Thời gian còn lại 0:06:59

Câu hỏi **5**

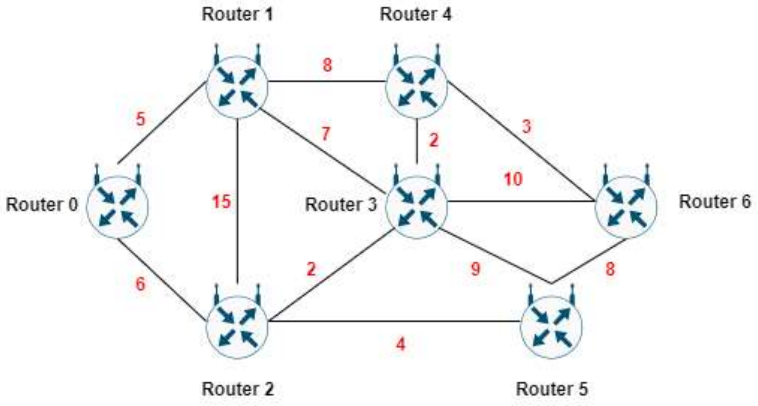
Không hoàn thành

Chấm điểm của 2,00

In computer networking, routing is the process of selecting paths on a computer network to send data over. Routing is performed for many types of networks, including telephone networks, inter-networks, Internet, and transport networks.

Routing indicates the direction, the movement of packets (data) addressed from their source network, towards the final destination through specialized hardware devices called **routers.** The routing process usually indicates the direction based on the routing table, which contains the best routes to different destinations on the network. Therefore, the construction of the routing table, which is organized in the router's memory, becomes extremely important for efficient routing.

Suppose the internet network is described as an undirected graph, with nodes being routers are connected to each other. To transmit a packet from router A to router B will cost w, corresponding to the weight of the edge connecting 2 vertices A and B. For example, we have the following internet network:



The cost of moving a packet from Router A → Router B is calculated as the sum of the weights of the edges joining the path. Assuming the packet needs to be sent from Router 0 → Router 6, it can go through several ways: 0 - 1 - 4 - 6, 0 - 1 - 3 - 6, 0 - 1 - 3 - 5 - 6, ... Where the path **0 - 2 - 3 - 4 - 6** for the lowest cost is **13** ((0, 2) = 6, (2, 3) = 2, (3, 4) = 2, (4, 6) = 3)

Class **Router** is used to store the information of a router, described on the following:

```
class Router
{
private:
    int size;
    list<pair<int, int>> adjList;
    vector<pair<int, int>> routingTable;

public:
    Router(){};
    Router(int R){};
    void pushEdge(int vertex, int weight)
    {
        adjList.push_back({vertex, weight});
        size++;
    }
    void pushRoute(int distance, int from)
    {
        routingTable.push_back({distance, from});
    }
    void printAdjacentRouter()
    {
        for (auto const &i : adjList)
        {
            cout << " -> " << i.first << " (weight: " << i.second << ")";
        }
    }
    void printRoutingTable()
    {
        for (size_t i = 0; i < routingTable.size(); i++)
        {
              cout << i << setw(15) << routingTable.at(i).first << setw(15) << routingTable.at(i).second
<< endl;

        }
    }
    int getSizeAdj() { return adjList.size(); }
    pair<int, int> getElementAdj(int idx)
    {
        {
            auto it = adjList.begin();
            advance(it, idx);
            return *it;
        }
    }
    pair<int, int> getElementRouting(int idx)
    {
        {
            auto it = routingTable.begin();
            advance(it, idx);
            return *it;
        }
    }
};
```

Where `adjList` is the list of connected router with weights (list<pair<int, int>>), `routingTable` is table store information of shortest path when transmitting packet (vector<pair<int, int>>)

Class **Network** is used to simulate network, described on the following:

```
class Network
{
private:
    int R;
    Router *router;

public:
    Network(int R)
    {
        this->R = R;
        this->router = new Router[R];
    }
    void addConnect(int u, int v, int w)
    {
        router[u].pushEdge(v, w);
        router[v].pushEdge(u, w);
    }
    void transferPacket(int src, int des)
    {
        vector<int> path;
        int cost = router[src].getElementRouting(des).first;
        while (true)
        {
            path.insert(path.begin(), des);
            des = router[src].getElementRouting(des).second;
            if (des == src)
            {
                break;
            }
        }
        cout << "Transfer path: " << src;
        for (auto const &i : path)
        {
            cout << " -> " << i;
        }
        cout << "\nTransfer cost: " << cost << "\n";
    }
    void printNetwork()
    {
        for (int r = 0; r < this->R; ++r)
        {
            cout << "\nAdjacency list of router " << r << "\nhead";
            router[r].printAdjacentRouter();
        }
    }
    void printRouting()
    {
        for (int r = 0; r < R; ++r)
        {
            cout << "\nRouting table of router " << r << endl;
            cout << "-----------------------------------" << endl;
            cout << "Router" << setw(15) << "Distance" << setw(15) << "Sent from" << endl;
            router[r].printRoutingTable();
        }
    }
    ~Network()
    {
        delete[] router;
    }
    void updateRouting();
};
```

Where `R` is number of router in the network (integer).

Create a function that update routing table to each router `(Hint: Dijkstra algorithm)`. Based on that table, we can find the shortest path from router A to router B and its value to transmit a packet.

Request: Implement function:

```
void Network::updateRouting()
```

Input: A weighted graph is represented by an adjacency list

Output: Routing table for each router

Note: In this exercise, the libraries `iostream`, `iomanip`, `stack`, `queue`, `map`, `vector`, `list`, `utility`, `climits`, `algorithm`, `string` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

Example explaination:

Routing table of Router 0

Router	Distance	Sent from
0	0	-1 (Started router)
1	5	0
2	6	0
3	8	2
4	10	3
5	10	2
6	13	4

We can find shortest path when transmit packer from Router 0 to Router 6:

- Packet transmit to Router 6 is sent from Router 4
- Packet transmit to Router 4 is sent from Router 3
- Packet transmit to Router 3 is sent from Router 2
- Packet transmit to Router 2 is sent from Router 0
- Router 0 is source -> Terminate

transmitPacket(0, 6) -> Transmit path: 0 - 2 - 3 - 4 - 6. Cost: 13

For example:

Test	Result

`int R = 7;`
`Network n(R);`
`n.addConnect(0, 1, 5);`
`n.addConnect(0, 2, 6);`
`n.addConnect(1, 2, 15);`
`n.addConnect(1, 3, 7);`
`n.addConnect(2, 3, 2);`
`n.addConnect(1, 4, 8);`
`n.addConnect(2, 5, 4);`
`n.addConnect(3, 4, 2);`
`n.addConnect(4, 6, 3);`
`n.addConnect(3, 6, 10);`
`n.addConnect(3, 5, 9);`
`n.addConnect(5, 6, 8);`
`n.updateRouting();`
`n.transmitPacket(0, 6);`
	`Transmit path: 0 -> 2 -> 3 -> 4 -> 6`
	`Transmit cost: 13`

Answer: (penalty regime: 0, 0, 0, 10, 20, ... %)

Reset answer

```
1  void Network::updateRouting()
2  {
3      // STUDENT'S ANSWER
4  }
```

Precheck Kiểm tra

◀ Testing

Chuyển tới...

Link google meet Final test (11/12/2021) ▶

Địa chỉ: Nhà A1- 268 Lý Thường Kiệt, Phường 14, Quận 10, Tp.HCM.
Email: elearning@hcmut.edu.vn