Royal University of Phnom Penh

Faculty of Engineering

Telecommunication and Electronics Engineering

Social Network

Taught by: Prof. Chhoeum Vantha

Presented by:

Heng Lymeng

Khav Srey Khim

Sai Pechvimeane

Ra Valenthina

Chhay Rinlengsenghong

Contents

I. Introduction

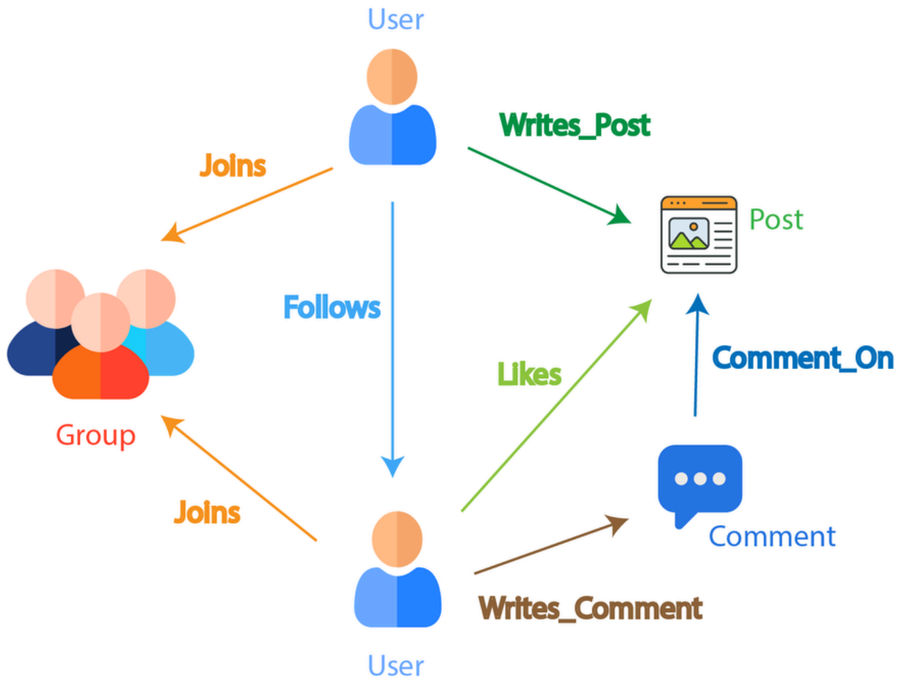
Presented by: Sai Pechvimeane

1.1 Overview and Definition

- Social network modeled as an undirected weighted graph
- Users as nodes, friendships as edges with weights
- Simulates connections and interactions in a social environment

1.2 Importance and Applications

- Social media platforms and communication networks rely on graph models
- Understanding user connectivity improves recommendations and messaging
- Weighted relationships represent strength or closeness between users



1.3 Motivation for the Project

- Explore graph algorithms in a real-world inspired context
- Build a simple social network system with messaging and friendship management
- Learn practical use of data structures and algorithms

1.4 Problem Statement

- Social networks involve complex, weighted relationships between users.
- Without features like shortest path, BFS, and DFS, exploring connections is difficult.
- A solution is needed that models the network as a weighted undirected graph and supports analysis.

1.5 Objectives of the Project

- Design user and network classes to manage users and friendships
- Support weighted undirected friendships
- Implement Dijkstra, BFS, DFS algorithms for graph operations
- Provide a messaging system respecting friendship constraints

1.6 Project Scope

- Focus on backend logic using Python
- No UI, command-line output-based interaction
- Small-sized networks (TEED Gen10) for demonstration

II. Methodology

Presented by: Chhay Rinlengsenghong

2.1 Tools and Technologies

- **Python 3:** The language used to build the social network, supporting modern features and efficient data structures for graph algorithms.
- **Built-in heapq for priority queue (Dijkstra):** Used in your shortest path feature to always pick the next user (vertex) with the smallest total friendship weight, making Dijkstra's algorithm efficient.
- **Standard data structures**

  _ **dictionaries(dict**): Stores users (self.users) and the friendship network (self.graph)

  _ **lists:** Holds ordered data like BFS/DFS queues and message inboxes.

