CSE 222 HOMEWORK 8: Graphs

AYKUT SERT 200104004104

Person Class

The Person class represents individuals in a social network. The Person class is used to store and access the information of individuals in the social network.

Properties:

- **String name**: It holds the person's name. It is of String data type.
- **İnt age**: It keeps the person's age. It is of int data type.
- **List<String> hobbies**: It keeps the person's hobbies. It is of List<String> data type.
- **Date timestamp**: It keeps track of when the person joined the network. It is of Date data type

Methods:

Constructor => This constructor creates a new Person object. It takes the person's name, age, hobbies and the time they joined the network as parameters. These parameters are assigned to the relevant fields.

public String getName() => Returns the person's name.

public int getAge() => Returns the person's age.

public List<String> getHobbies() => Returns a list of the person's hobbies.

public Date getTimestamp() => Returns the date and time the person joined the network.

public String toString() => Returns a String containing the person's name, age, and hobbies. This method presents the person's information in a readable format.

SocialNetwork Class

The SocialNetwork class holds the people in the social network and their friendship relationships. It uses two main data structures:

1. **people (Map<String, Person>):** A map that holds people's names as keys and Person objects as values.

2. **graph (Map<Person, List<Person>>)**: A map that keeps Person objects as keys and their friends as values.

Constructor => This constructor initializes the people and graph maps.

AddPerson => The addPerson method is used to add a new person to the social network. The parameters of the method are the name, age and list of hobbies of the person to be added. First, the validity of the entered data is checked; If the name is empty, the age is zero or less, or the list of hobbies is null, an invalid input message is printed and the method terminates. When valid data is provided, the current date and time information (timestamp) is retrieved and a new Person object is created with this information. The new Person object is added to the people map with its name and added to the graph map with an empty friends list. Information that the contact has been successfully added is printed on the console, along with their name and addition timestamp. This method allows new people to be added to the social network smoothly and gives feedback to the user.

RemovePerson => The removePerson method is used to remove a person from the social network. The parameter of the method is the name of the person to be removed. First, the validity of the entered name is checked; If the name is empty or null, an invalid input message is printed and the method terminates. When a valid name is provided, the Person object matching the specified name is extracted from the people map. If the person is found, that person is also removed from the graph map. It is then also removed from the friend lists of other people on the network. Information that the contact has been successfully removed is printed on the console along with their name. If the person is not found, a message that the specified person cannot be found is printed. This method ensures accurate and complete removal of people from the social network and provides feedback to the user.

AddFriendShip => The addFriendship method is used to add friendship between two people in the social network. The parameters of the method are the names of the two people to be friends with (name1 and name2). First, the validity of both names is checked; If one or both of the names are empty or null, an invalid input message is printed and the method terminates. When valid names are provided, Person objects (person1 and person2) matching the names are retrieved from the people map. If both people are found, they add each other to both people's friend lists in the graph map. Information about successfully adding friends is printed to the console along with their names. If one or both of the contacts cannot be found, a message that the specified contacts cannot be found is printed. This method allows people to become friends with each other on the social network and notifies the user of the situation.

RemoveFriendShip => The removeFriendship method is used to remove friendship between two people in the social network. The parameters of the method are the names

of the two people to be unfriended (name1 and name2). First, the validity of both names is checked; If one or both of the names are empty or null, an invalid input message is printed and the method terminates. When valid names are provided, Person objects (person1 and person2) matching the names are retrieved from the people map. If both people are found, the graph map removes each other from both people's friend lists. Information about successful friend removals is printed to the console along with their names. If one or both of the contacts cannot be found, a message that the specified contacts cannot be found is printed. This method allows people to end their friendship relations with each other on the social network and notifies the user of the situation.

FindShortestPath => The findShortestPath method is used to find the shortest path between two people on a social network. The parameters of the method are the names of the two people (name1 and name2) to whom the path will be found. First, Person objects (start and end) matching names are retrieved from the people map. If both contacts are found, the bfsForPath method is called to look for the shortest path between the start and end points. This method finds the shortest path between two people using the Breadth-First Search (BFS) algorithm and returns it as a list. If a path is found, the names of the people on the path are printed on the console, separated by " -> ".If the path is not found, a message that no connection was found between the specified contacts is printed. If one or both of the contacts cannot be found, a message is printed stating that the contacts cannot be found. This method allows finding the shortest connection path between two people in the social network and reporting it to the user in a clear way.

