Assignment 4: Routing Algorithms

Assignment goals

1. Enhance understanding of the purpose and functionality of routing algorithms.
2. Learn the differences between Link State and Distance Vector routing algorithms

Background

In computer networks, routing algorithms are essential for determining the path that data packets take through the network. Two primary routing algorithms are Link State routing and Distance Vector routing.

Link State Algorithm: Each router maintains a complete view of the network topology. Routers periodically broadcast their link state information to their neighbors. All routers use this information to build a global view of the network and compute the shortest paths using Dijkstra's algorithm.

Link-State (LS) Algorithm for Source Node u

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of u
5        then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for each neighbor v of w and not in N':
12         D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    least path cost to w plus cost from w to v */
15 until N'= N
```

Distance Vector Algorithm: Each router maintains a table that contains the distance to other routers and the next hop to reach them. Routers periodically exchange their distance vector tables with their neighbors. By continually updating these tables, routers eventually find the shortest paths to other routers.

Distance-Vector (DV) Algorithm

At each node, x:

```
1  Initialization:
2     for all destinations y in N:
3        D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
4     for each neighbor w
5        D_w(y) = ? for all destinations y in N
6     for each neighbor w
7        send distance vector D_x = [D_x(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       D_x(y) = min_v{c(x,v) + D_v(y)}
15
16    if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```

Task 1 – Link State Algorithm

Given a network topology G containing several routers, write a program to implement the Link State Routing Algorithm to compute the shortest path from router S to all other routers.

Input:

The first line contains two integers, N and M, representing the number of nodes (routers) and the number of edges (links) in the graph, respectively.

The next M lines each contain three integers x, y, and z, representing the cost of the link from router x to router y.

The M+2 line contains an integer S, representing the router from which the shortest distances to all other routers need to be calculated.

Output:

Output N lines, each line containing an integer. The i-th line represents the shortest path length from router S to router i.

Constraints:

N and M are not greater than 100, and $1 \leq x, y, z, S \leq N$.

Example Input:

4 5
1 2 6
1 3 4
1 4 1
2 4 4

3 4 2
1

Example Output:

0
5
3
1

Task 2 – Distance Vector Algorithm

Given a network topology G containing several routers, write a program to implement the Distance Vector Routing Algorithm to compute the shortest path from router S to router T.

Input:

The first line contains two integers, N and M, representing the number of nodes (routers) and the number of edges (links) in the graph, respectively.

The next M lines each contain three integers x, y, and z, representing the cost of the link from router x to router y.

The M+2 line contains two integers S, T.

Output:

An integer representing the shortest distance from router S to router T.

Constraints:

N and M are not greater than 100, and $1 \leq x, y, z, S, T \leq N$.

Example Input:

4 5
1 2 6
1 3 4
1 4 1
2 4 4
3 4 2
1 3

Example Output:

3

Submission

※ OS: Ubuntu/Linux/MacOs/Windows.

※ Programming language : C/C++/python3.

※ Submit your source code (**Files LS.* correspond to task 1, and files DV.* correspond to task 2**), along with a document (in md/pdf format) that includes screenshots of the program execution results for the example data, as well as instructions on how to compile and run the program. Compress all files into zip/rar/7z, and submit through Web Learning (网络学堂).

※ Note: During actual evaluation, testing will be conducted on data other than the provided samples.

Due date

June 3rd, 23:59 Beijing Time