OSPF路由模拟系统

学号：212207140

姓名：戴龙星

一、数据对象

数据对象1	数据对象名称：	管理员顺序表
	数据元素定义：	<管理员名称， 账号， 密码>
	数据结构：	顺序表
	储存结构：	数组
	基本操作：	初始化、查找、登录、登出
	数据来源：	administrator.txt文件导入导出

数据对象2	数据对象名称：	用户排序二叉树
	数据元素定义：	<用户名， 账号， 密码>
	数据结构：	二叉树
	储存结构：	二叉排序树
	基本操作：	初始化、插入、删除、取值、查找、注册、修改或查看密码、登录、登出
	数据来源：	user.txt文件导入导出

数据对象3	数据对象名称：	网络拓扑总图
	数据元素定义：	<路由器地址、权值>
	数据结构：	图
	储存结构：	邻接表
	基本操作：	由网络拓扑图类提供函数接口
	数据来源：	AllNetWork.txt文件导入导出

数据对象3	数据对象名称：	路由平衡二叉树
	数据元素定义：	<路由器地址、网络拓扑图、路由表>

数据对象3	数据对象名称：	路由平衡二叉树
	数据结构：	二叉树
	储存结构：	平衡二叉树
	基本操作：	初始化平衡二叉树，添加、查找、删除路由器，添加、删除、修改路由器权值，输出最短路径、路径长度、路由器内网络拓扑图，字典序显示路由器地址

数据对象4	数据对象名称：	网络拓扑图
	数据元素定义：	<路由器地址、权值>
	数据结构：	图
	储存结构：	邻接表
	基本操作：	初始化图，添加、删除路由器地址，添加、删除、修改权值，删除与某路由器不连通的所有路由器，显示网络拓扑图、快速排序显示权值

数据对象5	数据对象名称：	路由表
	数据元素定义：	<路由器地址、下一跳路由器地址>
	数据结构：	线性表
	储存结构：	链表
	基本操作：	初始化、构造路由表、显示两路由器最短路径、路径长度，删除路由表

二、系统功能

序号	名称	实现算法
1	管理员验证登录功能	利用顺序查找法根据用户输入的账号密码进行查找验证，验证失败返回主菜单，成功进入管理员面板
2	用户注册功能	利用排序二叉树的插入算法将用户输入的用户名、账号、密码插入排序二叉树，插入成功后返回主菜单

序号	名称	实现算法
3	用户验证登录功能	利用排序二叉树查找法根据用户输入的账号密码进行查找验证，验证失败返回主菜单，成功进入用户面板
4	查看网络拓扑总图	遍历邻接表输出网络拓扑总图，空图则显示图已空
5	查看与某路由器相邻的所有路由器	根据用户输入的路由器地址A，遍历链表输出相邻路由器，遍历失败显示”A不存在“
6	查看两路由器间的最短路径，以及经过的路由器	第一步：根据用户输入的起始路由地址A、终点路由地址B，遍历路由表输出最短路径 第二步：反向遍历路由表显示经过的各节点 遍历失败输出”无法显示A到B的最短路径“
7	查看某路由器内部的网络拓扑图	根据用户输入的路由器地址A，遍历邻接表输出其内部的网络拓扑图，遍历失败显示”该路由器不存在“
8	查看某路由器内部的路由表	根据用户输入的路由器地址A，遍历输出其内部的邻接表，遍历失败显示”该路由器不存在“
9	添加路由器	根据用户输入的路由器地址A，利用平衡二叉树的插入算法添加A
10	添加权值	第一步：通过起点路由器地址、终点路由器地址、权值，在网络拓扑总图中，以及在各自的网络拓扑图中将两路由器连接。第二步：连接后，两路由器交换现有信息，在各自的网络拓扑图中补充新的顶点和权值、并更新路由表。任选其一进行泛洪传播
11	修改权值	第一步：通过起点路由器地址、终点路由器地址、权值，在网络拓扑总图中，以及两者都在各自的网络拓扑图中修改两路由器之间权值 第二步：更新路由表。任选其一进行泛洪传播 修改失败显示”修改失败“
12	删除权值	第一步：通过起点路由器地址、终点路由器地址、权值，在网络拓扑总图中，以及两者都在各自的网络拓扑图中，将两路由器之间的权值删除 第二部：把不与自己联通的路由器信息全部删除，更新路由表。两者进行泛洪传播 删除失败显示”删除失败“
13	删除某路由器的所有邻接边	通过路由器地址A，在网络拓扑总图，以及将路由器内的所有邻接边逐个删除
14	删除某路由器	第一步：通过路由器地址，删除网络拓扑总图内路由器的所有邻接边和路由器地址 第二步：在平衡二叉树中删除路由器
15	查看用户信息	通过中序递归用户排序二叉树输出用户信息
16	查看网络的线路拥堵情况	通过快速排序将网络拓扑总图的权值从小到达排序输出
17	注销用户	通过用户名中序递归排序二叉树删除用户信息

