项目说明文档

数据结构课程设计

——电网建设造价模拟系统

作 者 姓 名： 李佳诺

学 号： 1751188

指 导 教 师： 张 颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 目录

1 分析 3

1.1 背景分析 3

1.2 功能分析 3

2 设计 4

2.1 算法设计 4

2.2 数据结构设计 4

2.3 类结构设计 4

3 设计 6

3.1 快速排序quickSort（） 6

**3.2 找到自己的爸爸功能findFather(char x)** 9

3.3 找到自己祖宗功能Union（char a, char b） 10

3.4 循环功能loop（） 12

4 测试 15

4.1 功能性测试 15

4.1.1系统初始化界面测试 15

4.1.2 创建电网顶点功能测试 15

4.1.3 添加电网的边功能测试 15

4.1.4 构造最小生成树功能测试 16

4.1.5 显示最小生成树功能测试： 16

4.1.6 退出程序功能测试： 16

4.2 出错性测试 17

4.2.1 操作数出错测试 17

4.2.2 顶点个数出错测试 18

# 1 分析

## 1.1 背景分析

项目简介

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。

## 1.2 功能分析

项目功能要求：

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。由项目需求可以知道，若用定点表示城市，用边表示城市之间的通信线路，边上的权值表示线路对应的造价，就可以将这一问题表示成一个带权图。为了建立成本最低的通信网络，就要找出该网络的一棵最小生成树。

# 2 设计

## 2.1 算法设计

按照定义，若连通的网络由n个定点组成，则其生成树比含有n个顶点，n-1条边。因此，构造最小生成树的准则有3条：

1. 只能使用该网络中的边来构造
2. 只能使用恰好n-1条边来连接网络中的n个顶点
3. 选用的这n-1条边不能构成回路

在这里，我们采用的是Kruscal算法，即先将所有的边按照权值从小到大排好序，然后依从小到大的次序从这边集合中每次取一条放入生成树的边集合，并且放入的这条边要满足：使得生成树中的边数少于n-1,并且不能构成回路。

## 2.2 数据结构设计

本题主要用到的是图这种数据结构，并且用数组作为存储结构，进行快速的开发。

## 2.3 类结构设计

边类（Edge）

//边的结构，起点、终点、权值

struct Edge

{

int value;

char begin;

char end;

};

电力网络类（ElecNetwork）

//电力网络的构建

class ElecNetwork

{

private:

Edge \_network[1000];

int \_father[1000]; //利用并查集查找爸爸

int \_count; //顶点的个数

public:

void createFather();

void createNetwork();

void menu(); //打印主菜单

void loop(); //主循环

void display();

void generateMinTree(); //产生最小生成树

char findFather(char x); //有关并查集的两个函数 查找与合并

void Union(char a, char b);

void quickSort(Edge network[], int left, int right); //将边的权值快速排序

};

# 3 设计

## 快速排序quickSort（）

**快速排序流程图**

开始

将List划分为两个子序列LeftList和RightList

对LeftList进行快速排序

RightList进行快速排序

List的长度>1?

将LeftList和RightList合并为新的List

结束

是

否

通过一趟排序将待排序记录分割成独立的两部分，其中一部分记录的关键字均比另一部分记录的关键字小，则可分别对这两部分记录继续进行排序，以达到整个序列有序的目的。

**代码如下**：

//快速排序 用于找最小的生成边

void ElecNetwork::quickSort(Edge \*a, int left, int right)

{

int i, j, mid;

Edge temp;

if (left >= right)

return;

i = left;

j = right;

mid = (i + j) >> 1;

do

{

while (a[mid].value > a[i].value) i++;

while (a[mid].value < a[j].value) j--;

//如果找到两个数的话 进行交换

if (i <= j)

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

j--;

}

} while (i <= j);

quickSort(a, left, j);

quickSort(a, i, right);

}

## 3.2 找到自己的爸爸功能findFather(char x)

**找到自己的爸爸功能流程图**

开始

结束

\_father[x]=findFather(\_father[x])

输入 x

\_father[x]<0

是

否

**代码如下**：

//找到自己的爸爸

char ElecNetwork::findFather(char x)

{

if (\_father[x] < 0)

return x;

else

return \_father[x] = findFather(\_father[x]);

}

## ****找到自己祖宗功能Union（char a, char b）****

**找到自己的祖宗功能流程图**

开始

father1=findFather(a);

\_father[father2]=father1;

father2=findFather(b);

结束

**代码如下**：

//找到自己的祖宗(连通 or 不连通)

void ElecNetwork::Union(char a, char b)

{

char father1, father2;

father1 = findFather(a);

father2 = findFather(b);

\_father[father2] = father1;

}

## ****循环功能loop（）****

**循环功能流程图**

开始

creatFather()

creatNetwork()

generateMinTree()

display()

退出

结束

**代码如下：**

//循环功能

void ElecNetwork::loop()

{

char command = 'A';

while (command != 'E')

{

cout << "\n请选择操作:";

cin >> command;

switch (command)

{

case 'A':

createFather();

break;

case 'B':

createNetwork();

break;

case 'C':

generateMinTree();

break;

case 'D':

display();

break;

case 'E':

cout << "感谢您使用本系统，祝你福如东海，寿比南山!" << endl;

break;

default:

cout << "输入错误！请输入有效的操作数!" << endl;

break;

}

}

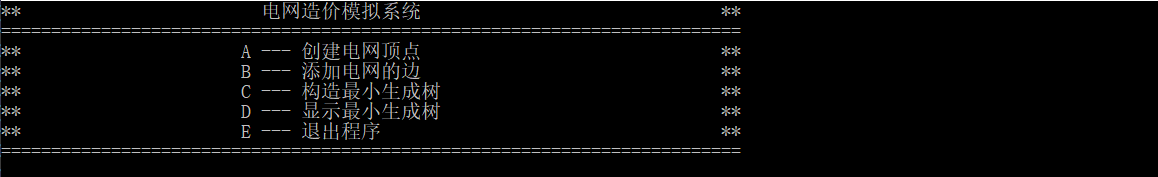
}

# **4 测试**

## 4.1 功能性测试

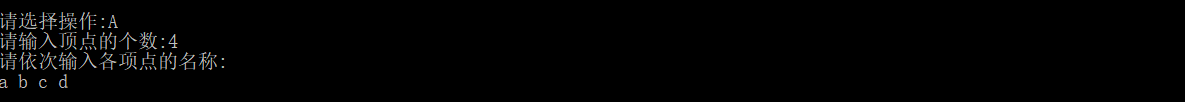
### 4.1.1系统初始化界面测试

查找本电网系统的界面是否可以成功生成，实验结果如下：



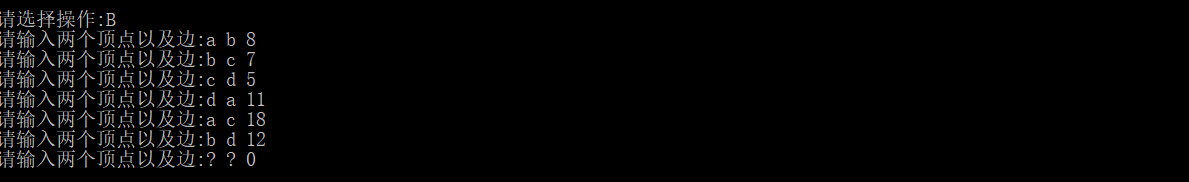
### 4.1.2 创建电网顶点功能测试

验证是否可以成功实现创建电网功能，实验结果如下：



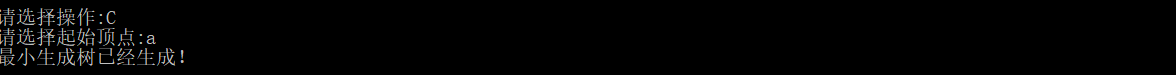
### 4.1.3 添加电网的边功能测试

验证是否可以成功地添加电网的边，实验结果如下：



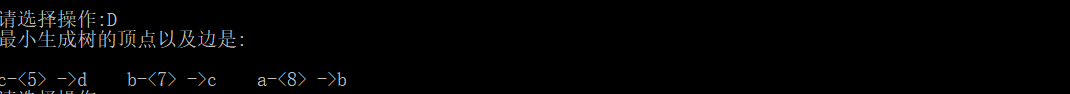
### 4.1.4 构造最小生成树功能测试

验证是否可以在构造完成最小生成树后，显示已经构造完毕的信息，实验结果如下：



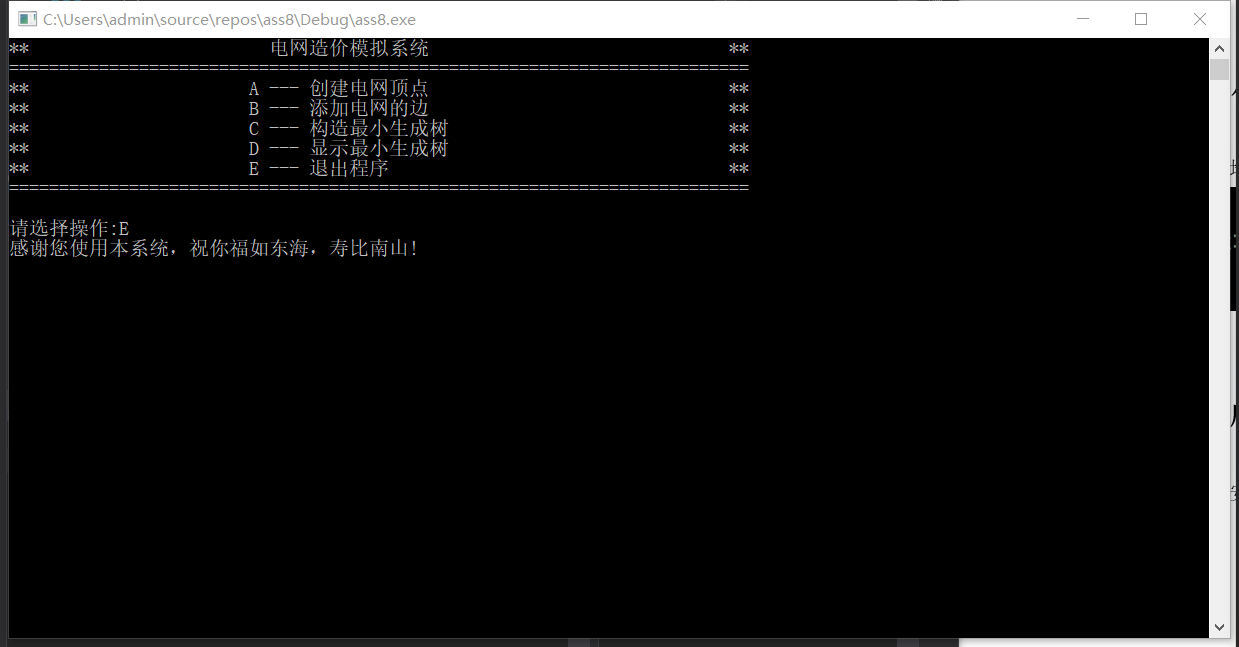
### 4.1.5 显示最小生成树功能测试：

验证是否可以正确地按照要求显示出来最小生成树，实验结果如下：



### 4.1.6 退出程序功能测试：

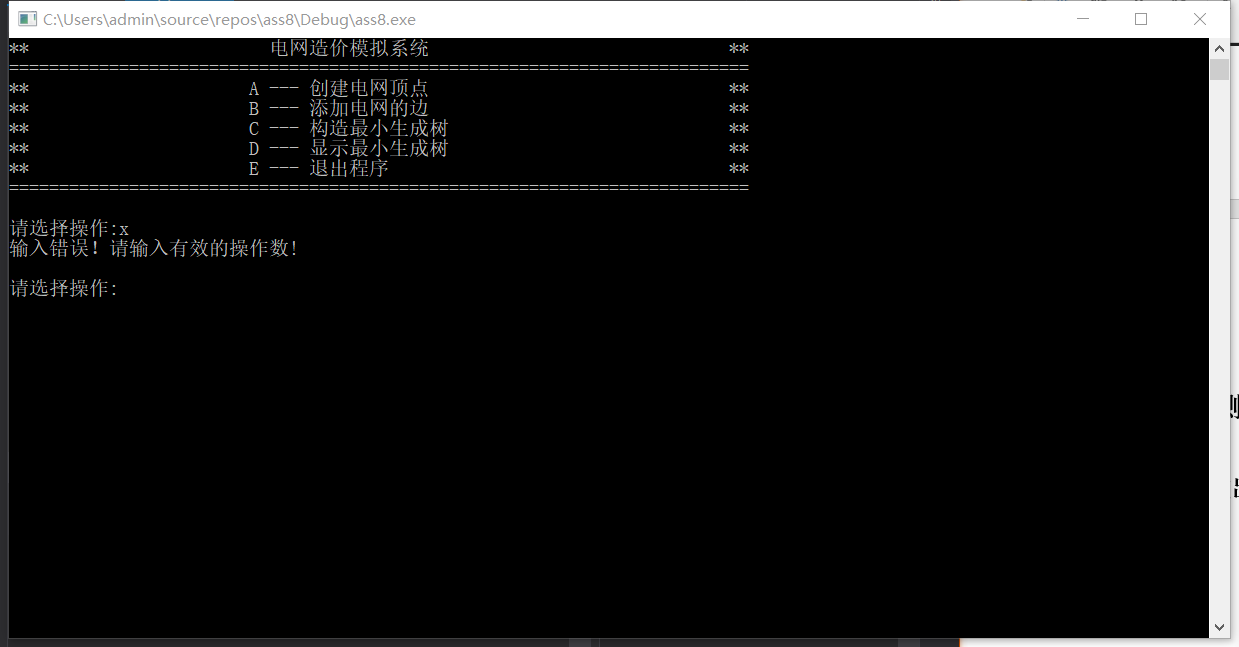
验证是否可以成功安全地退出系统，实验结果如下：



## 4.2 出错性测试

### 4.2.1 操作数出错测试

如果用户输入的操作数不存在的话，系统不会崩溃，并且会提示用户重新输入，实验结果如下：



### 4.2.2 顶点个数出错测试

如果用户输入的顶点数小于2，则无法构成电网，实验结果如下：

