重庆邮电大学

学生实验实习报告册

学年学期：2023-2024学年□春■秋学期

课 程 名 称：通信软件基础

学 生 学 院：通信与信息工程学院

专 业 班 级：01042301

学 生 学 号：2023211281

学 生 姓 名：丁同勖

联 系 电 话：18019582857

重庆邮电大学教务处制

课程名称	通信软件基础	课程编号	A2010771
实验地点	YF317	实验时间	2024/11/25
校外指导教师		校内指导教师	王华华
实验名称	链路状态路由协议的系统仿真功能		
评阅人签字		成绩	

1. 功能描述

该程序模拟了一个小型路由网络（包含 6 个路由器），通过加载邻接矩阵表示的网络拓扑结构，实现以下功能：

a) **加载路由器间的连接数据：**

- 从文件中读取每个路由器的连接情况，并进行**洪泛**，构建邻接矩阵。
- 支持基本的错误处理，例如文件不存在或数据非法。

b) **计算最短路径：**

- 使用 **Dijkstra 算法**计算每个路由器的**路由表**（包括**目标路由的距离**和**下一跳**信息）。

c) **展示网络信息：**

- 输出路由器的 **IP 地址**及其**邻接矩阵**。
- 显示每个路由器的**路由表**。

d) **模拟路由间数据传输：**

- 根据计算出的路由表，模拟从一个路由器向另一个路由器传输数据，并**展示路径**。

2. 实验原理

该程序的实验原理基于**计算机网络中的路由与数据转发机制**，主要目标是模拟路由器之间的通信与数据转发路径的选择，程序实现的核心原理如下：

a) 网络拓扑构建

- **网络模型：**
  - 使用**图论模型**来模拟网络拓扑，路由器作为图的节点，路由器之间的链路作为图的边。
  - **权重**表示链路代价，如传输延迟、带宽消耗等。较小的权重代表链路质量更高或更优。
- **邻接矩阵：**
  - 用二维数组 adjMatrix[N][N] 表示网络，其中：

- adjMatrix[i][j] 是路由器 i 和 j 之间的链路代价。
- 如果两个路由器之间无直接连接，设置为无穷大 INF。

- **文件读取：**
  - 从每个路由器的配置文件（如 Router0.txt）中读取其直接连接的其他路由器及其权重，动态填充邻接矩阵，模拟网络环境的洪泛与初始化。

