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： 张诚睿

学 号： 2150998

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](#_Toc122644775)

[1.1 项目背景分析 1](#_Toc122644776)

[1.2 功能分析 1](#_Toc122644777)

[2 设计 1](#_Toc122644778)

[2.1 数据结构设计 1](#_Toc122644779)

[2.2 类具体成员与操作设计 2](#_Toc122644780)

[2.4 main函数设计 4](#_Toc122644781)

[3 实现 4](#_Toc122644782)

[3.1 创建电网顶点功能的实现 4](#_Toc122644783)

[3.1.1 功能流程图 4](#_Toc122644784)

[3.1.2 核心代码 5](#_Toc122644785)

[3.1.3 效果演示 5](#_Toc122644786)

[3.2 添加电网的边功能的实现 6](#_Toc122644787)

[3.2.1 功能流程图 6](#_Toc122644788)

[3.2.2 核心代码 6](#_Toc122644789)

[3.2.3 效果演示 7](#_Toc122644790)

[3.3 构造最小生成树功能的实现 7](#_Toc122644791)

[3.3.1 功能流程图 7](#_Toc122644792)

[3.3.2 核心代码 8](#_Toc122644793)

[3.3.3 效果演示 8](#_Toc122644794)

[3.4 显示最小生成树功能的实现 9](#_Toc122644795)

[3.4.1 功能流程图 9](#_Toc122644796)

[3.4.2 核心代码 9](#_Toc122644797)

[3.4.3 效果演示 9](#_Toc122644798)

[4 测试 10](#_Toc122644799)

[4.1 功能测试 10](#_Toc122644800)

[4.1.1求解一般电网最小生成树功能测试 10](#_Toc122644801)

[4.2 边界测试 11](#_Toc122644802)

[4.2.1 整个图已经是最小生成树 11](#_Toc122644803)

[4.2.2 有的点不在起点开始的最小生成树里 11](#_Toc122644804)

[4.3 出错测试 11](#_Toc122644805)

[4.3.1 顶点个数不是正整数 11](#_Toc122644806)

[4.3.2 边的权值为负数或字符 12](#_Toc122644807)

[4.3.3 输入顶点名字不在图里 12](#_Toc122644808)

[4.3.5 添加的边已经被添加过，有重复 13](#_Toc122644809)

[4.3.6 起始顶点不存在 13](#_Toc122644810)

# 1 分析

## 1.1 项目背景分析

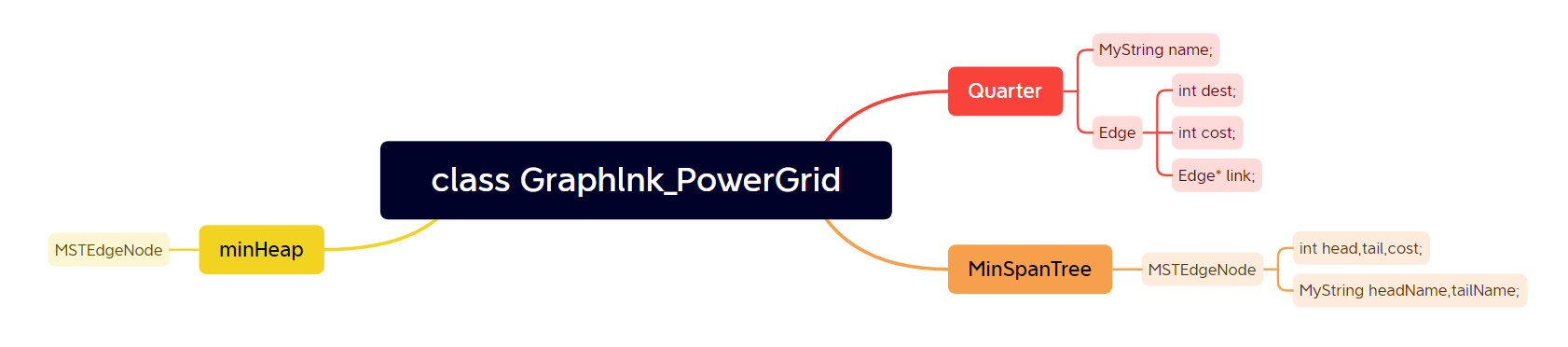
假设一个城市有n个小区，要实现n个小区之间的电网都能够相互接通，构造这个城市n个小区之间的电网，使总工程造价最低。在此基础上，要设计一个能够满足要求的造价方案来为实际工程作参考。