三、系统测试

1、核心代码

1、**利用dijkstra算法，构造路由表**

```
//根据路由器的地址以及它的网络拓扑图构建路由表
void Make_routerlist(Graph& G, const string& name)
{
    routerlist->name = name;
    //通过网络拓扑图和路由器地址初始化路径结构表
    Path P = CreatePath(G, name);
    //迪杰斯特拉构造Path表
    Dijkstra(P, G);
    //通过Path表构造路由表
    _make_routers(P);
}

//通过网络拓扑图初始化路径结构表
Path CreatePath(Graph& G, const string& name)
{
    //初始化头节点
    Path P = new PNode{};

    P->name = name;
    P->dist = 0;
    P->flag = true;
    P->parent = "";
    P->next = nullptr;

    Vert t = G.head->vp;
    while (t)
    {
        //将与本地路由器联通的且不是本地路由器的路由器插入Path
        if (t != G.FindV(name) && G.IsConnectivity(name, t->name))
        {
            Path pp = new PNode{};

            //元素赋值
            pp->name = t->name;
            pp->dist = G.Getpower(name, t->name);
            pp->flag = false;
            pp->parent = name;

            //头插法插入P链表
            pp->next = P->next;
            P->next = pp;
        }
        t = t->vp;
    }
    return P;
}

//迪杰斯特拉寻找单源最短路径
void Dijkstra(Path P, Graph& G)
{
```

