电网建设造价模拟系统

问题描述

假设一个城市有n个小区，要实现n个小区之间的电网都能够相互接通，构造这个城市n个小区之间的电网，使总工程造价最低。请设计一个能够满足要求的造价方案。

项目要求

在每个小区之间都可以设置一条电网线路，都要付出相应的经济代价。n个小区之间最多可以有n（n-1）/2条线路，选择其中的n-1条使总的耗费最少。

算法设计

用连通图表示各小区以及小区之间的电网线路，网的顶点表示小区，边表示两个小区之间的线路，赋予边的权值即为相应耗费。对于n个顶点的连通网可以建立多个不同的生成树，即电路网。要求得到耗费最少的电路网线，即为构造连通网中的最小生成树。小区之间的距离由邻接矩阵表示，若两个小区之间不存在通路，将相应边权值设为无穷大。

使用Prim算法求解最小生成树。

功能实现及代码分析

1、类和结构体设计

Vertex表示顶点，Net类为项目抽象出的类，包含邻接表生成、最小生成树计算、操作、输出最小生成树等函数。

```
class MinCost {
public:
    Node MSTree[100];
    int size;    //当前的容量值
    MinCost() { size = 0; }

    void insert(Node *&node) {
        MSTree[size].start = node->start;
        MSTree[size].end = node->end;
        MSTree[size].cost = node->cost;
        ++size;
    }
};

struct Vertex {
    int num;     //顶点序号
    string city; //小区名
    Edge *first;
};

class Net {
public:
    int size;           //顶点个数
    Vertex *vertArray;  //邻接表顶点表
    MinCost MST;

    Net() { size = 0; }
    ~Net() {}
    void create();
    void add();
    int findVertArray(string city);
    bool prim();
    void operation();
    void printMST();
};
```

2、数据结构设计

邻接矩阵表示图，用一对顶点下标表示一条边。

```
struct Edge {
    int num;     //次顶点序号
    float cost;  //边的开销
    Edge *next;  //指向下一个邻接顶点

};

class Node {
public:
    int start;   //头结点序号
    int end;     //尾节点序号
    float cost;  //造价

    Node() {}    //构造函数
    Node(int start, int end, float cost) {

        this->start = start;
        this->end = end;
        this->cost = cost;

    }

    ~Node() {}   //析构函数
};
```

3、进入项目

进入项目，引导用户输入并可多次重复选择操作。

```
int main() {
    cout << "**              电网造价模拟系统              **" << endl;
    cout << "==========================================" << endl;
    cout << "**            A --- 创建电网顶点            **" << endl;
    cout << "**            B --- 添加电网的边            **" << endl;
    cout << "**            C --- 构造最小生成树          **" << endl;
    cout << "**            D --- 显示最小生成树          **" << endl;
    cout << "**            E --- 退出    程序            **" << endl;
    cout << "==========================================\n" << endl;
    Net power;
    power.operation();
    return 0;
}

void Net::operation() {
    char strOpe;
    while (true) {
        cout << "请选择操作: ";
        cin >> strOpe;
        cin.sync(); //清除多余内容
        if (strOpe == 'A' || strOpe == 'a') {
            create();
        }
        if (strOpe == 'B' || strOpe == 'b') {
            add();
        }
        if (strOpe == 'C' || strOpe == 'c') {
            prim();
        }
        if (strOpe == 'D' || strOpe == 'd') {
            cout << "最小生成树的顶点及边为: " << endl;
            printMST();
        }
        if (strOpe == 'E' || strOpe == 'e') break;  //退出while(true)
        putchar(10);
    }
}
```

4、创建电网顶点

创建顶点并储存。

```
void Net::create() {                //创建电网顶点
    cout << "请输入顶点的数目:";
    cin >> size;
    cin.sync();                     //清除多余的输入
    if (size >= 50) {
        cout << "超过最大容量！" << endl;
        return;
    }
    vertArray = new Vertex[size];

    cout << "请依次输入各顶点名称:" << endl;;
    for (int i = 0; i < size; ++i) {
        vertArray[i].num = i;       //顶点序号
        cin >> vertArray[i].city;   //输入顶点的值
        vertArray[i].first = NULL;
    }
    cin.sync();
}
```

