Project Report: Wikispeedia

DATA

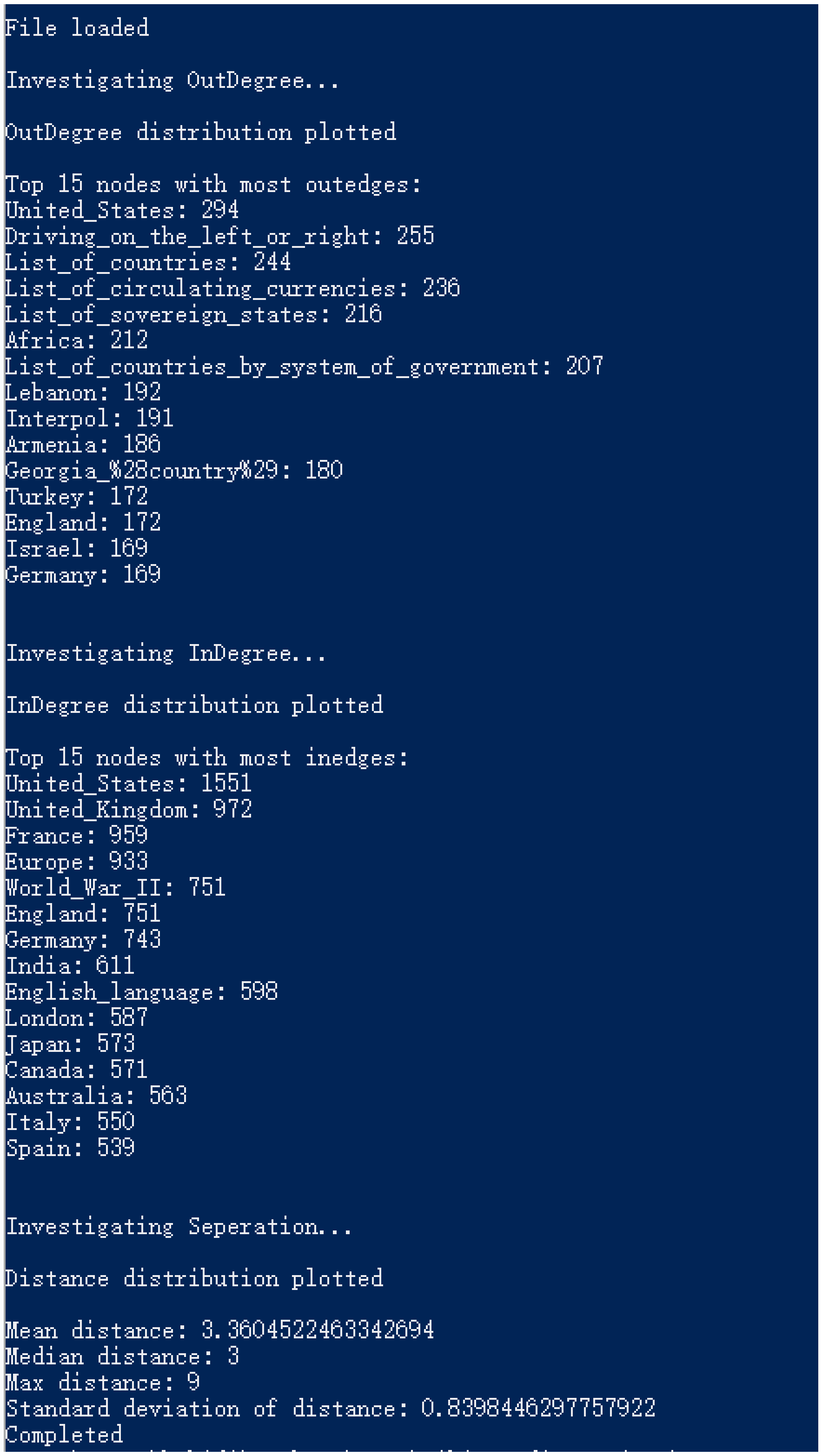
Wikispeedia is a game by The Data Science Lab in the School of Computer and Communication Sciences at EPFL. It's based on the 2007 Wikipedia Selection for schools, which contains around 4,600 articles. The goal of the game is to reach a random wikipedia article from another using embedded links in the articles, with as few redirections as possible. This project explores the Wikispeedia navigation paths dataset that is available on the Stanford Large Network Dataset Collection. The data can be downloaded from : https://snap.stanford.edu/data/wikispeedia.html. ("wikispeedia_paths-and-graph.tar.gz")
This project only uses "links.tsv" , which can be downloaded with the url above. "links.tsv" contains all links between the 2007 Wikipedia Selection for schools articles. Each row has a starting node and a target node. The "link.tsv" is already contained in the project folder.
This is a sample line: `City	Toronto`

