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： 赵子昱

学 号： 1951459

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](#_Toc60096115)

[1.1 项目简介 1](#_Toc60096116)

[1.2 功能分析 1](#_Toc60096117)

[2 设计 2](#_Toc60096118)

[2.1 数据结构设计 2](#_Toc60096119)

[2.2 类结构设计 2](#_Toc60096120)

[2.3 成员与操作设计 2](#_Toc60096121)

[2.4 系统设计 3](#_Toc60096122)

[3 实现 4](#_Toc60096123)

[3.1 创建电网顶点功能**BuildNodes()**的实现 4](#_Toc60096124)

[3.1.1 创建电网顶点功能代码 4](#_Toc60096125)

[3.1.2 创建电网顶点功能截屏示例 5](#_Toc60096126)

[3.2 建立电网的边功能**AddEdge ()**的实现 5](#_Toc60096127)

[3.2.1 建立电网的边功能代码 5](#_Toc60096128)

[3.2.2 建立电网的边功能截屏示例 6](#_Toc60096129)

[3.3 构造最小生成树功能**BuildTree()**的实现 6](#_Toc60096130)

[3.3.1 构造最小生成树功能代码 6](#_Toc60096131)

[3.3.2 构造最小生成树功能截图示例 8](#_Toc60096132)

[3.4 显示最小生成树功能**PrintMenu()**的实现 8](#_Toc60096133)

[3.4.1 显示最小生成树功能代码 8](#_Toc60096134)

[3.4.2 显示最小生成树功能截屏示例 9](#_Toc60096135)

[3.5 主函数的实现 9](#_Toc60096136)

[3.5.1 主函数代码 9](#_Toc60096137)

[3.5.2 主函数截屏示例 11](#_Toc60096138)

[4 测试 12](#_Toc60096139)

[4.1 功能测试 12](#_Toc60096140)

[4.1.1 测试1 12](#_Toc60096141)

[4.1.2 测试2 14](#_Toc60096142)

[4.2 边界测试 16](#_Toc60096143)

[4.2.1 图中只有两个顶点一条边 16](#_Toc60096144)

[4.3 出错测试 18](#_Toc60096145)

[4.3.1 创建边先于创建顶点 18](#_Toc60096146)

[4.3.2 构造最小生成树先于创建顶点或边 18](#_Toc60096147)

[4.3.3 显示最小生成树先于创建生成树 19](#_Toc60096148)

# 1 分析

## 1.1 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。

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。

## 1.2 功能分析

首先,每个小区当作一个顶点,那么就有n个顶点。两个小区之间设置一条电网线路相互联通且有相应的经济代价，可抽象为一个有权无向图模型。

所以问题就成了求一个加权无向图的最小生成树。所以首先需要用户输入小区即顶点数，并且给每个小区进行命名以方便查看。然后用户将添加若干个电网线路以及他们之间的花费。而程序就负责找出这个加权无向图的最小生成树，并且显示出来。

# 2 设计

## 2.1 数据结构设计

项目中采用Kruskal算法，用最小堆和并查集进行辅助操作，定义了**Edge**类用来储存图中的边，电网类（**ElectricityNet**）用来封装电网的定点、边及各种操作。

## 2.2 类结构设计

并查集UFSet与最小堆MinHeap的实现此处不再赘述。Edge类为电网的边，储存了边的起点、终点与权值，并重载了>、<、= 。**ElectricityNet**类为电网，储存定点的名称、顶点数量、顶点的并查集、存储边的最小堆、构成生成树的边。

## 2.3 成员与操作设计

**Edge类**

1. **class** Edge {
2. **friend** **class** ElectricityNet;
3. **public**:
4. Edge() {};
5. Edge(**int** start, **int** end, **float** cost) :Start(start), End(end), Cost(cost) {};
6. **void** Set(**int** start, **int** end, **float** cost) { Start = start, End = end, Cost = cost; }
7. **bool** operator<(**const** Edge &right) { **return** Cost < right.Cost; }
8. **bool** operator>(**const** Edge &right) { **return** Cost > right.Cost; }
9. **bool** operator==(**const** Edge &right) { **return** Cost > right.Cost&&Start == right.Cost&&End == right.End; }
10. **private**:
11. **float** Cost;
12. //权值
14. **int** Start, End;
15. //起点、终点（顶点下标）
16. };