  _ **sets:** Tracks visited users during BFS/DFS to prevent revisiting the same person.

2.2 Modeling (Graph)

- **Users represented as nodes (Vertices)**: Each user in the social network is a node in the graph. Example: "Lymeng", "Thina", "Khim" are nodes.
- Friendships represented as weighted undirected edges:

_ **Weighted:** The number on the edge represents the “strength” or “closeness” of the friendship (lower weight could mean stronger connection).

_ **Undirected:** Friendship is mutual; if A is friends with B, then B is also friends with A.

- **Data structure**: dictionary of dictionaries {user: {friend: weight}}:

_"Lymeng": {"Khim": 4, "Vin": 5}

_ "Vin": {"Lymeng": 5, "Khim": 4}

2.3 Algorithms Implemented

User Operation:

- **add_user:** adding a user to the Social Group.
- **get_user_info:** getting user information (bio in our case)
- **are_friends:** Check if they are friend or not.
- **remove_user:** Remove a user from Social Group.
- **remove_friendship:** End the connection between two users.
- **add_friendship:** add a connection between two users with a weight indicating the closeness between both users (set to default as 1).
- **search_user:** Searching for a user on the Social Group.

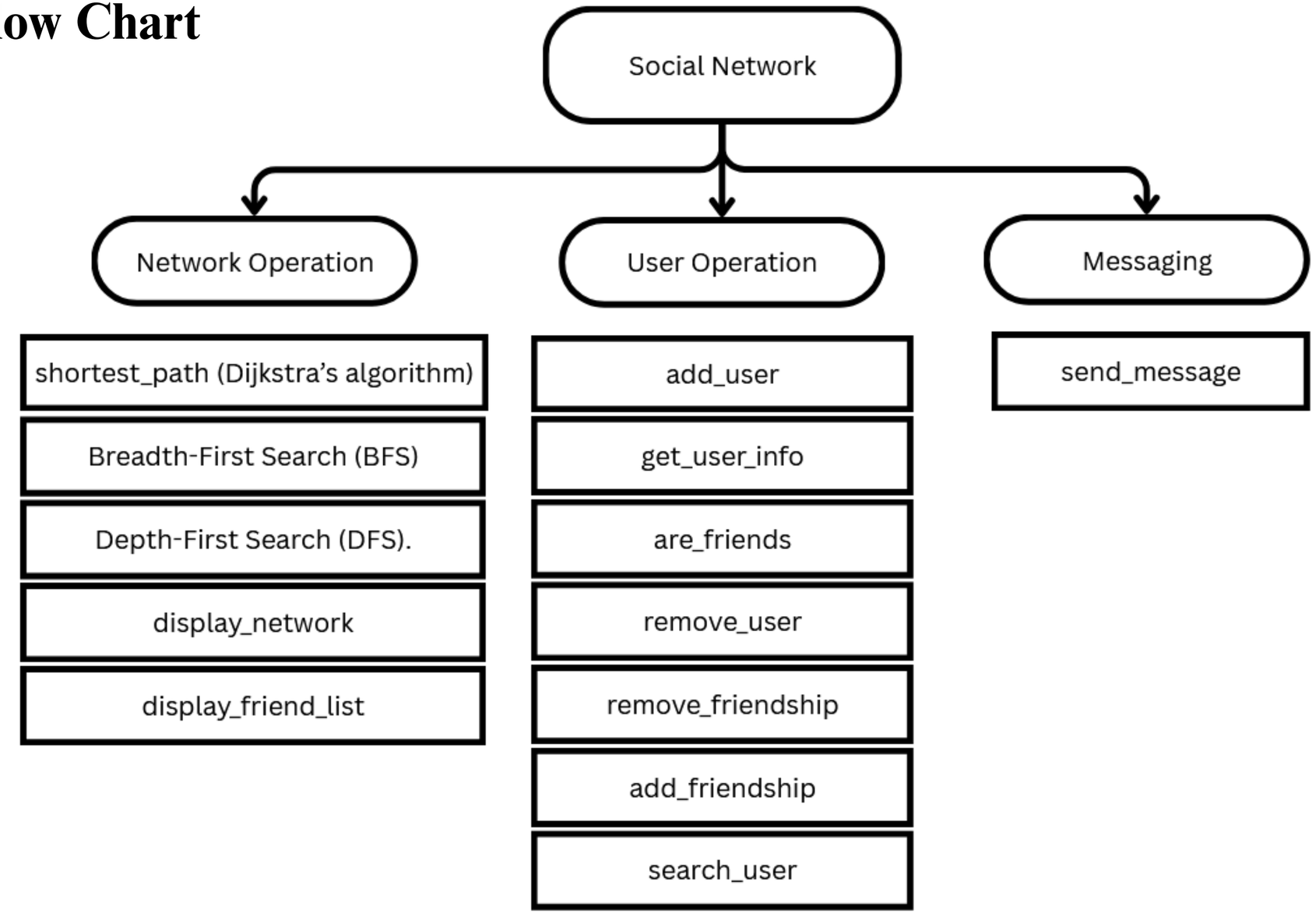
Network Operation:

- **shortest_path:** Using **Dijkstra's algorithm** to compute the shortest weighted path between users in the network, optimizing for minimum total edge weight.
- **Breadth-First Search (BFS):** Traverses the network level by level, exploring all users at the current distance before moving deeper.
- **Depth-First Search (DFS):** Explores the network deeply, following one path to its end before backtracking to explore others.
- **display_network:** to display the whole social network topology.
- **display_friend_list:** to display each user's list of friends with weights.

Messaging:

- **send_message:** Friendship Checks and Auto-Creating validates if users are connected (Weighted undirected graph) before messaging; automatically creates friendships when users send messages if none exist.

2.4 Flow Chart



Flow Chart of Processing

2.5 Simulation Scenarios

- Adding users and friendships with varied weights: Create a social network by adding users and connecting them with friendships, assigning varied weights to edges.

```python
# Example Usage
if __name__ == "__main__":
    TEED_GEN10 = SocialNetwork()

    # Adding users
    print("==================== Add user ====================")
    Add_user = [
        ("Lymeng", "Hello I'm Lymeng!"), ("Thina", "Hello I'm Thina!"),
        ("Khim", "Hello I'm Khim!"), ("Vimean", "Hello I'm Vimean!"),
        ("Hong", "Hello I'm Hong!"), ("Srun", "Hello I'm Srun!"),
        ("Vin", "Hello I'm Vin!"), ("Lida", "Hello I'm Lida!"),
        ("Chamrong", "Hello I'm Chamrong!"), ("Sambat", "Hello I'm Sambat!"),
        ("Lita", "Hello I'm Lita!"), ("Chinmi", "Hello I'm Chinmi!"),
        ("Nika", "Hello I'm Nika!"), ("Keam", "Hello I'm keam!"),
        ("Nut", "Hello I'm Nut!"), ("Leakena", "Hello I'm Leakena!"),
        ("Panhavorn", "Hello I'm Panhavorn!"), ("Raksa", "Hello I'm Raksa!"),
        ("Sokmeng", "Hello I'm Sokmeng")
    ]
    for username, bio in Add_user:
        TEED_GEN10.add_user(username, bio)
```

```python
# Adding friendships with default weight=1
print("\n================== Add friendships ================")
friendships = [
    ("Lymeng", "Khim", 4), ("Khim", "Vin", 4), ("Vimean", "Lida", 5),
    ("Chamrong", "Vin", 3), ("Chamrong", "Thina", 5), ("Chinmi", "Khim", 2),
    ("Thina", "Chinmi", 2), ("Khim", "Hong", 4), ("Hong", "Srun", 2),
    ("Srun", "Sambat", 2), ("Sambat", "Lita", 1), ("Chinmi", "Hong", 2),
    ("Vin", "Lymeng", 5), ("Lida", "Khim", 5), ("Lita", "Hong", 1),
    ("Lida", "Lymeng", 3), ("Hong", "Vimean", 1), ("Lymeng", "Thina", 5),
    ("Lida", "Hong", 3), ("Chamrong", "Hong", 2), ("Chinmi", "Vin", 4),
    ("Lymeng", "Chinmi", 3), ("Lymeng", "Chamrong", 3), ("Thina", "Hong", 4),
    ("Keam", "Nika", 1), ("Raksa", "Srun", 3), ("Sokmeng", "Chamrong", 4),
    ("Panhavorn", "Vimean", 3), ("Leakena", "Lida", 2), ("Sokmeng", "Nika", 2),
    ("Lymeng", "Nut", 2)
]
for user1, user2, weight in friendships:
    TEED_GEN10.add_friendship(user1, user2, weight)  # weight=1 default
```