## 1.2 功能分析

本项的目的在于实现对输入的小区和电网造价进行计算并求解出连接所有小区的造价最低的电网网络，输入者应当做适当抽象使小区名称要求不能过长。求解完成后，可以输出以某一小区开始的Prim最小生成树，使用过程中可以添加新的带权路径或更换初始顶点来生成新的Prim生成树。

# 2 设计

## 2.1 数据结构设计



核心数据结构设计如上，通过类-Graphlnk\_PowerGrid实现总体操作，用Quarter类型的动态申请来的数组及后续Edge搭建的链表存放图的邻接表，实现高效的读取使用；类内还包括一个MinSpanTree来存放最小生成树，每条边都用MSTEdgeNode来存储表示，方便快速获取顶点名字和其在Quarter数组的地址以及边的权值。另有一个最小堆类-minHeap，实现以权值为特征码的MSTEdgeNode类型数据的最小堆处理，在求解最小生成树时用于挑选权值最小边。

## 2.2 类具体成员与操作设计

/\*

实现不长于10个字符的字符串操作

\*/

class MyString {

private:

char content[10]; // 内容指针

public:

MyString()

{

for (int i = 0; i < 10; i++)

{

content[i] = '\0';

}

};// 无参

friend ostream& operator<<(ostream& os, MyString& target); // 输出内容

friend istream& operator>>(istream& is, MyString& target); // 读取内容，遇空格、回车停止

bool operator==(const MyString& matched); // 判断两字符串是否一致

bool operator==(const char\* str);

};

/\*

\* 边节点

\*/

struct Edge {

int dest; // 边另一个顶点位置

int cost; // 边权值—花费

Edge\* link = NULL; // 下一条边链指针

};

/\*

\* 小区顶点

\*/

struct Quarter {

MyString name; // 小区名称，不限长度与中英文、符号等

Edge\* adj = NULL;

};

/\*

\* 最小生成树的边

\*/

struct MSTEdgeNode {

int head, tail; // 头尾顶点

MyString headName, tailName;

int cost; // 权值

};

/\*

\* 最小堆

\*/

class minHeap {

private:

MSTEdgeNode\* heap;

int size; // 所有的边的总数，来规定堆的大小

int last; // 已作为堆使用的数组的最后一位

public:

minHeap() :heap(NULL), size(0), last(0) {};

~minHeap() { delete heap; };

void init(int maxSize);

void Insert(MSTEdgeNode item); // 添加进最小堆

bool Remove(MSTEdgeNode& out); // 取出

};

class MinSpanTree {

private:

MSTEdgeNode\* content; // 动态申请结果

int size; // n-1条边

int Current; // 当前位置

public:

MinSpanTree() :content(NULL), size(0), Current(0) {};

~MinSpanTree() { delete content; };

void init(int max);

void Insert(MSTEdgeNode target);

void Print();

void clean(); // 清空已有内容，重新初始化

};

class Graphlnk\_PowerGrid {

private:

int size;

Quarter\* NodeTable; // 动态申请顶点表

MinSpanTree answer;

int EdgeSize;

bool NoEdge(int VF, int VS);

bool SearchName(MyString target, int& Pos);

public:

Graphlnk\_PowerGrid() :size(0), NodeTable(NULL), answer(), EdgeSize(0) {};

~Graphlnk\_PowerGrid() { free(NodeTable); };

void init();

void CreateVertex(); // 创建顶点

void InsertEdge(); // 添加边

void Prim(); // 构造最小生成树

void ShowTree(); // 显示最小生成树

void Insert(int VF\_Pos, int VS\_Pos, int cost);

};

## 2.4 main函数设计

首先建立Graphlnk\_PowerGrid对象QuarterGrid并初始化，之后要求用户输入指令，根据指令选择不同功能的函数或退出程序，若输入为非法指令，会要求重新输入直至正确。

# 3 实现

## 3.1 创建电网顶点功能的实现

### 3.1.1 功能流程图

### 3.1.2 核心代码

NodeTable = new Quarter[size];

if (NodeTable==NULL)

{

cout << "No Memory" << endl;

exit(EXIT\_FAILURE);