USE

"wikispeedia_project" in the depository is the folder containing the entire project(sized 3MB). After downloading it, you can run `cargo test` and `cargo run` right away. This project tries to accomplish two questions provided in the default project prompts: Six degrees of separation and Degree distributions.
It will generate 3 plots as png under the root menu:

- Distribution of shortest distance between all nC2 pairs of nodes
- Distribution of incoming vertex degrees
- Distribution of outgoing vertex degrees.

It also prints the nodes with highest degrees, and statistics on the distribution of distance. This is a sample terminal output:



```
File loaded

Investigating OutDegree...

OutDegree distribution plotted

Top 15 nodes with most outedges:
United_States: 294
Driving_on_the_left_or_right: 255
List_of_countries: 244
List_of_circulating_currencies: 236
List_of_sovereign_states: 216
Africa: 212
List_of_countries_by_system_of_government: 207
Lebanon: 192
Interpol: 191
Armenia: 186
Georgia_%28country%29: 180
Turkey: 172
England: 172
Israel: 169
Germany: 169


Investigating InDegree...

InDegree distribution plotted

Top 15 nodes with most inedges:
United_States: 1551
United_Kingdom: 972
France: 959
Europe: 933
World_War_II: 751
England: 751
Germany: 743
India: 611
English_language: 598
London: 587
Japan: 573
Canada: 571
Australia: 563
Italy: 550
Spain: 539


Investigating Seperation...

Distance distribution plotted

Mean distance: 3.3604522463342694
Median distance: 3
Max distance: 9
Standard deviation of distance: 0.8398446297757922
Completed
```

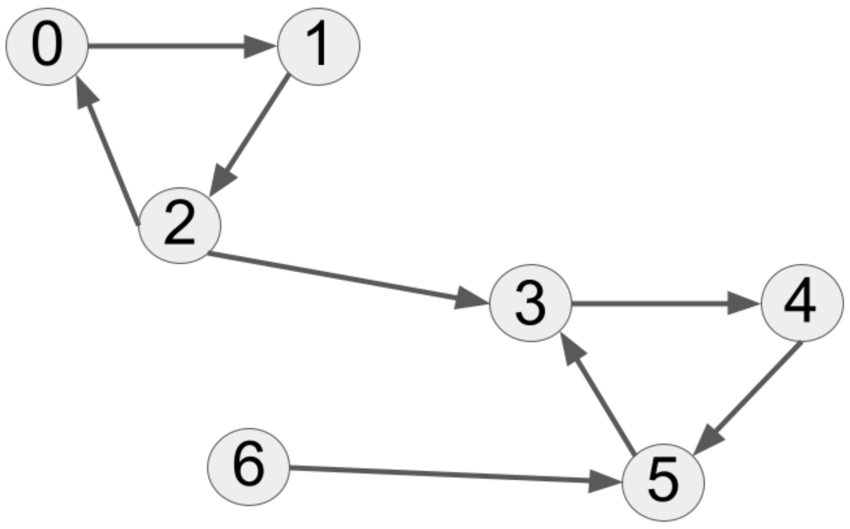
CODE

The project is separated into 4 .rs files. It has one dependency: plotters.

