target\debug\googleproject.exe`
Ignoring line: # Directed graph (each unordered pair of nodes is saved once): web-Google.txt , skipping: ParseIntError { kind: InvalidDigit
Ignoring line: # Webgraph from the Google programming contest, 2002, skipping: ParseIntError { kind: InvalidDigit }
Ignoring line: # Nodes: 875713 Edges: 5105039, skipping: ParseIntError { kind: InvalidDigit }
Ignoring line: # FromNodeId     ToNodeId, skipping: ParseIntError { kind: InvalidDigit }
Number of vertices: 19729
Number of edges: 218924
Average degree: 11.10
Degree distribution:
Degree 52: 9 nodes
Degree 75: 5 nodes
Degree 60: 5 nodes
Degree 81: 1 nodes
Degree 85: 1 nodes
Degree 18: 755 nodes
Degree 23: 142 nodes
Degree 7: 966 nodes
Degree 92: 2 nodes
Degree 29: 32 nodes
Degree 12: 827 nodes
Degree 107: 1 nodes
Degree 38: 14 nodes
Degree 96: 2 nodes
Degree 5: 926 nodes
Degree 71: 4 nodes
Degree 59: 2 nodes
Degree 51: 6 nodes
Degree 37: 17 nodes
Degree 97: 1 nodes
Degree 94: 1 nodes
Degree 67: 2 nodes
Degree 108: 2 nodes
```



```
Node     Degree   Centrality
203748   372      0.018856447688564478
305229   372      0.018856447688564478
768091   330      0.01672749391727494
156950   257      0.013027169505271696
885728   256      0.012976480129764802
685695   248      0.012570965125709651
302733   216      0.010948905109489052
915273   213      0.01079683698296837
285814   210      0.010644768856447688
575171   191      0.009681670721816707
458892   190      0.009630981346309813
512821   175      0.008870640713706407
738994   163      0.008262368207623681
899299   160      0.008110300081103
144662   130      0.0065896188158961885
595971   128      0.006488240064882401
655155   125      0.00633617193836172
83679    122      0.006184103811841038
665666   116      0.005879967558799676
820130   114      0.005778588807785888
314427   113      0.0057278994322789946
633292   112      0.0056772100567721
357952   111      0.005626520681265207
420984   108      0.005474452554744526
766209   108      0.005474452554744526
623655   107      0.0054237631792376315
206688   102      0.005170316301703163
550067   102      0.005170316301703163
536300   101      0.005119626926196269
869115   101      0.005119626926196269
47823    100      0.005068937550689375
366151   100      0.005068937550689375
58321    100      0.005068937550689375
681352   99       0.005018248175182482
384249   97       0.004916869424168694
763584   96       0.004866180048661801
724176   96       0.004866180048661801
91785    95       0.004815490673154906
233513   94       0.004764801297648013
822200   93       0.0047141119221411195
368262   92       0.004663422546634225
321966   92       0.004663422546634225
158258   90       0.004562043795620438
472339   86       0.004359286293592863
848635   85       0.004308596918085969
567756   84       0.004257907542579075
510853   84       0.004257907542579075
700787   83       0.004207218167072182
750938   83       0.004207218167072182
128994   82       0.004156528791565288
```

Test.rs output.



```
C:\Users\alhod\googleproject>cargo test
   Compiling googleproject v0.1.0 (C:\Users\alhod\googleproject)
    Finished test [unoptimized + debuginfo] target(s) in 0.74s
     Running unittests src\main.rs (target\debug\deps\googleproject-96038b945583a42c.exe)

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

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