CSE225-CSE2025 Data Structures PROJECT #2
Graph Implementation - Dijkstra's Algorithm
(Due : 12.06.2021 23:59)

In this project, you are expected to develop a program in C programming language to simulate a social network using a graph. The goal of this project is to implement a min Heap and Dijkstra's algorithm for computing the shortest paths in the graph which represents a social network.

First, you will build a graph from a given input file. In the input file, edges between nodes and weight of each edge are provided. Vertices of the graph represent the social network users (i.e., user A, user B, user C, user D, user E). In addition, edges of the graph and their corresponding weights represent the relations between users and intensity of the relations. You can use either adjacency matrix or adjacency list representation to store the vertices and weights.

After building the graph, you should find and print the shortest path between two nodes which are given as input parameter using Dijkstra's algorithm. An example input file and corresponding graph is shown in Figure 1. In Figure 1, first line of the input file has the format "A, B, 3" and this means that user A and user B are connected users and the intensity of this connection is equal to 3. Similarly, line "A, C, 5" indicates that user A and user B are connected users and the intensity is equal to 5.

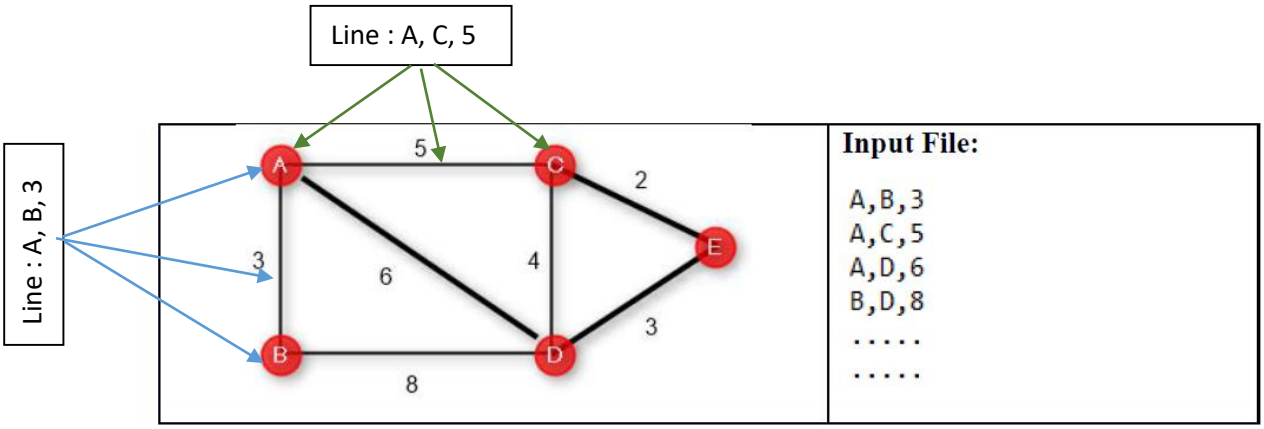

Figure 1. Sample input file and corresponding graph

Implementation Details:

1. Name of the input file will be taken as an input from the user.
2. You will show a menu to user.
   - In this menu, name of the input file to be read asked to the user. When the user enters name of the input file, the content of the file will be read and adjacency matrix/adjacency list will be created accordingly. Therefore, you will write a **function to read the input file**.
   - The user can see the content of adjacency matrix/list that is constructed according to the input file. As a result, you will write a **function to print the content of the adjacency matrix/list**.
   - Moreover, the user can select to use Dijkstra's algorithm to calculate the shortest path between two vertices (between source and destination vertices). Consequently, you will write a **function to implement Dijkstra's algorithm** to find the shortest path between two vertices. The source and destination vertices will be the parameters of the function.

- The menu will appear again and again until the user selects the exit option.

Menu will show 4 options to the user:

1. **Read file:** if the user enters 1, the name of the input file will be taken from the user and content of it will be read.
    - **Name of the input file:** input1.txt (i.e., here the user selects reading a file by entering 1 and name of the input file is asked to him/her. Then, he/she enters the name of the file which is input1.txt)
2. **Show adjacency matrix/list**: if the user enters 2, the content of the adjacency matrix/list will be printed to screen. As a result, you will write a **function to print the content of the adjacency matrix/list**. A sample output for the adjacency matrix is given below.

	A	**B**	**C**	**D**	**E**
A	**-**	**3**	**5**	**6**	**-**
B	**3**	**-**	**-**	**8**	**-**
C	**5**	**-**	**-**	**4**	**2**
D	**6**	**-**	**4**	**-**	**3**
E	**-**	**-**	**2**	**3**	**-**

If you are using adjacency list, a sample output can be similar to the one below.

A : B, 3 C, 5 D,6
B : ……
.
.
.

3. **Find shortest path:** if the user enters 3, then the Dijkstra's algorithm will be used to calculate the shortest path between the source and destination vertices. Source and destination vertices will be taken as input from the user.
    - **Enter the source vertex:** B
    - **Enter the destination vertex:** E (i.e., the user is asked to enter source and destination vertices and he/she enters vertex B as the source vertex and the vertex E as the destination vertex)

When the user enters the source and destination vertices the shortest path should be calculated between these nodes and **the path itself (including the vertices on the path) and its length should be printed.**

4. **Exit:** if the user enters 4 the program will terminate.

3. Comments are important.
    - Write your (and other group members') name, surname and student id as a comment in the beginning of the source code.
    - You should use meaningful function and variable names.

- You should write adequate and explanatory comments to explain implementation details, especially implementation details of the Dijkstra's algorithm, shortest path calculation, functionality of the functions and their parameters etc.

What to Submit:

1. Source code of your program. Write your (and other group members') name, surname and student id as a comment in the beginning of the source code (please do not use Turkish characters in the comment part). Name of the source file will be student ids of the group members (i.e., 1500001_1500012_1500007.c)

2. A Detailed report:

a. that explains the functions and their parameters

b. test each menu item and provide **screenshots** for execution of each of them (show both selection of them and output of them)

i. reading file option

ii. showing content of the adjacency matrix/list

iii. finding the shortest path.

c. If some of parts above do not work partially/at all, please indicate these cases.

d. Write complete and incomplete parts of the project.

Submission Rules:

1. You will submit the project through google classroom before the deadline. Late submissions will not be accepted and evaluated.
2. Please avoid plagiarism. Any attempt at plagiarism will result in strict disciplinary rules being enforced. The code should be totally your own work.
3. It is your responsibility to submit all files as requested. Please do not forget to submit your report as a pdf file.

Good luck!!!