项目8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： 谭欢秘

学 号： 1954185

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](#_Toc60248564)

[1.1 背景分析 1](#_Toc60248565)

[1.2 功能分析 1](#_Toc60248566)

[2 设计 1](#_Toc60248567)

[2.1 数据结构设计 1](#_Toc60248568)

[2.2 类结构设计 1](#_Toc60248569)

[2.3 成员与操作设计 2](#_Toc60248570)

[3.1.1 hmfNode类成员： 2](#_Toc60248571)

[3.1.2 HuffmanTree类成员： 2](#_Toc60248572)

[2.4 主函数设计 3](#_Toc60248573)

[3 实现 3](#_Toc60248574)

[3.1 前序遍历树功能的实现 3](#_Toc60248575)

[3.1.1 前序遍历树功能流程图 3](#_Toc60248576)

[3.1.2 前序遍历树功能核心代码 4](#_Toc60248577)

[3.1.3 preOrder()函数实现时需要注意的细节 4](#_Toc60248578)

[3.2 对还未加入树中节点权值排序功能的实现 4](#_Toc60248579)

[3.2.1 权值排序功能核心代码 4](#_Toc60248580)

[3.2.2 sort(int len)函数实现时需要注意的细节 5](#_Toc60248581)

[3.3 构造霍夫曼树功能的实现 5](#_Toc60248582)

[3.3.1 构造霍夫曼树功能流程图 5](#_Toc60248583)

[3.3.2 构造霍夫曼树功能核心代码 6](#_Toc60248584)

[3.4初始化功能的实现 6](#_Toc60248585)

[3.4.1 初始化功能流程图 6](#_Toc60248586)

[3.4.2 初始化功能核心代码 7](#_Toc60248587)

[3.4.3 initialize()函数实现时需要注意的细节 7](#_Toc60248588)

[3.5 主函数的实现 7](#_Toc60248589)

[3.5.1 主函数代码 7](#_Toc60248590)

[3.5.2 总体系统截屏示例 8](#_Toc60248591)

[4 测试 8](#_Toc60248592)

[4.1 功能测试 8](#_Toc60248593)

[4.1.1 初始化功能测试 8](#_Toc60248594)

[4.1.2森林排序、子树合并功能测试 8](#_Toc60248595)

[4.1.3 生成霍夫曼树功能测试 9](#_Toc60248596)

[4.2 边界测试 10](#_Toc60248597)

[4.2.1 输入木头总块数不合法 10](#_Toc60248598)

[4.2.2输入木头长度不合法 11](#_Toc60248599)

[5 总结 12](#_Toc60248600)

[5.1 遇到的错误 12](#_Toc60248601)

[5.1.1 xx函数中 12](#_Toc60248602)

[5.2 改进与优化 12](#_Toc60248603)

[5.3 项目心得 12](#_Toc60248604)

# 1 分析

## 背景分析

一个城市里有n个小区，要实现n个小区之间的电网都能够相互接通，可以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，但要求选择其中的n-1条连通整个城市且使总的耗费最少。这样一个问题其实就是求解最小生成树的问题。一个连通图G的某一无环连通子图T若可以覆盖G中所有的顶点，则称其为G的一棵生成树(spanning tree)，生成树既是“禁止环路”前提下的极大子图，也是“保持连通” 前提下的最小子图。最小生成树则是对带权网络而言的。若图G为一带权网络，则每一棵生成树的成本(cost)即为其所包含的每条边权重的总和。在G 的所有生成树中，成本最低的就是最小生成树(minimum spanning tree, MST)。本题搭建小区间电网、搭建电网有不同开销的背景就可以抽象为一个带权网络，寻找最优惠的最少搭建线路的方案就是寻找这个网络的最小生成树。

## 1.2 功能分析

城市里各个小区可看作数据结构 图中的顶点，所以要有能初始化顶点、存储各个小区信息的功能。此外，图中还有重要的数据成员——边，所以本系统还需要能接受小区间电网线路的输入并存储信息。完成图的初始化之后，就是用Kruskal或者Prim等算法找到网络中的最小生成树。

综上所述，本系统至少应有初始化输入顶点、边的功能，以及寻找最小生成树的功能。本系统选择了用Prim算法生成最小生成树。

# 2 设计

## 2.1 数据结构设计

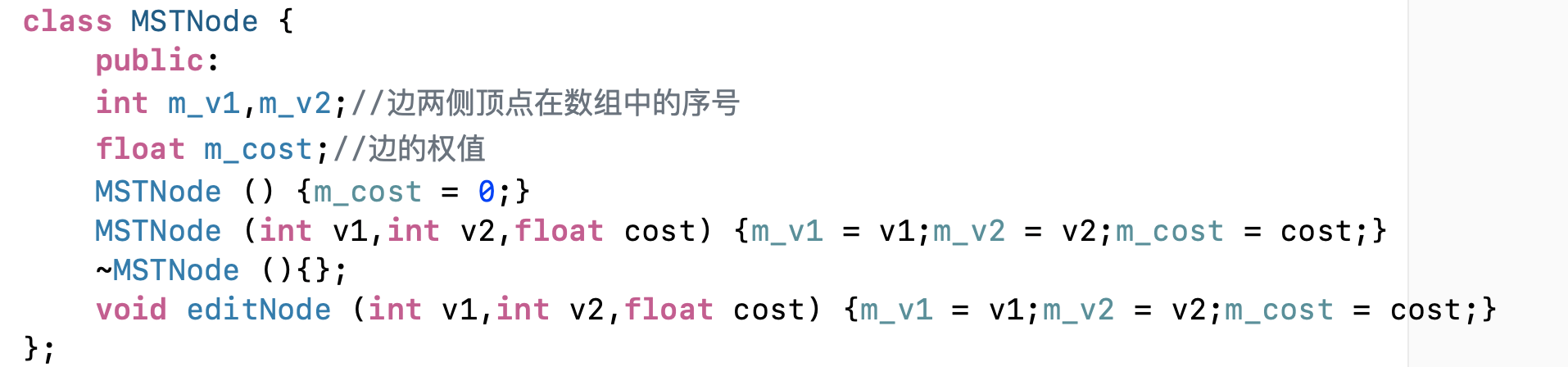
如上功能分析所述，小区间电网通路的关系是多对多的，所以应采用图的数据结构，且两个小区间若有电网线路就是连通的，所以应采用无向图。在存储方式上，考虑到不涉及顶点的增添、删除，所以选择了用邻接矩阵的存储方式来保存无向图的信息。

