项目说明文档

数据结构课程设计

——电网建设造价系统

作 者 姓 名： 罗佳瑞

学 号： 1952528

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 1](#_Toc2201)

[1.1 背景分析 1](#_Toc19135)

[1.2 功能分析 1](#_Toc15238)

[2 设计 2](#_Toc1080)

[2.1 数据结构设计 2](#_Toc22272)

[2.2 类结构设计 2](#_Toc7747)

[2.3 成员与操作设计 2](#_Toc30485)

[2.4 系统设计 3](#_Toc22656)

[3 实现 4](#_Toc18827)

[3.1 创建功能的实现 4](#_Toc27576)

[3.1.1 创建功能流程图 4](#_Toc17849)

[3.1.2 创建功能核心代码 4](#_Toc21709)

[3.1.3 创建功能截屏示例 5](#_Toc29753)

[3.2 构造最小生成树的实现 6](#_Toc1069)

[3.2.1 构造最小生成树流程图 6](#_Toc19731)

[3.2.2 构造最小生成树核心代码 6](#_Toc32205)

[3.2.3 构造最小生成树截屏示例 9](#_Toc10317)

[3.3 显示最小生成树的实现 9](#_Toc5617)

[3.3.1 显示最小生成树核心代码 9](#_Toc23566)

[3.3.1 显示最小生成树截图示例 10](#_Toc6480)

[3.4 总体系统的实现 10](#_Toc15533)

[3.4.1 总体系统流程图 10](#_Toc25437)

[3.4.2 总体系统核心代码 10](#_Toc5732)

[3.4.3 总体系统截屏示例 12](#_Toc15284)

[4 测试 13](#_Toc13704)

[4.1 功能测试 13](#_Toc23541)

[4.2 边界测试 13](#_Toc31077)

[4.3 出错测试 13](#_Toc18848)

# 1 分析

## 1.1 背景分析

一个城市中，在每个小区之间设置电网线路，都要付出相应的经济代价。为可以实现n个小区之间的电网都能够相互接通，同时使总工程造价最低，现通过电网建设造价系统设计一个能够满足要求的造价方案，构造这个城市n个小区之间的电网。

## 1.2 功能分析

电网造价系统首先应该有的功能就是输入个顶点及边的信息，对电网进行初始化。其次，还应该有生成最小耗费路径的功能，在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。并且能将其显示。最后，电网建设造价系统软件还应该确保软件可以正常打开与关闭关闭。

综上所述，一个考试报名系统至少应该具有输入、构建、输出、退出的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统需要求解最小耗费路径，因此考虑利用最小生成树数据结构，通过最小生成树的构建与遍历得到最小耗费路径。

## 2.2 类结构设计

树结构的链表一般包括两个抽象数据类型——树结点类（edgeNode）与生成树类（Tree），而两个类之间的耦合关系可以采用嵌套关系。为方便处理，本程序采用class描述树结点类（edgeNode），使生成树类（Tree）可以访问树结点；同时，为保证访问的安全，设置唯一运行函数对树类中的函数进行调用。