**ElectricityNet类**

1. **class** ElectricityNet {
2. **public**:
3. ElectricityNet() :Node(NULL), NeedEdges(NULL){}
4. ~ElectricityNet() { **delete**[]Node, **delete**[]NeedEdges; }
6. **void** BuildNodes();
7. //创建电网顶点
9. **void** AddEdge();
10. //添加电网的边
12. **void** BuildTree();
13. //构造最小生成树
15. **void** PrintTree();
16. //显示最小生成树
18. **void** PrintMenu();
19. //输出可选操作列表
21. **private**:
22. string \*Node;
23. //节点名称
25. MinHeap<Edge> UnUsedEdges;
26. //储存未构建生成树的边
28. UFSets NodeSet;
29. //顶点并查集
31. Edge \*NeedEdges;
32. //生成树中的边
34. **int** NodeNum;
35. //顶点数
37. };

## 2.4 系统设计

系统首先创建一个ElectricityNet对象net，然后调用PrintMenu()函数输出菜单，让用户输入操作码，然后应用switch结构根据操作码调用相应的函数实现不同功能。

# 3 实现

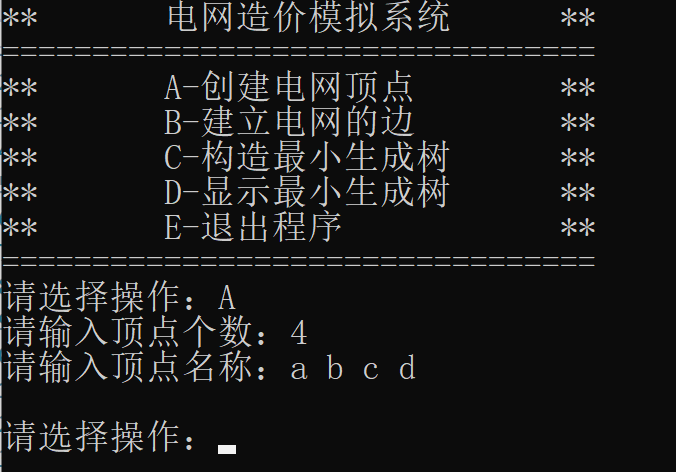
## 3.1 创建电网顶点功能**BuildNodes()**的实现

首先让用户输入顶点个数，如果个数不合理（小于2）会给出提示让用户重新输入，直到输入合法。让用户输入各顶点名称，实现对各顶点名称的初始化。并根据顶点个数初始化顶点并查集、保存边的最小堆与最小生成树所需的边链表。

### 3.1.1 创建电网顶点功能代码

1. **void** ElectricityNet::BuildNodes() {
2. cout << "请输入顶点个数：";
3. cin >> NodeNum;
4. **while** (NodeNum < 2) {
5. cout << "顶点个数至少为2！" << endl;
6. cin >> NodeNum;
7. }
8. NodeSet.SetSize(NodeNum);           //初始化顶点并查集
9. UnUsedEdges.SetSize(NodeNum\*(NodeNum - 1) / 2);//初始化边的最小堆
10. **if** (Node != NULL) { **delete** Node; }
11. cout << "请输入顶点名称：";
12. Node = **new** string[NodeNum];         //初始化顶点名称数组
13. NeedEdges = **new** Edge[NodeNum - 1];  //初始化组成最小生成树的边数组
14. **for** (**int** i = 0; i < NodeNum; i++) { cin >> Node[i]; }
15. }