5、添加电网边

根据用户输入添加电网的边，对边的两个顶点进行合法性判断，若不存在，报错并退出操作。

```
void Net::add() {                //添加电网边
    string temp;
    int index = 0;

    for (int i = 0; i < size * (size - 1) / 2; ++i) {
        cout << "请输入两个顶点及边：";
        cin >> temp;
        if (temp == "?") break;  //?表示停止输入
        index = findVertArray(temp);
        if (index == -1) {
            cout << "没有该初始顶点" << endl;
            return;
        }

        Edge *temp1 = new Edge;
        cin >> temp;
        index = findVertArray(temp);
        if (index == -1) {
            cout << "没有该初始顶点" << endl;
            return;
        }
        cin >> temp1->cost;
        temp1->num = findVertArray(temp);
        temp1->next = NULL;

        if ((vertArray[index].first) == NULL) {
            vertArray[index].first = temp1;
        } else {
            temp1->next = (vertArray[index].first)->next;
            vertArray[index].first->next = temp1;
        }

        Edge *temp2 = new Edge;
        temp2->num = vertArray[index].num;
        temp2->cost = temp1->cost;
        temp2->next = NULL;

        index = temp1->num;  //插入另一个点为起点的邻接表
        if ((vertArray[index].first) == NULL) {
            vertArray[index].first = temp2;
        } else {
            temp2->next = (vertArray[index].first)->next;
            vertArray[index].first->next = temp2;
        }
    }
    cout << "边导入完毕！" << endl;
}
```

非法输入样例演示

```
请选择操作: A
请输入顶点的数目:3
请依次输入各顶点名称:
a b c

请选择操作: B
请输入两个顶点及边: c d 4
没有该初始顶点

请选择操作:
```

6、构造最小生成树

通过Prim算法构造电网中的最小生成树。对于输入的起始顶点进行合法性判断，若不存在，报错并退出操作。

```
bool Net::prim() {     //Prim算法求最小生成树
    string temp;
    int index = 0;
    cout << "请输入起始顶点:  ";
    cin >> temp;
    index = findVertArray(temp);
    if (index == -1) {
        cout << "没有该初始顶点" << endl;
        return false;
    }

    int lowCost[50], near[50];;
    for (int i = 0; i < size; ++i) {  //初始化near, lowCost
        near[i] = -2;
        lowCost[i] = 9999;  //9999表示无穷大
    }

    int minIndex = index;
    near[index] = -1;  //标记起始点

    Edge *edgeFirst;
    bool createOver = false;
    while (!createOver) {
        for (int i = 0; i < size; ++i) {
            edgeFirst = vertArray[i].first;
            if (near[i] == -1) continue;
            while (edgeFirst != NULL) {
                if ((near[edgeFirst->num] == -1) && (edgeFirst->cost < lowCost[i])) {
                    near[i] = edgeFirst->num;
                    lowCost[i] = edgeFirst->cost;
                }
                edgeFirst = edgeFirst->next;
            }
        }

        for (int i = 0; i < size; ++i) {
            if ((lowCost[i] < lowCost[minIndex]) && (near[i] != -1)) {
                minIndex = i;
            }
        }

        if (lowCost[minIndex] < 9999 && near[minIndex] != -1) {
            Node *tmepNode = new Node(minIndex, near[minIndex], lowCost[minIndex]);
            MST.insert(tmepNode);
            near[minIndex] = -1;
            minIndex = index;  //index是起始顶点号
        }

        for (int i = 0; i < size; ++i) {
            if (near[i] != -1) break;
            if (i == size - 1) createOver = true;
        }
    }
    cout << "已生成Prim最小生成树!" << endl;
    return true;
}
```

非法输入样例演示

```
请选择操作: A
请输入顶点的数目:3
请依次输入各顶点名称:
A B C

请选择操作: C
请输入起始顶点:  R
没有该初始顶点
```

7、显示最小生成树

遍历顶点数组输出构造结果。

```
void Net::printMST() {  //显示最小生成树
    for (int i = 0; i < MST.size; ++i) {
        cout << vertArray[MST.MSTree[i].start].city
             << "-<"
             << MST.MSTree[i].cost
             << ">" << "->"
             << vertArray[MST.MSTree[i].end].city
             << "\t";
    }
}
```

8、辅助函数

在顶点数组中寻找对应的小区，若存在，返回数组下标，若不存在返回-1。

```
int Net::findVertArray(string city) {
    for (int i = 0; i < size; ++i) {
        if (vertArray[i].city == city) {
            return i;
        }
    }
    return -1;
}
```

用例演示

```
**              电网造价模拟系统              **
==========================================
**            A --- 创建电网顶点              **
**            B --- 添加电网的边              **
**            C --- 构造最小生成树            **
**            D --- 显示最小生成树            **
**            E --- 退出    程序              **
==========================================

请选择操作: A
请输入顶点的数目:4
请依次输入各顶点名称:
a b c d

请选择操作: B
请输入两个顶点及边: a b 8
请输入两个顶点及边: b c 7
请输入两个顶点及边: c d 5
请输入两个顶点及边: d a 11
请输入两个顶点及边: a c 18
请输入两个顶点及边: b d 12
边导入完毕!

请选择操作: C
请输入起始顶点:  a
已生成Prim最小生成树!

请选择操作: D
最小生成树的顶点及边为:
b-<8>->a    c-<7>->b    d-<5>->c
请选择操作: E
```

```
Process finished with exit code 0
```