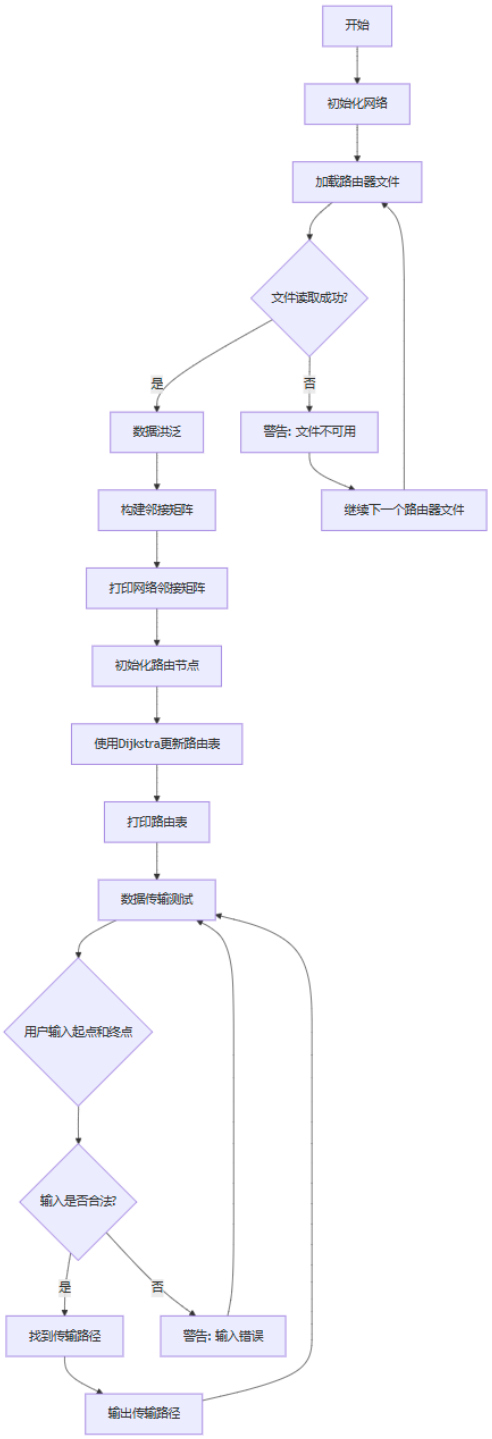
b) **最短路径计算**

- **路由表生成：**
  - 每个路由器需要维护一个**路由表**，记录：
    i. 到所有其他路由器的最短距离。
    ii. 到目标路由器的下一跳路由器。
- **Dijkstra 算法：**
  - 使用单源最短路径算法计算路由表，具体步骤：
    i. 将起点路由器到自身的距离设为 0，其他路由器距离设为 INF。
    ii. 通过优先队列选择当前距离最短的路由器，更新其邻居的距离。
    iii. 持续重复上述过程，直至所有路由器的最短路径被计算。
- **路径构造：**
  - 在计算最短路径时，通过维护前驱节点（predecessors），记录从源路由到目标路由的路径。

c) **路由与数据转发**

- **数据转发路径：**
  - 模拟从源路由器到目标路由器的数据转发。
  - 根据路由表逐跳（next hop）查找下一节点，直至到达目标。
- **转发机制：**
  - 如果目标可达，输出完整的传输路径。
  - 如果不可达（例如网络中断或未建立连接），输出警告。

