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： 李子涵

学 号： 1851892

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](#_Toc10760)

[1.1 背景分析 1](#_Toc19050)

[1.2 功能分析 1](#_Toc25458)

[2 设计 1](#_Toc24382)

[2.1 数据结构设计 1](#_Toc14047)

[2.2 类结构设计 1](#_Toc12258)

[2.3 成员与操作设计 1](#_Toc25176)

[2.4 系统设计 3](#_Toc23273)

[3 实现 3](#_Toc18448)

[3.1 添加结点功能的实现 3](#_Toc10236)

[3.1.1 添加结点功能流程图 3](#_Toc20035)

[3.1.2 添加结点功能核心代码 4](#_Toc5306)

[3.1.3 添加结点功能截屏示例 5](#_Toc13088)

[3.2 添加边功能的实现 6](#_Toc27000)

[3.2.1 添加边功能流程图 6](#_Toc7430)

[3.2.2 添加边功能核心代码 6](#_Toc7437)

[3.2.3 添加边功能截屏示例 7](#_Toc18458)

[3.3 生成Prim最小二叉树功能的实现 8](#_Toc4708)

[3.3.1 生成Prim最小二叉树功能流程图 8](#_Toc11112)

[3.3.2 生成Prim最小二叉树功能核心代码 8](#_Toc21603)

[3.3.3 生成Prim最小二叉树功能截图示例 10](#_Toc10831)

[3.4 输出功能的实现 11](#_Toc28451)

[3.4.1 输出功能流程图 11](#_Toc24904)

[3.4.2 输出功能核心代码 11](#_Toc19367)

[3.4.3 输出功能截屏示例 12](#_Toc532)

[3.6 总体系统的实现 13](#_Toc5731)

[3.6.1 总体系统流程图 13](#_Toc10785)

[3.6.2 总体系统核心代码 13](#_Toc22167)

[3.6.3 总体系统截屏示例 17](#_Toc19732)

[4 测试 18](#_Toc23712)

[4.1 功能测试 18](#_Toc28601)

[4.1.1 建立顶点功能测试 18](#_Toc24592)

[4.1.2 建立边功能测试 18](#_Toc30800)

[4.1.3生成功能测试 19](#_Toc307)

[4.1.4 输出功能测试 20](#_Toc2388)

[4.2 边界测试 21](#_Toc388)

[4.2.1 输入顶点数量过少 21](#_Toc15465)

[4.2.2 输入边数量不足以构成最小生成树 22](#_Toc25035)

[4.3 出错测试 23](#_Toc28197)

[4.3.1 顶点名称错误 23](#_Toc6024)

[4.3.2 操作码错误 23](#_Toc9280)

[4.3.3 插入位置不存在 23](#_Toc18145)

[4.3.4 未成功生成prim最小二叉树就输出 24](#_Toc6738)

# 1 分析

## 1.1 背景分析

城市建设中电网的铺设是很重要的一环。在不同小区之间铺设电网，需要的经济造价不同。如何选择经济成本最低的铺设路线，是很具有现实意义的问题。而该问题可以由特定的算法简单模拟，使用软件系统对该设计进行统一化的处理，既可以节省人力，又可以快速准确地得到最省成本的建设方案。

因此，设计一个电网建设造价模拟系统很有必要。

## 1.2 功能分析

一个简易的电网建设造价模拟系统，首先应该有的功能是录入城市小区信息，即输入各小区名称和相邻之间的电网造价。其次，系统需要运算得出最低成本的建造设计方案并输出。同时，如果输入信息有误，需要给出提示。

综上所述，一电网建设造价模拟系统至少应该具有输入、输出、运算、退出的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求在不同结点之间创建边，最符合的数据结构就是图。同时，由于电网路线有造价要求，并且并不要求方向，因此应该采用带权无向图。

## 2.2 类结构设计

图通常可以用邻接矩阵或邻接表表示。为了节省不必要的储存空间开销，本系统采用邻接表。用图类储存代表小区的结构，小区结点结构存储有小区名和边的头结点。此外，还有用于筛选MST边的辅助优先队列，以及对应辅助结构。

## 2.3 成员与操作设计

**图类（graph）**

private:

vertex \*NodeTable;//head link

int num;//quantity of cities

public:

graph(int);//using the quantity of cities to build the graph

~graph();

int get\_pos(const string);//find the position in the node table

double get\_cost(int&, int&);//get the cost between two nodes

bool insert\_edge(int&, int&, double&);//insert an edge

int get\_first(int&);//get the first neighbor of the node

int get\_next(int&, int&);//get the next neighbor of a neighbor of the node

int get\_num() { return num; }//get the quantity of cities

void output(int n) { cout << NodeTable[n].name; }//print the name of specified node

void build();//build the edges

**优先队列类（pile）**

private:

ednode\* front;//the front pointer,doesn't include data

ednode\* rear;//points to the last node

public:

pile();//constructor

~pile();//destructor

bool insert(ednode\*);//insert a new node in an increasing order

bool pop(ednode\*&);//if empty,return 0,else remove the smallest node

bool is\_empty() { return front == rear; }//to judge whether the pile is empty

void print(graph &G);//print all the node with the node names in graph G

void clear();

struct edge

{

int dest=-1;//another node

double cost = Max;//cost of the edge

edge\* link=NULL;//next edge

};

struct vertex

{

string name="null";//name of the city

edge\* adj=NULL;//head link of edges

};

struct ednode

{

int head=-1;//inside the tree

int tail=-1;//outside the tree

ednode\* link=NULL;

double cost=Max;//cost of the edge

};

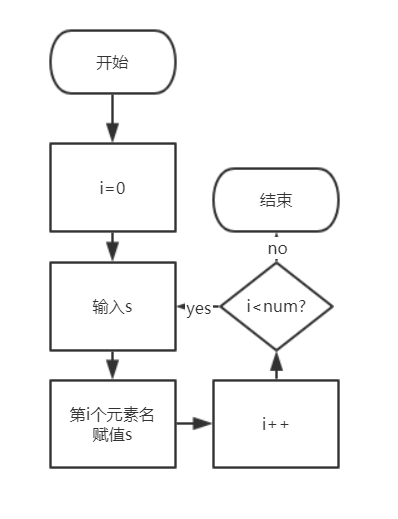
## 2.4 系统设计

首先使用start函数实现屏幕初始化，然后依次录入小区数量、结点名称、边的权值，最后用Prim算法生成最小二叉树并输出。

# 3 实现

## 3.1 添加结点功能的实现

### 3.1.1 添加结点功能流程图



### 3.1.2 添加结点功能核心代码

string s;

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

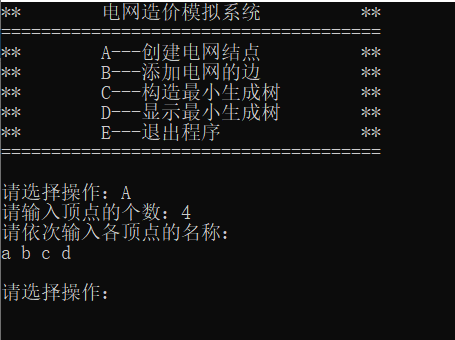
{

cin >> s;

NodeTable[i].name = s;

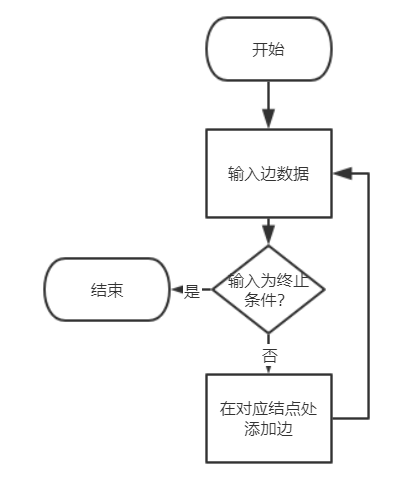
}

### 3.1.3 添加结点功能截屏示例



## 3.2 添加边功能的实现

### 3.2.1 添加边功能流程图



### 3.2.2 添加边功能核心代码

if (u == v || u >= num || v >= num)

return 0;

edge\* p = new edge{ v,c,NodeTable[u].adj };

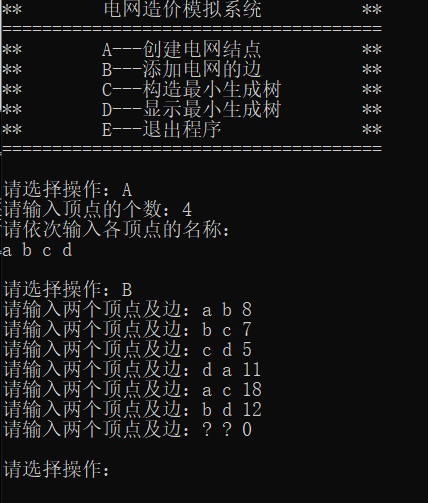
NodeTable[u].adj = p;

p = new edge{ u,c,NodeTable[v].adj };

NodeTable[v].adj = p;

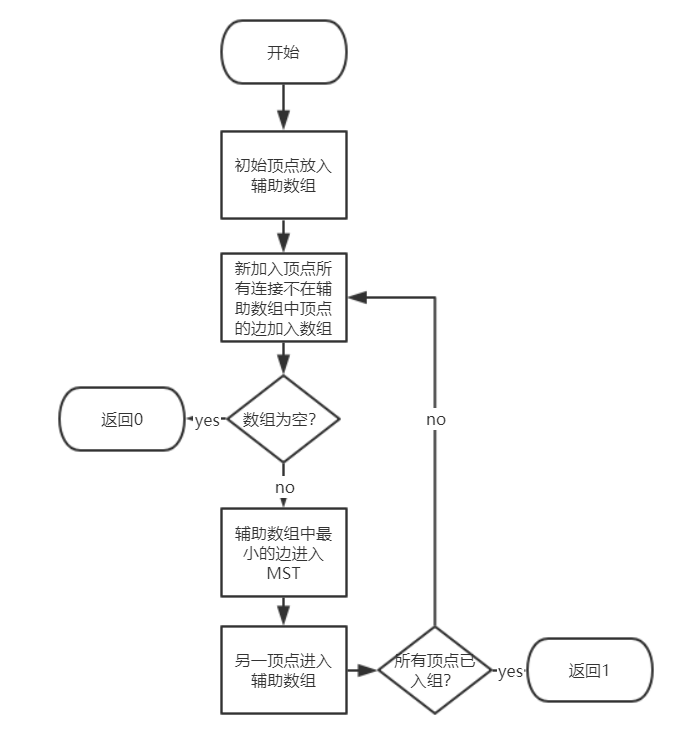
return 1;

### 3.2.3 添加边功能截屏示例



## 3.3 生成Prim最小二叉树功能的实现

### 3.3.1 生成Prim最小二叉树功能流程图



### 3.3.2 生成Prim最小二叉树功能核心代码

int n = G.get\_num();//quantity of nodes

pile H;//Auxiliary list containing alternative edges

bool \*Vmst = new bool[n];//auxiliary array,to mark whether a node is in the MST

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

Vmst[i] = 0;

Vmst[u0] = 1;//the starting node is inside the tree

int u=u0,count = 1;//u is the newly inserted node to the MST,count marks the quantity of nodes already in the MST

do

{

int v = G.get\_first(u);

while (v != -1)//put each edge between the unmarked nodes and the current node into H

{

if (Vmst[v] == 0)

{

ednode\* ed = new ednode;

ed->head = u;

ed->tail = v;

ed->cost = G.get\_cost(u, v);

H.insert(ed);

}

v = G.get\_next(u, v);

}

if (H.is\_empty() && count < n)

return 0;

while (!H.is\_empty() && count < n)

{

ednode\* ed=NULL;

H.pop(ed);

if (Vmst[ed->tail] == 0)

{

MST.insert(ed);

Vmst[ed->tail]=1;

u = ed->tail;

count++;

break;

}

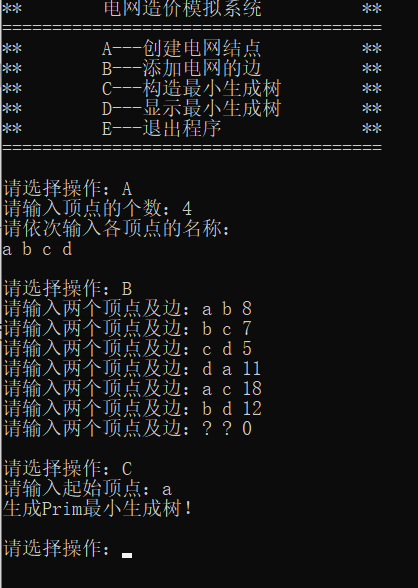
}

}

while (count < n);

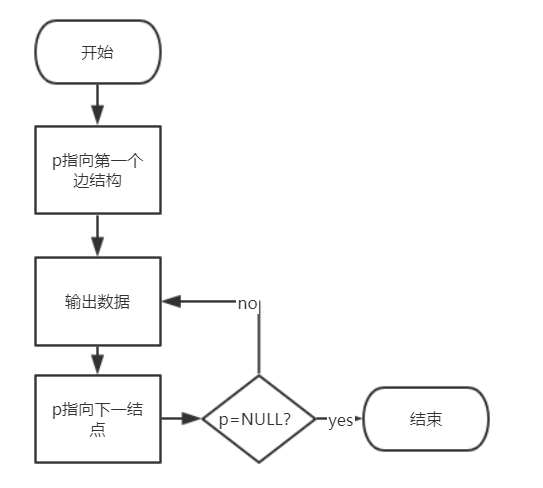
、return 1;

### 3.3.3 生成Prim最小二叉树功能截图示例



## 3.4 输出功能的实现

### 3.4.1 输出功能流程图



### 3.4.2 输出功能核心代码

void pile::print(graph &G)

{

ednode\* p=front->link;

while (p != NULL)

{

G.output(p->head);

cout << "-" << '(' << p->cost << ')' << "->";

G.output(p->tail);

cout << '\t';

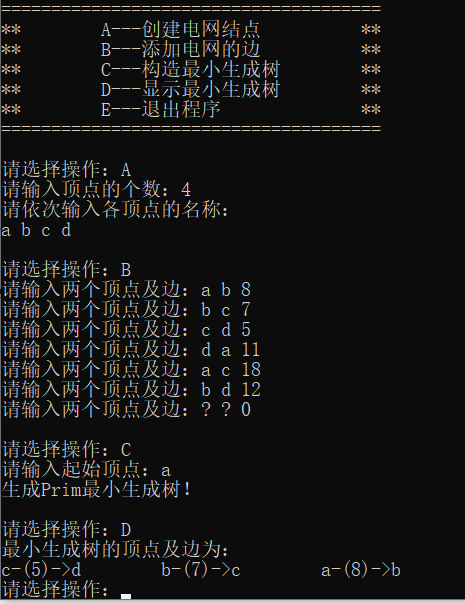
p = p->link;

}

}

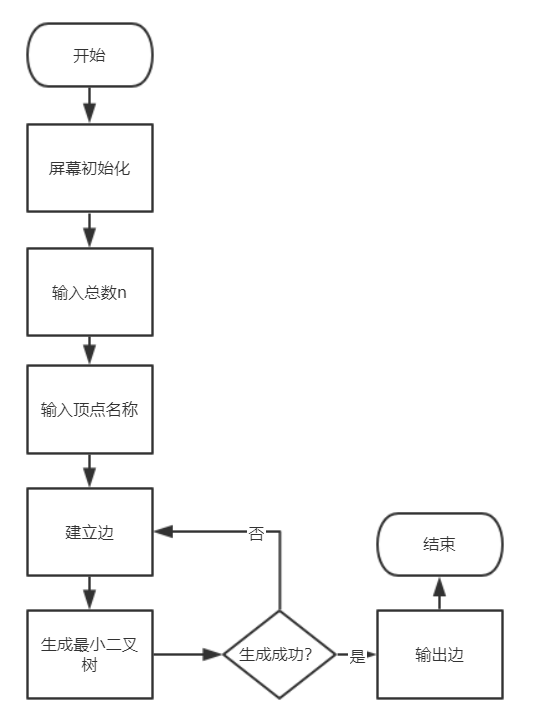
void output(int n) { cout << NodeTable[n].name; }

### 3.4.3 输出功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

start();

char op;

cout << "请选择操作：";

cin >> op;

while (op != 'A')

{

cout << "请先创建结点：";

cin >> op;

}

int n;

cout << "请输入顶点的个数：";

cin >> n;

while (n <= 1)

{

cout << "顶点数量至少为2！请重新输入：" << endl;

cin >> n;

}

graph G(n);

cout << "请依次输入各顶点的名称："<<endl;

G.build();

pile H,MST;

bool flag=0;//build MST

cout << endl<<"请选择操作：";

while (cin >> op, op != 'E')

{

string ustart;

int u0;

switch (op)

{

case'A':

cout << "顶点建立已完成！" << endl;

break;

case'B':

while (1)

{

cout << "请输入两个顶点及边：";

string su, sv;

cin >> su >> sv;

double co;

cin >> co;

if (co == 0)

break;

int u = G.get\_pos(su), v = G.get\_pos(sv);

if (u == -1 || v == -1)

{

cout << "顶点输入错误！" << endl;

continue;

}

G.insert\_edge(u, v, co);

}

break;

case'C':

cout << "请输入起始顶点：";

cin >> ustart;

u0=G.get\_pos(ustart);

if (u0 == -1)

{

cout << "顶点不存在！";

break;

}

if (Prim(G, u0, MST))

{

cout << "生成Prim最小生成树！" << endl;

flag = 1;

}

else

{

cout << "边数量不足以生成Prim最小生成树,请添加边！" << endl;

MST.clear();

}

break;

case'D':

if (!flag)

{

cout << "请先生成Prim最小生成树！" << endl;

break;

}

cout << "最小生成树的顶点及边为：" << endl;

MST.print(G);

break;

default:

cout << "错误的操作！" << endl;

break;

}

cout <<endl<< "请选择操作：";

}

return 0;

### 3.6.3 总体系统截屏示例

# 

# 4 测试

## 4.1 功能测试

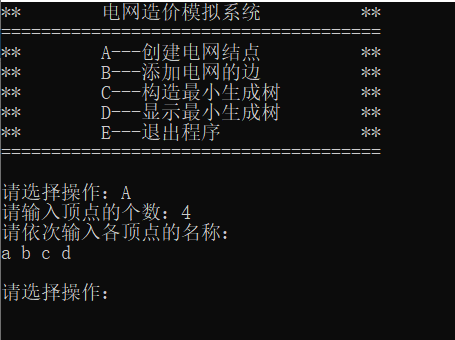
### 4.1.1 建立顶点功能测试

**测试用例**：4 a b c d

**预期结果**：

a b c d

**实验结果**



### 4.1.2 建立边功能测试

**测试用例：**

**a b 8**

**b c 7**

**c d 5**

**d a 11**

**a c 18**

**b d 12**

**? ? 0**

**预期结果：**

**a b 8**

**b c 7**

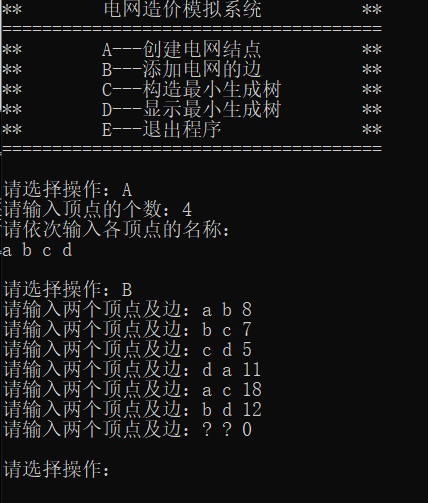
**c d 5**

**d a 11**

**a c 18**

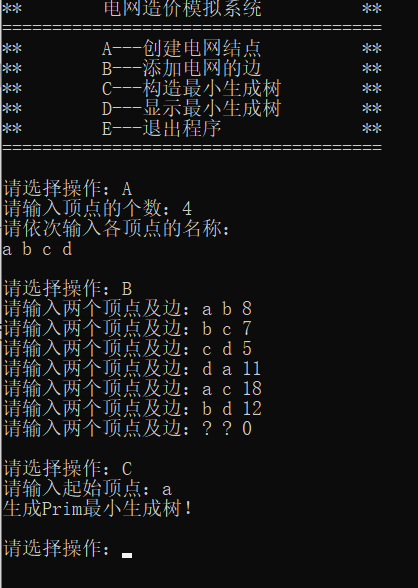
**b d 12**

**实验结果：**



### 4.1.3生成功能测试

**实验结果：**

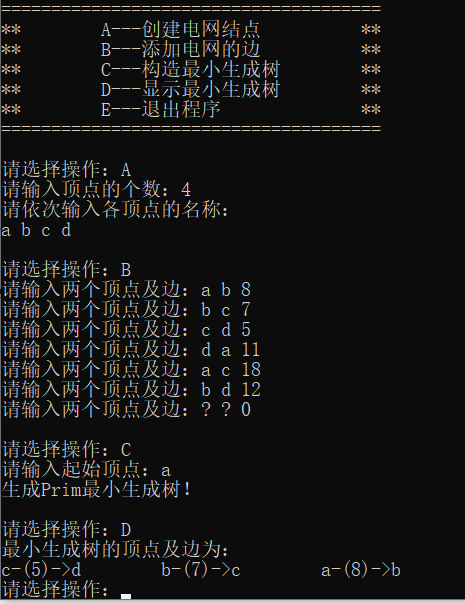


### 4.1.4 输出功能测试

**预期结果：**

c-(5)->d b-(7)->c a-(8)->b

**实验结果：**



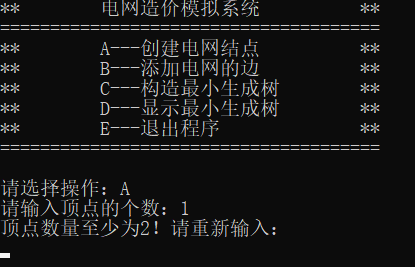
## 4.2 边界测试

### 4.2.1 输入顶点数量过少

**测试用例：**输入1

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**



### 4.2.2 输入边数量不足以构成最小生成树

**测试用例：**

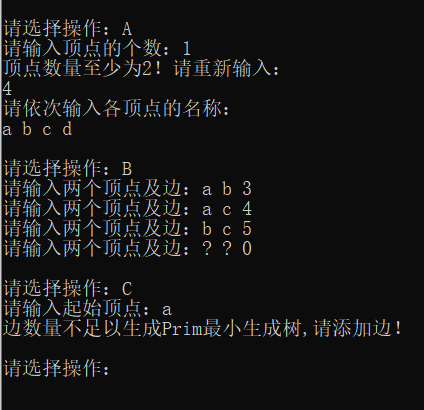
**a b 3**

**a c 4**

**b c 5**

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**



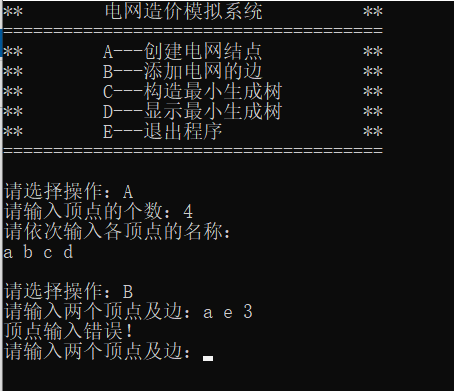
## 4.3 出错测试

### 4.3.1 顶点名称错误

**测试用例：**输入不存在的顶点

**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 操作码错误

**测试用例：**输入操作码错误

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

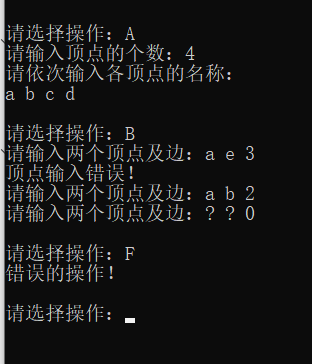
**实验结果：**

### 4.3.3 插入位置不存在

**测试用例：**链表里有两条记录，向链表的第四个位置插入结点

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**



### 4.3.4 未成功生成prim最小二叉树就输出

**测试用例：输出**

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

