Assignment 3:

Due Date: March 27th, 11:59 p.m.

Individual assignment

No Group work or group submission

Objectives:

- Implementing a Link State routing algorithm for an autonomous system in python

Description

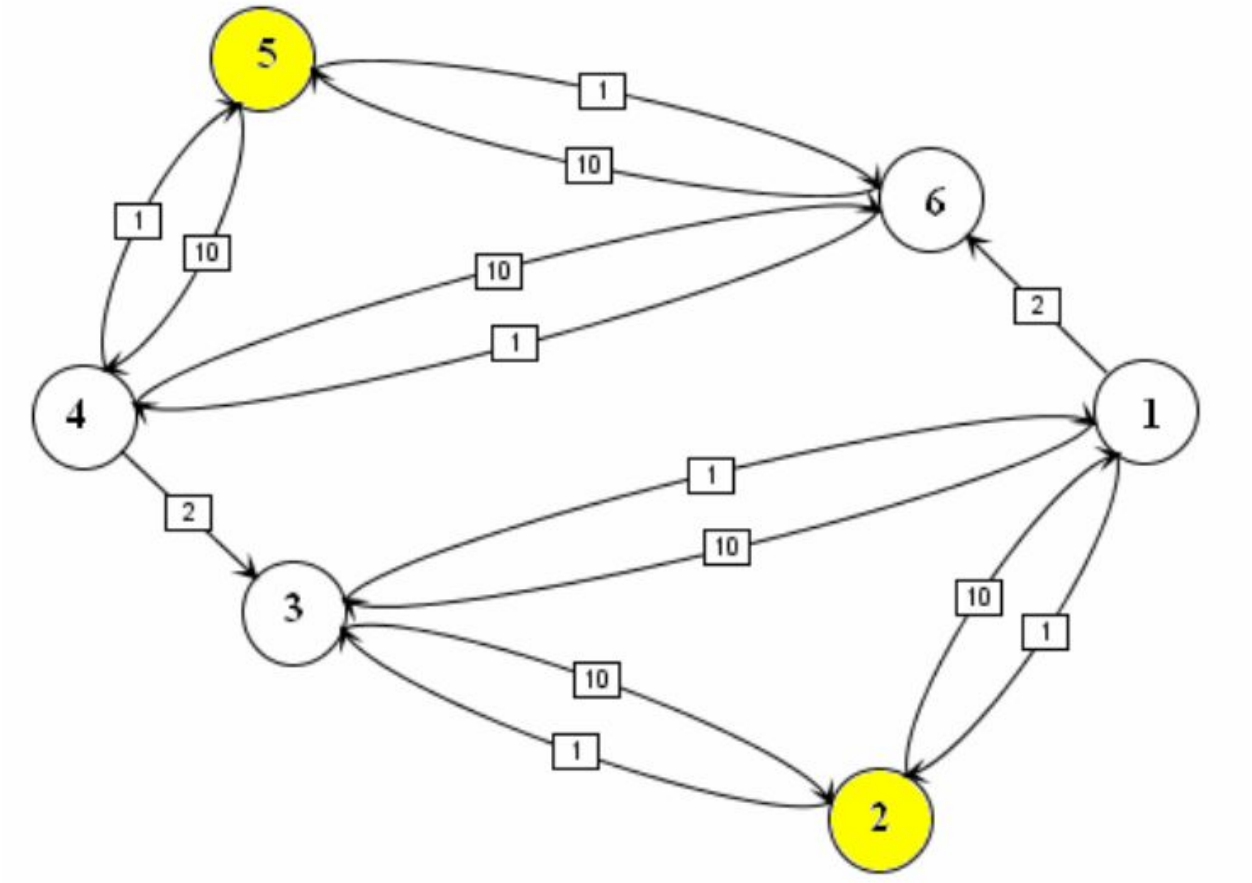
In this assignment you will implement a core functionality of the Link State routing algorithm for an autonomous system. The input to the algorithm is a weighted directed graph with **n** vertices (*labeled with 1, 2, 3,..., n*) that represents topology of a particular autonomous system. Each vertice represents a router or a gateway router in the network. The gateway routers are specified in the input as well. The output should print the forwarding table for each router other than the gateway routers. Each entry in a forwarding table shows the next hop node along the shortest path route from that node (source) to a gateway router (destination).