3. 编程思路（软件流程图）



4. 代码

==========main.cpp==========

```
#include <iostream>
#include <vector>
#include <queue>
#include <fstream>
#include <sstream>
#include <climits>
#include <iomanip>
```

```
using namespace std;

const int INF = INT_MAX;
const int NODE_COUNT = 6; // 路由数量
const std::string ErrorMsg = "\033[31m[ERR]\033[0m\t";
const std::string InformationMsg = "\033[32m[INFO]\033[0m\t";
const std::string WarningMsg = "\033[33m[WARN]\033[0m\t";

void printIP(int index) {
    unsigned int Addr = 0xC0A80001 + index;
    cout << Addr / 256 / 256 / 256 << "." << Addr / 256 / 256 % 256 << "." << Addr / 256 % 256 <<
"." << Addr % 256;
}

class Network {
public:
    int adjMatrix[NODE_COUNT][NODE_COUNT];

    friend void printIP(int index);

    Network() {
        for (int i = 0; i < NODE_COUNT; ++i)
            for (int j = 0; j < NODE_COUNT; ++j)
                adjMatrix[i][j] = (i == j) ? 0 : INF; // 网络初始化，除了自身和自身的链接为 0 ，
其余均为最大值
    }

    void loadFromFiles() {

        int src;
        for (src = 0; src < NODE_COUNT; ++src) {
            string fileName = "Router" + to_string(src) + ".txt";
            ifstream file(fileName);

            if (!file) {
                cerr << WarningMsg <<"无法读取文件 " << fileName << endl;
                continue;
            }

            cout << InformationMsg << "Router " << src << " 数据洪泛开始" << endl;

            string line;
            while (getline(file, line)) {
                stringstream ss(line);
                int dest, weight;
                ss >> dest >> weight;
```

```
                if (src >= 0 && src < NODE_COUNT && dest >= 0 && dest < NODE_COUNT) {
                    adjMatrix[src][dest] = weight;
                }
                else {
                    cerr << ErrorMsg <<"非法数据存在 " << fileName << ": " << line << endl;
                }
            }
            file.close();
            cout << InformationMsg << "Router " << src << " 数据洪泛结束" << endl;
        }
        cout << endl;
    }

    void printNetwork() {
        for (int i = 0; i < 6; ++i) {
            cout << InformationMsg << "Router " << i << " 的 IP : ";
            printIP(i);
            cout << endl;
        }
        cout << endl << InformationMsg << "邻接矩阵:\n" << endl;
        cout << "\033[47;30m" << setw(18) << "Router 0" << setw(10) << "Router 1" << setw(10) <<
"Router 2" << setw(10) << "Router 3" << setw(10) << "Router 4" << setw(10) << "Router 5" <<
"\033[47;0m\n";
        for (int i = 0; i < NODE_COUNT; ++i) {
            cout << "\033[47;30mRouter " << i << ":\033[47;0m" << setw(6) << left << "";
            for (int j = 0; j < NODE_COUNT; ++j) {
                if (adjMatrix[i][j] == INF)
                    cout << setw(9) << left << "INF";
                else
                    cout << setw(9) << left << adjMatrix[i][j];
            }
            cout << endl;
        }
        cout << endl;
    }
};

class Node {
public:
    int id;
    vector<int> distances;
    vector<int> predecessors;

    Node(int id) : id(id), distances(NODE_COUNT, INF), predecessors(NODE_COUNT, -1) {}
```

```
    friend void printIP(int index);

    void updateRoutingTable(Network& network) {
        distances.assign(NODE_COUNT, INF);
        predecessors.assign(NODE_COUNT, -1);
        distances[id] = 0;

        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<>> pq;
        pq.emplace(0, id);

        while (!pq.empty()) {
            int dist = pq.top().first;
            int current = pq.top().second;
            pq.pop();

            if (dist > distances[current]) continue;

            for (int neighbor = 0; neighbor < NODE_COUNT; ++neighbor) {
                if (network.adjMatrix[current][neighbor] != INF) {
                    int newDist = dist + network.adjMatrix[current][neighbor];
                    if (newDist < distances[neighbor]) {
                        distances[neighbor] = newDist;
                        predecessors[neighbor] = current;
                        pq.emplace(newDist, neighbor);
                    }
                }
            }
        }
    }

    void printRoutingTable() {
        cout << InformationMsg << "Router " << id << " 的路由表:\n";
        cout << "\033[47;30m" << setw(16) << left << "本机" << setw(16) << left << "目标" << setw(8)
<< left << "距离" << setw(14) << left << "下一跳" << "\033[47;0m\n";
        for (int i = 0; i < NODE_COUNT; ++i) {
            printIP(id);
            cout << "\t";
            printIP(i);
            cout << "\t" << (distances[i] == INF ? "INF" : to_string(distances[i])) << "\t";
            if (predecessors[i] == -1) {
                cout << "-" << endl;
            }
            else {
                printIP(predecessors[i]);
                cout << endl;
            }
```

```
        }
        cout << endl;
    }

    void sendData(int destination) {
        cout << InformationMsg << "Router " << id << " 正在向 Router " << destination << " 发送数据.\n";
        vector<int> path;
        int current = destination;

        while (current != id && current != -1) {
            path.push_back(current);
            current = predecessors[current];
        }

        if (current == -1) {
            cout << WarningMsg << "没有合适的路径!" << endl;
            return;
        }

        path.push_back(id);
        reverse(path.begin(), path.end());

        cout << InformationMsg << "传输路径: ";
        for (auto& node : path) {
            printIP(node);
            cout << " -> ";
        }
        cout << "成功\n" << endl;
    }
};

int main() {
    Network network;

    vector<Node> nodes;
    for (int i = 0; i < NODE_COUNT; ++i) { // 新建路由节点，模拟网络中的各个路由
        nodes.emplace_back(i);
    }

    network.loadFromFiles(); // 网络洪泛
    network.printNetwork(); // 输出洪泛后的邻接矩阵

    // 根据洪泛结果更新路由表
    for (auto& node : nodes) {
        node.updateRoutingTable(network);
```

