CO323 - Lab 07
Graph Algorithms in Networking: Dijkstra's Algorithm and Bellman-Ford Algorithm

Background:

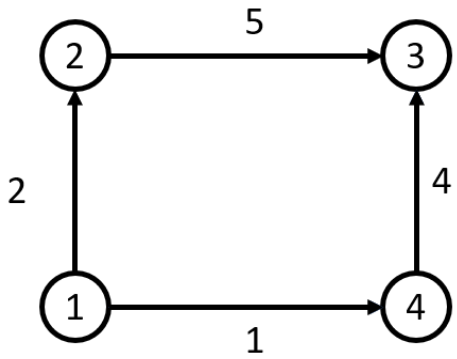
Networks inherently can be represented as graphs due to their nature. Hence, graph theory and related algorithms can be used directly to solve some problems in networking. For example, we can use graph algorithms to find the optimal paths between two network nodes so that the routing cost is minimum. We first need to assign each connection between nodes and represent it as a weighted graph. Then we can apply one of the shortest path-finding algorithms to find an optimum path.

There are multiple shorted path-finding algorithms. We will consider Dijkstra's algorithm and Bellman-Ford's algorithm for this lab.

1) Dijkstra's Algorithm
   - A simple and efficient algorithm
   - Single source shortest path-finding algorithm
   - Works on both directed and undirected graphs
   - All edges **must have nonnegative weights**.
2) Bellman-Ford Algorithm
   - Slower than Dijkstra's algorithm
   - Single source shortest path-finding algorithm
   - Can **work with negative weights**.

Input and Output Format:

Consider the graph shown below.



Running the scripts

```
$ python3 dijkstras_algorithm.py < input1.txt
```

The format of the input.txt is as follows:

```
rows cols
Adjacency_Matrix
```

e.g:

```
4 4
0 2 inf 1
inf 0 5 inf
inf inf 0 inf
inf inf 4 0
```

It should output this kind of output when we run Dijkstra's algorithm.

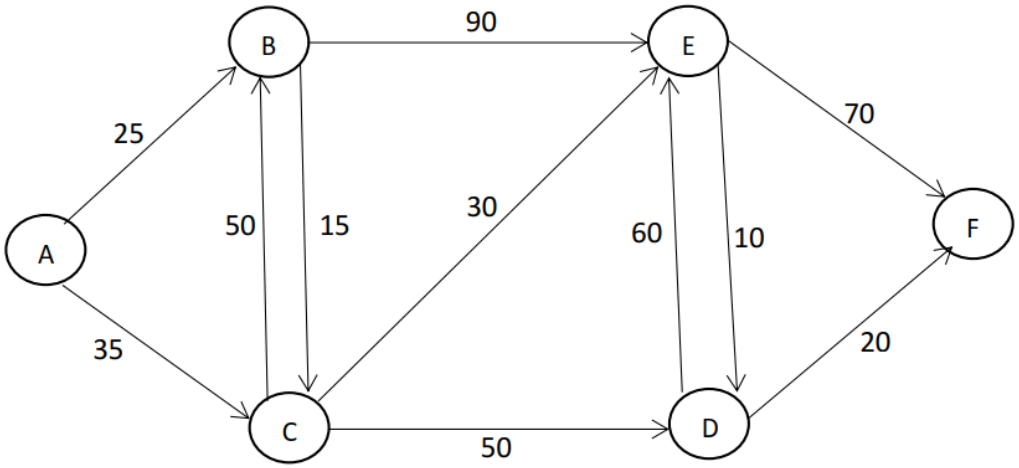
```
Input Adjacency Matrix size: 4x4
Input Adjacency Matrix:
[0.0, 2.0, inf, 1.0]
[inf, 0.0, 5.0, inf]
[inf, inf, 0.0, inf]
[inf, inf, 4.0, 0.0]

Dijkstra's Algorithm Output:
Shortest Distance from 1 to 2: 2
Shortest Distance from 1 to 3: 5
Shortest Distance from 1 to 4: 1
```

There should be a similar output for Bellman-Ford Algorithm.

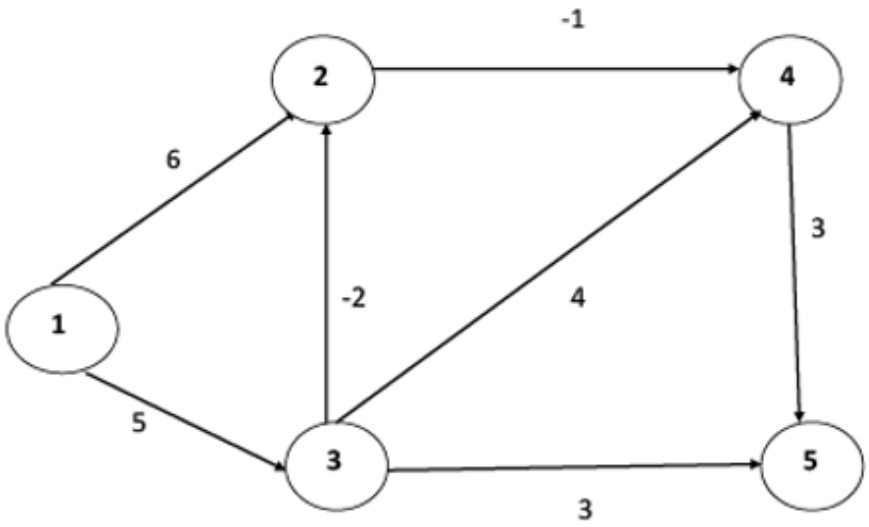
Exercises:

1. Consider the following graph



   a. Represent the above network using an Adjacency Matrix.
   b. Considering node A as the source node, implement Dijkstra's algorithm using Python to find the shortest distances from node 1 to all other nodes for a given Adjacency matrix.

2. Consider the following graph



   a. Represent the above network using an Adjacency Matrix.

b. Considering node 1 as the source node, implement the Bellman-Ford algorithm using Python to find the shortest distances from node 1 to all other nodes for a given Adjacency matrix

Create a small report renamed as E17XXX_report.pdf (XXX is your E Number) including the explanations for your approach (explain the algorithms/pseudocodes etc.), and graphs representation used.

Submit a zip file **E17XXX_Lab07.zip** (XXX is your E Number) which contains the following.

- **E17XXX_report.pdf**
- **dijkstras_algorithm.py**
- **bellman_ford_algorithm.py**
- **input1.txt**