## 2.3 成员与操作设计

**树结点类（edgeNode）**

**公有成员：**

char tail, head;//生成树各边的两顶点

float cost;//生成树各边的权值

**公有操作：**

void edgeNode::assign(int h, int t, float c);//构建结点

void edgeNode::operator =(edgeNode& R);//重载=

bool edgeNode::operator <(edgeNode& R);//重载<

bool edgeNode::operator >(edgeNode& R);//重载>

**生成树类（Tree）**

**私有成员：**

edgeNode \*root;//生成树的头结点

int n, m;//生成树的顶点数、边数

**私有操作：**

void Tree::create();//初始化树

void Tree::print();//打印函数

void Tree::add(char element, char\* set);//将元素加入集合

void Tree::unite(char\*\* set, int num1, int num2, int setnum);//合并两集合

int Tree::subject(char element, char\*\* set, int num);//查找该元素是否在某集合中

void Tree::Keuskal();//Kruskal算法生成最小生成树

**公有操作：**

void Tree::run();//运行函数

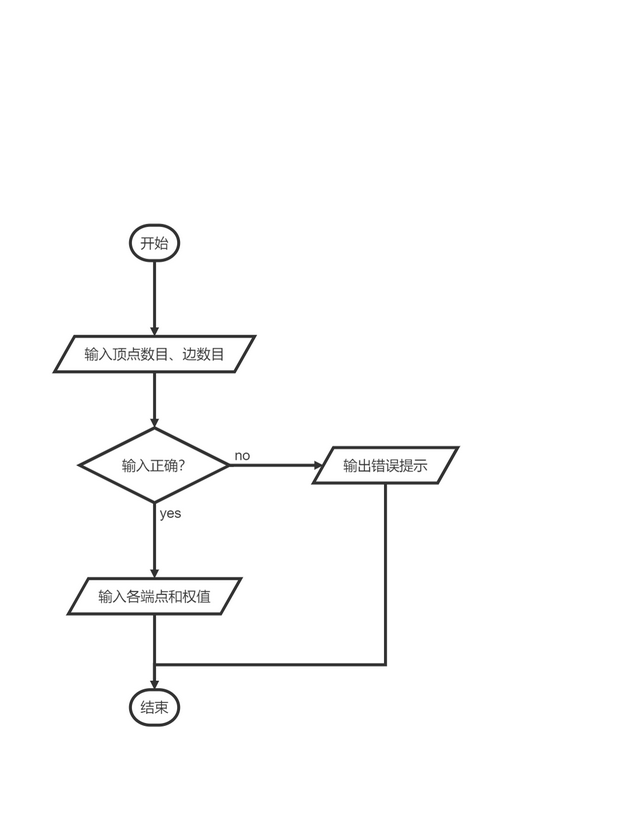
## 2.4 系统设计

系统首先打印提示用户输入操作选择，根据用户所输入的操作码（op）执行树类Tree对应的成员函数。

# 3 实现

## 3.1 创建功能的实现

### 3.1.1 创建功能流程图



### 3.1.2 创建功能核心代码

void Tree::create() {//初始化树

cout << "请输入顶点数目:" ; //输入所求的顶点数目和边数

cin >> n;

cout << "请输入边数目:" ;

cin >> m;

root = new edgeNode[m];

if (m > 0 && n > 0)//判断是否是空树

{

for (int i = 0; i < m; i++)//输入两端点和权值

{

cout << "请输入两个顶点以及边(以空格分隔各个数):";

char head, tail;

float cost;

cin >> head >> tail >> cost;

root[i].assign(head, tail, cost);

}

}

else

{

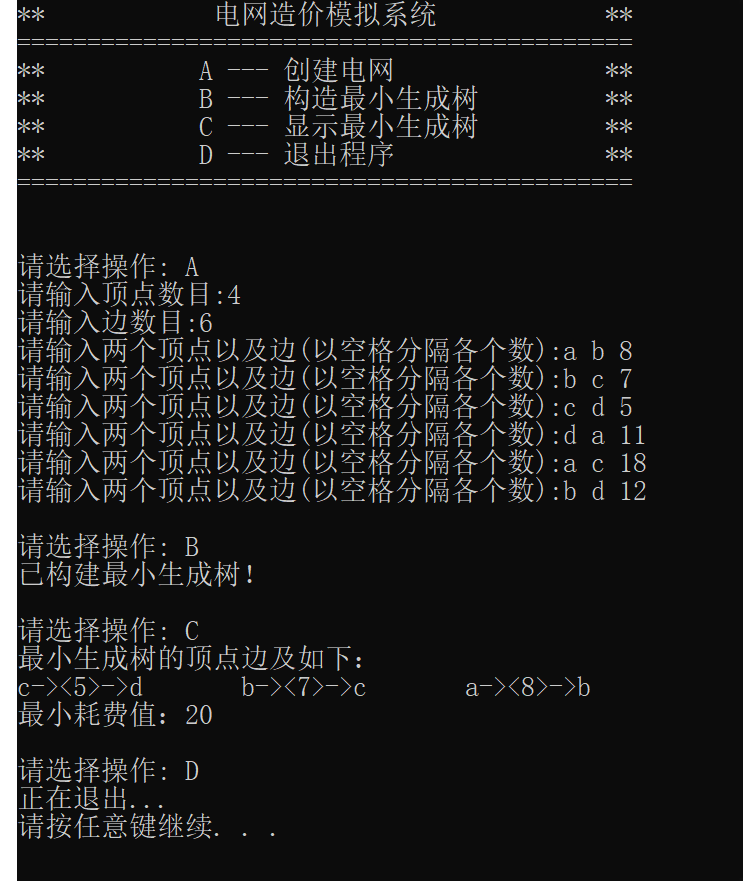
cout << "输入错误" << endl;

exit(1);