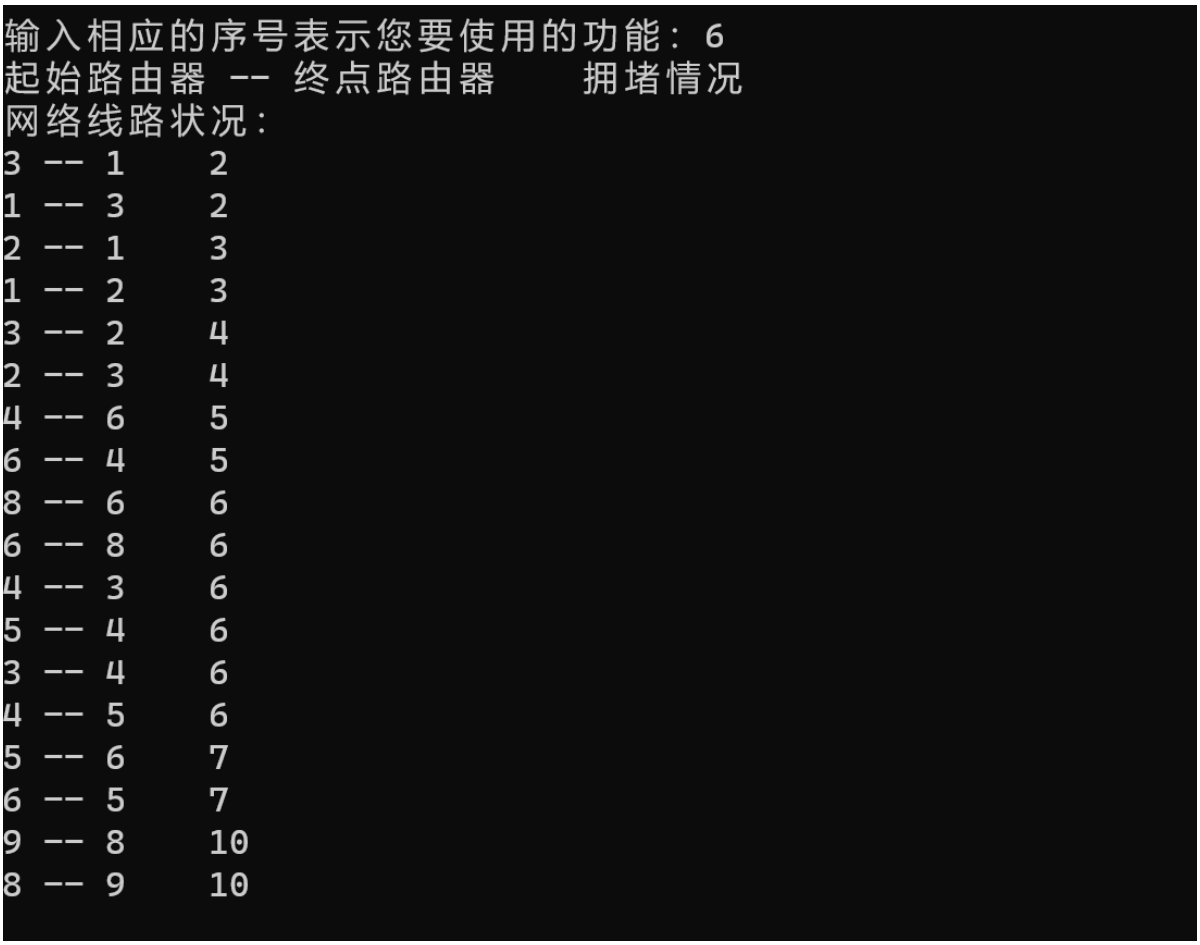
```
    //图中与本地路由器联通的路由器都被标记，则退出
    if (IsFullFlag(P))
        return;

    //找未被标记的与本地路由器dist最小的路由器，并将其标记
    Path minp = FindMinPath(P);

    //更新dist
    UpDist(P, minp, G);
    Dijkstra(P, G);
}
```

2、**利用广度优先搜索，将路由表数据进行全网泛洪**

```
//用bps模拟的路由器报文泛洪传播函数
bool Flood_Fill(const string& name)
{
    //创建一个队列
    Queue q = CreateQueue();
    //树内无name直接返回false
    if (!FindRAVL(head, name))
        return false;
    //找到路由器在树中的位置
    RAVL Rp = FindRAVL(head, name);
    //将路由器入队
    PushQ(q, Rp->R);

    //循环出队入队复制网图
    while (!IsEmptyQ(q))
    {
        //出队
        Router R = PopQ(q);
        //找到路由器在自己的网图中的位置
        Edge E = R.network.FindV(R.name)->ep;

        //寻找邻接顶点
        while (E)
        {
            //用邻接点的路由名寻找邻接路由在树中的位置
            Rp = FindRAVL(head, E->name);
            //如果该邻接路由未被标记
            if (!Rp->visited)
            {
                //标记
                Rp->visited = true;
                //将路由的网图赋给邻接顶点的网图
                Rp->R.network = R.network;
                //邻接节点更新路由表
                Rp->R.rlist.deleteRouterList();
                Rp->R.rlist.Make_routerlist(Rp->R.network, Rp->R.name);
                //入队
                PushQ(q, Rp->R);
            }
            E = E->ep;
        }
```

```
    }
    //重置标记
    ReserVisited(head);
    return true;
}
```