SuggestFriends => The suggestFriends method is used to suggest people that a person can add as friends on a social network. The parameters of the method are the name of the person to whom the suggestion will be made and the maximum number of suggestions (maxSuggestions). First, the Person object (person) matching the name is retrieved from the people map. If the person is not found, the method ends by printing the relevant message.

If the person is found, potential friends are identified among other people. This determination is based on people who have mutual friends and hobbies who are not on their current friends list. For each candidate, the variable mutualFriends calculates the number of mutual friends and the variable commonHobbies calculates the number of mutual hobbies. Based on these two values a score is calculated (mutualFriends + 0.5 * commonHobbies). Candidates with scores greater than 0 are added to a recommendation list.

Suggestions are sorted in descending order by score. The suggestions are then printed to the console, up to the specified maximum number of suggestions. Each suggestion includes the person's name, score, number of mutual friends, and number of common hobbies. This method helps the user to identify and recommend suitable people to make new friends on the social network.

CountClusters => The countClusters method is used to find the number of clusters in the social network and the list of people in each cluster. The method keeps the list of people visited by creating a cluster called visited and keeps the list of people in each cluster by creating a list called clusters.

For each person in the graph map, a cluster is created if the person has not been visited before. This cluster is expanded using the BFS algorithm using the bfs method. This is accomplished by scanning the list of one's friends and visiting each friend. The BFS algorithm finds all contacts that are linked to a particular contact.

When a cluster is complete (bfs complete), it is added to the list of clusters. After all contacts are scanned, the size of the clusters list indicates the number of clusters and the list of contacts in each cluster is printed to the console.

The purpose of this method is to identify different friendship groups in the social network and list the members of each group. This helps the user better understand the communities on the network.

empty clusters => When printing a cluster, people who did not have friends were included in the cluster.

BFS => The bfs method is used to find connections between people in a social network using the breadth-first search (BFS) algorithm. The parameters of the method are defined as a starting point (start), a set of visited people (visited) and a list of found people (cluster).

The BFS algorithm starts with one person, scans that person's direct friends, and then scans those friends' friends, and so on. For each person, the BFS algorithm finds that person's direct friends and adds them to a queue. These contacts are then marked as visited, and friends belonging to these contacts are processed in the same way.

This process is completed when the queue is empty and the cluster list contains all contacts that are reachable from the starting point person. In this way, the BFS algorithm finds all contacts that are linked to a person in the social network.

The purpose of this method is to find all contacts linked to a particular contact and add these contacts to the cluster list. This helps us better understand the communities and connections in the social network.

BFSforPath => The bfsForPath method is used to find the shortest path between two specific contacts in the social network using the breadth-first search (BFS) algorithm.

The parameters of the method are defined as a person (start) which is the starting point and a person (end) which is the end point.

The BFS algorithm tries to find the shortest path from the starting point to the ending point. During this process, for each person, the BFS algorithm finds that person's direct friends and adds them to a queue. These contacts are then marked as visited, and friends belonging to these contacts are processed in the same way.

Once the path from the start point to the end point is found, the BFS algorithm returns a list containing the entire path. This list includes contacts from the starting point to the ending point. If the link is not found, the method returns null.

The purpose of this method is to find the shortest path between two given individuals and return a list containing that path. This helps us better understand the connections and distances in the social network.

Suggestion Class

The Suggestion class is used to represent a person's suggested friends. This class includes attributes such as person, score, number of mutual friends, and number of common hobbies for each recommendation.

Properties:

1. **Person person:** It is a Person object that represents the suggested person.
2. **Double score:** It is the score of the recommended person. This score is calculated by a combination of the number of mutual friends and the number of common hobbies.
3. **İnt mutualFriends:** Represents the number of mutual friends between the recommended person and the original person.
4. **Long commonHobbies:** It represents the number of common hobbies between the suggested person and the original person.

Methods:

1. **getPerson()** => Returns the suggested contact.
2. **getScore()** => Returns the score of the recommendation.
3. **getMutualFriends()** => Returns the number of mutual friends between the suggested contact and the original contact.
4. **getCommonHobbies()** => Returns the number of common hobbies between the suggested contact and the original contact.

The purpose of this class is to represent friends recommended to a person and store the score and properties of each recommendation. This is used when creating and sorting the list of suggested friends.