}

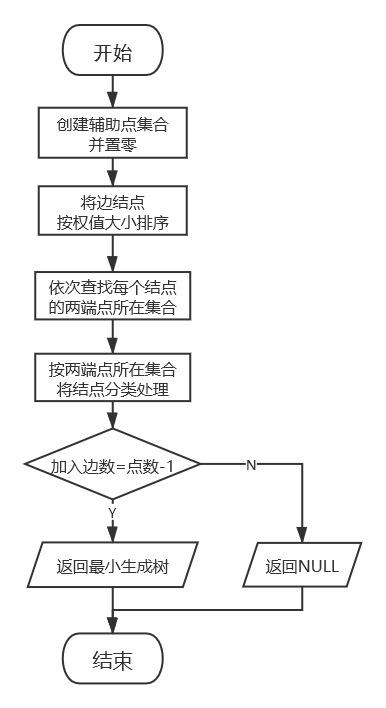
}

### 3.1.3 创建功能截屏示例



## 3.2 构造最小生成树的实现

### 3.2.1 构造最小生成树流程图



### 3.2.2 构造最小生成树核心代码

//从属关系判断

int Tree::subject(char element, char\*\* set, int num) {

int i, j;

for (i = 0;i < num;i++)

for (j = 0; j < n; j++)

if (element == set[i][j])//判断元素属于某个集合

return i;//返回集合序号

return -1;//不属于任何集合返回-1

}

//元素加入到集合中

void Tree::add(char element, char\* set) {

int i;

for (i = 0;i < n;i++)

if (set[i] == 0)

{

set[i] = element;

return;

}

exit(1);

}

//集合合并

void Tree::unite(char\*\* set, int num1, int num2, int setnum) {

int i, j, k, min = (num1 <= num2 ? num1 : num2), max = (num1 > num2 ? num1 : num2);

for (i = 0;i < n;i++)

if (set[min][i] == 0)

{

for (j = 0;set[max][j] != 0 && j < n;i++, j++)

set[min][i] = set[max][j];

for (k = max;k < setnum;k++)

set[k] = set[k + 1];

return;

}

exit(1);}

//Kruskal算法生成最小生成树

void Tree::Keuskal() {

char\*\* node = new char\* [n];

for (int i = 0; i < n; i++)

node[i] = new char[n];//辅助点集合

for (int i = 0;i < n;i++)//数组置零

for (int j = 0;j < n;j++)

node[i][j] = 0;

edgeNode\* minTree = new edgeNode[n - 1];//建立最小树

int set = 0, count = 0;//记集合数、边数（直到n-1）

for (int i = 0;i < m;i++)//按权值将边排序

for (int j = i;j < m;j++)

if (root[j] < root[i])

{

edgeNode temp;

temp = root[i];

root[i] = root[j];

root[j] = temp;

}

for (int i = 0;i < m;i++)

{

if (count < n) {

int h = subject(root[i].head, node, set), t = subject(root[i].tail, node, set);//查找其所属集合

if (h == -1 && t == -1)//两端点均不在集合中，则新建一集合

{

node[set][0] = root[i].head;

node[set][1] = root[i].tail;

set++;

minTree[count] = root[i];

count++;

}

else if (h == t)//两端点在一个集合里，不将该边加入，处理下一条边

continue;

else if (h == -1 && t != -1)//一端点在集合中，另一端点不在，则将另一端点加入

{

add(root[i].head, node[t]);

minTree[count] = root[i];

count++;

}

else if (h != -1 && t == -1)//一端点在集合中，另一端点不在，则将另一端点加入

{

add(root[i].tail, node[h]);

minTree[count] = root[i];

count++;

