

DSA Final Project

**Group Members:**

Iruj Zulfiqar

**(SP23-BCS-032)**

Maimoona Mustafvi

**(SP23-BCS-039)**

**Class:** BCS-3A

**Instructor:**

Dr. Inayat Ur Rehman

**Dated:** 28th May, 2024

**Instructor’s Name:**

Mr. Muhammad Ali

**Mini LinkedIn**

**Overview**

This project simulates a social network where users can connect with each other, send messages, and view messages. The network is represented as a graph, where users are nodes and connections between users are edges. Messages sent between users are managed using a max-heap structure to ensure the most recent messages are easily accessible.

**Components**

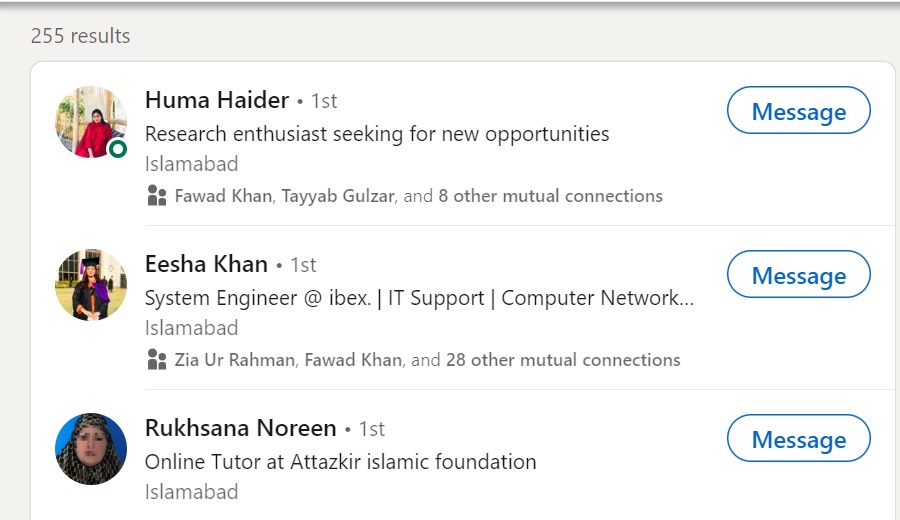
**Data Structures**

1. **Messages**: Represents a message with content and a timestamp.
2. **LinkedInUser**: Represents a user in the network with a username, skills, an index, and a list of messages. The messages are managed as a max-heap.
3. **Graph**: A vector of vectors of linkedinuser objects, representing the connections between users.
4. **Heap**: Sorted messages stored in user’s inbox.

**Functions**

1. **addEdge**: Adds a bidirectional edge between two users in the graph.
2. **printGraph**: Prints the connections between all users in the graph.
3. **BFS**: Performs a breadth-first search on the graph starting from a specified node, printing **the level of each visited node**.
4. **sendMsg**: Sends a message from one user to another, adding the message to both users' heaps and adjusting the heaps accordingly.
5. **viewMsg**: Prints all messages for a specified user.
6. **createAccount**: Creates a new user profile.
7. **searchUser**: Search a user by username and view their profile.

**Implementation:**



A screenshot of a computer

Description automatically generated

A screenshot of a social media profile

Description automatically generated

**Levels in BFS**

In the context of BFS, the "level" of a node (user) refers to its distance from the starting node (the node from which BFS is initiated). The distance is measured in terms of the number of edges traversed.

* **Level 0**: The starting node itself.
* **Level 1**: Nodes directly connected to the starting node.**(1st Degree connections)**
* **Level 2**: Nodes connected to the Level 1 nodes.**(2nd Degree connections)**
* **Level 3**: Nodes connected to the Level 2 nodes.**(3rd Degree connections)**
* **Level others**: Nodes at levels greater than 3.

A diagram of a network

Description automatically generated

A diagram of a level

Description automatically generated

**Output:**

A blue background with white dots

Description automatically generated

**Main Objects/Structures in LinkedIn**

* **LinkedIn User Profile :** Takes username and skills and sets it to an index for ease of access.

**A computer screen shot of text

Description automatically generated**

**This is the constructor of our main class that is LinkedIn User. It takes the username and skills of user as arguments and sets them accordingly. It takes the array of skills and copies elements one by one.**

* **Messages:** Enter message content and timestamp.

