Pennsylvania Road Network Analyzer (PRNA)

Introduction

Pennsylvania Road Network Analyzer (PRNA) is a comprehensive network analysis project of road network of Pennsylvania. The project aims to provide some basic analysis of the road network conditions in Pennsylvania (total number of connections, mean, median, mode, maximum, minimum the node having the most number of connections, the percentage of both directional node pair in all connections) and some easily applicable tools to find the connection between different nodes (the shortest path between two nodes, the distance between two nodes). PRNA can be applied in Urban and Transportation Planning, Navigation Systems Services, and Policy Making about the traffic condition. In urban planning, especially when designing or analyzing road networks, it's crucial to identify intersections (nodes) that have the most connections. These intersections are typically high-traffic areas and are crucial for efficient traffic flow. For the navigation services can use this data to provide real-time traffic updates and alternate routes to drivers, helping them avoid congested areas. Policymakers can use data on the most connected nodes to strategize long-term traffic management and road network expansion plans.

Data Set

Download here: https://snap.stanford.edu/data/roadNet-PA.html

We are using Pennsylvania Road Network from Stanford Road Network Analysis Project. This is a road network of Pennsylvania. Intersections and endpoints are represented by nodes, and the roads connecting these intersections or endpoints are represented by undirected edges. This dataset contain 1088092 nodes and 1541898 edges in total. The brief statistics of this large dataset is as below:

Property	Value
Nodes in largest WCC	1087562
Edges in largest WCC	1541514

Nodes in largest SCC	10875562
Edge in largest SCC	1541514
Average clustering coefficient	0.0465
Number of triangles	67150
Fraction of closed triangles	0.02062
Diameter (longest shortest path)	786
90-percentile effective diameter	530

The main portions of the road network are captured within the largest Weakly Connected Component (WCC) and Strongly Connected Component (SCC), encompassing over 99% of both nodes and edges. The network's average clustering coefficient stands at 0.0465, which indicates a moderate level of clustering. Furthermore, the presence of 67,150 triangles and a closed triangle fraction of 0.02062 points to a notable number of interlinked clusters within the network. In terms of network reach, the longest shortest path (diameter) between any two nodes is 786, and the effective diameter at the 90th percentile is 530, reflecting a comparatively short average path length across the network. These metrics offer valuable insights for directing the analysis towards particularly significant or interesting sections of the road network.

Getting Started

1. Install Rust: Before running the Rust code, you should download the Rust in your system. Install instruction: https://www.rust-lang.org/learn/get-started
2. Clone the repository:

   ```
   git clone https://github.com/hLoeyCC/PRNA.git
   ```

3. Compile and run the project: Choose any of these below

   ```
   cargo run --release
   ```

   ```
   cargo run
   ```

4. Now you can explore the code having project up and running! Feel free to download and make any changes or additions to it.

Project Module

- `main.rs`
  - `read_road_data`: Reads road data from a file and converts it into a vector of ‘RoadRecord’ structs.
- `adjacencylist_convert.rs`
  - Convert the data from the set of RoadRecord into the form of adjacency list
  - `collect_unique_nodes`: Take a slice of ‘RoadRecord’ and return a set of ‘HashSet’. This function ensure every unique node without outgoing connection is collected.
  - `create_adjacency_list`: Take the data from ‘RoadRecord’ and convert into adjacency list for further use in Breadth First Search. It first initializes the ‘adj_list’ as an empty ‘HashMap’ and iterates through each nodes and insert into ‘adj_list’.
- `algorithms.rs`
  - `shortest_path_bfs`: This function is used for finding the shortest path between two nodes in the graph by bfs. By putting the data converted to adjacency list, the starting node and the ending node, the function would return all the nodes passing by the shortest path as list of ‘usize’.
  - `find_nodes_with_most_neighbors`: This function is used for finding any number of nodes having the most number of connections (e.g. can be used to find the top 3 nodes having most connections). It starts with calculating the number of nodes connected by each node (e.g. both A → B and C→A counting as the connections of A) and sorting in descending order. By inputting the adjacency list and number of top nodes you want to find, the function returns the ID number of the top nodes.
  - `nodes_info`: This function computes the basic information of mean, median, mode, maximum and minimum number of connections the nodes having by inputting the adjacency list.

- `both_direction_nodes`: The function is used to find all pairs of nodes that can go both directions (e.g. A ↔ B). It iterates through each connection and sees if the opposite connection also exist . The function ensures that each bidirectional pair is included only once (e.g. if (A, B) has already been contained in the vector, (B, A) would not be contained). This function will be used to calculate the percentage of bidirectional node pairs taking in all connections in the dataset.
- `count_total_connections`: This function is for counting all connections in dataset. This function will be used to calculate the percentage of bidirectional node pairs taking in all connections in the dataset.

