Programming Assignment #3

This project is designed to model a social network as a weighted directed graph. Each node represents a student in an alternate fictional universe, and directed edges represent social connections. The edges are weighed by the number of days a student waits before introducing someone in their network. The project demonstrates graph-related algorithms and operations, including building the graph from CSV files and enabling dynamic modifications.

Features

The system provides the following functionality through a command-line interface:

1. **User Operations**:
    a. **Print Network List for a Student**: Displays the students directly connected to the selected student.
    b. **Find Quickest Path Between Two Students**:
        i. Computes the shortest path (in terms of days) from one student to another using **Dijkstra's Algorithm**.
        ii. Outputs both the total days required, and the number of nodes traversed.
    c. **Disconnect Two Students**:
        i. Removes a connection (edge) between a student and another.
    d. **Set Wait Days**:
        i. Set the wait time (edge weight) for a student's connections.
    e. **Print The Full List of Students:**
        i. Shows all of the students in the graph and prints their information (wait time, network, etc.)
2. **Dynamic Updates**:
    a. The graph updates dynamically after any of the modifications (operations 3, 4, or 5).
    b. Users can repeat operations on the updated graph until they choose to exit.

How to Run

Prerequisites

- **Java Development Kit (JDK)** version 8 or higher.
- A terminal or an IDE for compiling and running Java programs.

How to use

You will be given a command list to complete operations on the graph. Options include:

Operations Menu

1. **Print Network List for a Student**:
    a. Enter the enrollment number of the student to view their direct network.
2. **Find Quickest Path Between Two Students**:
    a. Enter the enrollment numbers of the source and target students.
    b. Outputs:
        i. The shortest path from one student to another (shown using each student's corresponding enrollment number)
        ii. Total time (in days) to navigate the path.
        iii. The number of nodes on the path.
3. **Disconnect Two Students**:
    a. Enter the enrollment numbers of the source and target students.
    b. Removes the edge from the source to the target.
4. **Set Wait Days**:
    a. Enter the enrollment number of the student.
    b. Specify the number of days to set their wait time to.
5. **Print A Full List of Students:**
    a. Doesn't need any extra information, just prints the list.
6. **Exit the Program**:
    a. Ends the program.

Graph Operations Overview

Data Structure

- **Graph**: Modeled as a weighted, directed graph.
- **Nodes**: Represent students.
- **Edges**: Represent connections, with weights corresponding to wait days.

Algorithms Used

- **Dijkstra's Algorithm**:
    - Computes the shortest path from one node to another based on edge weights (wait days).

Files Structure

1. **`SocialNetwork.java`**:
    a. Implements the graph data structure and supports graph-related operations.
    b. Handles building the graph, modifying edges, and computing shortest paths.
2. **`Introduction.java`**:
    a. Provides the user interface, allowing users to interact with the program and execute operations.

Authors

- Khalil El-abbassi -kelabb1@lsu.edu (Tester)
- Benjamin Gutowski -bgutow1@lsu.edu (Lead Programmer)
- Robert Ngo -rngo4@lsu.edu (Manager)
- Timothy Posley -tposle1@lse.edu (Documentation Writer)