**Collaborators**: None

**Dataset and Purpose**
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. The purpose of this project was to examine the network connections of actors and dramas.

I constructed an undirected graph with dramas and actors as nodes, connecting dramas that share actors. The project computes all the shortest path distances between nodes and then calculates the average, mode, and maximum of these distances.

**Output and Analysis**
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
```

**Average Distance:** The average distance is around 6.48. Lower average distance indicates that many dramas share similar actors. Higher average represents a broader network.

**Median and Mode:** The median and the mode are both 6, indicating that the most common distance among dramas is six steps.

**Maximum:** The maximum distance tells us the furthest network, which is 14.

I also created a test to verify the mode function.

```rust
#[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:

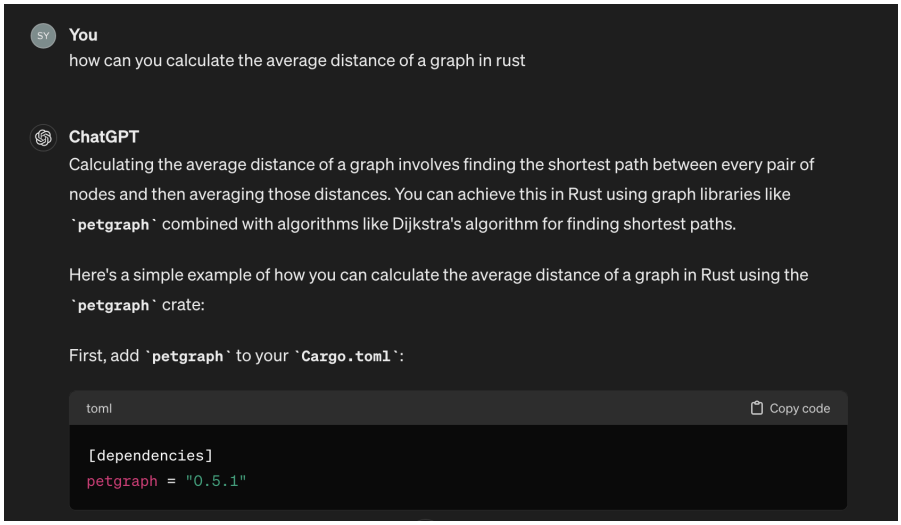
```
(base) shinyeongpark@crc-dot1x-nat-10-239-4-131 project % cargo test
   Compiling project v0.1.0 (/Users/                    oject/project)
    Finished test [unoptimized + deb  Open file in editor (cmd + click)
     Running unittests src/main.rs (target/debug/deps/project-def6ecd6406f9def)

running 1 test
test test ... ok

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

**Resources:**

- 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:**

- I used ChatGPT (version 3.5) for guidance. I used it as a foundation for the functions.
- Here is the entire exchange with tool:



**You**
how can you calculate the average distance of a graph in rust

**ChatGPT**
Calculating the average distance of a graph involves finding the shortest path between every pair of nodes and then averaging those distances. You can achieve this in Rust using graph libraries like `petgraph` combined with algorithms like Dijkstra's algorithm for finding shortest paths.

Here's a simple example of how you can calculate the average distance of a graph in Rust using the `petgraph` crate:

First, add `petgraph` to your `Cargo.toml`:

toml — Copy code

```
[dependencies]
petgraph = "0.5.1"
```

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 for my project, 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.