Katherine Beaty
DS210 Fall 2023
Final Project Report

Dataset:

The dataset I used consists of 1,912 nodes and features an undirected graph of Portuguese-speaking Twitch streamers. The nodes are the users themselves and the edges are the links of mutual friendship between the two users. This data was collected in May 2018.
The dataset can be found here:
https://www.kaggle.com/datasets/andreagarritano/twitch-social-networks/data (under the file name "PTBR_edges.csv")

How To Run:

The code can be run using cargo run or cargo run –release and expects user input in order to execute fully. The code will terminate under certain conditions specified in the main function.

Code Specifics:

The code is split into 3 modules and a main.rs file. There is an additional module for tests.

src/readfile.rs

This module includes the public Graph struct where n is the number of vertices in a graph and outages is an adjacency list for all the vertices in the form of a hashmap. The implementation for the Graph in this file includes a new function. This function creates a new graph with default values. The next function in this file is called read_file and it takes in the path of a text file and populates the graph's outedges adjacency list accordingly. It also determines the number of vertices by finding the maximum node number found in the file while accounting for the possibility of repeating nodes. The imputed file is assumed to have undirected data; therefore, the graph is undirected.

src/bfs.rs

This module contains the function to compute the Breadth-First Search algorithm and contains a public struct to hold the degrees from one node (the starting node) to all the other nodes in the graph. The compute_bfs function, to reiterate, computes the Breadth-First Search on an imputed graph and displays the shortest path distances from a specified starting vertex to all the other vertices in the graph.

src/friendrecs.rs

This module contains the function to find friend recommendations for a user based on a specified distance. This function uses the compute_bfs function in order to acquire the distances and then collects all the vertices with the specified distance.

src/main.rs

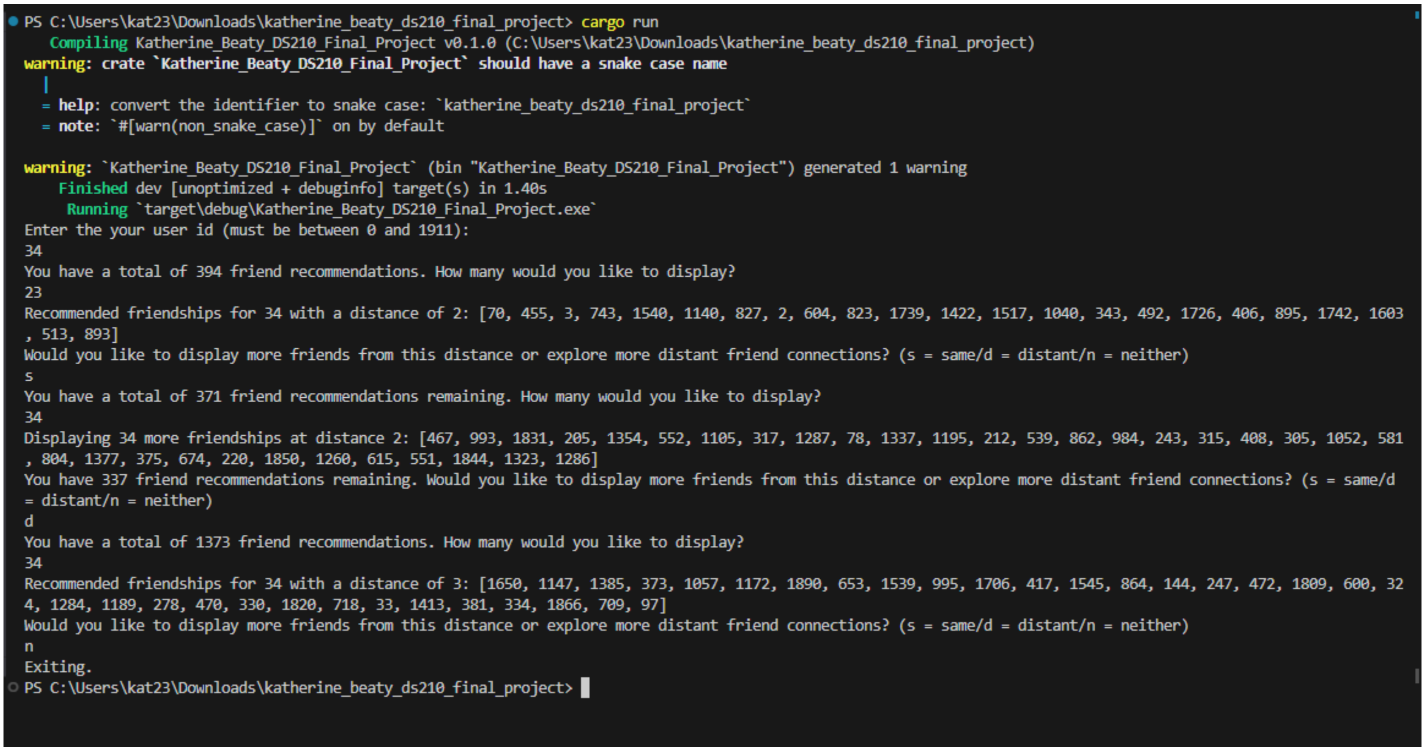
The main module has a function called user_input that states the basic layout for numeric user input and handles errors. The errors it can handle are non-numeric entries or an entry outside of the specified range. The main function contains the code to ask the user for their user ID, how many friend recommendations they would like to display, and if they would like to explore more distant friend connections, more friends within the same distance, or neither. If they choose the same then it displays the remaining number of friends they can view and allows the user to input the number of friends they want to see from that list. Choosing more distance allows the user to see the next distance away of friendship. Choosing neither exits the code.