3、**利用快速排序，将网络拓扑图的权值从小到达排序输出**




```
//对权值进行排序并打印
void SortPrintEdge() {
    const int MAX_EDGE = 1000;  //假设边的最大数
    WeightEdge edges[MAX_EDGE];
    int size = 0;
    //获取所有边
    GetAllEdges(edges, size);

    //使用快排排序
    QuickSortEdges(edges, 0, size - 1);

    cout << "网络线路状况：\n";
    if (size > 0) {
        cout << edges[0].from << " -- " << edges[0].to << "    " <<
edges[0].weight << endl;
    }

    for (int i = 1; i < size; i++) {
        if (edges[i].from != edges[i - 1].from || edges[i].to != edges[i - 1].to
|| edges[i].weight != edges[i - 1].weight)
```

```
            cout << edges[i].from << " -- " << edges[i].to << "    " <<
edges[i].weight << std::endl;
    }
}

//获取图中所有边
void GetAllEdges(WeightEdge* edges, int& size) {
    size = 0;
    Vert p = head;
    while (p->vp) {
        p = p->vp;
        Edge t = p->ep;
        while (t) {
            edges[size].from = p->name;
            edges[size].to = t->name;
            edges[size].weight = t->power;
            size++;
            t = t->ep;
        }
    }
}

//快速排序
void QuickSortEdges(WeightEdge* edges, int low, int high) {
    if (low < high) {
        int pivot = low;
        int i = low + 1;
        int j = high;

        while (i <= j) {
            if (edges[i].weight < edges[pivot].weight)
                i++;
            else if (edges[j].weight > edges[pivot].weight)
                j--;
            else
            {
                swap(edges[i], edges[j]);
                i++;
                j--;
            }
        }

        swap(edges[pivot], edges[j]);

        QuickSortEdges(edges, low, j - 1);
        QuickSortEdges(edges, j + 1, high);
    }
}
```

2、功能演示

1、主菜单演示：

2、管理员登录，以及管理员面板：

```
普通用户注册请输入C
退出OSPF路由模拟系统请输入D
您的输入： A
尊敬的管理员，请输入您的账号以及密码登录本平台
账号： 12454fdrg
密码： 45drgd
登录失败！


如果您需要以管理员的身份登录，请输入A。
普通用户登录请输入B
普通用户注册请输入C
退出OSPF路由模拟系统请输入D

您的输入： A
尊敬的管理员，请输入您的账号以及密码登录本平台
账号： tina888
密码： 123abc
登录成功！

尊敬的管理员，您有以下功能的使用权限：

通用功能：
1、查看网络拓扑总图
2、查看与某路由器相邻的所有路由器
3、查看两路由器间的最短路径，以及经过的路由器
4、查看某路由器内部的网络拓扑图
5、查看某路由器内部的路由表
6、退出登录

管理员专用功能：
7、添加路由器
8、添加权值
9、修改权值
10、删除权值
11、删除某路由器的所有邻接边
12、删除某路由器
13、查看用户信息
14、查看网络的线路拥堵情况
15、注销用户
```

3、查看网络拓扑图

```
输入相应的序号表示您要使用的功能：1
网络拓扑总图：
9----->8
8----->6----->9
6----->8----->4----->5
5----->6----->4
4----->3----->6----->5
3----->4----->1----->2
2----->3----->1
1----->3----->2
```