- reader.rs : Parse the data file and store it as a graph. The graph struct is also defined here. The Graph struct borrows from the lecture notes, and stores the graph in form of adjacency lists. The reader uses a hash set to keep track of unique nodes(wikipedia article). The ith unique node is stored in the graph as i, and a hashmap keeps track of which index corresponds to which article. It returns the hashmap, the graph, and also the reversed graph.
- distribution.rs : Responsible for distribution of vertex degrees. It uses a hash map whose key is the number of edges a node has, and value is the number of such nodes. It uses the crate plotters to plot the distribution of vertex degrees. It also contains a function that prints the nodes with most edges. The two functions deal with Incoming vertex degrees or outgoing vertex degrees, depending on the input graph.
- seperation.rs : Responsible for degrees of separations. It contains a BFS function (borrowed from lecture notes). The distance_distribution function iterates all nC2 pairs of nodes in the graph to calculate the shortest distance between each pair of nodes(if reachable), and stores the distance in a vector. Then we can plot the distribution of the distance and get statistics such as mean and standard deviation.
- main.rs : calls functions from modules and print results.

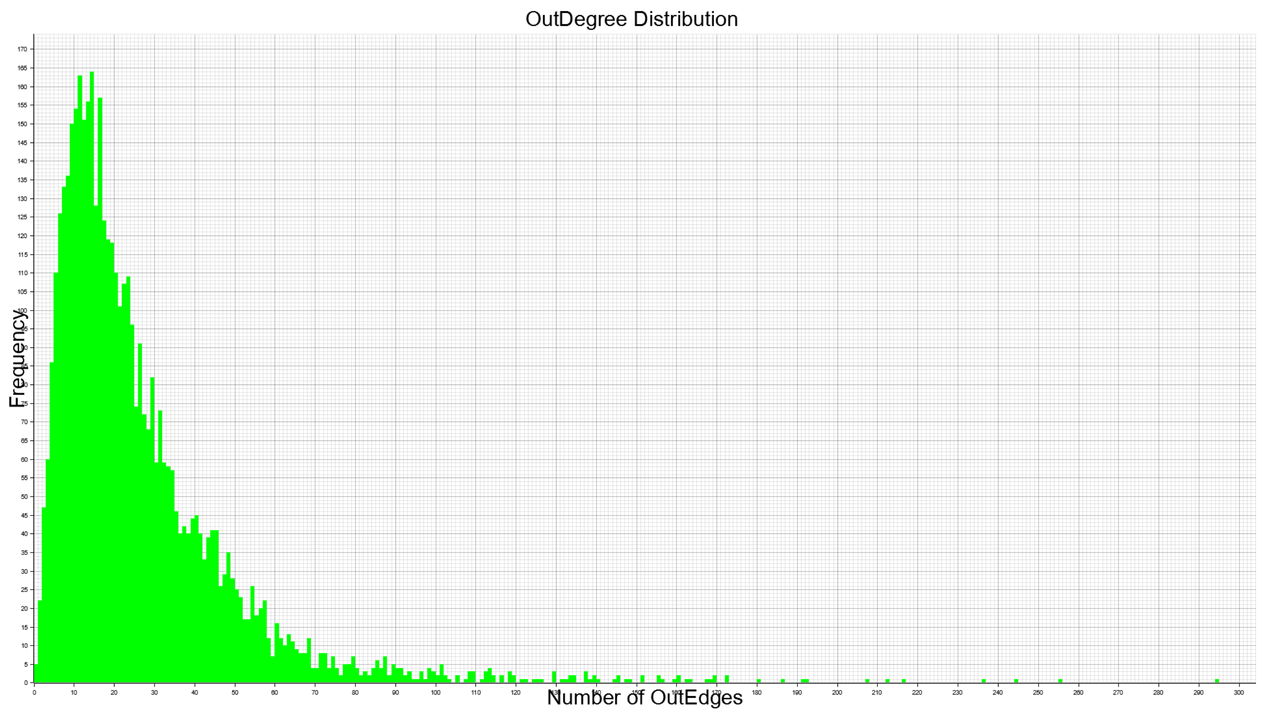
TESTS

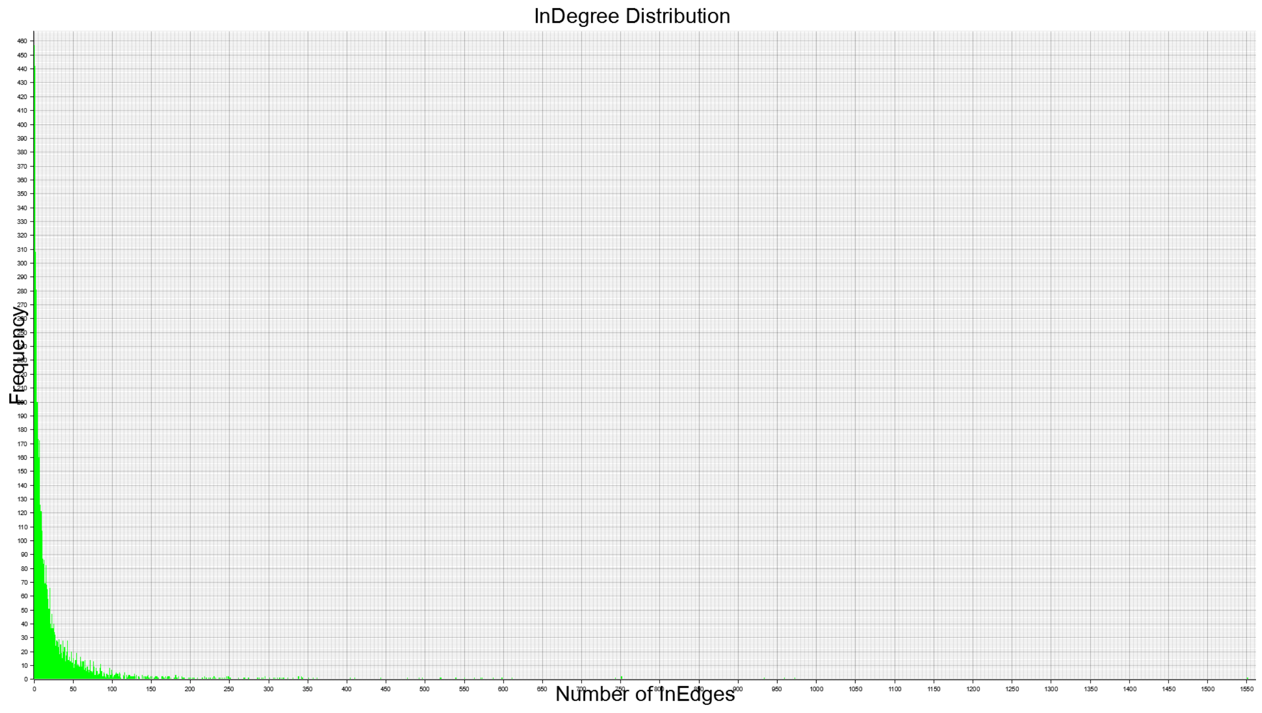
This project has 4 test functions to test different methods: reading data, analyzing degrees of vertex, and analyzing distance distribution. It uses testdata.tsv(included in the project folder), which comes from this graph from lecture 28:



The test compares my result with ground truth, I manually checked the distribution plots - they are correct.

RESULTS





The vertex degree distributions show a power-law distribution, especially the Incoming degree, which fits the general assumption for social networks. I believe this wikipedia dataset has a similar nature as a social network - A very small number of people have a very large number of links.

