Gayoung Ko

# Final Project Write-Up

I am using files representing social media friend connections with about 10,000 entries. Before the existence of Facebook, there was a famous Korean social media platform called 'Cyworld,' which pretty much works the same as Facebook. It had a concept of 'Ilchon,' which means close familial relations. If one requests a friend request and the other accepts it, they are basically the 'Ilchon,' which represents the 1 degree of separation. There can be different degrees of separation, in which people's direct followers are 1 degree of separation, and those who are further apart have higher numbers of degrees of separation. For example, if I have a follower named John, he is the 1st degree, with a distance of 1. From my perspective, John's follower, Joanna, will be the 2nd degree, with a distance of 2. It is the same concept as the degree of kinship/separation within families but in social media. *Using the file, my objective for this project is to 1) assess individuals ' degree of separation and their further relationships based on their connections and followers on Cyworld and 2) determine the degree of relationships of 2 specific individuals on Cyworld.*

Before using the files, I removed one outlying information with an unusual value of a person's age of -5, which doesn't make sense, and confirmed that this outlier would be an obstacle. With samples of 999 individuals, I've created nodes using struct, implemented constructors, and read user_table.csv and friends_table.csv. I've converted the file into a CSV, read the file, created nodes, and started coding using Hashing. I've registered nodes in Hash Maps to efficiently get access to each Node since index information is absent in the current dataset. Utilizing hashing allows me to give each Node an 'id' number to increase the efficiency of finding each. In addition, it increases the efficiency of searching since I am using an extensive

dataset. Hashing has a time complexity of O(1), in which the time it takes will not depend on the input size but will always be in a constant time with order O(1).

To achieve my first objective, I have used BFS (Breadth-First Search) to categorize people based on the degrees of separation, it is efficient to use it because it explores a graph by prioritizing breadth instead of depth— ensuring that it explores nodes in the same level (let's say the distance of 1, 1st degree of relationship) before moving to the next deeper level (2nd degree of relationship, and so on). To achieve my second objective, I have similarly used BFS to find the shortest and quickest path that connects two specific individuals.

I will explain each of the rs files I've created and summarize what I did for each function. I've also added a few markdown comments on my code.

**data_preprocessing.rs:** This file does a job of data preprocessing that handles possible outliers and errors and returns a graph.

After defining a struct called Node,

1. delete_outlier function opens and reads the CSV file, removes outliers, creates a Node for each user, and creates HashMap that stores the Nodes instances with unique IDs. If everything is successful, it returns; otherwise, it gives you an error.
2. create_user_graph function opens and reads the file, connects nodes, and returns a graph represented as a HashMap. It will not return a graph if an error occurs.

**degree_of_kinship.rs:** This file allows us to understand the number of people in a certain degree of relationship/kinship/separation between the two specific nodes and their path from one to another.

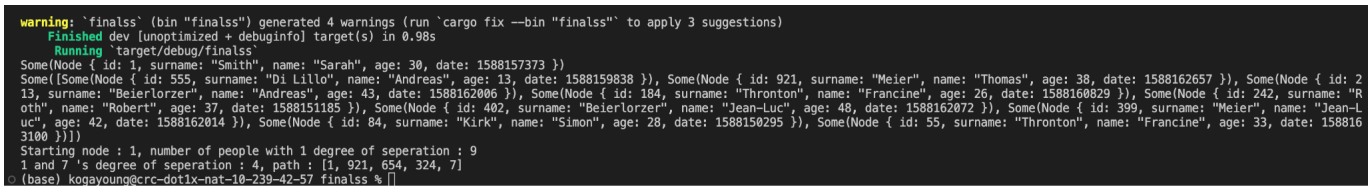
1. get_number_of_people_in_degree function uses BFS to visit possible nodes in the created graph. When it visits, it keeps track of which Node it visited, and when it reaches the desired Node, it terminates and returns the count. The count is the number of people with a certain degree of separation.
2. find_relationship_distance_with_path function also uses BFS to find the shortest distance between two assigned nodes by exploring the graph, keeping track of visited nodes, storing the path, and returning the path and the relationship distance.

**test._function.rs:** This test file is used to test if user data and graph is successfully made.

**main.rs:** This file imports the data_preprocessing file, the degree of kinship file, and the test_function file. After grabbing the CSV file, the first user_data_hashmap grabs the user's information and uses this in the test function tests to see if I can successfully create a graph as a HashMap. graph_hashmap then proceeds to use the information to create a graph, and the test proceeds to see if I can successfully find the starting node, the degree of separation, and the number of people at a certain degree of separation from the starting node. I have set the degree as 1 and the starting node as 1.

Then, I tested for a degree of separation between two individuals. I've set 1 as the 1st person and 7 as the second person to find the degree of separation and its path.

**Results**:



Gayoung Ko

- Starting Node: 1, number of people with 1 degree of separation: 9
- 1 and 7's degree of separation: 4, path : [1, 921, 654, 324, 7]

The results show that :

1. From the starting node of 1, there are 9 people with a degree of separation of 1.
2. 1 and 7's degree of separation is 4, and the path it took is [1, 921, 654, 324, 7]

**Sources:**


**https://doc.rust-lang.org/book/ch05-01-defining-structs.html**

**https://en.wikipedia.org/wiki/Six_degrees_of_separation**

**https://en.wikipedia.org/wiki/Cyworld**