- Finding shortest paths between the users: Use algorithms like Dijkstra's or BFS to find the shortest path (minimum total weight or hops) between two users in the network.
- Traversing the network with BFS and DFS: Implement Breadth-First Search (BFS) for level-order exploration and Depth-First Search (DFS) for deep exploration of the network.
- Sending messages between users with friendship validation: Enable message exchange between users, validating that a friendship exists before allowing communication.

III. Result and Analysis

Presented by: Ra Valenthina

III. Result and Analysis

- Successfully added 19 users and multiple weighted friendships

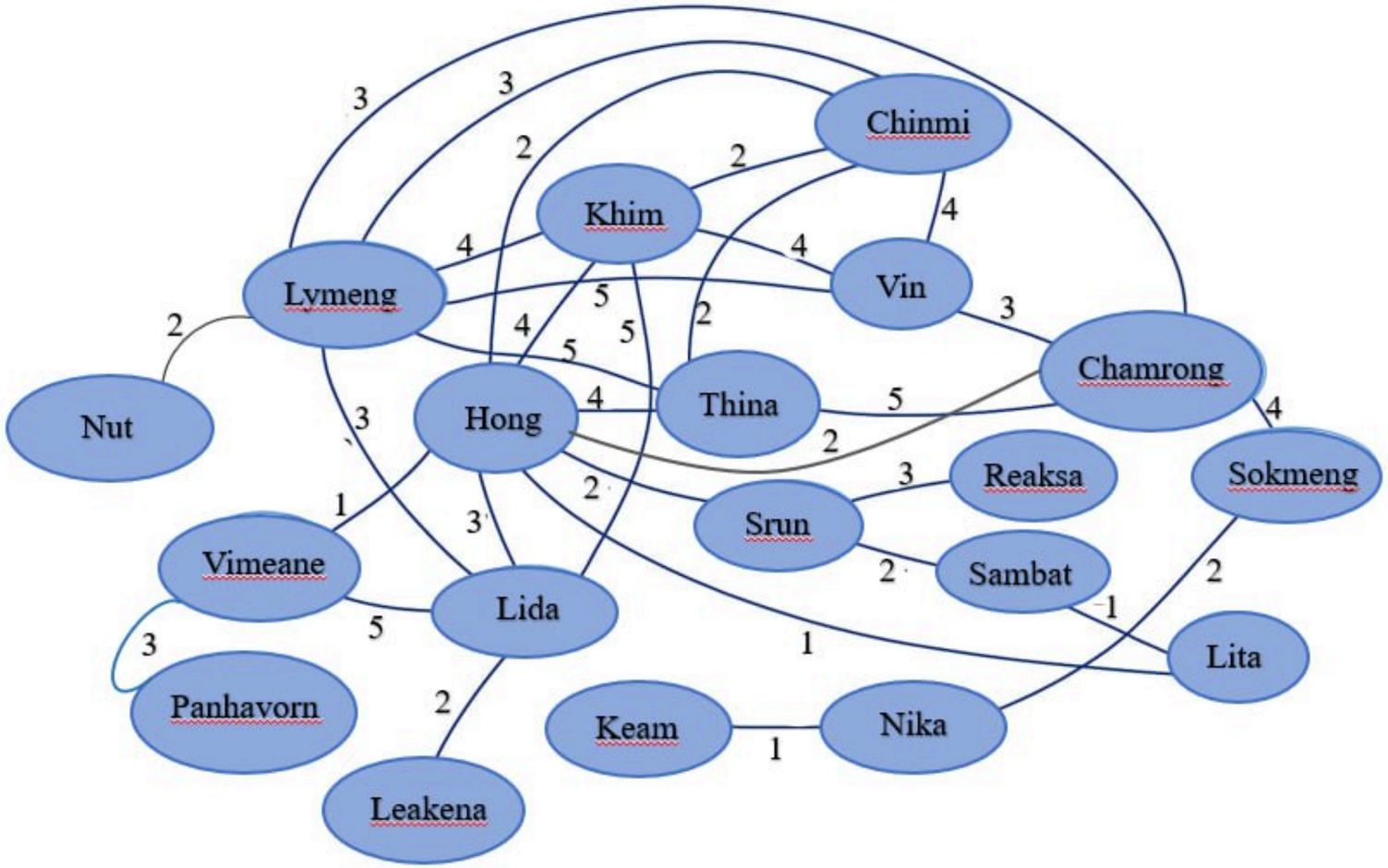
```
==================== Add user ====================
User Lymeng        --> added successfully.
User Thina         --> added successfully.
User Khim          --> added successfully.
User Vimean        --> added successfully.
User Hong          --> added successfully.
User Srun          --> added successfully.
User Vin           --> added successfully.
User Lida          --> added successfully.
User Chamrong      --> added successfully.
User Sambat        --> added successfully.
User Lita          --> added successfully.
User Chinmi        --> added successfully.
User Nika          --> added successfully.
User Keam          --> added successfully.
User Nut           --> added successfully.
User Leakena       --> added successfully.
User Panhavorn     --> added successfully.
User Raksa         --> added successfully.
