项目说明文档

数据结构课程设计

——电网建设造价模拟系统

同济大学

Tongji University

目 录

[1 分析 1](#_Toc60161245)

[1.1 项目简介 1](#_Toc60161246)

[1.2 功能分析 1](#_Toc60161247)

[2 设计 1](#_Toc60161248)

[2.1 数据结构设计 1](#_Toc60161249)

[2.2 类结构设计 2](#_Toc60161250)

[2.3 成员与操作设计 2](#_Toc60161251)

[2.4 系统设计 3](#_Toc60161252)

[3 实现 4](#_Toc60161253)

[3.1 图构造函数的实现 4](#_Toc60161254)

[3.1.1 图构造函数流程图 4](#_Toc60161255)

[3.1.2 图构造函数核心代码 5](#_Toc60161256)

[3.2 prim算法的实现 6](#_Toc60161257)

[3.2.1 prim算法的原理和思想： 6](#_Toc60161258)

[3.2.2 prim算法流程图 7](#_Toc60161259)

[3.2.3 prim 算法代码 9](#_Toc60161260)

[4 测试 10](#_Toc60161261)

[4.1 功能测试 10](#_Toc60161262)

[4.1.1 普通测试1 10](#_Toc60161263)

[4.1.2 普通测试2 11](#_Toc60161264)

[4.1.3 普通测试3 13](#_Toc60161265)

[4.2 边界测试 16](#_Toc60161266)

[4.2.1 平凡图 16](#_Toc60161267)

[4.2.2 不连通图 16](#_Toc60161268)

[4.3 出错测试 17](#_Toc60161269)

[4.3.1输入顶点数/边数错误 17](#_Toc60161270)

[4.3.2 输入非法数据 18](#_Toc60161271)

[4.3.3 输入顶点不存在 18](#_Toc60161272)

# 1 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

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。

所给数据构成一个无向网，要求这个无向网的最小生成树，可以采用prim算法和kruskal算法，本程序采用kruskal算法。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，本项目需要读入一个无向网，所以需要先建立一个图结构，之后在图结构的基础上调用prim算法解决问题。为了程序编写方便，在存储结构上采用邻接矩阵。

## 2.2 类结构设计

为了处理最小生成树问题，本程序在普通的图类中新增了一个closedge结构体，用以存储相邻顶点名称以及该顶点到已连通顶点集U的最小距离。(如果U在该顶点集内，则距离置为0)

## 2.3 成员与操作设计

**最短边类（closedge）**

**成员：**

char adjvex;

//该下标所指顶点与连通集U取最小距离是的相邻顶点

//如果目前没有与连通集U相连，则默认是起始顶点

int lowcost;

//该下标所指顶点与连通集U的最小距离

//不连通则默认是无穷大

**图类（Graph）**

**私有成员：**

char \*vexs; //储存顶点的名称

int \*arcs; //储存边的长度

closedge\* close; //储存与当前已连通顶点集的最短相邻路径

int vexnum, arcnum; //储存顶点数量，边的数量

**公有操作：**

int LocateVex(char);

//根据顶点名称找到顶点的下标

Graph();

//构造函数，输入顶点数，边，边数和边，由此构建出一个图

Graph(const Graph&);

//赋值构造函数

void MiniSpanTree\_Prim();

//使用prim算法求最小生成树

## 2.4 系统设计

打印标语，通过用户输入构建图，调用prim算法求出最小生成树并输出，最后打印结束标语。

# 3 实现

## 3.1 图构造函数的实现

### 3.1.1 图构造函数流程图

开始

读入顶点数

申请内存

将临界矩阵先赋值为无穷大

读入顶点名称

读入边数

读入边

结束

说明：该过程看似简单，其实包含一些细节问题。逻辑上的细节，应先将邻接矩阵的所有值赋为无穷大，再读入边，作为非无穷大的边，用户还有可能输入自环，在图的理论上，这自然是允许的，但是在实际工程上，这是不可能的（为什么要在小区内部修从自己连到自己的电网），此时很有可能是用户手误导致错误输入，故应该输出提示信息；还有可能用户多次同时输入同一条边，同时输入的长度不同，在实际中，可能是代表有多种不同的修法；但也有可能是用户手误，故也应该给出提示信息，并且取多次输入的最小值作为边长。