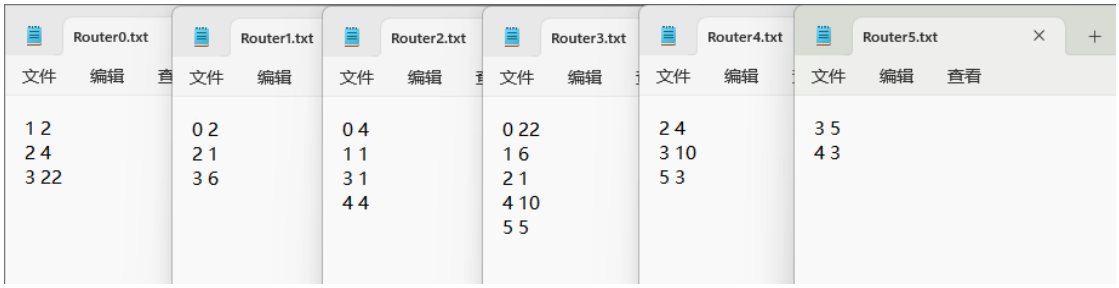
```
    }

    // 输出路由表
    for (auto& node : nodes) {
        node.printRoutingTable();
    }

    // 数据传输路径测试
    while (1) {
        int src = -1, dest = -1;
        cout << InformationMsg << "开始进行路由间通信路径测试，请输入起始路由（数字 0-5 ）：";
        cin >> src;
        cout << InformationMsg << "请输入目标路由（数字 0-5 ）：";
        cin >> dest;
        if (src<6 && src>-1 && dest<6 && dest>-1)
            nodes[src].sendData(dest);
        else
            cout << WarningMsg << "路由范围不正确！\n" << endl;
    }
    return 0;
}
```

