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： 李圣

学 号： 1952723

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji Universit

目录

[1 分析 1](#_Toc20978)

[1.1 背景分析 1](#_Toc17946)

[1.2 功能分析 1](#_Toc574)

[2 设计 1](#_Toc2284)

[2.1 文件结构设计 1](#_Toc29889)

[2.2 数据结构设计 1](#_Toc24081)

[2.3 类结构设计 1](#_Toc22118)

[2.4 成员与函数设计 2](#_Toc24717)

[3 实现 7](#_Toc30933)

[3.1 添加顶点功能的实现 7](#_Toc6965)

[3.1.1 添加顶点功能流程图 7](#_Toc1051)

[3.1.2 添加顶点功能核心代码 8](#_Toc25671)

[3.1.3 添加顶点功能截屏示例 8](#_Toc4698)

[3.2 添加边的实现 9](#_Toc32463)

[3.2.1 添加边功能流程图 9](#_Toc28181)

[3.2.2 添加边功能核心代码 10](#_Toc30790)

[3.2.3 添加边功能截屏示例 11](#_Toc12253)

[3.3 最小生成树功能的实现 12](#_Toc11270)

[3.3.1 最小生成树功能流程图 12](#_Toc7787)

[3.3.2 最小生成树功能核心代码 13](#_Toc25486)

[3.3.3 最小生成树功能截屏示例 14](#_Toc3169)

[3.4 显示最小生成树功能的实现 15](#_Toc421)

[3.4.1 显示最小生成树功能流程图 15](#_Toc2382)

[3.4.2 显示最小生成树功能核心代码 15](#_Toc22498)

[3.4.3 显示最小生成树功能截屏示例 16](#_Toc10159)

[4 测试 16](#_Toc287)

[4.1 鲁棒性测试 16](#_Toc26615)

[4.1.1 正常案例测试 16](#_Toc12516)

[4.1.2 错误指令测试 18](#_Toc6381)

[4.1.3 funA测试 18](#_Toc9937)

[4.1.4 funB测试 19](#_Toc16350)

[4.1.5 funC测试 22](#_Toc11903)

[4.1.6 funD测试 23](#_Toc28530)

# 分析

## 背景分析

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。

## 功能分析

考试报名系统至少应该具有的基本功能：

1. 输入命令
2. 设置电网
3. 构造电网
4. 显示电网

# 设计

## 文件结构设计

表达式求解的过程中需要使用Vector，因此该项目包含两个文件，如下：

main.cpp

MyVector.h

其中main.cpp是主程序，且main.cpp调用了MyVector.h

## 数据结构设计

## 类结构设计

考虑到功能需要，除了Vector相关的类以外，拟定设计以下类：

class Graph

class Side

class MST

class Guide

其中Graph是图类，Side是边类，MST是最小生成树类，Guide是输入指引类

## 成员与函数设计

**MyVectorIterator类：**

template<typename val\_type>

using iterator = MyVectorIterator;

using pointer = val\_type\*;

using reference = val\_type&;

**私有成员：**

pointer \_ptr;

int \_pos;

**公有成员：**

MyVectorIterator(pointer ptr = nullptr, int pos = -1)；

~MyVectorIterator()；

int getPos()；

bool operator==(const iterator& it)const；

bool operator!=(const iterator& it)const；

iterator& operator++()；

iterator& operator++(int)；

iterator& operator--()；

iterator& operator--(int)；

reference operator\*()；

pointer operator->()；

**MyVector类：**

template<typename val\_type>

using vector = MyVector<val\_type>;

using reference = val\_type&;

using iterator = MyVectorIterator<val\_type>;

**私有成员：**

int \_size;

int \_capacity;

val\_type\* \_data;

void enlarge(int newcapacity);

**公有成员：**

MyVector(int capacity = DEFAULT\_CAPACITY)；

MyVector(const vector& vec)；

~MyVector()；

bool empty()；

int size()；

int length()；

int capacity()；

void resize(int newsize,val\_type val=val\_type())；

void insert(iterator pos, const val\_type& val)；

val\_type erase(iterator pos)；

void push\_back(val\_type val)；

val\_type pop\_back()；

vector& operator=(vector& vec)；

vector& operator=(vector&& vec)；

iterator find(val\_type val)；

reference at(int pos)；

reference operator[](int pos)；

reference front()；

reference back()；

iterator begin()；

iterator end()；

void clear()；

**Side类：**

**私有成员：**

string \_start;

string \_end;

int \_val;