User Sokmeng       --> added successfully.
```

```
================= Add friendships ================
Added connection: Lymeng       --(4)-- Khim
Added connection: Khim         --(4)-- Vin
Added connection: Vimean       --(5)-- Lida
Added connection: Chamrong     --(3)-- Vin
Added connection: Chamrong     --(5)-- Thina
Added connection: Chinmi       --(2)-- Khim
Added connection: Thina        --(2)-- Chinmi
Added connection: Khim         --(4)-- Hong
Added connection: Hong         --(2)-- Srun
Added connection: Srun         --(2)-- Sambat
Added connection: Sambat       --(1)-- Lita
Added connection: Chinmi       --(2)-- Hong
Added connection: Vin          --(5)-- Lymeng
Added connection: Lida         --(5)-- Khim
Added connection: Lita         --(1)-- Hong
Added connection: Lida         --(3)-- Lymeng
Added connection: Hong         --(1)-- Vimean
Added connection: Lymeng       --(5)-- Thina
Added connection: Lida         --(3)-- Hong
Added connection: Chamrong     --(2)-- Hong
Added connection: Chinmi       --(4)-- Vin
Added connection: Lymeng       --(3)-- Chinmi
Added connection: Lymeng       --(3)-- Chamrong
Added connection: Thina        --(4)-- Hong
Added connection: Keam         --(1)-- Nika
Added connection: Raksa        --(3)-- Srun
Added connection: Sokmeng      --(4)-- Chamrong
Added connection: Panhavorn    --(3)-- Vimean
Added connection: Leakena      --(2)-- Lida
Added connection: Sokmeng      --(2)-- Nika
Added connection: Lymeng       --(2)-- Nut
```

