CSE222 – Homework 8 PDF Report- RECEP FURKAN AKIN

findShortestPath function

- The method takes two parameters: **startName** and **endName**. These are the names of the two people between whom we want to find the shortest path.
- It retrieves the **Person** objects corresponding to **startName** and **endName** from the people map.
- If either start or end is null, it means one or both people are not found in the network. In this case, it prints an error message and returns.
- It initializes three data structures: **prev** (a map to store the previous person in the path for each person), **queue** (a queue for BFS), and **visited** (a set to store visited persons).
- It adds the **start** person to the **queue** and marks it as visited by adding it to the **visited** set.
- It enters a loop that continues until the **queue** is empty. In each iteration:
    - It dequeues a person (**current**) from the queue.
    - If **current** is the **end** person, it means a path has been found. It then calls the **printPath** method to print the path and returns.
    - If **current** is not the **end** person, it goes through each friend (**neighbor**) of **current**. If a **neighbor** has not been visited, it adds the **neighbor** to the **queue**, marks it as visited, and sets **current** as its previous person in the **prev** map.
- If the loop finishes without finding a path, it means there is no path between **startName** and **endName**. It then prints a message indicating this.

countClusters function

- It initializes a **Set** called **visited** to keep track of the **Person** objects that have been visited.
- It also initializes a counter **count** to keep track of the number of clusters.
- It prints a message indicating that it's starting to count the clusters.
- It then starts a loop over all **Person** objects in the **people** map.
- For each **Person**, it checks if the **Person** has been visited. If not, it means this **Person** is part of a new cluster.
- It initializes a **List** called cluster to store the **Person** objects in the new cluster.
- It performs a Breadth-First Search (BFS) starting from the current **Person** to find all **Person** objects in the same cluster. The **bfs** method takes three parameters: the starting

Person, the **visited** set, and the **cluster** list. It adds all **Person** objects in the same cluster to the **cluster** list and marks them as visited.

- It increments the cluster count.
- It prints the number of the cluster and the names of all **Person** objects in the cluster.
- It prints an empty line for separation between clusters.
- After the loop, it prints the total number of clusters.

bfs function

- The method takes three parameters: **start** (the starting person), **visited** (a set of people who have been visited), and **cluster** (a list of people in the current cluster).
- It initializes a **Queue** called **queue** for BFS.
- It adds the **start** person to the **queue** and marks it as visited by adding it to the **visited** set.
- It enters a loop that continues until the **queue** is empty. In each iteration:
    - It dequeues a person (**current**) from the **queue**.
    - It adds **current** to the **cluster** list because current is in the same cluster as **start**.
    - It goes through each friend (**neighbor**) of **current**. If a **neighbor** has not been visited, it adds the **neighbor** to the **queue** and marks it as visited. This is because **neighbor** is also in the same cluster as **start**.
- The method doesn't return anything. Instead, it modifies the **visited** set and the **cluster** list in-place. After the method finishes, all people in the same cluster as **start** will be in the **cluster** list and marked as visited in the **visited** set.

The time complexity of a Breadth-First Search (BFS) is O(V + E), where V is the number of vertices (people in this case) and E is the number of edges (friendships in this case). This is because each person and each friendship are processed exactly once.

The space complexity of BFS is O(V), where V is the number of vertices. This is because in the worst-case scenario, all vertices could be in the queue at the same time. In this case, the **visited** set, the **queue**, and the **cluster** list can all potentially store all the people, so the space complexity is O(V).

suggestFriends function

- The method takes two parameters: **name** (the name of the person for whom to suggest friends) and **maxSuggestions** (the maximum number of suggestions to make).
- It retrieves the **Person** object corresponding to name from the **people** map.
- If **person** is **null**, it means the person is not found in the network. In this case, it prints an error message and returns.
- It initializes a **Map** called **scores** to store the scores for potential friends. The score is calculated based on the number of mutual friends and common hobbies.
- It starts a loop over all **Person** objects in the **people** map. For each **Person** (**potentialFriend**):
    - If **potentialFriend** is not already a friend of **person** and is not **person** itself, it calculates a score for **potentialFriend**.
    - It initializes a **List** called **mutualFriends** to store the mutual friends of **person** and **potentialFriend**. It adds the number of mutual friends to the score.
    - It initializes a **List** called **commonHobbies** to store the common hobbies of **person** and **potentialFriend**. It adds half the number of common hobbies to the score.
    - It stores the score for **potentialFriend** in the **scores** map.
- It initializes a **List** called **sortedScores** to store the scores sorted in descending order.
- It prints the suggestions. For each suggestion, it prints the name of the potential friend and the score.