**公有成员：**

Side();

Side(string start, string end, int val);

friend ostream& operator<<(ostream& os, Side side);

**MST类：**

**私有成员：**

MyVector<Side> \_sides;

**公有成员：**

MST();

friend ostream& operator<<(ostream& os, MST mst);

void clear();

void push\_back(Side side);

bool empty();

**Graph类：**

**私有成员：**

int \_size;

MyVector<string> \_points;

MyVector<MyVector<int>> \_sides;

MST \_mst;

**公有成员：**

Graph(int size = 0);

void clear();

void resize(int size);

void addSide(string pa, string pb, int val);

bool findMST(string pbegin);

MST getMST();

void showPoints();

**Guide类：**

**私有成员：**

Graph \_graph;

**公有成员：**

Guide();

void guider();

bool loop();

void funA();

void funB();

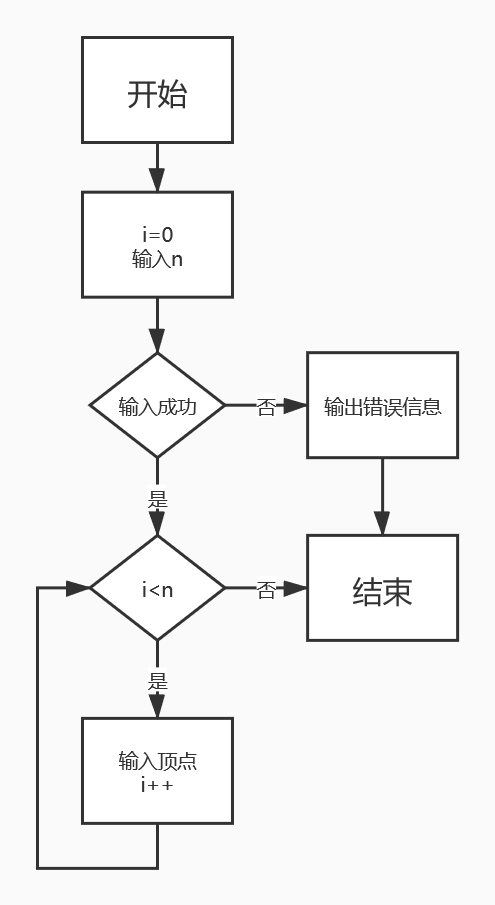
void funC();

void funD();