- Displayed network with weighted connections

```
================= Display network ================
        Social Network Connections:
Lymeng        follows --> Khim(4), Vin(5), Lida(3), Thina(5), Chinmi(3), Chamrong(3), Nut(2)
Thina         follows --> Chamrong(5), Chinmi(2), Lymeng(5), Hong(4)
Khim          follows --> Lymeng(4), Vin(4), Chinmi(2), Hong(4), Lida(5)
Vimean        follows --> Lida(5), Hong(1), Panhavorn(3)
Hong          follows --> Khim(4), Srun(2), Chinmi(2), Lita(1), Vimean(1), Lida(3), Chamrong(2), Thina(4)
Srun          follows --> Hong(2), Sambat(2), Raksa(3)
Vin           follows --> Khim(4), Chamrong(3), Lymeng(5), Chinmi(4)
Lida          follows --> Vimean(5), Khim(5), Lymeng(3), Hong(3), Leakena(2)
Chamrong      follows --> Vin(3), Thina(5), Hong(2), Lymeng(3), Sokmeng(4)
Sambat        follows --> Srun(2), Lita(1)
Lita          follows --> Sambat(1), Hong(1)
Chinmi        follows --> Khim(2), Thina(2), Hong(2), Vin(4), Lymeng(3)
Nika          follows --> Keam(1), Sokmeng(2)
Keam          follows --> Nika(1)
Nut           follows --> Lymeng(2)
Leakena       follows --> Lida(2)
Panhavorn     follows --> Vimean(3)
Raksa         follows --> Srun(3)
Sokmeng       follows --> Chamrong(4), Nika(2)
```

- Displayed network with weighted connections



III. Result and Analysis

- Found shortest paths reflecting minimum weighted distance from Meng to Srun

```
================== Shortest path ====================
Shortest path Lymeng --> Srun: ['Lymeng', 'Chamrong', 'Hong', 'Srun']
```

- BFS & DFS traversals demonstrated network reachability

```
==================== BFS & DFS ======================
BFS traversal: Lymeng Khim Vin Lida Thina Chinmi Chamrong Nut Hong Vimean Leakena Sokmeng Srun Lita Panhavorn Nika Sambat Raksa Keam
DFS traversal: Lida Vimean Hong Khim Lymeng Vin Chamrong Thina Chinmi Sokmeng Nika Keam Nut Srun Sambat Lita Raksa Panhavorn Leakena
```

- Searching for User (Hong & Panha)

```
=================== Search a user ==================
Hong          --> Hello I'm here!
Panha         --> Hello I'm not here!!
```

- Messaging enforced friend relationships, auto-created friendships where missing

```
==================== Send message ==================
No, Lida and Sambat are not friends.
Added connection: Lida         --(1)-- Sambat
Sambat       already connected to Lida.
Friendship automatically added between Lida and Sambat.
Message sent from Lida to Sambat: Hi, CEO handsome boys & beautiful girls
```

- Viewing User Info

```
================= User information ================
User: Lymeng, Bio: Hello I'm Lymeng!
Lymeng       follows --> Khim(4), Vin(5), Lida(3), Thina(5), Chinmi(3), Chamrong(3), Nut(2)
User: Thina, Bio: Hello I'm Thina!
Thina        follows --> Chamrong(5), Chinmi(2), Lymeng(5), Hong(4)
```

- User and friendship removals updated network correctly

```
================== Remove user ====================
User Thina --> removed successfully.

================ Remove friendship ================
Removed connection: Lymeng -- Khim
===================================================
```