### 3.1.2 创建电网顶点功能截屏示例



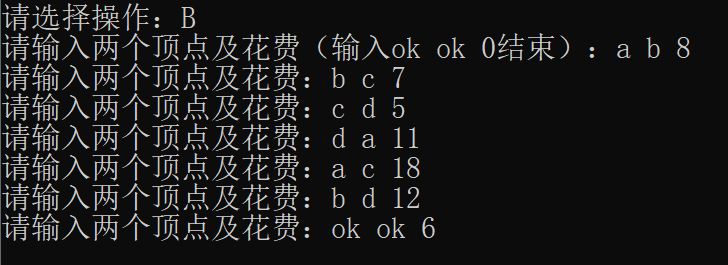
## 3.2 建立电网的边功能**AddEdge ()**的实现

首先检查顶点名称链表是否为空，若为空说明还没有创建顶点，给出错误提示。否则让用户依次输入每条边的两个顶点及花费。

### 3.2.1 建立电网的边功能代码

1. **void** ElectricityNet::AddEdge() {
2. **if** (!Node) {
3. cout << "请先创建电网节点！" << endl;
4. **return**;
5. }
6. string start, end;
7. **int** startNum = 0, endNum = 0;
8. **float** cost;
9. Edge e;
10. cout << "请输入两个顶点及花费（输入ok ok 0结束）：";
11. cin >> start >> end >> cost;
12. **while** (start != "ok") {
13. **while** (Node[startNum] != start) { startNum++; }
14. **while** (Node[endNum] != end) { endNum++; }
15. e.Set(startNum, endNum, cost);
16. UnUsedEdges.Insert(e);
17. startNum = endNum = 0;
18. cout << "请输入两个顶点及花费：";
19. cin >> start >> end >> cost;
20. }
21. }

### 3.2.2 建立电网的边功能截屏示例



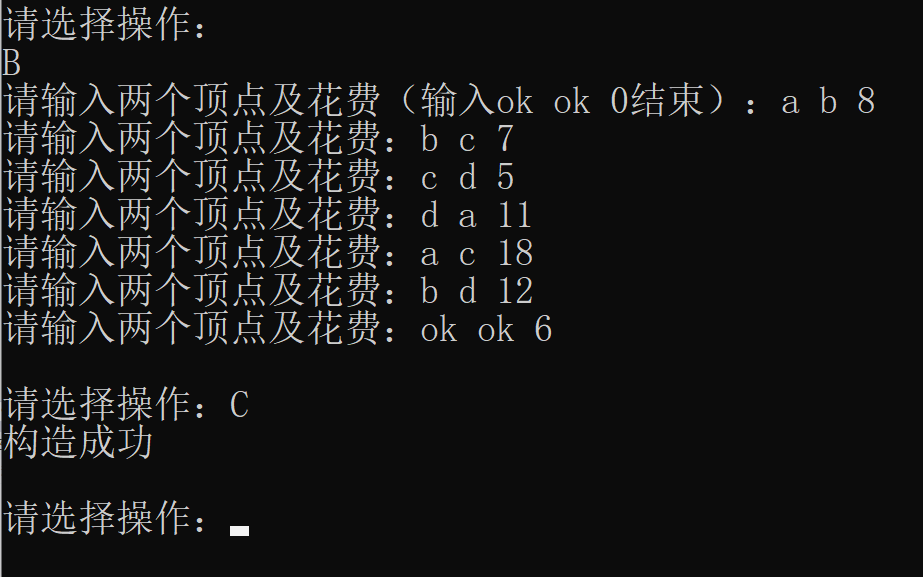
## 3.3 构造最小生成树功能**BuildTree()**的实现

首先检测边的最小堆是否为空，若为空说明还没有建立边，这是给出错误提示。否则根据Kruskal算法，每次最小堆中弹出一条边，若这条边的两个顶点不在同一个类中则合并他们，直到最小堆为空。最后还要检查是否已经组成连通图，若没有组成说明边不合理，给出错误提示让用户重新输入边，并重新初始化各辅助存储变量。

### 3.3.1 构造最小生成树功能代码

1. **void** ElectricityNet::BuildTree() {
2. **if** (UnUsedEdges.IsEmpty()) {
3. cout << "请先建立电网的边！" << endl;
4. **return**;
5. }
6. Edge e;
7. **int** ParStart, ParEnd;
8. **int** currentEdges = 0;
9. **while** (!UnUsedEdges.IsEmpty()) {
10. e = UnUsedEdges.Pop();
11. ParStart = NodeSet.Find(e.Start), ParEnd = NodeSet.Find(e.End);
12. **if** (ParEnd != ParStart) {
13. //如果两个顶点不在同一类，则选中这条边，并合并顶点所在的类
14. NeedEdges[currentEdges] = e;
15. NodeSet.Union(ParEnd, ParStart);
16. currentEdges++;
17. }
18. }
19. **if** (!NodeSet.IsConnected()) {
20. //如果没有连通，给出错误提示，并重新初始化各辅助存储变量
21. cout << "电网未连通，请重新建立电网的边！" << endl;
22. NodeSet.SetSize(NodeNum);
23. UnUsedEdges.SetSize(NodeNum\*(NodeNum - 1) / 2);
24. **delete**[]NeedEdges;
25. NeedEdges = **new** Edge[NodeNum - 1];
26. **return**;
27. }
28. **else** {
29. cout << "构造成功" << endl;
30. }
31. }