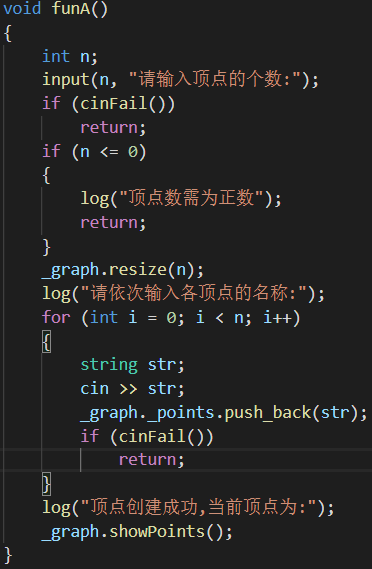
# 实现

## 添加顶点功能的实现

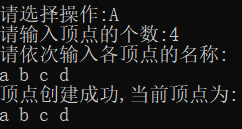
### 添加顶点功能流程图



### 添加顶点功能核心代码

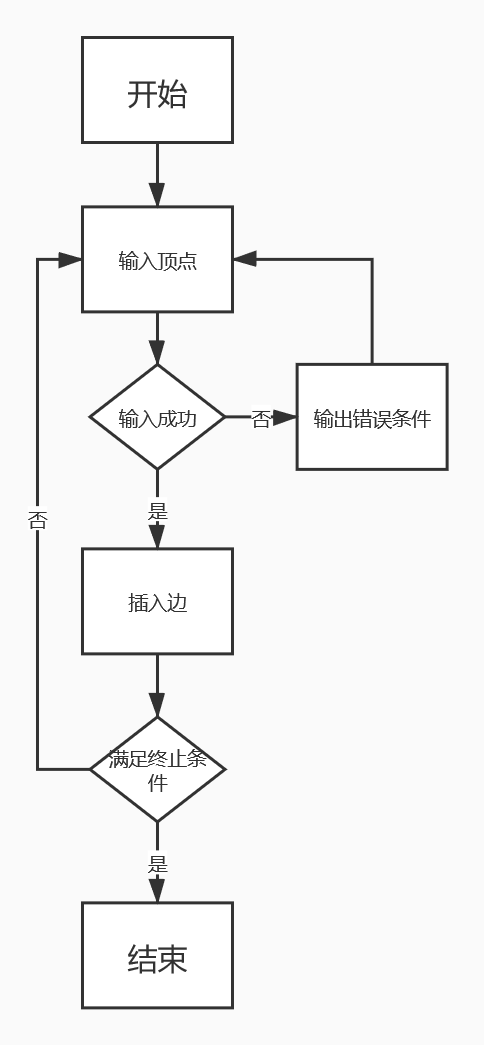


### 添加顶点功能截屏示例



## 添加边的实现

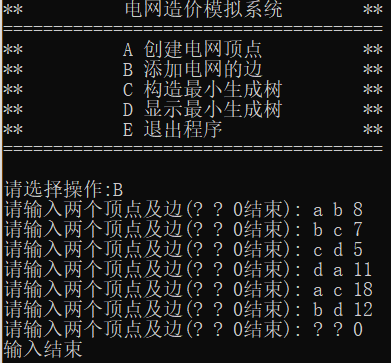
### 添加边功能流程图



### 添加边功能核心代码

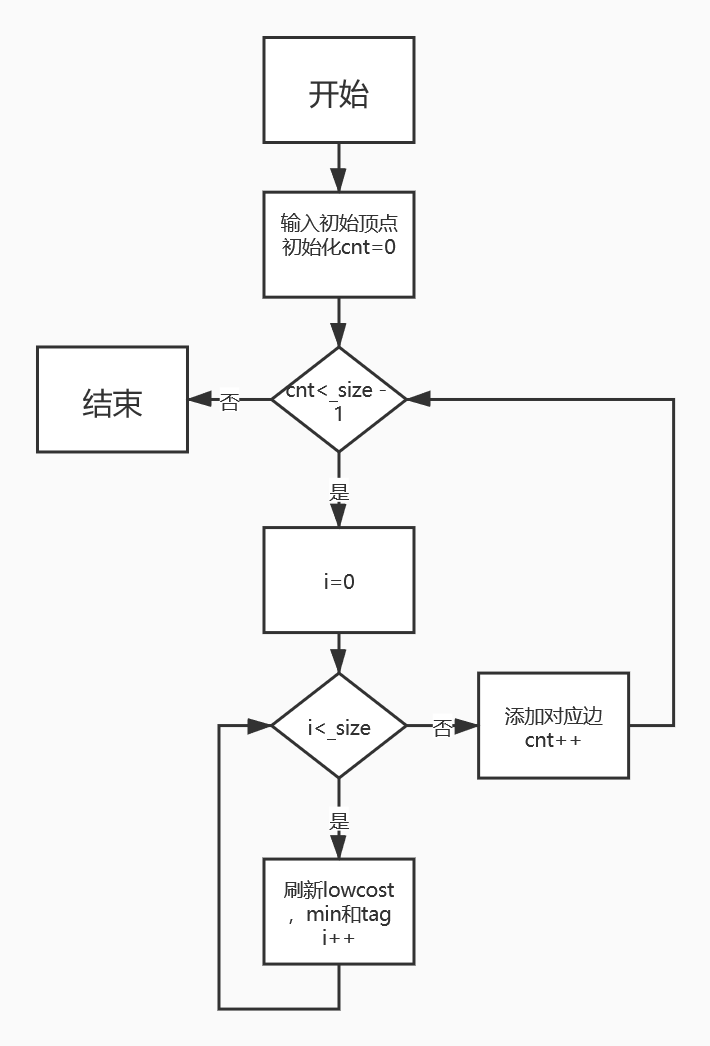


### 添加边功能截屏示例

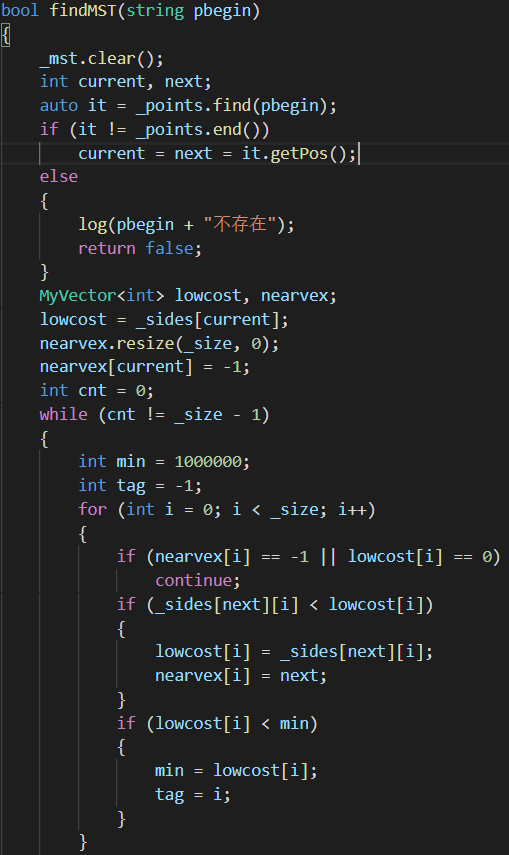


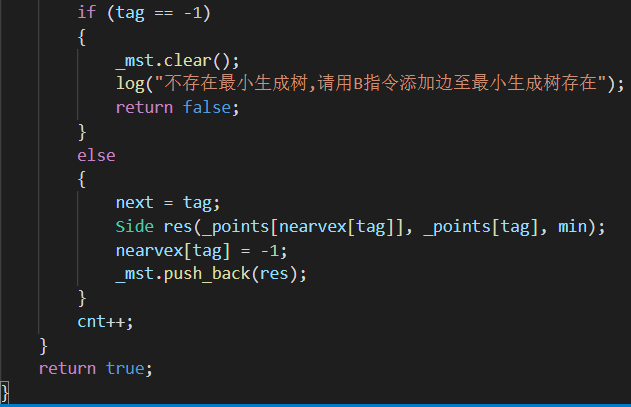
## 最小生成树功能的实现

### 最小生成树功能流程图

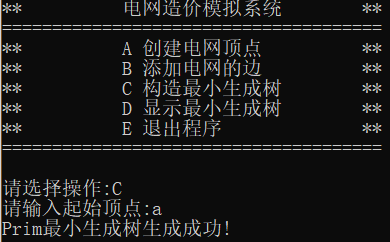