- Displayed network with weighted connections after Deletion

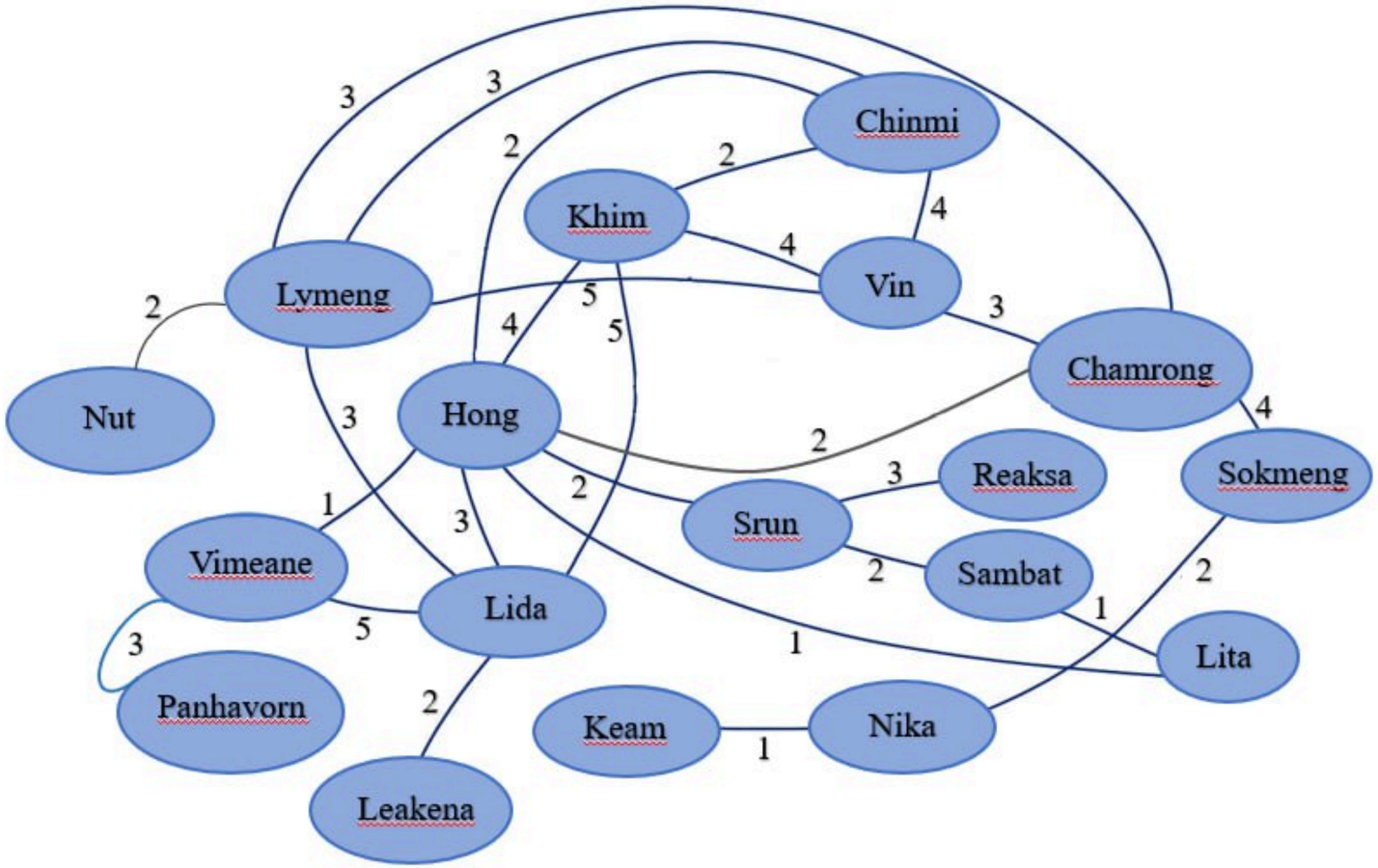
```
================ Display Network ===============
        Social Network Connections:
Lymeng       follows --> Vin(5), Lida(3), Chinmi(3), Chamrong(3), Nut(2)
Khim         follows --> Vin(4), Chinmi(2), Hong(4), Lida(5)
Vimean       follows --> Lida(5), Hong(1), Panhavorn(3)
Hong         follows --> Khim(4), Srun(2), Chinmi(2), Lita(1), Vimean(1), Lida(3), Chamrong(2)
Srun         follows --> Hong(2), Sambat(2), Raksa(3)
Vin          follows --> Khim(4), Chamrong(3), Lymeng(5), Chinmi(4)
Lida         follows --> Vimean(5), Khim(5), Lymeng(3), Hong(3), Leakena(2), Sambat(1)
Chamrong     follows --> Vin(3), Hong(2), Lymeng(3), Sokmeng(4)
Sambat       follows --> Srun(2), Lita(1), Lida(1)
Lita         follows --> Sambat(1), Hong(1)
Chinmi       follows --> Khim(2), Hong(2), Vin(4), Lymeng(3)
Nika         follows --> Keam(1), Sokmeng(2)
Keam         follows --> Nika(1)
Nut          follows --> Lymeng(2)
Leakena      follows --> Lida(2)
Panhavorn    follows --> Vimean(3)
Raksa        follows --> Srun(3)
Sokmeng      follows --> Chamrong(4), Nika(2)

================================================
```