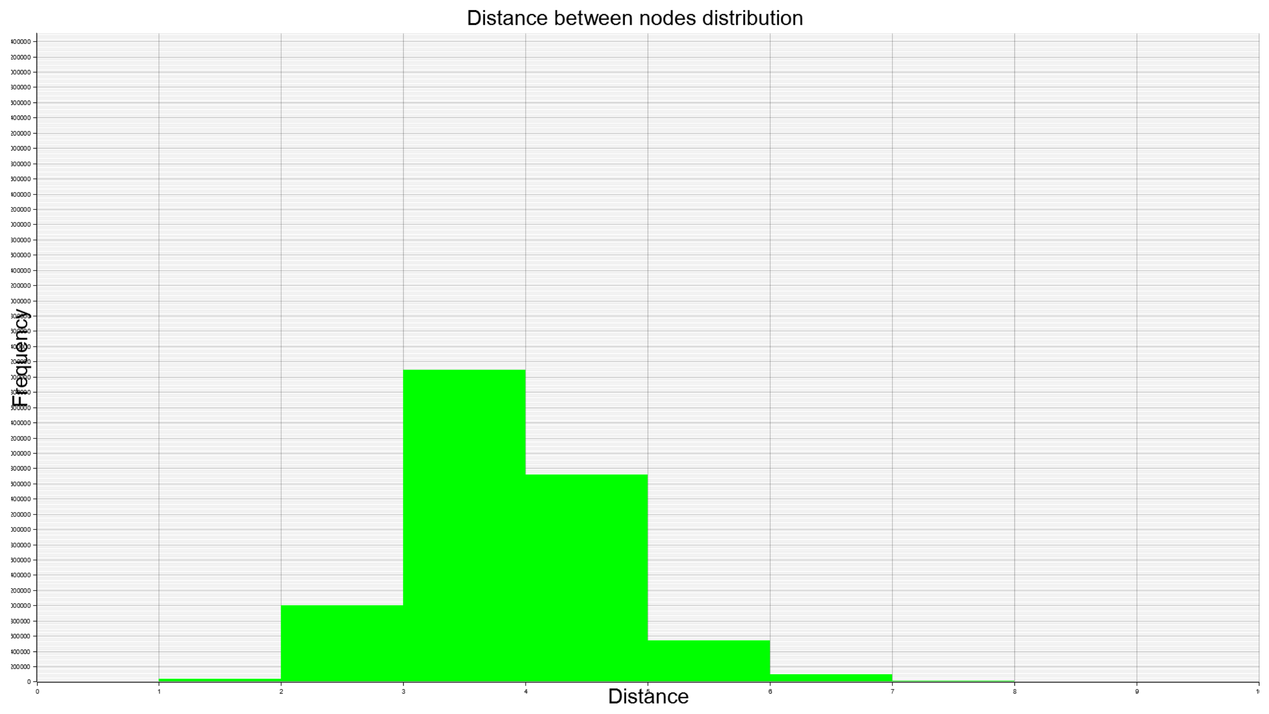
- The 15 nodes with most outgoing edges are:

  United_States: 294
  Driving_on_the_left_or_right: 255
  List_of_countries: 244
  List_of_circulating_currencies: 236
  List_of_sovereign_states: 216
  Africa: 212
  List_of_countries_by_system_of_government: 207
  Lebanon: 192
  Interpol: 191
  Armenia: 186
  Georgia_(country): 180
  Turkey: 172
  England: 172
  Israel: 169
  Germany: 169

- The 15 nodes with most outgoing edges are:

  United_States: 1551
  United_Kingdom: 972
  France: 959
  Europe: 933
  World_War_II: 751
  England: 751
  Germany: 743
  India: 611
  English_language: 598
  London: 587
  Japan: 573
  Canada: 571
  Australia: 563
  Italy: 550
  Spain: 539

These nodes are mainly countries and List, which is not a surprise. One interesting exception: "Driving_on_the_left_or_right"[1] is ranked so high because has outgoing edges to almost all countries.



[1] https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/d/Driving_on_the_left_or_right.htm

The distribution of distance between pairs of nodes has a:

- Mean = 3.36
- Median = 3
- Max = 9

This graph has a (at very max) nine degrees of separation: Knowledge is indeed interconnected.

RESOURCES

DS210 lecture notes

https://doc.rust-lang.org/rust-by-example/

https://github.com/plotters-rs/plotters

https://math.stackexchange.com/questions/3668702/shortest-distance-between-all-pairs-of-nodes-in-graph

https://www.programiz.com/dsa/floyd-warshall-algorithm

https://stackoverflow.com/questions/72201066/how-to-invert-a-hashmapstring-myenum-error-fromiterator-not-implement

https://robinmoussu.gitlab.io/blog/post/2021-03-25_rust_iterators_tips_and_tricks/