Here some output screenshots:

```
Friend suggestions for John Smith:
Suggestions for John Smith:
John Brown with score 3.0
Daniel Rodriguez with score 2.5
Emily Martinez with score 2.0
Katie Johnson with score 2.0
John Williams with score 2.0
```

```
John Johnson
Chris Miller
Emily Smith
Tom Garcia
Sara Rodriguez

Cluster 2:
Mike Brown

Cluster 3:
Jane Williams

Cluster 4:
Chris Smith
Sara Jones

Cluster 5:
Katie Brown

Cluster 6:
Jane Miller

Cluster 7:
Tom Rodriguez
Emily Garcia
Tom Davis

Cluster 8:
Daniel Miller

Cluster 9:
Sara Davis

Total clusters: 9
```

```
recep@DESKTOP-USOAJ0L:/mnt/c/Users/DELL/Desktop/cse
javac -g Test.java Person.java SocialNetworkGraph.java
java Test
Person added: Mike Davis (Age: 64, Hobbies: [Gaming])
Person added: Mike Davis (Age: 64, Hobbies: [Gaming])
Person added: John Brown (Age: 44, Hobbies: [Hiking, Hiking, Photography])
Person added: John Brown (Age: 44, Hobbies: [Hiking, Hiking, Photography])
Person added: Katie Jones (Age: 72, Hobbies: [Hiking])
Person added: Katie Jones (Age: 72, Hobbies: [Hiking])
Person added: Emily Martinez (Age: 67, Hobbies: [Art])
Person added: Emily Martinez (Age: 67, Hobbies: [Art])
Person added: Chris Williams (Age: 44, Hobbies: [Reading, Traveling])
Person added: Chris Williams (Age: 44, Hobbies: [Reading, Traveling])
Person added: John Brown (Age: 66, Hobbies: [Cooking, Art, Hiking])
Person added: John Brown (Age: 66, Hobbies: [Cooking, Art, Hiking])
Person added: Katie Johnson (Age: 45, Hobbies: [Music, Music, Gaming])
recep@DESKTOP-USOAJ0L:/mnt/c/Users/DELL/Desktop/cse222/assignments/hw8$ make run
javac -g Main.java Person.java SocialNetworkGraph.java
java Main
===== Social Network Analysis Menu =====
1. Add a person
2. Remove a person
3. Add a friendship
4. Remove a friendship
5. Find the shortest path between two people
6. Suggest friends for a person
7. Count clusters
8. Exit
Please select an option:
```

```
Counting clusters in the social network...
Cluster 1:
Sara Smith
Daniel Martinez
Tom Williams
John Rodriguez
Daniel Rodriguez
Jane Johnson
Alex Jones
Sara Brown
John Davis
Chris Davis
```

```
Friendship added between Jane Davis and John Smith
Friendship added between Jane Davis and John Smith
Friendship added between John Davis and Chris Garcia
Friendship added between John Davis and Chris Garcia
Friendship added between Sara Garcia and John Smith
Friendship added between Sara Garcia and John Smith
Friendship added between Chris Johnson and Emily Martinez
Friendship added between Chris Johnson and Emily Martinez
Friendship added between Jane Garcia and Chris Garcia
Friendship added between Jane Garcia and Chris Garcia
Friendship added between Tom Johnson and Mike Johnson
Friendship added between Tom Johnson and Mike Johnson
Friendship added between Alex Martinez and Katie Jones
```

Note: If you run `make test`, the script will generate random test cases that exercise all functions in the system. This includes adding a large number of people, establishing friendships, generating friend suggestions, and counting clusters. The script ensures comprehensive testing of the social network's functionality.

```
Shortest path between John Brown and Jane Garcia:
Shortest path: John Brown->Chris Garcia->Jane Garcia
```