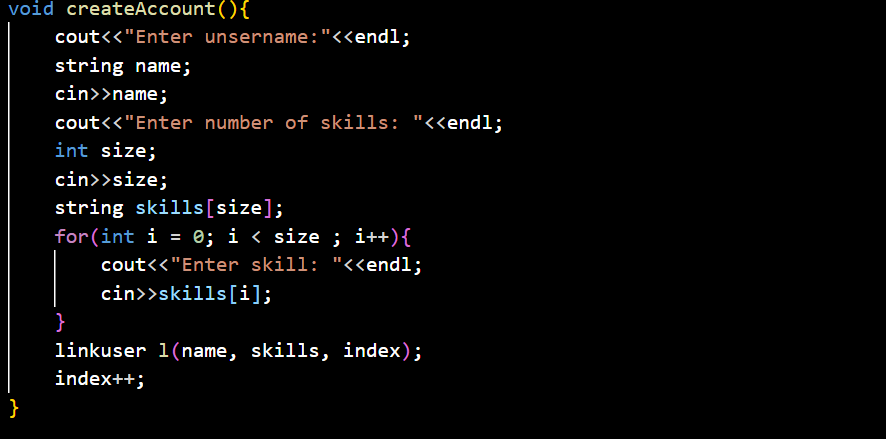
**A screen shot of a computer code

Description automatically generated**

***Messages Heap per User:***

**In-Depth View of Functions:**

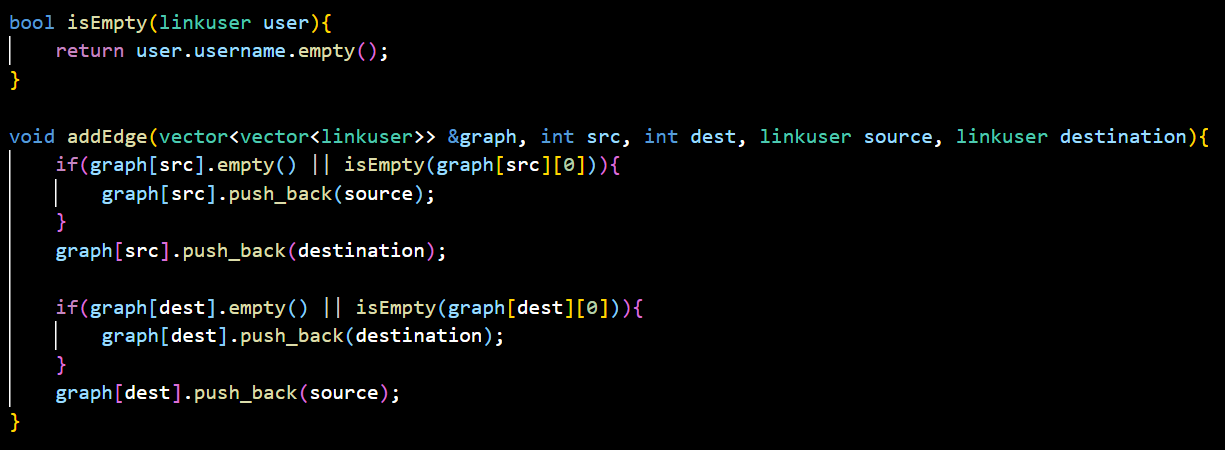
* **createAccount:** Make new user profile.

****

* **BFS:** View followers by degree.



* **addEdge:** Connect with users on LinkedIn.



* **sendMsg:** Send messages to connections.

A computer screen with text on it

Description automatically generated

This function takes a message and source index and destination index as arguments and stores the same message to their respective owners’ messages heap.

A computer code on a black background

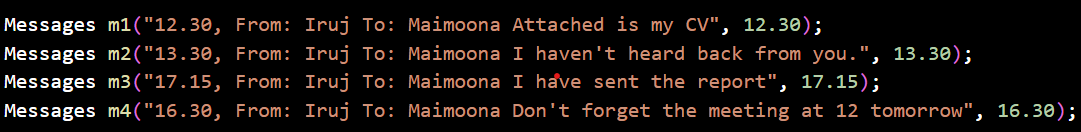
Description automatically generated

This function adjusts during insertion the messages in the respective owner’s heap prioritizing max timestamp message to be on top.

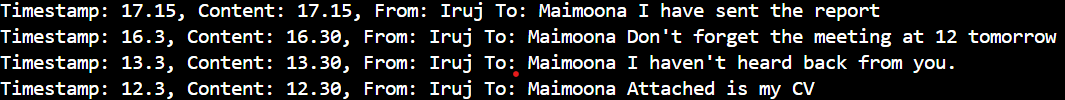
* **viewMsgs:** View messages from domain of a particular user:



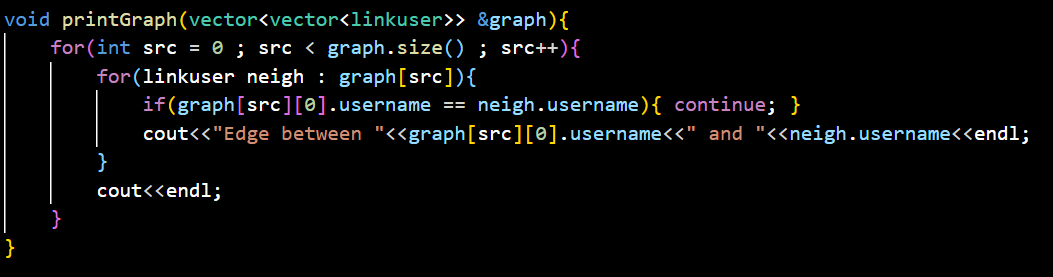
**Messages sent in terminal:**



**Output: Retrieved on the basis of timestamp:**



* **printGraph:** See how users are connected across the network.



* **searchUser:** Search a user and view their profile.