### 最小生成树功能核心代码



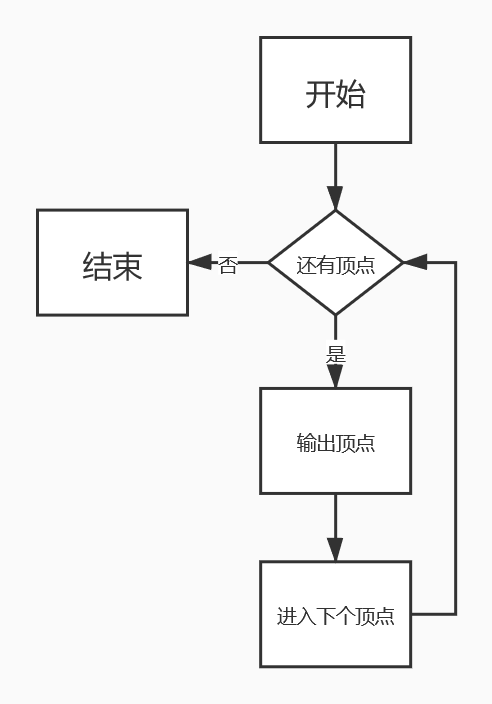


### 最小生成树功能截屏示例

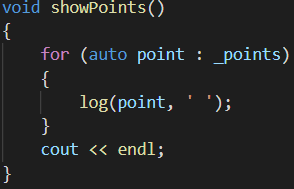


## 显示最小生成树功能的实现

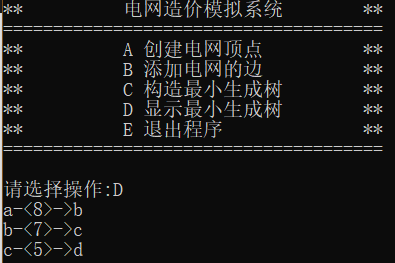
### 显示最小生成树功能流程图



### 显示最小生成树功能核心代码



### 显示最小生成树功能截屏示例



# 测试

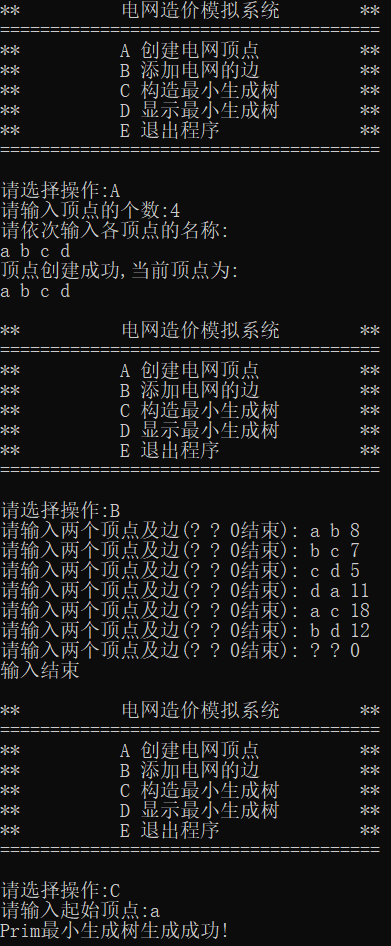
## 鲁棒性测试

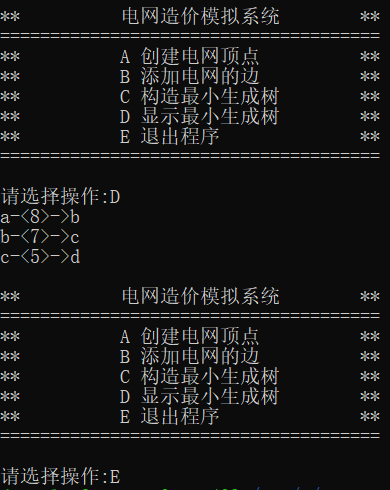
### 正常案例测试

**测试用例：**一般情况

**预期结果：**对应的正确结果。

**实验结果：**



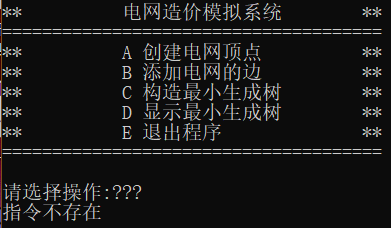


### 错误指令测试

**测试用例：**指令错误

**预期结果：**给出提示，程序继续进行

**实验结果：**

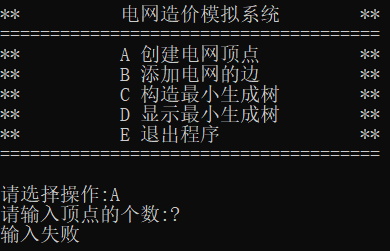


