

Dataset used: [http://konect.cc/networks/subelj\\_euroroad/](http://konect.cc/networks/subelj_euroroad/)

## **DS 210 Project Report**

Paul Lee

The dataset I am using for the project is the international E-road network, a road network located mostly in Europe. I decided to use this dataset because road networks are fit to map and find degrees of separation between the nodes. Each node represents a designated code for each city and roads in Europe and the edges represent the connection between two nodes denoting that they are connected by an E-road. The objective of this project is to examine the distances and degrees of connection between nodes, specifically focusing on determining the shortest number of E-roads required to travel from one city to another in Europe.

My final project file contains two modules called 'analysis' and 'graph\_helpers'. In 'analysis', I created a RoadNetwork instance with a predefined road graph. This graph forms the fundamental structure of the analysis. The first algorithm I have used is Breadth-First Search (BFS) algorithm. It finds the shortest path from an origin city to a destination city. Next, I needed to understand the connectivity between the different parts of the road network. The 'evaluate degrees' function calculates the degrees of separation between all pairs of nodes (cities) in the graph through iterating over each possible pair of cities, and uses the BFS algorithm to determine the level of separation between them across the entire road network. In the 'graph\_helpers' module, I utilized the petgraph crate to find the minimum path, in terms of the number of edges, between two cities in a graph. In terms of calculating the shortest distance, I used [Dijkstra's algorithm](#). It finds the shortest paths from the starting city to all other nodes in the graph.

The main.rs opens up the dataset I have imported in src and constructs an undirected graph of the data. A HashMap is used to efficiently track and reference city nodes in the graph.

As the file is read line by line, cities are added to the graph, and connections are established between them. Once the graph is constructed, the analysis::RoadNetwork module is used to analyze the network. This analysis includes calculating degrees of separation between cities and summarizing the connectivity of the network. I have added some statistical information about the network as well, such as the total number of connections, the average degree of separation between cities, and the standard deviation of these degrees of separation.

```
0 degrees of separation: 1174 connections.
1 degrees of separation: 2834 connections.
2 degrees of separation: 5318 connections.
3 degrees of separation: 9232 connections.
4 degrees of separation: 14244 connections.
5 degrees of separation: 19634 connections.
6 degrees of separation: 25144 connections.
7 degrees of separation: 30844 connections.
8 degrees of separation: 36222 connections.
9 degrees of separation: 41680 connections.
10 degrees of separation: 46202 connections.
11 degrees of separation: 48848 connections.
12 degrees of separation: 50210 connections.
13 degrees of separation: 50062 connections.
14 degrees of separation: 49124 connections.
15 degrees of separation: 47800 connections.
16 degrees of separation: 46340 connections.
17 degrees of separation: 44732 connections.
18 degrees of separation: 43206 connections.
19 degrees of separation: 41590 connections.
20 degrees of separation: 39834 connections.
21 degrees of separation: 37748 connections.
22 degrees of separation: 35256 connections.
23 degrees of separation: 32500 connections.
24 degrees of separation: 29542 connections.
25 degrees of separation: 26376 connections.
26 degrees of separation: 23498 connections.
27 degrees of separation: 21058 connections.
28 degrees of separation: 18940 connections.
29 degrees of separation: 17396 connections.
30 degrees of separation: 16232 connections.
31 degrees of separation: 15226 connections.
32 degrees of separation: 14248 connections.
33 degrees of separation: 13280 connections.
34 degrees of separation: 12184 connections.
35 degrees of separation: 10934 connections.
36 degrees of separation: 9610 connections.
37 degrees of separation: 8472 connections.
38 degrees of separation: 7460 connections.
39 degrees of separation: 6412 connections.
40 degrees of separation: 5484 connections.
41 degrees of separation: 4774 connections.
42 degrees of separation: 4080 connections.
43 degrees of separation: 3412 connections.
44 degrees of separation: 2908 connections.
45 degrees of separation: 2444 connections.
46 degrees of separation: 1956 connections.
47 degrees of separation: 1522 connections.
48 degrees of separation: 1178 connections.
51 degrees of separation: 484 connections.
52 degrees of separation: 350 connections.
53 degrees of separation: 264 connections.
54 degrees of separation: 194 connections.
55 degrees of separation: 144 connections.
56 degrees of separation: 100 connections.
57 degrees of separation: 64 connections.
58 degrees of separation: 42 connections.
59 degrees of separation: 26 connections.
60 degrees of separation: 16 connections.
61 degrees of separation: 8 connections.
62 degrees of separation: 2 connections.
Total connections: 1081660
Average degree of separation: 18.35
Standard deviation of degrees of separation: 9.43
```

Here are the outputs of the main function. As you can see, 25,144 connections of cities can reach each other within 6 different E-roads' combinations. This is 2.3246% of the total network between the cities. In order to reach the maximum connections you have to go through 12 degrees of separation, which are 50,210 connections. The average degree of separation is 18.35 with standard deviation of 9.43.

Sources:

<https://gist.github.com/vTurbine/16fbb99225ad4c0ac80b24855dd61a7c>

<https://www.sotr.blog/articles/breadth-first-search>

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

<https://www.sheshbabu.com/posts/rust-module-system/>

<https://crates.io/crates/petgraph>

<https://users.rust-lang.org/t/using-petgraph-graphmap-to-create-sub-graphs/79041>

<https://users.rust-lang.org/t/advice-on-architecture-and-testing-in-rust/70630>