Some Outputs:

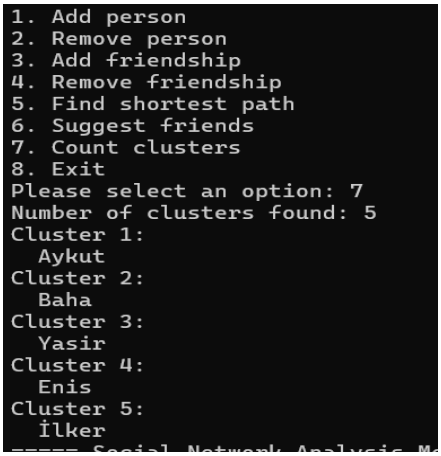
Test Case in homework pdf

```
Person added: John Doe (Timestamp: Wed May 29 17:31:25 TRT 2024)
Person added: Jane Smith (Timestamp: Wed May 29 17:31:25 TRT 2024)
Person added: Alice Johnson (Timestamp: Wed May 29 17:31:25 TRT 2024)
Person added: Bob Brown (Timestamp: Wed May 29 17:31:25 TRT 2024)
Person added: Emily Davis (Timestamp: Wed May 29 17:31:25 TRT 2024)
Person added: Frank Wilson (Timestamp: Wed May 29 17:31:25 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Number of clusters found: 2
Cluster 1:
  John Doe
  Jane Smith
  Alice Johnson
  Bob Brown
Cluster 2:
  Emily Davis
  Frank Wilson
```

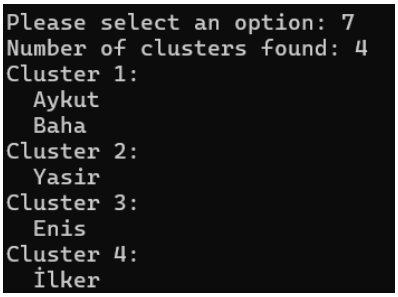
AddPerson and RemovePerson

```
===== 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: 1
Enter name: Aykut Sert
Enter age: 23
Enter hobbies (comma-separated): fitness
Person added: Aykut Sert (Timestamp: Wed May 29 19:09:24 TRT 2024)
===== 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: 2
Enter name: Aykut Sert
Enter person's timestamp: Wed May 29 19:09:24 TRT 2024
Person removed: Aykut Sert
===== Social Network Analysis Menu =====
```

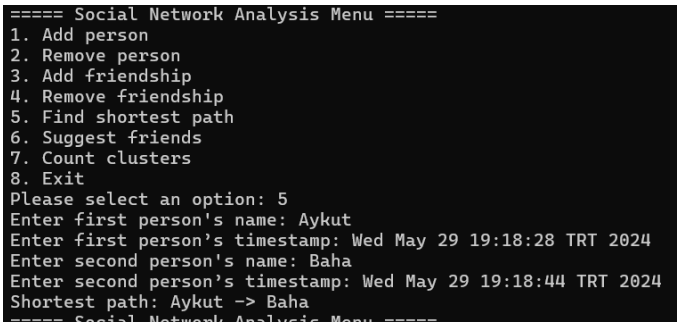
Cluster 1:



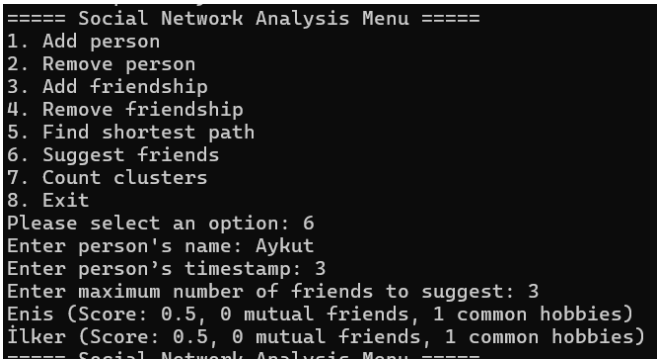
Cluster 2:



ShortestPath:



Suggestion:



RemoveFriendShip:

```
Please select an option: 3
Enter first person's name: Aykut
Enter first person’s timestamp:  2024-05-29 20:05:16
Enter second person's name: Baha
Enter second person’s timestamp: 2024-05-29 20:05:21
Friendship added between Aykut and Baha
===== 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: 7
Number of clusters found: 1
Cluster 1:
  Aykut
  Baha
===== Social Network Analysis Menu =====
```

```
Enter first person's name: Aykut
Enter first person’s timestamp: 2024-05-29 20:05:16
Enter second person's name: Baha
Enter second person’s timestamp: 2024-05-29 20:05:21
Friendship removed between Aykut and Baha
===== 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: 7
Number of clusters found: 2
Cluster 1:
  Aykut
Cluster 2:
  Baha
===== Social Network Analysis Menu =====
```