5. **调试及结果（截图或视频附件）**

数据由文件读入：



执行结果：

D:\个人文件\作业\第三学期_大

```
[INFO]  Router 0 数据洪泛开始
[INFO]  Router 0 数据洪泛结束
[INFO]  Router 1 数据洪泛开始
[INFO]  Router 1 数据洪泛结束
[INFO]  Router 2 数据洪泛开始
[INFO]  Router 2 数据洪泛结束
[INFO]  Router 3 数据洪泛开始
[INFO]  Router 3 数据洪泛结束
[INFO]  Router 4 数据洪泛开始
[INFO]  Router 4 数据洪泛结束
[INFO]  Router 5 数据洪泛开始
[INFO]  Router 5 数据洪泛结束

[INFO]  Router 0 的 IP : 192.168.0.1
[INFO]  Router 1 的 IP : 192.168.0.2
[INFO]  Router 2 的 IP : 192.168.0.3
[INFO]  Router 3 的 IP : 192.168.0.4
[INFO]  Router 4 的 IP : 192.168.0.5
[INFO]  Router 5 的 IP : 192.168.0.6

[INFO]  邻接矩阵:

          Router 0  Router 1  Router 2  Router 3  Router 4  Router 5
Router 0:       0         2         4        22       INF       INF
Router 1:       2         0         1         6       INF       INF
Router 2:       4         1         0         1         4       INF
Router 3:      22         6         1         0        10         5
Router 4:     INF       INF         4        10         0         3
Router 5:     INF       INF       INF         5         3         0

[INFO]  Router 0 的路由表:
本机            目标            距离    下一跳
192.168.0.1     192.168.0.1     0       -
192.168.0.1     192.168.0.2     2       192.168.0.1
192.168.0.1     192.168.0.3     3       192.168.0.2
192.168.0.1     192.168.0.4     4       192.168.0.3
192.168.0.1     192.168.0.5     7       192.168.0.3
192.168.0.1     192.168.0.6     9       192.168.0.4

[INFO]  Router 1 的路由表:
本机            目标            距离    下一跳
192.168.0.2     192.168.0.1     2       192.168.0.2
192.168.0.2     192.168.0.2     0       -
192.168.0.2     192.168.0.3     1       192.168.0.2
192.168.0.2     192.168.0.4     2       192.168.0.3
192.168.0.2     192.168.0.5     5       192.168.0.3
192.168.0.2     192.168.0.6     7       192.168.0.4

[INFO]  Router 2 的路由表:
本机            目标            距离    下一跳
192.168.0.3     192.168.0.1     3       192.168.0.2
192.168.0.3     192.168.0.2     1       192.168.0.3
192.168.0.3     192.168.0.3     0       -
192.168.0.3     192.168.0.4     1       192.168.0.3
192.168.0.3     192.168.0.5     4       192.168.0.3
192.168.0.3     192.168.0.6     6       192.168.0.4

[INFO]  Router 3 的路由表:
本机            目标            距离    下一跳
192.168.0.4     192.168.0.1     4       192.168.0.2
192.168.0.4     192.168.0.2     2       192.168.0.3
192.168.0.4     192.168.0.3     1       192.168.0.4
192.168.0.4     192.168.0.4     0       -
192.168.0.4     192.168.0.5     5       192.168.0.3
192.168.0.4     192.168.0.6     5       192.168.0.4

[INFO]  Router 4 的路由表:
本机            目标            距离    下一跳
192.168.0.5     192.168.0.1     7       192.168.0.2
192.168.0.5     192.168.0.2     5       192.168.0.3
192.168.0.5     192.168.0.3     4       192.168.0.5
192.168.0.5     192.168.0.4     5       192.168.0.3
192.168.0.5     192.168.0.5     0       -
192.168.0.5     192.168.0.6     3       192.168.0.5

[INFO]  Router 5 的路由表:
本机            目标            距离    下一跳
192.168.0.6     192.168.0.1     9       192.168.0.2
192.168.0.6     192.168.0.2     7       192.168.0.3
192.168.0.6     192.168.0.3     6       192.168.0.4
192.168.0.6     192.168.0.4     5       192.168.0.6
192.168.0.6     192.168.0.5     3       192.168.0.6
192.168.0.6     192.168.0.6     0       -

[INFO]  开始进行路由间通信路径测试, 请输入起始路由 (数字 0-5 ) : 0
[INFO]  请输入目标路由 (数字 0-5 ) : 5
[INFO]  Router 0 正在向 Router 5 发送数据.
[INFO]  传输路径: 192.168.0.1 -> 192.168.0.2 -> 192.168.0.3 -> 192.168.0.4 -> 192.168.0.6 -> 成功

[INFO]  开始进行路由间通信路径测试, 请输入起始路由 (数字 0-5 ) : 5
[INFO]  请输入目标路由 (数字 0-5 ) : 0
[INFO]  Router 5 正在向 Router 0 发送数据.
[INFO]  传输路径: 192.168.0.6 -> 192.168.0.4 -> 192.168.0.3 -> 192.168.0.2 -> 192.168.0.1 -> 成功

[INFO]  开始进行路由间通信路径测试, 请输入起始路由 (数字 0-5 ) : 2
[INFO]  请输入目标路由 (数字 0-5 ) : 4
[INFO]  Router 2 正在向 Router 4 发送数据.
[INFO]  传输路径: 192.168.0.3 -> 192.168.0.5 -> 成功

[INFO]  开始进行路由间通信路径测试, 请输入起始路由 (数字 0-5 ) : 4
[INFO]  请输入目标路由 (数字 0-5 ) : 9
[WARN]  路由范围不正确!

[INFO]  开始进行路由间通信路径测试, 请输入起始路由 (数字 0-5 ) :
```