- Comparison

```
Social Network Connections:
Lymeng      follows --> Vin(5), Lida(3), Chinmi(3), Chamrong(3), Nut(2)
Khim        follows --> Vin(4), Chinmi(2), Hong(4), Lida(5)
Vimean      follows --> Lida(5), Hong(1), Panhavorn(3)
Hong        follows --> Khim(4), Srun(2), Chinmi(2), Lita(1), Vimean(1), Lida(3), Chamrong(2)
Srun        follows --> Hong(2), Sambat(2), Raksa(3)
Vin         follows --> Khim(4), Chamrong(3), Lymeng(5), Chinmi(4)
Lida        follows --> Vimean(5), Khim(5), Lymeng(3), Hong(3), Leakena(2), Sambat(1)
Chamrong    follows --> Vin(3), Hong(2), Lymeng(3), Sokmeng(4)
Sambat      follows --> Srun(2), Lita(1), Lida(1)
Lita        follows --> Sambat(1), Hong(1)
Chinmi      follows --> Khim(2), Hong(2), Vin(4), Lymeng(3)
Nika        follows --> Keam(1), Sokmeng(2)
Keam        follows --> Nika(1)
Nut         follows --> Lymeng(2)
Leakena     follows --> Lida(2)
Panhavorn   follows --> Vimean(3)
Raksa       follows --> Srun(3)
Sokmeng     follows --> Chamrong(4), Nika(2)
```

```
Social Network Connections:
Lymeng      follows --> Khim(4), Vin(5), Lida(3), Thina(5), Chinmi(3), Chamrong(3), Nut(2)
Thina       follows --> Chamrong(5), Chinmi(2), Lymeng(5), Hong(4)
Khim        follows --> Lymeng(4), Vin(4), Chinmi(2), Hong(4), Lida(5)
Vimean      follows --> Lida(5), Hong(1), Panhavorn(3)
Hong        follows --> Khim(4), Srun(2), Chinmi(2), Lita(1), Vimean(1), Lida(3), Chamrong(2), Thina(4)
Srun        follows --> Hong(2), Sambat(2), Raksa(3)
Vin         follows --> Khim(4), Chamrong(3), Lymeng(5), Chinmi(4)
Lida        follows --> Vimean(5), Khim(5), Lymeng(3), Hong(3), Leakena(2)
Chamrong    follows --> Vin(3), Thina(5), Hong(2), Lymeng(3), Sokmeng(4)
Sambat      follows --> Srun(2), Lita(1)
Lita        follows --> Sambat(1), Hong(1)
Chinmi      follows --> Khim(2), Thina(2), Hong(2), Vin(4), Lymeng(3)
Nika        follows --> Keam(1), Sokmeng(2)
Keam        follows --> Nika(1)
Nut         follows --> Lymeng(2)
Leakena     follows --> Lida(2)
Panhavorn   follows --> Vimean(3)
Raksa       follows --> Srun(3)
Sokmeng     follows --> Chamrong(4), Nika(2)
```

- Displayed network with weighted connections after Deletion



IV. Conclusion

Presented by: Ra Valenthina

IV. Conclusion

- Social network graph effectively modeled with weighted undirected graph
- Dijkstra, BFS, DFS algorithms provided core analysis and traversal features
- Messaging system integrated well with friendship constraints
- System demonstrates basic social network functionality with efficient data structures

Future Work

- Scale to larger networks with performance optimizations
- Add features like friend recommendations and mutual friends detection
- Develop GUI or web interface for user interaction
- Support different message types and privacy settings

THANK YOU FOR YOUR ATTENTION!

Questions and comments?