4、查看与某路由器相邻的所有路由器

```
输入相应的序号表示您要使用的功能：2
输入路由器地址来查看它的邻接路由器：4
3 6 5
```

5、查看两路由器间的最短路径，以及经过的路由器

```
输入相应的序号表示您要使用的功能：3
输入起始路由器和终点路由器来查看最短路径
起始路由器地址：123
终点路由器地址：432
无法显示 123 到 432 的最短路径

输入相应的序号表示您要使用的功能：3
输入起始路由器和终点路由器来查看最短路径
起始路由器地址：1
终点路由器地址：9
1 到 9 的最短路径所经过的路由器有：
1 -> 3 -> 4 -> 6 -> 8 -> 9

最短路径的总权值为：29
```

6、查看某路由器内部的网络拓扑图

```
输入相应的序号表示您要使用的功能：4
输入路由器地址来查看它内部的网络拓扑图：132
该路由器不存在！

输入相应的序号表示您要使用的功能：1
网络拓扑总图：
9---->8
8---->6---->9
6---->8---->4---->5
5---->6---->4
4---->3---->6---->5
3---->4---->1---->2
2---->3---->1
1---->3---->2
```

7、查看某路由器内部的路由表

```
输入相应的序号表示您要使用的功能：5
输入路由器地址来查看它内部的路由表：123
路由器不存在！

输入相应的序号表示您要使用的功能：5
输入路由器地址来查看它内部的路由表：1
1 的路由表：
To ---> 9  NextJump: 3
To ---> 8  NextJump: 3
To ---> 5  NextJump: 3
To ---> 6  NextJump: 3
To ---> 4  NextJump: 3
To ---> 3  NextJump: 3
To ---> 2  NextJump: 2
```

8、添加路由器

```
输入相应的序号表示您要使用的功能：7
输入路由器地址来添加路由器：10
网络信息保存成功!
添加成功

输入相应的序号表示您要使用的功能：1
网络拓扑总图：
10
9---->8
8---->6---->9
6---->8---->4---->5
5---->6---->4
4---->3---->6---->5
3---->4---->1---->2
2---->3---->1
1---->3---->2
```

9、添加权值

```
输入相应的序号表示您要使用的功能：8
输入起始路由器和终点路由器的地址，以及它们之间的权值来添加权值
起始路由器地址：231
终点路由器地址：423
权值：231
添加失败

输入相应的序号表示您要使用的功能：8
输入起始路由器和终点路由器的地址，以及它们之间的权值来添加权值
起始路由器地址：1
终点路由器地址：10
权值：11
添加成功
网络信息保存成功!

输入相应的序号表示您要使用的功能：1
网络拓扑总图：
10---->1
9---->8
8---->6---->9
6---->8---->4---->5
5---->6---->4
4---->3---->6---->5
3---->4---->1---->2
2---->3---->1
1---->10---->3---->2
```

10、修改权值

```
输入相应的序号表示您要使用的功能：9
输入起始路由器和终点路由器的地址，以及新权值，来修改它们之间的权值
起始路由器地址：131
终点路由器地址：321
权值：123
修改失败

输入相应的序号表示您要使用的功能：9
输入起始路由器和终点路由器的地址，以及新权值，来修改它们之间的权值
起始路由器地址：1
终点路由器地址：10
权值：21
修改成功
网络信息保存成功!
```

11、删除权值

```
输入相应的序号表示您要使用的功能：10
输入起始路由器和终点路由器的地址,来删除它们之间的权值
起始路由器地址：1232
终点路由器地址：321
删除成功
网络信息保存成功!
```

