Homework#3

Due date:

2020.05.14 23 : 59

Homework Policy:(Read before you start to work)

1. 作業請勿抄襲，如果被發現，作業以零分計算
2. 如果作業上遇到困難可以討論，但是報告和程式碼的部分必須是你自己完成，並且請在作業的 pdf 檔註明討論的同學姓名及學號
3. 程式作業請於期限內至 ceiba 作業區上傳，格式為 zip 檔，解壓縮後應恰為一個以學號 _hw3 為名的資料夾，資料夾內有一以學號為檔名的 pdf 檔和兩個子資料夾，裡面分別裝有你第一二小題撰寫的程式碼以及一個 README 檔 (需詳述 compile 與執行的方式)，如下所示。

```
B0XXXXXXX_hw3
|-- B0XXXXXXX.pdf
|-- src_1
    |-- (your part 1 program)
    |-- README.md
|-- src_2
    |-- (your part 2 program)
    |-- README.md
```

Figure 1: folder tree

4. 逾期繳交一天內，分數 $\times\frac{2}{3}$，超過一天未滿兩天，分數 $\times\frac{1}{3}$，超過兩天則不予計分，請務必盡早開始，並努力完成。
5. 如有任何問題歡迎來信，並請在郵件的標題註明課程。範例:[2020ICN] 作業三問題
    - 學號末號　mod 3 = 0 林芹學 R08921047@ntu.edu.tw
    - 學號末號　mod 3 = 1 林宛霓 R08921055@ntu.edu.tw
    - 學號末號　mod 3 = 2 連潔琳 R08942159@ntu.edu.tw

[**Objective**]

In this homework, you are asked to develop a simulator to implement Link-State Routing Protocol. Your program should have two functions:

- Simulate the process of generating routing table for each router in a given network.
- Compute optimal path with least cost between any two specific routers.

[**Problem description**]

Suppose we have a network with arbitrary number of routers. The network topology is given by a matrix, called the original topology (graph) matrix, which only indicates the costs of links between all directly connected routers. We assume each router only knows its own information and has no knowledge about others at the beginning.

In this homework, you have to implement Link-State Routing Protocol. First, your program is required to create the state of the links by each router after the input file containing the network information which has been loaded. Then, by reading the topology matrix file, a network graph can be determined. A Dijkstra' s algorithm could be applied to find shortest path between two entities: source and destination nodes. Finally, your program should be able to output the routing table of any router.

1. [**Link-State Routing**] – 70% = 56%(8% for each test case) + 14% report a(Programming 56%) Consider the network topology below, in which R1,R2,...R6 represent the routers, and there is a value on each link representing the link cost between all directly connected routers.
   (e.g., the link cost between routers 1 and 2 is 3.)

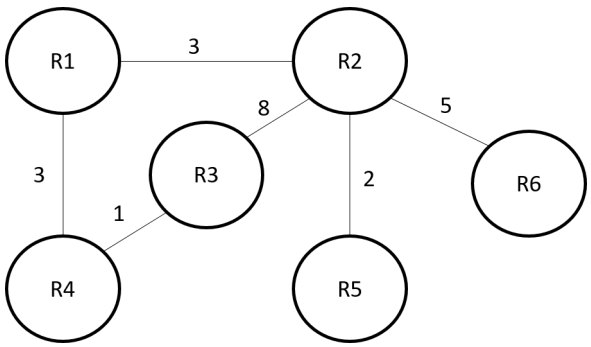

Figure 2: A sample network topology with costs on links

Based on the above network topology diagram which is an undirected graph, we set the cost between a router and itself to 0, the cost between two indirectly connected routers to -1. Value -1 is equal meaning of infinite value (∞) as link state routing algorithm discussed in the class. Then we have the original topology matrix shown as Table 1.

Table 1 will be given as an input in a text file form named input.txt. Please notice that this is only a sample table and your program is required to be able to process a network with arbitrary number of routers and costs.

Table 1: A sample original topology matrix

	R1	R2	R3	R4	R5	R6
R1	0	3	-1	3	-1	-1
R2	3	0	8	-1	2	5
R3	-1	8	0	1	-1	-1
R4	3	-1	1	0	-1	-1
R5	-1	2	-1	-1	0	-1
R6	-1	5	-1	-1	-1	0

- **Sample input**
  Take the network topology in Figure 2 as example, the format of input file input.txt will be a matrix as below. Row 1 represents the number of routers. Row 2 represents the direct link cost from router 1 to router 1-6, and row 3 represents the direct link cost from router 2 to router 1-6, etc. Hence, you should design your program that can accept input file of this format.
  6
  0 3 -1 3 -1 -1
  3 0 8 -1 2 5
  -1 8 0 1 -1 -1
  3 -1 1 0 -1 -1
  -1 2 -1 -1 0 -1
  -1 5 -1 -1 -1 0
- **Sample output**
  Your output should contain the routing table of all routers, including the cost(column 1) of the least-cost path from the source router to destination router, and the next router(column 2) along the least-cost path from the source.
  After running the program, you should produce a text file named [input file name]_out1.txt under the same directory. (e.g., if the input file is case1.txt, the output file should be case1_out1.txt.)
  Routing table of router 1:
  0 1
  3 2
  4 4
  3 4
  5 2
  8 2
  Routing table of router 2:
  3 1
  0 2
  7 3
  6 1
  2 5
  5 6
  .....

- **Notice!**
  Your program should be able to process the input matrix regardless of its size.

b(Report 16%)Your report should contain the following part:

- Detailed description of your part 1 program. (You may attach some important part of your code.)

2. [**Router removed**] – 30% = 24% + 6% report
   a(Programming 24%) If one of the routers been removed, the whole network topology will be changed. Hence, in this part, you will have to handle such condition and build a new routing table. For the router been removed, you should omit it while producing an output file. The output file should be named as [input file name]_out2.txt under the same directory. (e.g., if the input file is case1.txt, the output file should be case1_out2.txt.)
   e.g. router 2 has been removed:
   Routing table of router 1:
   0 1
   4 4
   3 4
   -1 -1
   -1 -1
   Routing table of router 3:
   4 4
   0 3
   1 4
   -1 -1
   -1 -1
   .....

   - **Notice!**
     If removing a router makes some routers cannot reach the other routers, then set both the cost and the router number in the routing table to -1.

   b(Report 6%)Your report should contain the following part:

   - Detailed description of your part 2 program. (You may attach some important part of your code.)

3. [**Program Testing**]
   a. Submission :
   You can write your code either with c/c++ or python, just make sure it can run, and submit the source code, a README file, and a report in the project package as mentioned above.
   b. Testing command :
   Your program should support the following user input commands, which marked by color red.The third command can just support in your part 2 program, while the first one and second one should support in both part 1 and 2.

- Load input file – lf [input file name].txt
- Produce an output file that contains the routing table of all routers(expect the one been removed) – of [output file name].txt
- Remove a router – rm r2 (e.g. remove router 2)

c. Testing rule :
(Part 1)TA will test your part 1 program by the two basic test cases provided to you and the other five test cases, each earns 8% of the score.
(Part 2)TA also tests your part 2 program by three test cases that a router been removed, and each earns 8% of the score as well.