### 3.3.2 构造最小生成树功能截图示例

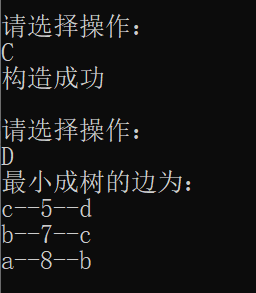


## 3.4 显示最小生成树功能**PrintMenu()**的实现

### 3.4.1 显示最小生成树功能代码

1. **void** ElectricityNet::PrintTree() {
2. **if** (!NeedEdges) {
3. cout << "请先构建最小生成树！" << endl;
4. **return**;
5. }
6. **int** tmp=NodeNum-1;
7. cout << "最小成树的边为：" << endl;
8. **for** (**int** i = 0; i < tmp; i++) {
9. cout << Node[NeedEdges[i].Start] << "--" << NeedEdges[i].Cost << "--" << Node[NeedEdges[i].End] << ' ' << endl;
10. }
11. }

### 3.4.2 显示最小生成树功能截屏示例



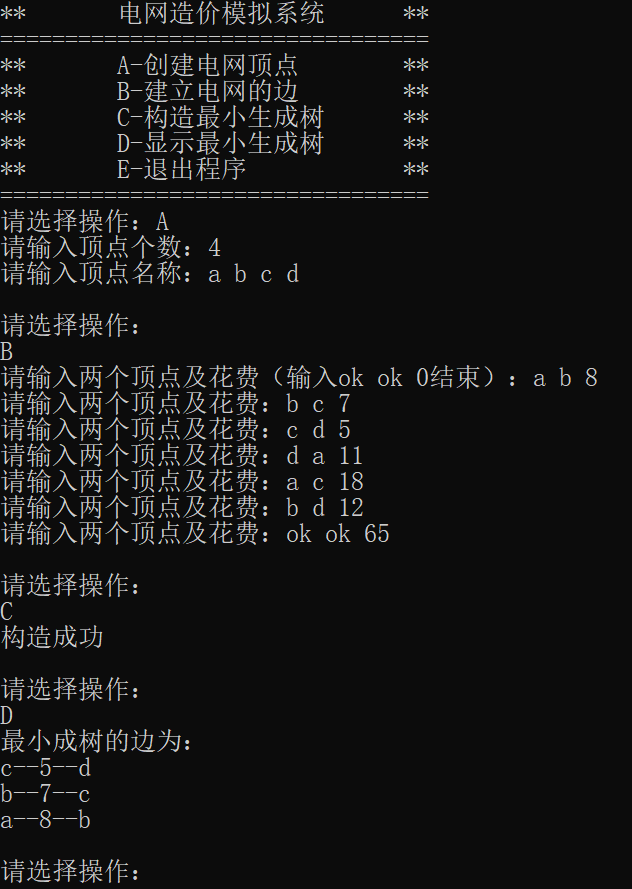
## 3.5 主函数的实现

系统首先创建一个ElectricityNet对象net，然后调用PrintMenu()函数输出菜单，让用户输入操作码，然后应用switch结构根据操作码调用相应的函数实现不同功能。

### 3.5.1 主函数代码

1. **int** main() {
2. ElectricityNet net;
3. net.PrintMenu();
4. **char** operation='a';
5. **while** (operation != 'E') {
6. cout << "请选择操作：";
7. cin >> operation;
8. **switch** (operation) {
9. **case** 'A':
10. net.BuildNodes();
11. **break**;
12. **case** 'B':
13. net.AddEdge();
14. **break**;
15. **case** 'C':
16. net.BuildTree();
17. **break**;
18. **case** 'D':
19. net.PrintTree();
20. **break**;
21. **default**:
22. operation = 'E';
23. **break**;
24. }
25. cout << endl;
26. }
27. system("pause");
28. **return** 0;
29. }