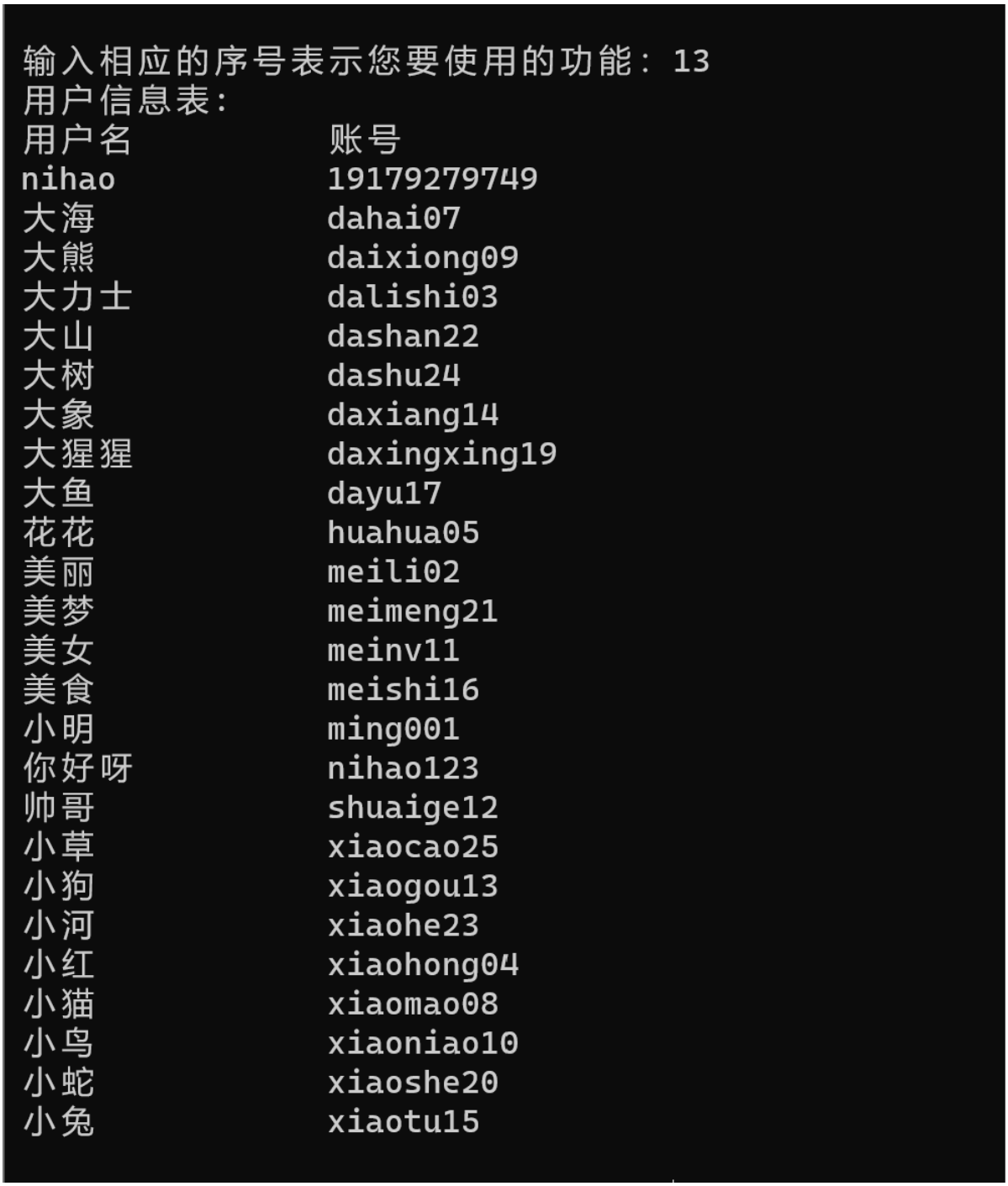
12、删除某路由器的所有邻接边

```
输入路由器地址来删除它的所有邻接边：10
网络信息保存成功!

输入相应的序号表示您要使用的功能：1
网络拓扑总图：
10
9---->8
8---->6---->9
6---->8---->4---->5
5---->6---->4
4---->3---->6---->5
3---->4---->1---->2
2---->3---->1
1---->3---->2
```