}

else if (h != -1 && t != -1)//两个端点分别在两个集合里，则将两个集合合并

{

unite(node, h, t, set);

set--;

minTree[count] = root[i];

count++;

}

}

}

if (count == n - 1)

{

root = minTree;

n = count;

}

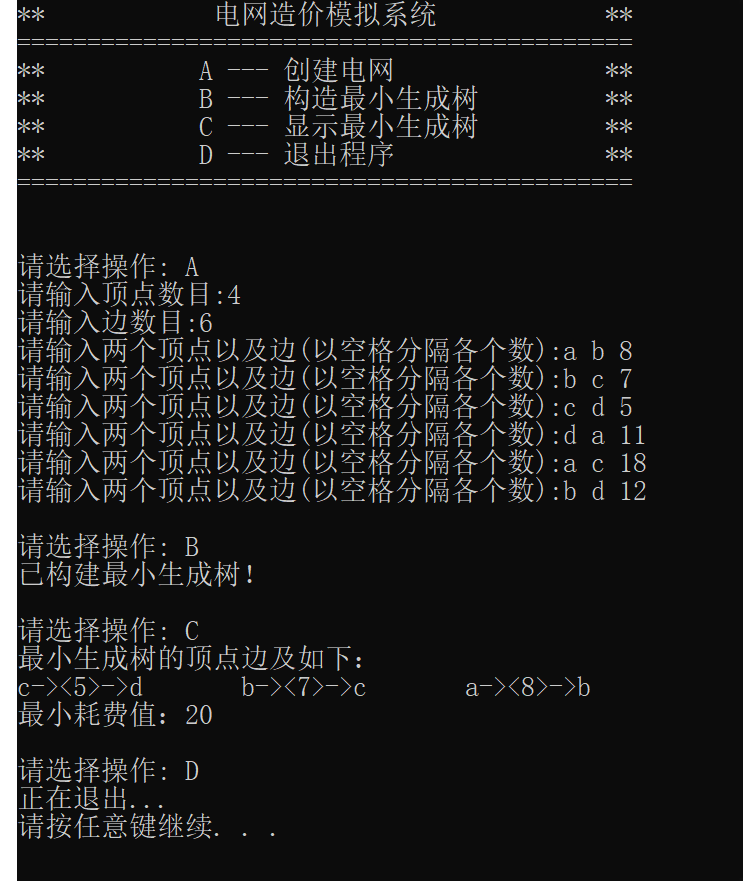
else

root = NULL;

cout << "已构建最小生成树！" << endl;

}

### 3.2.3 构造最小生成树截屏示例



## 3.3 显示最小生成树的实现

### 3.3.1 显示最小生成树核心代码

void Tree::print() {

float sum = 0;

if (root == NULL)

{

cout << "不存在最小生成树" << endl;

exit(1);

}

cout << "最小生成树的顶点边及如下：" << endl;

for (int i = 0;i < n;i++)

{

cout << root[i].head << "-><" <<root[i].cost<<">->"<< root[i].tail <<'\t';

sum += root[i].cost;

