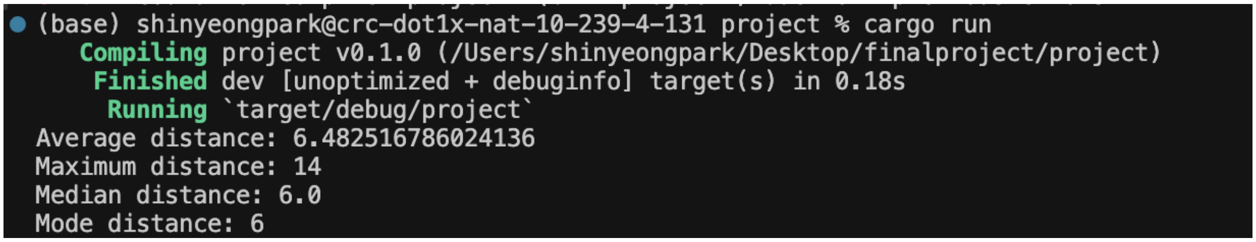
Collaborators: None

The dataset that I chose is the Korean Drama List that contains one column of 1278 unique dramas and one column of actors that are in the drama. Here is the link to the dataset: https://www.kaggle.com/datasets/iphigeniebera/korean-drama-list-about-740-unique-dramas?select=actors.csv.

This project constructs an undirected graph and uses drama names and actors as the nodes. There are edges between dramas that share similar actors. The project computes all the shortest path distances, the average, mode, and maximum of the distances.

Here is the output of the project:



```
● (base) shinyeongpark@crc-dot1x-nat-10-239-4-131 project % cargo run
     Compiling project v0.1.0 (/Users/shinyeongpark/Desktop/finalproject/project)
      Finished dev [unoptimized + debuginfo] target(s) in 0.18s
       Running `target/debug/project`
  Average distance: 6.482516786024136
  Maximum distance: 14
  Median distance: 6.0
  Mode distance: 6
```

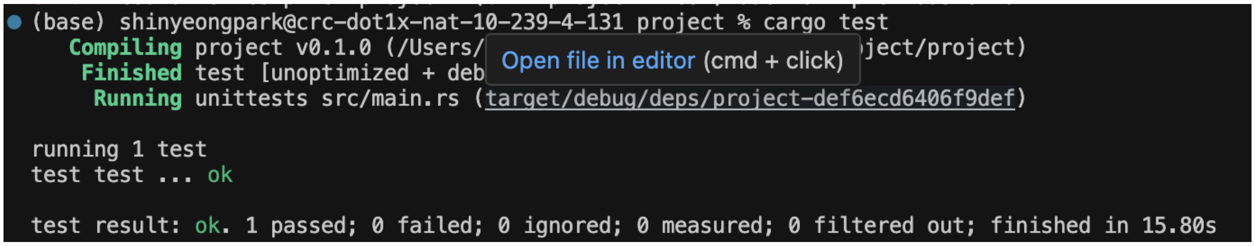
The average distance is around 6.48. Lower average distance means that a lot of dramas share similar actors. Higher distance represents a broad network between the dramas. The median and the mode are both 6. This means that a lot of the drama has a distance of 6. The maximum distance tells us the furthest network.

I created a test to check the mode function.

```
#[test]
fn test() {
let graph = file_graph("actors.csv");
let graph_distance = GraphDistances::new(graph.expect("REASON"));
let mode = graph_distance.unwrap().mode_distance;
assert_eq!(mode,6);
}
```

This checks if the mode of the dataset is 6.

Here is the output of the test:



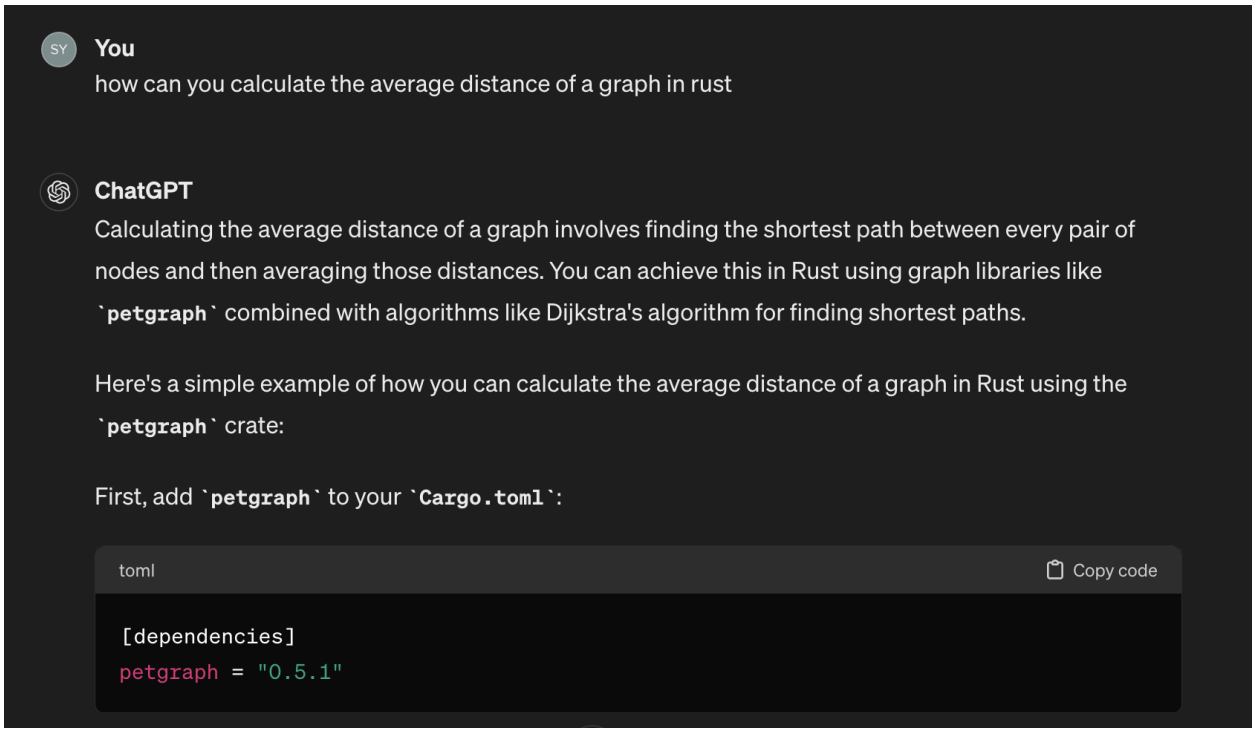
```
● (base) shinyeongpark@crc-dot1x-nat-10-239-4-131 project % cargo test
     Compiling project v0.1.0 (/Users/                              ject/project)
      Finished test [unoptimized + deb
       Running unittests src/main.rs (target/debug/deps/project-def6ecd6406f9def)

  running 1 test
  test test ... ok

  test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 15.80s
```

Websites used:

- https://docs.rs/petgraph/latest/petgraph/
- https://doc.rust-lang.org/rust-by-example/error/multiple_error_types/boxing_errors.html
- https://docs.rs/petgraph/latest/petgraph/algo/dijkstra/fn.dijkstra.html
- https://medium.com/@emanuelmechie/reading-csv-files-in-rust-4351aa2473bf
- https://codereview.stackexchange.com/questions/173338/calculate-mean-median-and-mode-in-rust

Chat GPT:

- In this project, I used chatgpt for the basis of the function because I wasn't sure where to start. I added on to the code that GPT provided to fit the context of my project.
- I used version 3.5, and here is the entire exchange with tool:
-



rust

Copy code

```rust
extern crate petgraph;

use petgraph::graph::{Graph, NodeIndex};
use petgraph::algo::dijkstra;
use std::collections::HashMap;

fn main() {
    // Create a graph
    let mut graph = Graph::<(), ()>::new();
    let node_a = graph.add_node(());
    let node_b = graph.add_node(());
    let node_c = graph.add_node(());
    let node_d = graph.add_node(());
    let node_e = graph.add_node(());

    graph.add_edge(node_a, node_b, ());
    graph.add_edge(node_b, node_c, ());
    graph.add_edge(node_c, node_d, ());
    graph.add_edge(node_d, node_e, ());
    graph.add_edge(node_a, node_e, ());
```

```rust
    // Calculate the average distance
    let average_distance = calculate_average_distance(&graph);
    println!("Average Distance: {}", average_distance);
}

fn calculate_average_distance(graph: &Graph<(), ()>) -> f64 {
    let mut total_distance = 0;
    let mut num_pairs = 0;

    // Iterate over all pairs of nodes
    for start_node in graph.node_indices() {
        let distances = dijkstra(graph, start_node, None, |_| 1);

        for (_, distance) in distances {
            total_distance += distance;
            num_pairs += 1;
        }
    }

    // Calculate the average distance
    if num_pairs > 0 {
        total_distance as f64 / num_pairs as f64
```

```rust
    // Calculate the average distance
    if num_pairs > 0 {
        total_distance as f64 / num_pairs as f64
    } else {
        0.0
    }
}
```

- This code uses dijsktra. Instead of using dijsktra, I created a function that computes all the distances. I used the code that does the calculation to find the average distance in my project.