}

cout << "请依次输入各顶点的名称:" << endl;

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

{

if (NodeTable[i].name == "\0") // 只输入新顶点名字

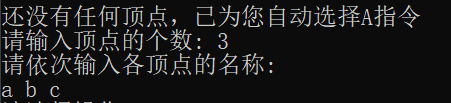
{

cin >> NodeTable[i].name;

}

}

### 3.1.3 效果演示



## 3.2 添加电网的边功能的实现

### 3.2.1 功能流程图

### 3.2.2 核心代码

while (true)

{

cout << "请输入两个顶点及边权值: " << endl;

MyString VertexF, VertexS;

int cost,VF\_Pos=0,VS\_Pos=0;

cin >> VertexF >> VertexS >> cost;

if (!cin.good() || cost < 0)

{

cout << "输入有误，请重新输入" << endl;

continue;

}

if (SearchName(VertexF,VF\_Pos) && SearchName(VertexS,VS\_Pos) && NoEdge(VF\_Pos,VS\_Pos)) // 名字都存在，且无边

{

/\* 无向图，来回添加一遍 \*/

Insert(VF\_Pos, VS\_Pos, cost);

Insert(VS\_Pos, VF\_Pos, cost);

EdgeSize++;

}

else

{

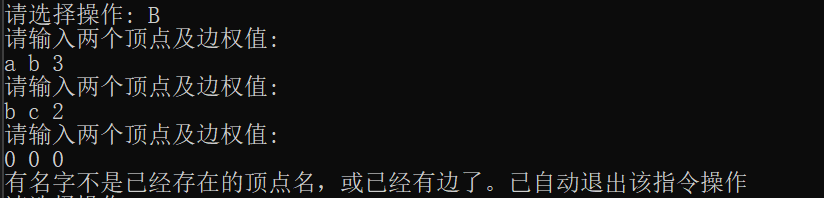
cout << "有名字不是已经存在的顶点名，或已经有边了。已自动退出该指令操作" << endl;

return;

}

}

### 3.2.3 效果演示



## 3.3 构造最小生成树功能的实现

### 3.3.1 功能流程图

### 3.3.2 核心代码

while (count <= size - 1)

{

Edge\* tmp = NodeTable[startPos].adj;

/\* 把startPos的所以边加入到堆中 \*/

while (tmp != NULL)

{

if (!Vmst[tmp->dest])

{

MSTEdgeNode ReadyToHeap;

ReadyToHeap.cost = tmp->cost;

ReadyToHeap.head = startPos;

ReadyToHeap.tail = tmp->dest;

ReadyToHeap.headName = NodeTable[startPos].name;

ReadyToHeap.tailName = NodeTable[tmp->dest].name;

Heap.Insert(ReadyToHeap);

}

tmp = tmp->link;

}

/\* 从堆中取出堆顶，取出的边加入结果中 \*/

MSTEdgeNode out;

while (Heap.Remove(out))

{

/\* 取出的元素需要判断是否可以加入结果 \*/

if (Vmst[out.head] && !Vmst[out.tail])

{ // 可以加入结果

answer.Insert(out);

Vmst[out.tail] = true;

count++;

startPos = out.tail;

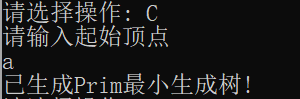
break;

}

}

}

### 3.3.3 效果演示



## 3.4 显示最小生成树功能的实现

### 3.4.1 功能流程图

### 3.4.2 核心代码

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

{

cout << " " << content[i].headName << "-<" << content[i].cost << ">->" << content[i].tailName;

}

cout << endl;