### 3.5.2 主函数截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 测试1

**测试用例**：

A

4

a b c d

B

a b 8

b c 7

c d 5

d a 11

a c 18

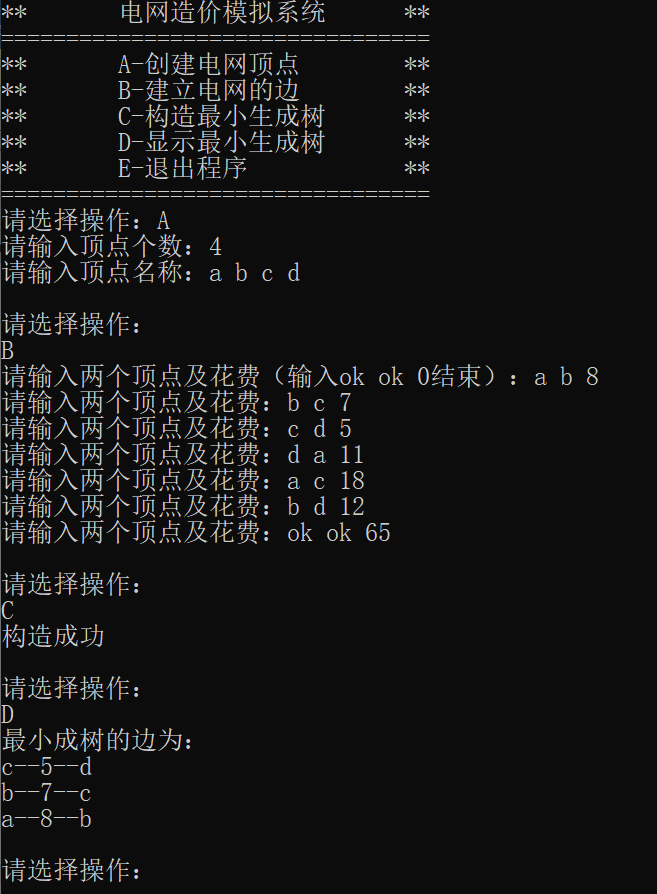
b d 12

ok ok 65

C

D

**实验结果**

****

### 4.1.2 测试2

**测试用例：**

A

8

0 1 2 3 4 5 6 7

B

4 5 0.35

4 7 0.37

5 7 0.28

0 7 0.16

1 5 0.32

0 4 0.38

2 3 0.17

1 7 0.19

0 2 0.26

1 2 0.36

1 3 0.29

2 7 0.34

6 2 0.40

3 6 0.52

6 0 0.58

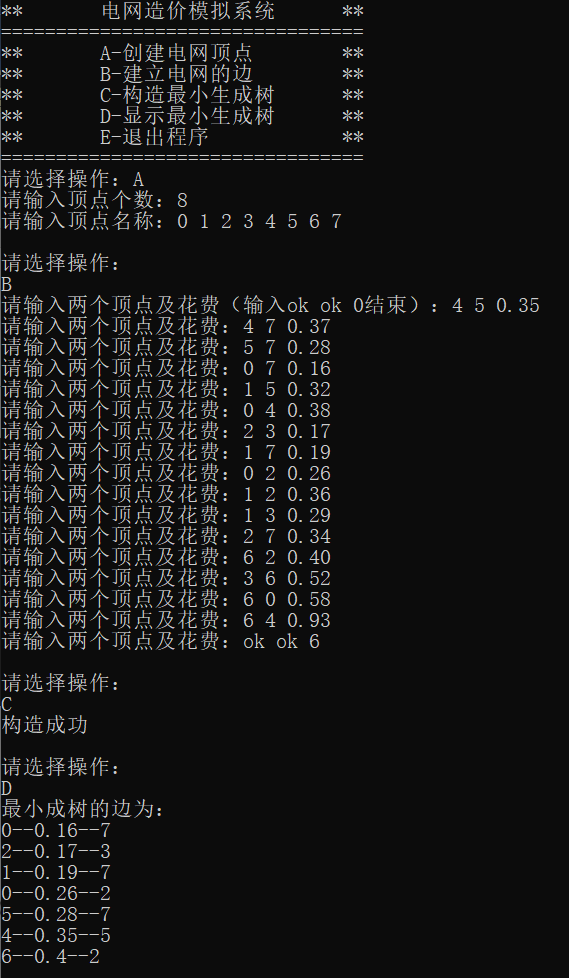
6 4 0.93

ok ok 6

C

D