Output:

The output will vary depending on user input but follows the general structure of:

- Asking the user for their user ID
- Asking the user to enter the number of friends they wish to see (distance = 2)
- Display the specified number of users from a shuffled list of friends
- Asking the user if they want to see friends in a more distant node, the same node, or neither
- Steps are repeated until code comes to a termination factor such as entering “n” for neither

Example Output:



```
PS C:\Users\kat23\Downloads\katherine_beaty_ds210_final_project> cargo run
   Compiling Katherine_Beaty_DS210_Final_Project v0.1.0 (C:\Users\kat23\Downloads\katherine_beaty_ds210_final_project)
warning: crate `Katherine_Beaty_DS210_Final_Project` should have a snake case name
  |
  = help: convert the identifier to snake case: `katherine_beaty_ds210_final_project`
  = note: `#[warn(non_snake_case)]` on by default

warning: `Katherine_Beaty_DS210_Final_Project` (bin "Katherine_Beaty_DS210_Final_Project") generated 1 warning
    Finished dev [unoptimized + debuginfo] target(s) in 1.40s
     Running `target\debug\Katherine_Beaty_DS210_Final_Project.exe`
Enter the your user id (must be between 0 and 1911):
34
You have a total of 394 friend recommendations. How many would you like to display?
23
Recommended friendships for 34 with a distance of 2: [70, 455, 3, 743, 1540, 1140, 827, 2, 604, 823, 1739, 1422, 1517, 1040, 343, 492, 1726, 406, 895, 1742, 1603
, 513, 893]
Would you like to display more friends from this distance or explore more distant friend connections? (s = same/d = distant/n = neither)
s
You have a total of 371 friend recommendations remaining. How many would you like to display?
34
Displaying 34 more friendships at distance 2: [467, 993, 1831, 205, 1354, 552, 1105, 317, 1287, 78, 1337, 1195, 212, 539, 862, 984, 243, 315, 408, 305, 1052, 581
, 804, 1377, 375, 674, 220, 1850, 1260, 615, 551, 1844, 1323, 1286]
You have 337 friend recommendations remaining. Would you like to display more friends from this distance or explore more distant friend connections? (s = same/d
= distant/n = neither)
d
You have a total of 1373 friend recommendations. How many would you like to display?
34
Recommended friendships for 34 with a distance of 3: [1650, 1147, 1385, 373, 1057, 1172, 1890, 653, 1539, 995, 1706, 417, 1545, 864, 144, 247, 472, 1809, 600, 32
4, 1284, 1189, 278, 470, 330, 1820, 718, 33, 1413, 381, 334, 1866, 709, 97]
Would you like to display more friends from this distance or explore more distant friend connections? (s = same/d = distant/n = neither)
n
Exiting.
PS C:\Users\kat23\Downloads\katherine_beaty_ds210_final_project> █
```

Tests:

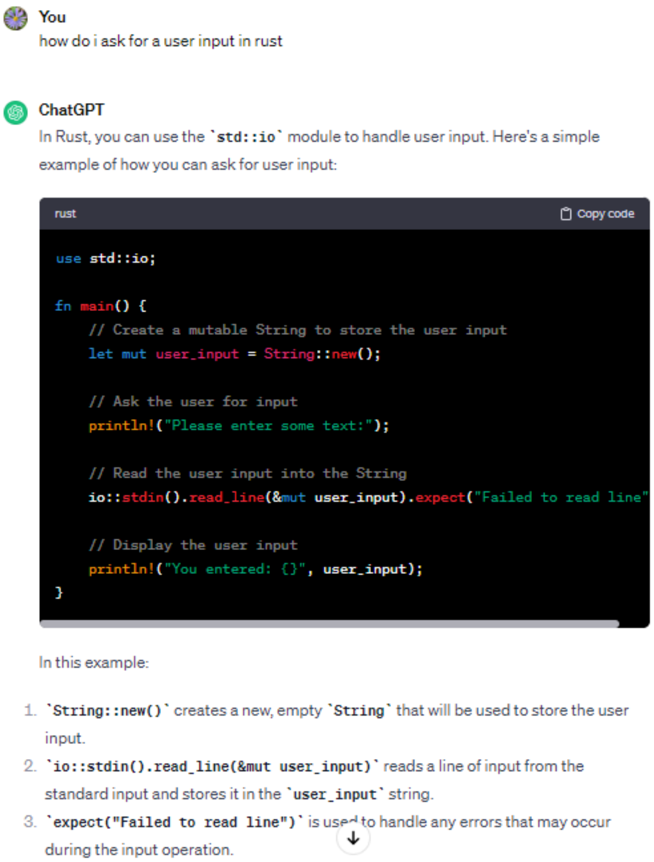
src/tests.rs

There are two tests implemented for this code. One checks the accuracy of the compute_bfs and the other checks the accuracy of the recommendations function. Both tests pass upon running cargo test.

Sources:

- https://web.mit.edu/rust-lang_v1.25/arch/amd64_ubuntu1404/share/doc/rust/html/book/second-edition/ch11-02-running-tests.html

- https://users.rust-lang.org/t/how-to-get-user-input/5176/8
- ChatGPT



You

how do i ask for a user input in rust

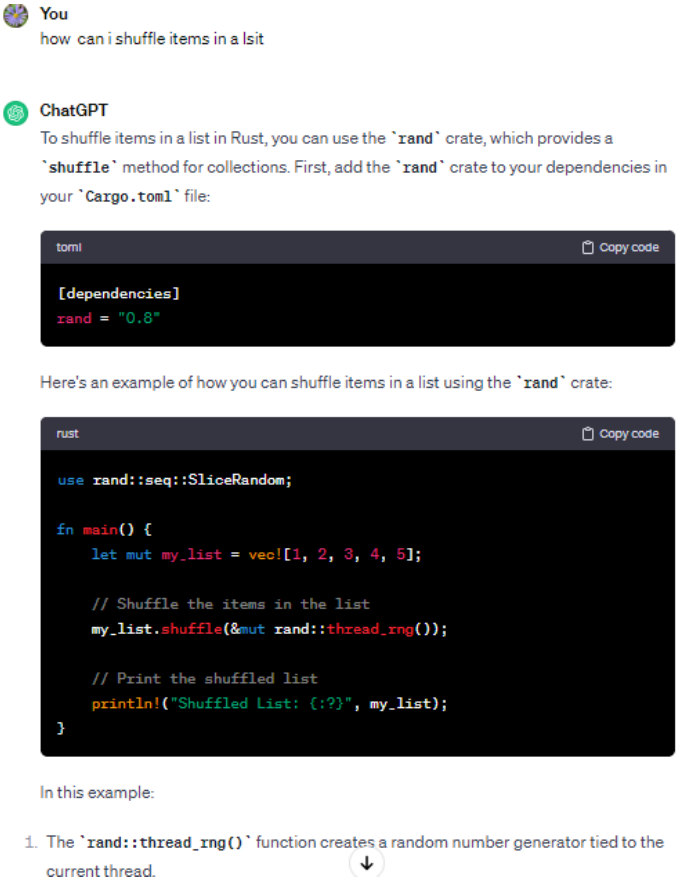
ChatGPT

In Rust, you can use the `std::io` module to handle user input. Here's a simple example of how you can ask for user input:

```rust
use std::io;

fn main() {
    // Create a mutable String to store the user input
    let mut user_input = String::new();

    // Ask the user for input
    println!("Please enter some text:");

    // Read the user input into the String
    io::stdin().read_line(&mut user_input).expect("Failed to read line"

    // Display the user input
    println!("You entered: {}", user_input);
}
```