同时，用户输入的顶点数，边数等参量可能不合法，在输入边时，两端的顶点可能并不存在，这些都应该给出提示信息，并且让用户重新输入。

### 3.1.2 图构造函数核心代码

Graph()

{

printf("请输入顶点数：\t");

vexnum = readInt(range);

//输入顶点数并检查读写

vexs = new char[vexnum];

close = new closedge[vexnum];

arcs = new int[vexnum \* vexnum];

//申请内存

for (int i = 0; i < vexnum \* vexnum; ++i)arcs[i] = INFINITY;

//邻接矩阵赋值为无穷

printf("请依次输入顶点的名称（用单字符表示）：\n");

for (int i = 0; i < vexnum; ++i) { cin>>vexs[i]; }

printf("请输入边数：\t");

arcnum = readInt(range);

//读入边数

for (int k = 0; k < arcnum; ++k)

{

...//读入边

arcs[j \* vexnum + i] = arcs[i \* vexnum + j] = MIN(arcs[i \* vexnum + j], len);

//关键点：无向图对于邻接矩阵要对称赋值

}

}

## 3.2 prim算法的实现

### 3.2.1 prim算法的原理和思想：

prim算法利用了MST性质:假设N=(V，{E})是一个连通网，U是顶点集V的一个非空子集。若(u,v)是一条具有最小权值（代价)的边，其中u∈U，v∈V-U，则必存在一棵包含边(u,v)的最小生成树。

MST证明：

可以用反证法证明之。假设网N的任何一棵最小生成树都不包含(u,v)。设T是连通网上的一棵最小生成树，当将边(u,v)加入到T中时，由生成树的定义，T中必存在一条包含(u,z)的回路。另一方面，由于T是生成树，则在T上必存在另一条边(u',v')，其中u'∈U，v'∈V-U，且u和u'之间，和v'之间均有路径相通。删去边(u',v')，便可消除上述回路，同时得到另一棵生成树T’。因为(u,v)的代价不高于(u',v')，则T;的代价亦不高于T，T’是包含(u,v)的一棵最小生成树。由此和假设矛盾。

prim算法的思想如下：

假设N=(V,{E})是连通网,TE是N上最小生成树中边的集合。算法从U={}()，TE=(}开始，重复执行下述操作：在所有u∈U,∈V-U的边(u,v)∈E中找一条代价最小的边(,)并入集合TE,同时并入U,直至U=V为止。此时TE中必有n—1条边,则T-(V,{TE})为N的最小生成树。

### 3.2.2 prim算法流程图

开始

读入起始顶点，将其作为当前连通网U

所有点与起始顶点的距离设为该顶点的lowcost

所有点都插入连通网？

所有未加入连通网的所有顶点中，找到lowcost最小的顶点k

没找到？

输出该图不是连通图

打印其相邻顶点与该顶点，作为最小连通树的一部分

将k点加入连通网，并且根据k所连接的点更新所有顶点的lowcost

结束

y

n

y

n

### 3.2.3 prim 算法代码

void MiniSpanTree\_Prim()

{

char u;

printf("请输入初始顶点：\t");

cin >> u;

//输入初始顶点

int k = LocateVex(u);

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

if (j != k) close[j] = { u,arcs[k \* vexnum + j] };

close[k].lowcost = 0;

for (int i = 1; i < vexnum; ++i)

{

int k = -2, MIN = INFINITY;

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

if (close[j].lowcost && close[j].lowcost < MIN)

k = j, MIN = close[j].lowcost;

//寻找lowcost最小的k

if (k==-1) { printf("图不连通！\n"); break; }

//cout << vexs[k] << endl;

printf("%c--<%d>-->%c\n", close[k].adjvex, close[k].lowcost, vexs[k]);

close[k].lowcost = 0;

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

if (arcs[k \* vexnum + j] < close[j].lowcost)close[j] = { vexs[k],arcs[k \* vexnum + j] };

}

}

# 4 测试

## 4.1 功能测试

### 4.1.1 普通测试1

**测试用例**：

4

a b c d

6

a b 8

b c 7

c d 5

d a 11

a c 18

b d 12

a

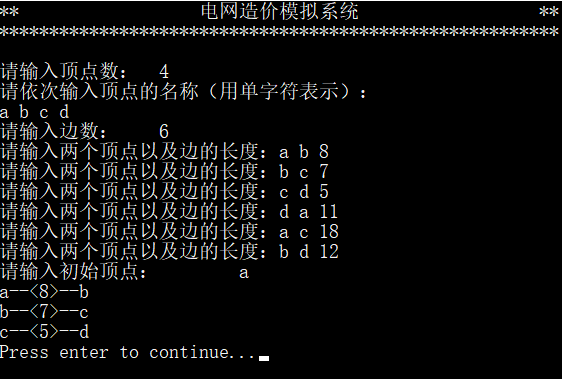
**预期结果**：

a--<8>-->b

b--<7>-->c

c--<5>-->d

**实验结果**



### 4.1.2 普通测试2

**测试用例：**

6

1 2 3 4 5 6

10

1 4 5

4 6 2

6 5 6

5 2 3

2 1 6

1 3 1

2 3 5

5 3 6

3 6 4

4 3 5

1

**预期结果：**

（数据结构课本上的样例）

1--<1>-->3

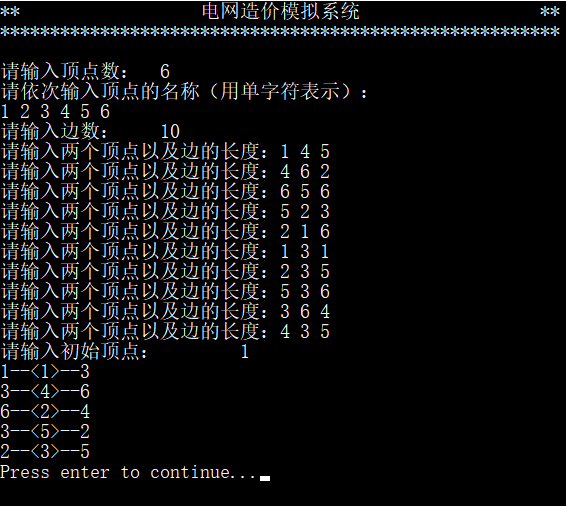
3--<4>-->6

6--<2>-->4

3--<5>-->2

2--<3>-->5

**实验结果：**



### 4.1.3 普通测试3

**测试用例：（包含自环和重复的边）**

5

1 2 3 4 5

18

2 4 276

3 3 435

3 4 608

2 4 860

1 2 318