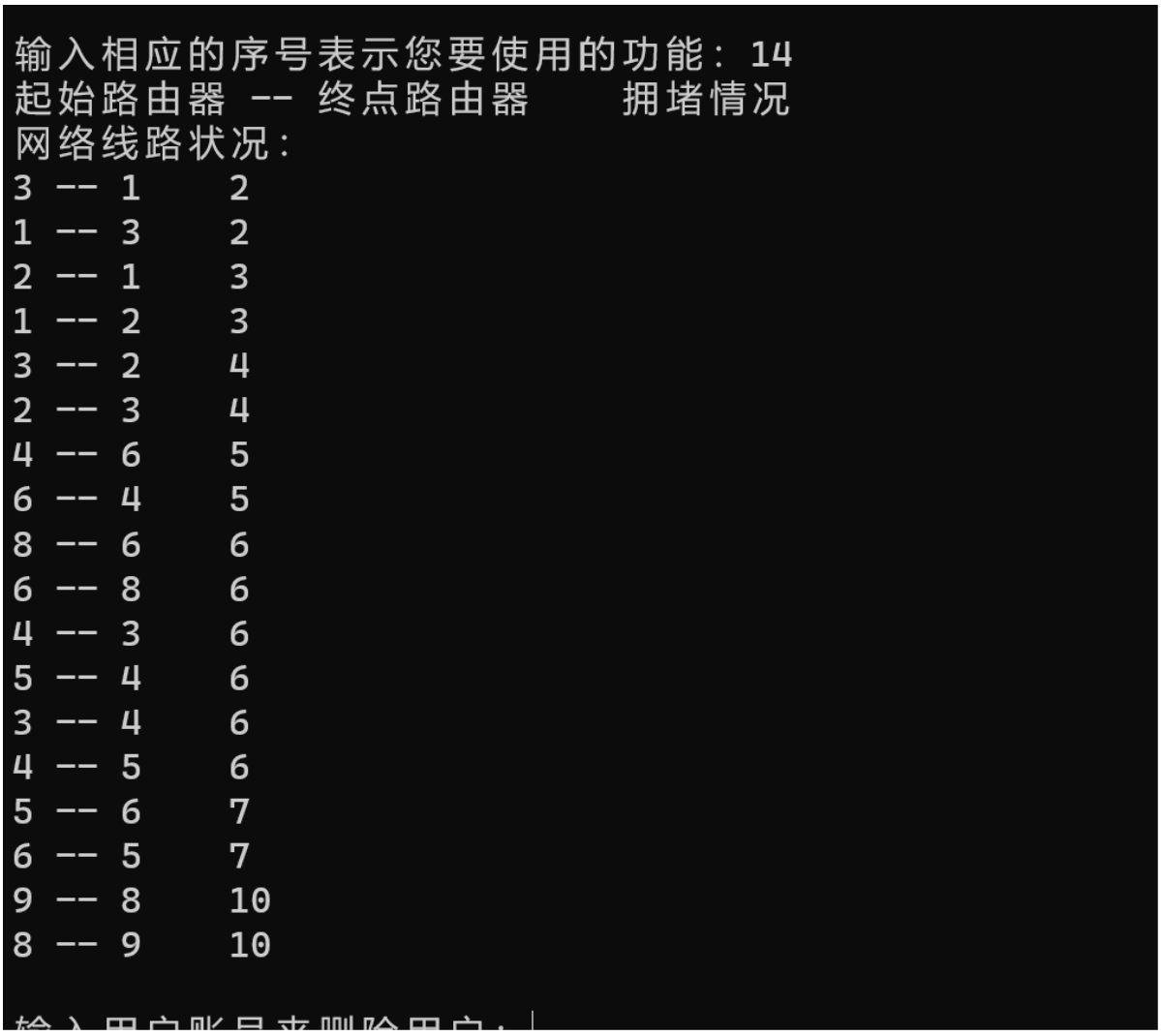
13、删除某路由器

```
输入相应的序号表示您要使用的功能：12
输入路由器地址来删除路由器：10
网络信息保存成功!

输入相应的序号表示您要使用的功能：1
网络拓扑总图：
9---->8
8---->6---->9
6---->8---->4---->5
5---->6---->4
4---->3---->6---->5
3---->4---->1---->2
2---->3---->1
1---->3---->2
```