Input

The first line in the input represents the number of vertices, n. It is followed by n lines representing the adjacency matrix of the graph with weights. Finally, there is one more line representing a list of **gateway routers** $x_1, \ldots, x_k$, where each x_i is a distinct number from the set $\{1,2,\ldots,n\}$. Example input:

```
 6
 0   1  10  -1  -1   2
 10  0   1  -1  -1  -1
 1  10   0  -1  -1  -1
-1  -1   2   0   1  10
-1  -1  -1  10   0   1
-1  -1  -1   1  10   0
 2 5
```

The above input represents the following graph with vertices 2 and 5 being gateway routers.



Output

The output consists of `n-k` forwarding tables (the forwarding tables for routers but not the gateway routers). The output for the example input above should consist of 4 forwarding tables – each containing 2 entries that correspond to least-cost path to gateway routers 2 and 5, as below:

```
Forwarding Table for 1
        To        Cost    Next Hop
         2           1           2
         5           4           6

Forwarding Table for 3
        To        Cost    Next Hop
         2           2           1
         5           5           1
```

```
Forwarding Table for 4
        To       Cost   Next Hop
         2          4          3
         5          1          5

Forwarding Table for 6
        To       Cost   Next Hop
         2          5          4
         5          2          4
```

Note

- Implement Dijkstra algorithm that takes a graph and one source vertex, determines least-cost path to all other vertices from the source and forms the forwarding table at the source. Call this algorithm **`n-k`** times to produce desired results.
- do not assume that every vertex is reachable from any other vertex (imagine graph with 2 connected components for example); if a particular destination is not reachable from the source print -1 as the “Cost” and -1 as the “Next Hop” destination
- you can assume correct input (no exceptions handling)
- Dijkstra algorithm allows one to compute not only single source shortest path weights, but also predecessor for every node along the shortest path from the source, which is useful when constructing forwarding table at the source

Bonus

Implement alternative forwarding: if there is an alternative path of the same weight from the source to a particular destination, it might be useful to find it too. The entry in the forwarding table in this case might have two “next hop” elements (separated by comma with no space in between) which is helpful for the alternative forwarding.

Submission Instructions:

- Submit one file named **`LinkState.py.txt`**
- **Do NOT submit eclipse project!**
- **NOTE: Make sure to use python version 3.5 or higher. Your program will be tested with python3.5.**
- Make sure to put your names on top of each python program file
- You must not import any python module. You should implement it using the built-in functionality in python.
- **your implementation must read input data from the system input and print the tables to the system output!!**
  Test cases will be run with input redirection as follows

  ```
  Python LinkState < test1.txt
  ```

  Where `test1.txt` contains a particular input.

- **As autograding will be used for marking this assignment, to get the same output as mine, use the following format for printing a forwarding table: Off course you need to replace the values (such as 1, 2, 3, etc.) with variables, but follow the same formatting**:

```
print("Forwarding Table for 1")
print("{:>10} {:>10} {:>10}".format("To", "Cost", "Next Hop"))
print("{:>10} {:>10} {:>10}".format("2", "1", "2"))
print("{:>10} {:>10} {:>10}".format("5", "4", "6"))
print()
print("Forwarding Table for 3")
print("{:>10} {:>10} {:>10}".format("To", "Cost", "Next Hop"))
print("{:>10} {:>10} {:>10}".format("2", "2", "1"))
print("{:>10} {:>10} {:>10}".format("5", "5", "1"))
print()
```