### funA测试

**测试用例：**个数输入错误

**预期结果：**给出提示，程序继续进行

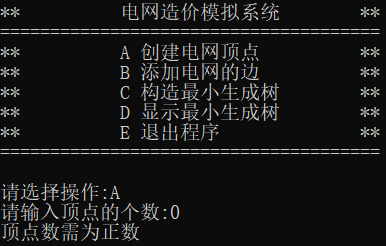
**实验结果：**



**测试用例：**个数不为正数

**预期结果：**给出提示，程序继续进行

**实验结果：**

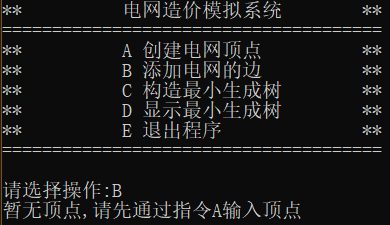


### funB测试

**测试用例：**暂无顶点

**预期结果：**给出提示，程序继续进行

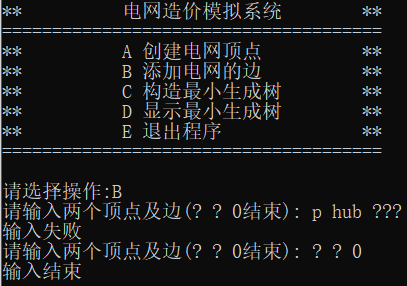
**实验结果：**



**测试用例：**边长错误

**预期结果：**给出提示，程序继续进行

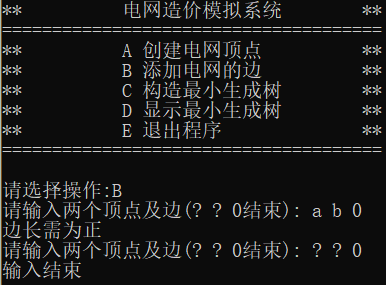
**实验结果：**



**测试用例：**边长不为正

**预期结果：**给出提示，程序继续进行

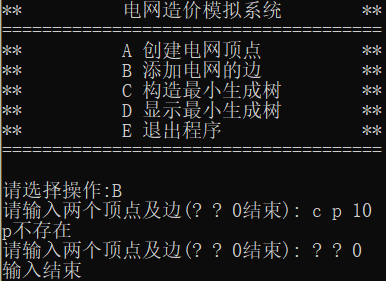