**实验结果：**



## 4.2 边界测试

### 4.2.1 图中只有两个顶点一条边

**测试用例**：

A

2

a b

B

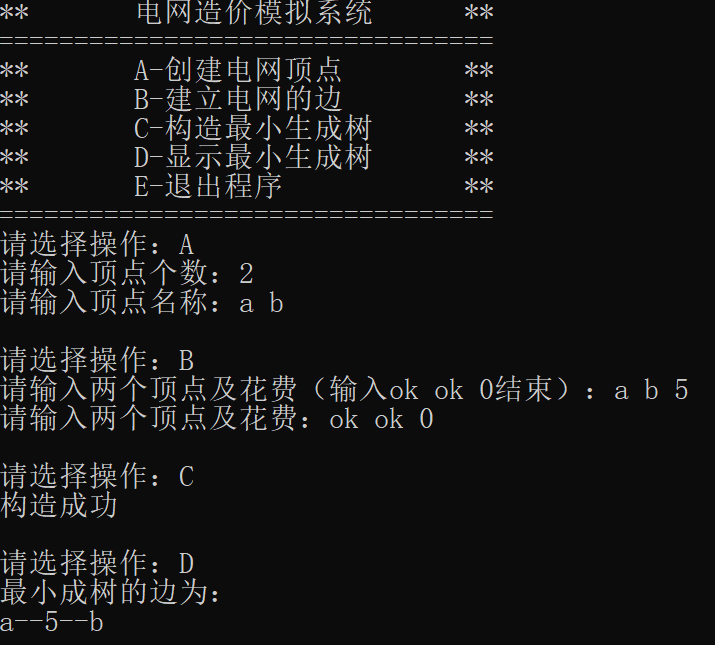
a b 5

ok ok 0

C

D

**实验结果**

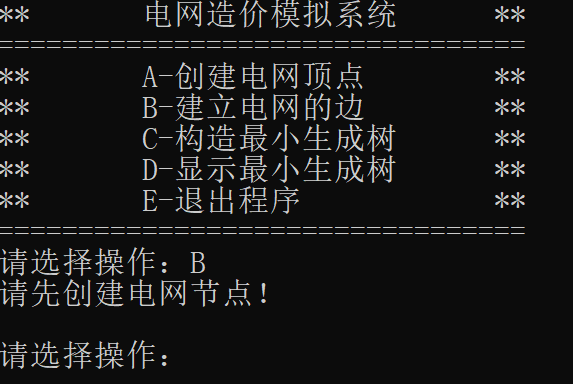


## 4.3 出错测试

### 4.3.1 创建边先于创建顶点

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

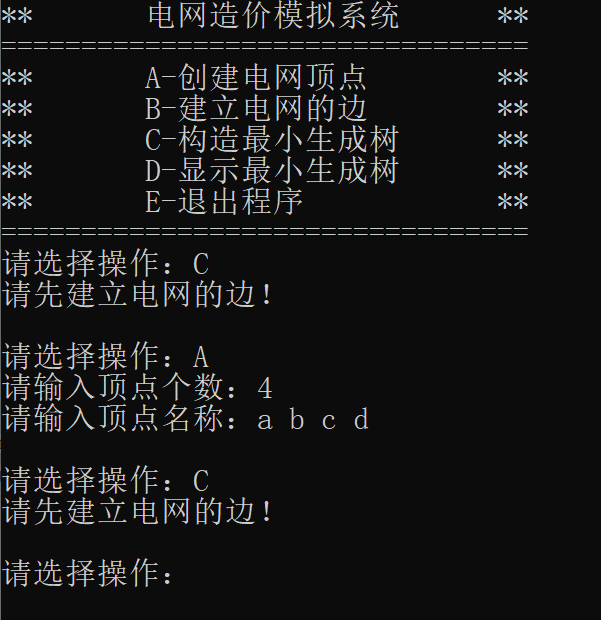
**实验结果：**

****

### 4.3.2 构造最小生成树先于创建顶点或边

**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 显示最小生成树先于创建生成树

**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：**