In this example:

1. `String::new()` creates a new, empty `String` that will be used to store the user input.
2. `io::stdin().read_line(&mut user_input)` reads a line of input from the standard input and stores it in the `user_input` string.
3. `expect("Failed to read line")` is used to handle any errors that may occur during the input operation.



You

how can i shuffle items in a lsit

ChatGPT

To shuffle items in a list in Rust, you can use the `rand` crate, which provides a `shuffle` method for collections. First, add the `rand` crate to your dependencies in your `Cargo.toml` file:

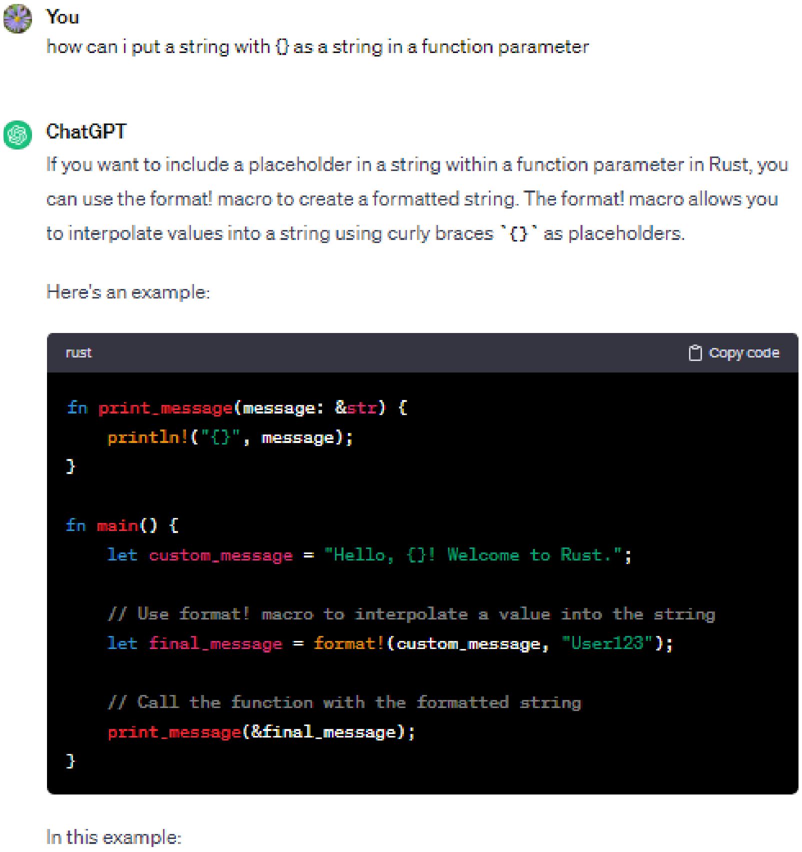
```toml
[dependencies]
rand = "0.8"
```

Here's an example of how you can shuffle items in a list using the `rand` crate:

```rust
use rand::seq::SliceRandom;

fn main() {
    let mut my_list = vec![1, 2, 3, 4, 5];

    // Shuffle the items in the list
    my_list.shuffle(&mut rand::thread_rng());

    // Print the shuffled list
    println!("Shuffled List: {:?}", my_list);
}
```

In this example:

1. The `rand::thread_rng()` function creates a random number generator tied to the current thread.



You

how can i put a string with {} as a string in a function parameter

ChatGPT

If you want to include a placeholder in a string within a function parameter in Rust, you can use the format! macro to create a formatted string. The format! macro allows you to interpolate values into a string using curly braces `{}` as placeholders.

Here's an example:

```rust
fn print_message(message: &str) {
    println!("{}", message);
}

fn main() {
    let custom_message = "Hello, {}! Welcome to Rust.";

    // Use format! macro to interpolate a value into the string
    let final_message = format!(custom_message, "User123");

    // Call the function with the formatted string
    print_message(&final_message);
}
```

In this example: