Lab 6

Graph

In this lab session, we will implement some algorithms to solve various problems for graph.
A **weighted undirected graph** (with no negative edges) is represented by an adjacency matrix provided in the file `graph.txt`.
For example, the graph visualized as in Figure 1 would be represented as follows:

```
5
0 0 8 3 0
0 0 6 0 0
8 6 0 4 0
3 0 4 0 5
0 0 0 5 0
```

where:

- The first line of the file contains an integer n (the number of vertices in the graph).
- Each of the following n lines of the file contains n integers representing the adjacency matrix. The value at position (i, j) represents the weight of the edge between vertex i and vertex j. If there is no edge between these vertices, the value is 0.

1 Exercise 1: Dijkstra's Algorithm

Implement Dijkstra's algorithm to find the shortest path from a source vertex to all other vertices in the graph. The source vertex can be read from console. For example:

Input:

```
Enter source vertex: 0
```

Output:

```
The shortest path from 0 to 1: 0 -> 3 -> 2 -> 1.
The shortest path from 0 to 2: 0 -> 3 -> 2.
The shortest path from 0 to 3: 0 -> 3.
The shortest path from 0 to 4: 0 -> 3 -> 4.
```

2 Exercise 2: Bellman-Ford Algorithm

Similar to Exercise 1 above, implement the Bellman-Ford Algorithm to find the shortest path from a source vertex to all other vertices in the graph. The source vertex can be read from the console.

3 Exercise 3: Prim's Algorithm

Implement Prim's algorithm to find the Minimum Spanning Tree of the graph. For example:

```
Edge    Weight
0 - 3   3
1 - 2   6
2 - 3   4
3 - 4   5
```

4 Exercise 4: Kruskal's Algorithm

Similar to Exercise 3 above, implement Kruskal's algorithm to find the Minimum Spanning Tree of the graph.

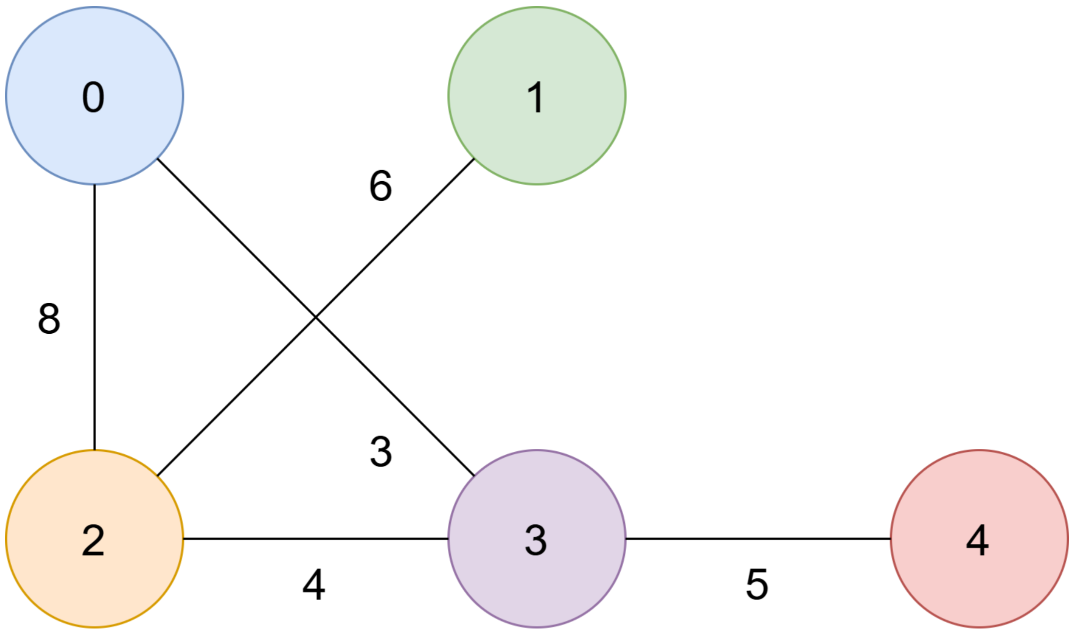

Figure 1: An example visualization of a graph with 5 vertices and 5 edges.

5 Submission

Your source code must be contributed in the form of a compressed file and named your submission according to the format `StudentID.zip`. Here is a detail of the directory organization:

```
StudentID
├── Exercise_1.cpp
├── Exercise_2.cpp
├── Exercise_3.cpp
└── Exercise_4.cpp
```

The end.