1 3 547

5 4 419

2 5 98

1 5 460

5 3 399

3 5 240

3 2 733

3 3 903

4 2 909

5 2 206

3 4 810

2 1 115

2 3 419

5

**预期结果：**

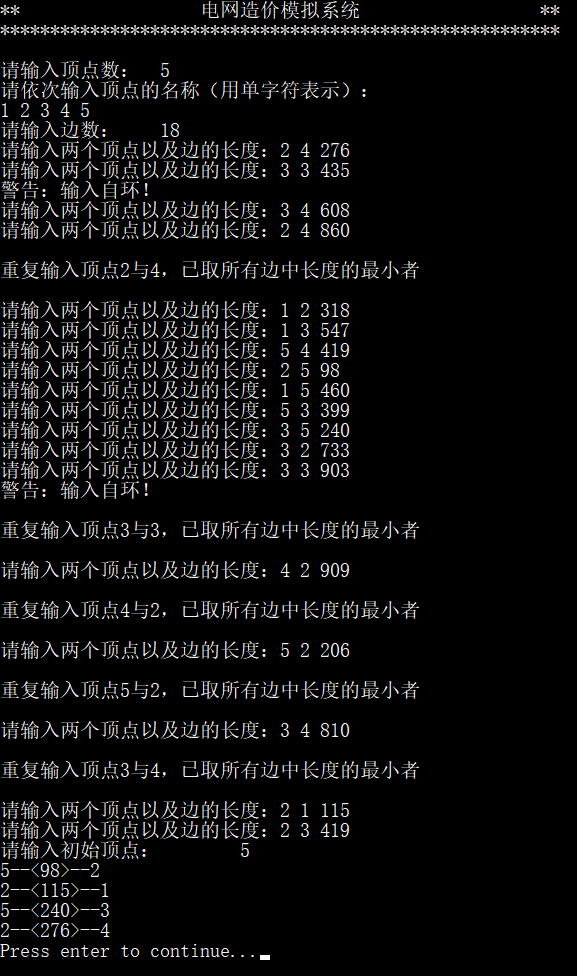
5--<98>--2

2--<115>--1

5--<240>--3

2--<276>--4

**实验结果：**



## 4.2 边界测试

### 4.2.1 平凡图

**测试用例：**

1

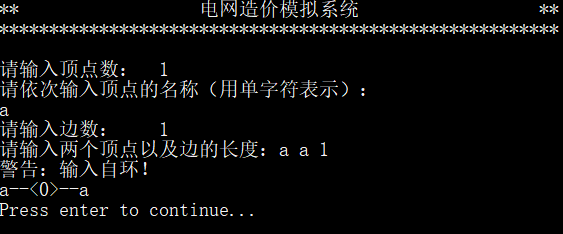
a

1

a a 1

**预期结果：**程序运行正常。

**实验结果：**



### 4.2.2 不连通图

**测试用例：**

4

A B C D

2

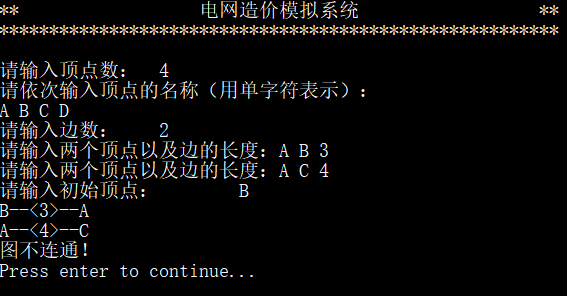
A B 3

A C 4

B

**预期结果：**提示输入为不连通图。

**实验结果：**



## 4.3 出错测试

### 4.3.1输入顶点数/边数错误

**测试用例：**

-1

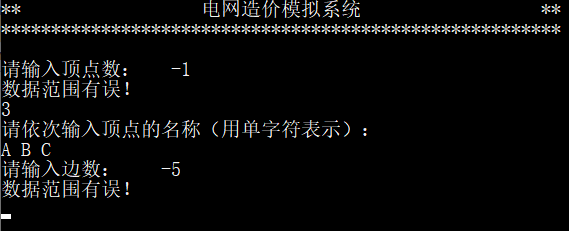
3

A B C

-5

**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：**

****

### 4.3.2 输入非法数据

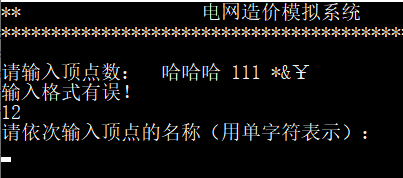
**测试用例：**

哈哈哈 111 \*&￥

12

**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：**

****

### 4.3.3 输入顶点不存在

**测试用例：**

4

A B C D

3

a

A B 12

**预期结果：**程序给出提示信息，指引用户进行正确输入。

**实验结果：**