**实验结果：**



**测试用例：**顶点不存在

**预期结果：**给出提示，程序继续进行

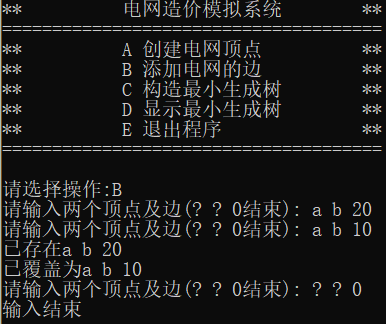
**实验结果：**



**测试用例：**边已存在

**预期结果：**给出提示，对边进行覆盖，程序继续进行

**实验结果：**

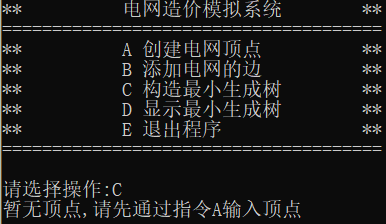


### funC测试

**测试用例：**暂无顶点

**预期结果：**给出提示，程序继续进行

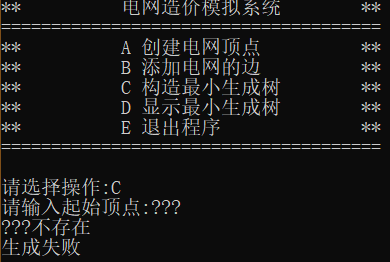
**实验结果：**



**测试用例：**顶点未找到

**预期结果：**给出提示，程序继续进行

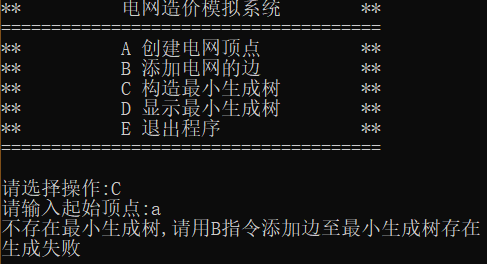
**实验结果：**



**测试用例：**生成失败

**预期结果：**给出提示，程序继续进行

**实验结果：**



### funD测试

**测试用例：**生成树不存在

**预期结果：**给出提示，程序继续进行

**实验结果：**