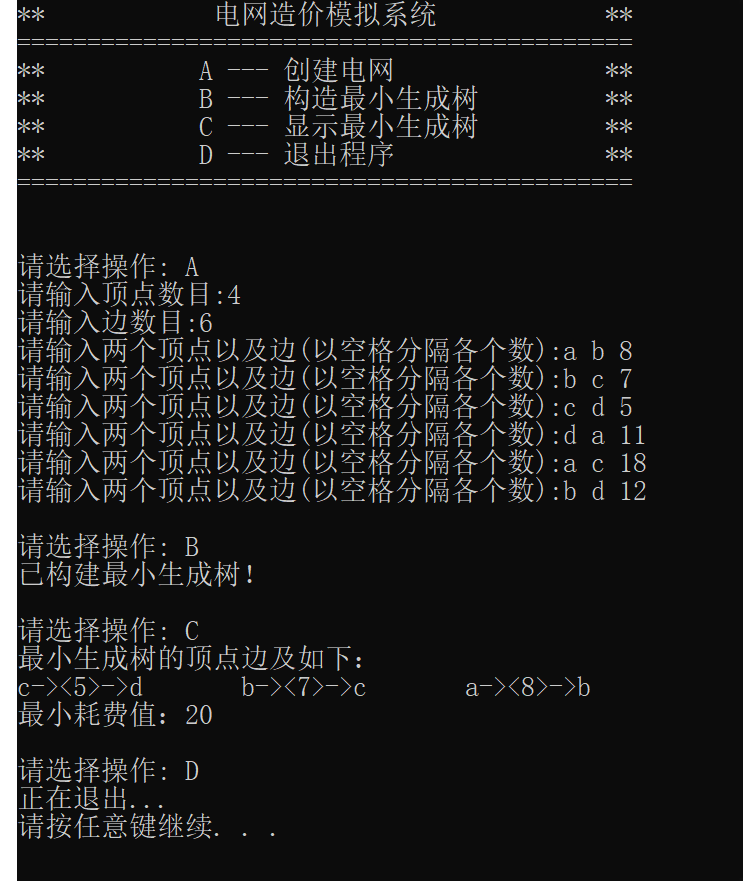
}

cout << "\n最小耗费值：" << sum << endl;

return;

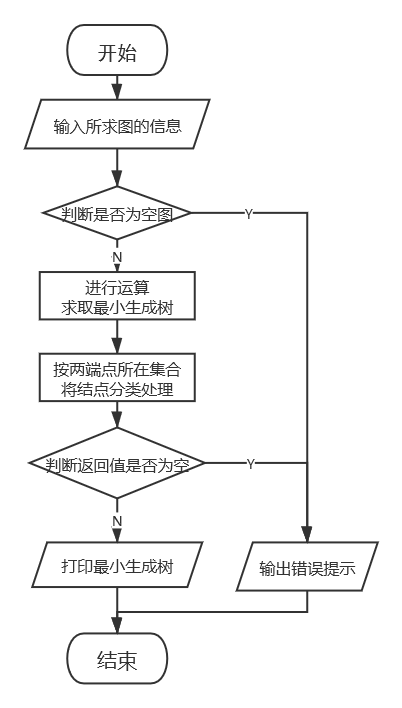
}

### 3.3.1 显示最小生成树截图示例



## 3.4 总体系统的实现

### 3.4.1 总体系统流程图



### 3.4.2 总体系统核心代码

void Tree::run() {

cout << "\*\* 电网造价模拟系统 \*\*" << endl

<< "============================================" << endl

<< "\*\* A --- 创建电网 \*\*" << endl

<< "\*\* B --- 构造最小生成树 \*\*" << endl

<< "\*\* C --- 显示最小生成树 \*\*" << endl

<< "\*\* D --- 退出程序 \*\*" << endl

<< "============================================" << endl

<< endl;

while (true) {

char op;

cout << "\n请选择操作: ";

cin >> op;

if (!cin.good()) {//输入错误

cin.clear();

op = -1;

}

cin.ignore(INT32\_MAX, '\n');

switch (op) {

case 'A':case 'a':create();break;

case 'B':case 'b':Keuskal();break;

case 'C':case 'c':print();break;

case 'D':case 'd':cout << "正在退出..." << endl;return;