14、查看用户信息



15、查看网络的线路拥堵情况


输入相应的序号表示您要使用的功能: 14
起始路由器 -- 终点路由器 拥堵情况
网络线路状况:
3 -- 1 2
1 -- 3 2
2 -- 1 3
1 -- 2 3
3 -- 2 4
2 -- 3 4
4 -- 6 5
6 -- 4 5
8 -- 6 6
6 -- 8 6
4 -- 3 6
5 -- 4 6
3 -- 4 6
4 -- 5 6
5 -- 6 7
6 -- 5 7
9 -- 8 10
8 -- 9 10

16、注销用户

```
输入相应的序号表示您要使用的功能：15
输入用户账号来删除用户：19179279749
用户信息保存成功！

输入相应的序号表示您要使用的功能：13
用户信息表：
用户名          账号
大海            dahai07
大熊            daixiong09
大力士          dalishi03
大山            dashan22
大树            dashu24
大象            daxiang14
大猩猩          daxingxing19
大鱼            dayu17
花花            huahua05
美丽            meili02
美梦            meimeng21
美女            meinv11
美食            meishi16
小明            ming001
你好呀          nihao123
帅哥            shuaige12
小草            xiaocao25
小狗            xiaogou13
小河            xiaohe23
小红            xiaohong04
小猫            xiaomao08
小鸟            xiaoniao10
小蛇            xiaoshe20
小兔            xiaotu15
```

17、用户登录、用户面板展示

```
如果您需要以管理员的身份登录，请输入A。
普通用户登录请输入B
普通用户注册请输入C
退出OSPF路由模拟系统请输入D

您的输入： B
尊敬的用户，请输入您的账号以及密码登录本平台
账号： xiaotu15
密码： 3424234
密码错误！


如果您需要以管理员的身份登录，请输入A。
普通用户登录请输入B
普通用户注册请输入C
退出OSPF路由模拟系统请输入D

您的输入： B
尊敬的用户，请输入您的账号以及密码登录本平台
账号： xiaotu15
密码： bunny123
登录成功！

尊敬的用户，您有以下功能的使用权限：

1、查看网络拓扑总图
2、查看与某路由器相邻的所有路由器
3、查看两路由器间的最短路径，以及经过的路由器
4、查看某路由器内部的网络拓扑图
5、查看某路由器内部的路由表
6、查看网络的拥堵情况
7、注销账户
8、退出登录
```

四、实验心得

课程的理论学习就是了解各种数据结构的概念，各种算法的思想，实验就是将这些东西一一实践，在实践的过程中还能巩固学习到的知识。

这次试验的难点有两个，一个是路由表的构造，一个是路由表的泛洪传播。

开始写大作业的时候，我并没有过多构思就直接编写，写这一步的时候完全不知道下一步是什么，完全凭感觉。这导致我每写一个函数之前写的代码都有bug，每次修复bug都耗费了巨量时间，我那个星期几乎每时每刻都在修bug。

后来我在路由表的泛洪传播的编写中，简单的认为只要将出现边权变化的路由器的路由表直接BFS传播。到很后面才知道我是完全错误的（原因不赘述了），这导致我写的很多代码都都得重新写。于是我干脆把所有的代码删掉，从外层到内层重新构思，列了很多表格，每一步都写在表里。然后从内核至外层编写代码，每次写了一个函数都会对它进行各方面的测试，以防影响后续代码。

从这次实验中，我学到了很多，其中最重要就是在写一个项目之前，要尽可能地将所有过程先用文字或表格写下来，再分析是否有逻辑错误。准备好一切后才能开始写代码，不然就会没有底，bug满天飞。