Results

Sample Outputs:

```
Mean: 2.8341316726894417, Median: 3, Mode: 3, Maximum: 9, Minimum: 1
The Shortest Path between node NO.7 and node NO.7634:: Some([7, 3998, 4006, 4004, 4012, 4013,
4007, 4002, 4020, 4019, 4023, 4021, 4051, 4050, 4049, 4047, 4048, 4312, 4325, 4040, 4039,
4031, 2963, 2962, 2976, 3576, 5226, 10868, 3766, 3205, 3039, 2641, 2155, 3048, 3497, 3440,
3075, 3109, 3438, 3112, 3504, 3507, 10423, 3508, 6292, 5957, 5272, 5954, 5956, 5988, 5989,
6125, 6126, 5406, 5405, 5438, 5432, 5444, 5453, 5478, 7613, 7614, 5529, 7634])
Most connected Node: [859326]
Two roads having most number of connected roads are Road 759553 and Road 859326
Distance between 2 most connected roads: 186
```

```
The number of node pairs going both directions: 1541898.0
```

```
Percentage of roads going both directions: 100.0%
```

Analysis:

Nodes Information:

The mean number of connections all nodes have is around 2.834, the median is 3 and the mode is also 3. It indicates that there are a great number of nodes having 3 other nodes connected with them. The maximum number of nodes a node connected to is 9 nodes and the minimum is 1 nodes.

Shortest Path Algorithm:

For the main function, we tried to find the shortest path between Node NO.7 and NO.7634 as example, the nodes that passing by the shortest path are: 7, 3998, 4006, 4004, 4012, 4013, 4007, 4002, 4020, 4019, 4023, 4021, 4051, 4050, 4049, 4047, 4048, 4312, 4325, 4040, 4039, 4031, 2963, 2962, 2976, 3576, 5226, 10868, 3766, 3205,

3039, 2641, 2155, 3048, 3497, 3440, 3075, 3109, 3438, 3112, 3504, 3507, 10423, 3508, 6292, 5957, 5272, 5954, 5956, 5988, 5989, 6125, 6126, 5406, 5405, 5438, 5432, 5444, 5453, 5478, 7613, 7614, 5529, 7634.

Number of Connections and distance:

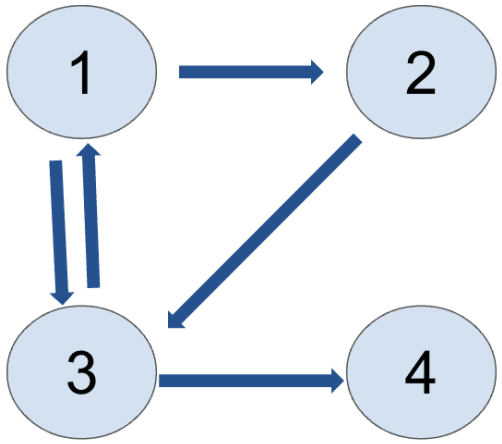
The node having the most number of connections is Node NO. 859326 and the second most is NO. 759553, and the distance between the is 186 nodes (since we don't have further information about the real length between each nodes in this dataset, we can only calculate it as the number of connecting roads between these two nodes). However, since there are several different nodes having the same number of neighbors, the top nodes may varied every time we run, which causes varied outcomes.

The node pairs going both directions:

There are 1541898 pairs of nodes in total number of 1541898 edges, and the percentage both directional pairs taking within all road connections is 100%, which indicated that the road network is undirected — all roads can go both directions.

Tests

- `create_linear_graph`: create s smaller simple directional graph to test our functions on `algorithms.rs` . The graph created is as below.



- `test_shortest_path_on_linear_graph`: apply `shortest_path_bfs` in `algorithms.rs` and check if the shortest path between node 1 and 4 is 1 → 3 → 4. And it does pass the

test.

- `test_most_neighbors_nodes`: apply `find_nodes_with_most_neighbors` in `mod algorithms` to find the top 2 most connected nodes and check if that's 1 and 3. The test passes.
- `test_info_correctnness`: apply `nodes_info` in `algorithms.rs` and check if the mean is 1.25, median is 1.5, mode is 2.0, maximum is 2.0, and minimum is 0.0. The test passes.
- `test_bidirectional_path`: apply `both_direction_nodes` in `algorithms.rs` and check if percentage of bidirectional node pairs is 40% (0.40). (total number of connections is 5 and number of bidirectional pair is 1 * 2; percentage is 2/5 = 0.40 = 40%).

Limitations

1. The dataset doesn't contain all road information with the state of Pennsylvania, so the analyzation we have above may have difference with real situation.
2. Since this dataset was published in 2009, there may be some updates until now, and it would affect the accuracy of our analysis.
3. The distance between 2 nodes can only be calculated as the number of connection since we don't have more information about exactly how long each road is.

Citation

- J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics 6(1) 29--123, 2009.
- https://gist.github.com/ayoisaiah/185fec1ca98ce44fca1308753182ff2b
- AI for error fixing.