CSE 222 HOMEWORK 8: Graphs

Gebze Technical University Computer Engineering Department

1 Introduction

This assignment involves the comprehensive implementation of a social network analysis system using graph data structures and algorithms. The objective is to simulate a social network and perform extensive analyses on the network, such as suggesting friends and counting clusters. Detailed instructions for each task will be provided to ensure clarity.

2 Assignment Requirements

2.1 Social Network Representation

Represent the social network using an undirected graph. Each node represents a person, and each edge represents a friendship. Each person should have the following attributes:

- **Name** - The person's name.
- **Age** - The person's age.
- **Hobbies** - A list of the person's hobbies.
- **Timestamp** - The date and time when the person joined the network.

3 OOP Design Specifications

3.1 Class Hierarchy

Develop the following classes for the social network analysis system:

- **Person**: Represents an individual in the network. Attributes include:
  - **Name** (String): The person's name.
  - **Age** (int): The person's age.
  - **Hobbies** (List<String>): A list of the person's hobbies.
  - **Timestamp** (DateTime): The date and time when the person joined the network.
- **SocialNetwork**: Represents the social network graph using an adjacency list. It includes:

  - **People** (Map<String, Person>): A map of people in the network, keyed by their names.
  - **Graph** (Map<Person, List<Person>>): An adjacency list representing friendships.
  - Methods to add/remove people, add/remove friendships, and perform analyses.

4 Core Features

Implement the following functionalities in detail:

4.1 Adding and Removing People

- **Add Person**:
  - Prompt the user for the person's name, age, and hobbies.
  - Add the person to the network with the current timestamp.
  - Display confirmation.
- **Remove Person**:
  - Prompt the user for the person's name and timestamp.
  - Remove the person and all associated friendships.
  - Display confirmation.

4.2 Adding and Removing Friendships

- **Add Friendship**:
  - Prompt the user for the names and timestamps of both people.
  - Add the friendship to the network.
  - Display confirmation.
- **Remove Friendship**:
  - Prompt the user for the names and timestamps of both people.
  - Remove the friendship from the network.
  - Display confirmation.

4.3 Finding Shortest Path

- Prompt the user for the names and timestamps of the start and end persons.
- Use Breadth-First Search (BFS) to find the shortest path.
- Display the path.

5 Advanced Features

Implement the following advanced functionalities:

5.1 Suggesting Friends

- Prompt the user for the person's name, timestamp, and the maximum number of friends to suggest (N).
- Suggest friends based on a score calculated from mutual friends and common hobbies.
- Calculate the score as follows:
  - Each mutual friend contributes 1 point.
  - Each common hobby contributes 0.5 points.
  - Total score = (number of mutual friends) * 1 + (number of common hobbies) * 0.5.
- Sort the candidates by their scores in descending order.
- Display the top N suggested friends with the score breakdown.

5.2 Counting Clusters

- Implement a Breadth-First Search (BFS) based algorithm.
- Display the number of clusters found.
- For each cluster, print out the names of the people in that cluster.

6 User Interface

Develop a command-line interface (CLI) to facilitate interaction. Below is the main menu template:

```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option:
```

Users interact by selecting an option and providing additional information as prompted.

7 Example Outputs

This section provides examples of expected outputs and interactions.

7.1 Adding a Person

```
Enter name: John Doe
Enter age: 25
Enter hobbies (comma-separated): reading,hiking,cooking
Person added: John Doe (Timestamp: 2024-05-29 10:30:00)
```

7.2 Adding a Friendship

```
Enter first person’s name: John Doe
Enter first person’s timestamp: 2024-05-29 10:30:00
Enter second person’s name: Jane Smith
Enter second person’s timestamp: 2024-05-28 14:45:00
Friendship added between John Doe and Jane Smith
```

7.3 Finding Shortest Path

```
Enter first person’s name: John Doe
Enter first person’s timestamp: 2024-05-29 10:30:00
Enter second person’s name: Alice Johnson
Enter second person’s timestamp: 2024-05-27 09:15:00
Shortest path: John Doe -> Jane Smith -> Bob Brown -> Alice Johnson
```

7.4 Suggesting Friends

```
Enter person’s name: John Doe
Enter person’s timestamp: 2024-05-29 10:30:00
Enter maximum number of friends to suggest: 3
Suggested friends for John Doe:

Alice Johnson (Score: 4.0, 2 mutual friends, 2 common hobbies)
Charlie Lee (Score: 3.5, 1 mutual friend, 5 common hobbies)
David Kim (Score: 2.5, 1 mutual friend, 3 common hobbies)
```

7.5 Counting Clusters

```
Counting clusters in the social network...
Number of clusters found: 3

Cluster 1:
John Doe
Jane Smith
Bob Brown

Cluster 2:
```

```
Alice Johnson
Charlie Lee
David Kim

Cluster 3:
Emily Davis
Frank Wilson
```

8 Submission Guidelines

Submissions must include complete JavaDoc documentation, a PDF report, and a Makefile for compilation. Code should be well-commented, detailing purpose, inputs, outputs, and side effects.

9 Evaluation Criteria

Projects will be evaluated on functionality, adherence to OOP principles, code quality, interface usability, and documentation completeness.

10 Frequently Asked Questions

1. **Q:** How should I handle invalid inputs or edge cases?
   A: Implement appropriate error handling mechanisms. Display informative error messages to the user and ensure the program continues running smoothly.

2. **Q:** Can I use external libraries for graph algorithms?
   A: No, you should implement the graph algorithms (BFS, shortest path) from scratch to demonstrate your understanding of the concepts.

3. **Q:** What should I do if a person or friendship is not found during removal?
   A: Display a message indicating that the person or friendship was not found in the network. The program should continue running without any errors.

4. **Q:** How should I handle cases where there is no path between two people?
   A: When finding the shortest path, if no path exists between the two people, display a message indicating that there is no connection between them.

5. **Q:** How should I handle empty clusters while counting clusters?
   A: If an empty cluster is encountered (a person with no friends), you can either include it in the count or ignore it based on the requirements. Clearly document your approach in the code and report.