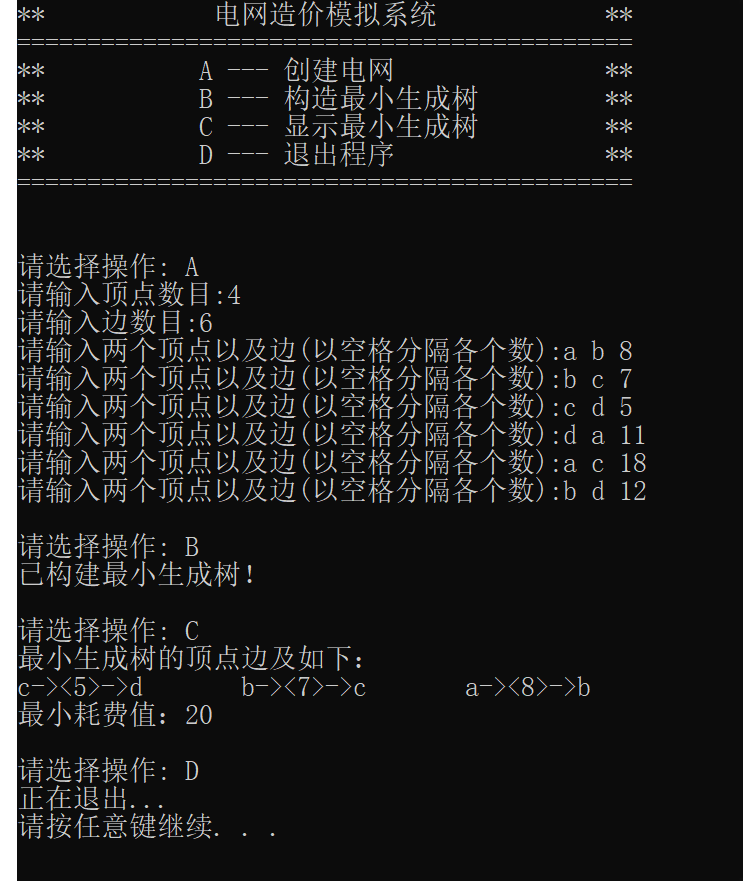
default:cerr << "输入有误，请重新输入!" << endl;break;

}

}

}

### 3.4.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

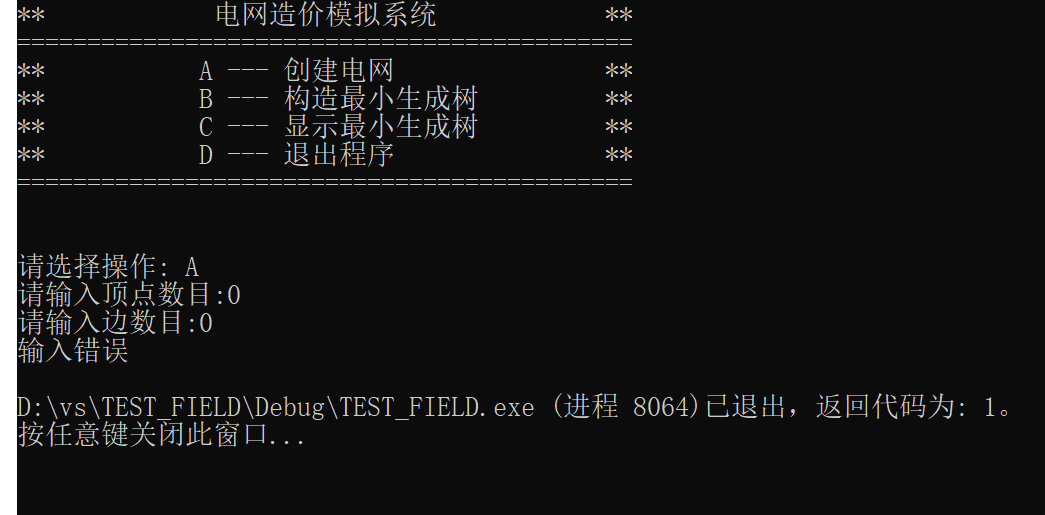
**见各功能截屏示例**

## 4.2 边界测试

**测试用例：**顶点数、边数非正整数

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

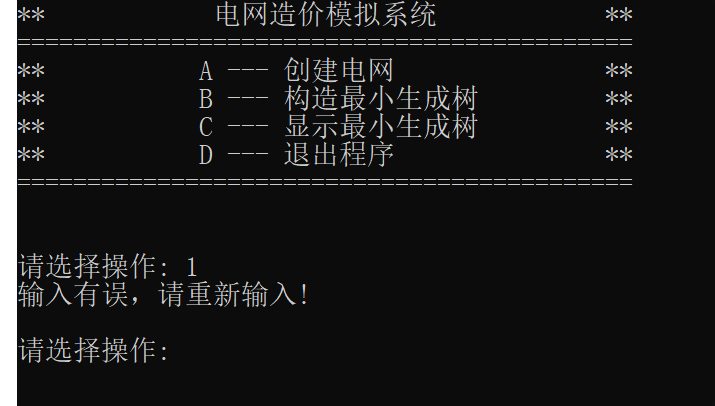


## 4.3 出错测试

**测试用例：**错误操作码

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

****