### 3.4.3 效果演示



# 4 测试

## 4.1 功能测试

### 4.1.1求解一般电网最小生成树功能测试

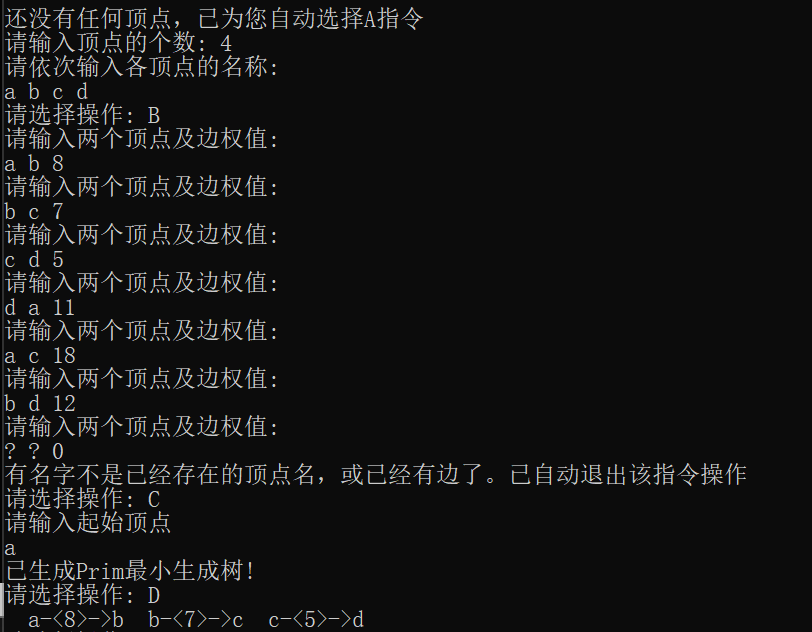
**测试用例**：

4个顶点a、b、c、d；权值分别为a b 8、b c 7、c d 5、d a 11、a c 18、b d 12；以a为起始顶点

**预期结果**：

a-<8>->b b-<7>->c c-<5>->d

**实验结果:**



## 4.2 边界测试

### 4.2.1 整个图已经是最小生成树

**测试用例：**

4个顶点a、b、c、d；权值分别为a b 8、b c 7、c d 5；以a为起始顶点

**预期结果：**

a-<8>->b b-<7>->c c-<5>->d

**实验结果：**



### 4.2.2 有的点不在起点开始的最小生成树里

**测试用例：**

4个顶点a、b、c、d；权值分别为a b 8、b c 7；以a为起始顶点（d不被连接）

**预期结果：**

提示添加新的边直至所有点都在起点开始的最小生成树里

**实验结果：**



## 4.3 出错测试

### 4.3.1 顶点个数不是正整数

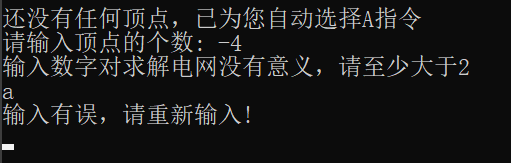
**测试用例：**

-4 a

**预期结果：**

输出提示信息，要求重新输入直至正确，程序不崩溃

**实验结果：**



### 4.3.2 边的权值为负数或字符

**测试用例：**

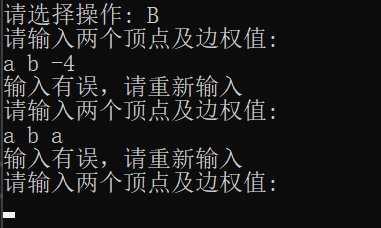
a b -4

a b a

**预期结果：**

输出提示信息，要求重新输入直至正确，程序不崩溃

**实验结果：**



### 4.3.3 输入顶点名字不在图里

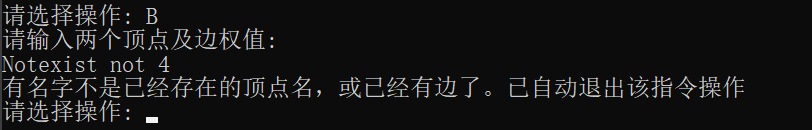
**测试用例：**

Notexist not 4

**预期结果：**

输出提示信息，自动退出该指令，程序不崩溃

**实验结果：**



### 4.3.5 添加的边已经被添加过，有重复

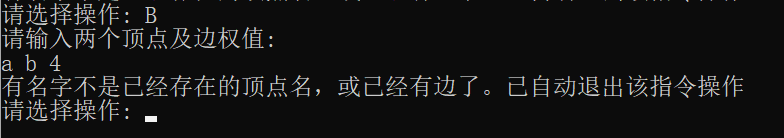
**测试用例：**

a b 4

**预期结果：**

输出提示信息，自动退出该指令，程序不崩溃

**实验结果：**



### 4.3.6 起始顶点不存在

**测试用例：**

5

**预期结果：**

输出提示信息，要求重新输入直至正确，程序不崩溃

**实验结果：**