## 2.2 类结构设计

为了实现在带权无向图中寻找最小生成树，我主要用了一个Graph类。考虑到图是一个常用的数据结构，所以采用了模版类，以便在以后图存储的内容发生变化时还能继续沿用。我还在Graph类的私有成员里嵌套定义了一个MSTNode类作为图的节点类。此外在输入边的信息过程中，还需要辅助的队列结构，所以本程序还包括了之前写的Queue模版类头文件。

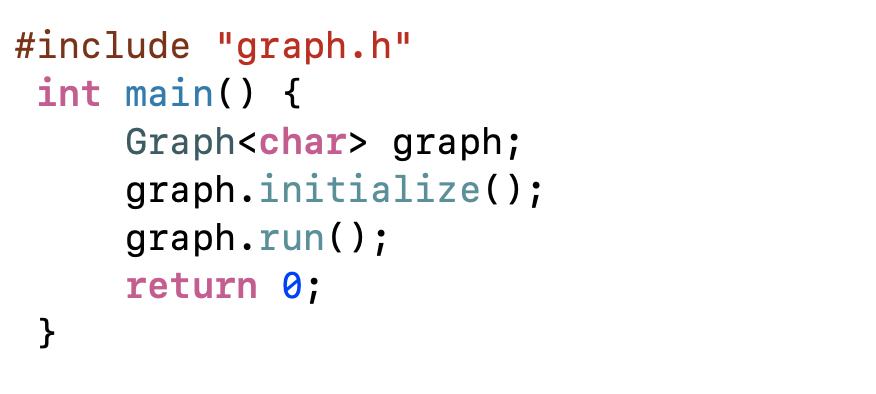
## 2.3 成员与操作设计

### 3.1.1 MSTNode类成员：

因为MSTNode类已经是Graph类的私有成员了，所以将MSTNode类的所有成员都声明为public的，也不会破坏其封装性。

### 3.1.2 Graph类成员：

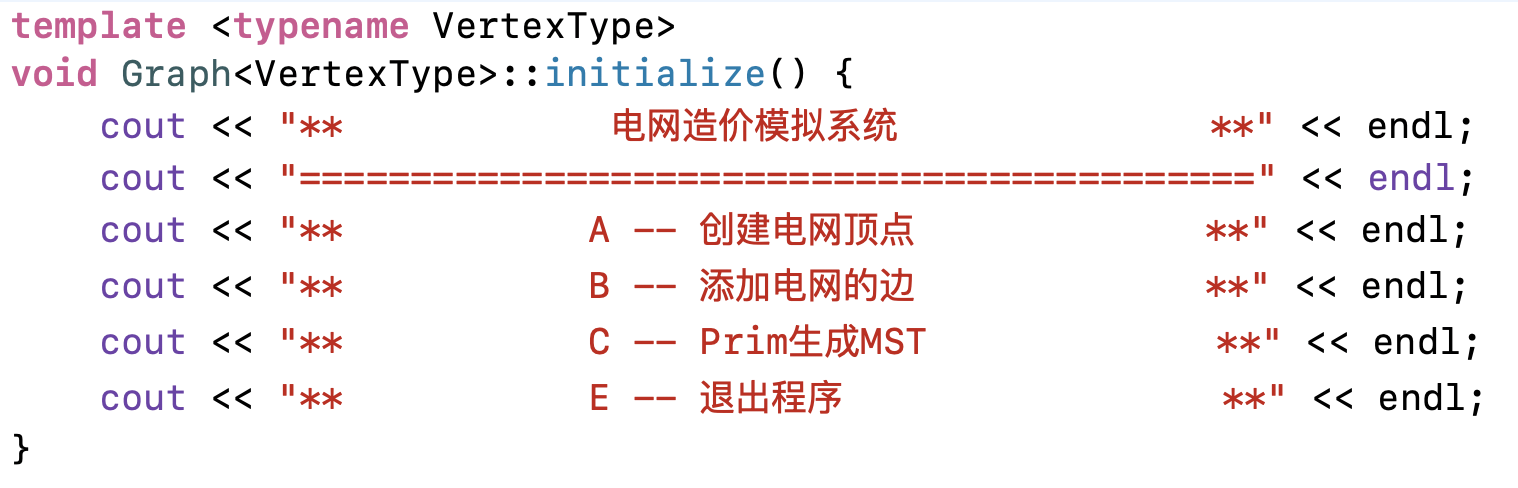
## 2.4 主函数设计

声明一个Graph<char>类的对象graph。首先调用graph.initialize()函数实现初始化、然后调用graph.run()，可以选择操作输入顶点、边的信息，可以选择操作生成最小生成树并打印。

# 3 实现

## 3.1 系统初始化功能的实现

### 3.1.1 初始化功能代码

主要是initialize() run()函数承担了系统初始化、运行的功能。它们代码如下

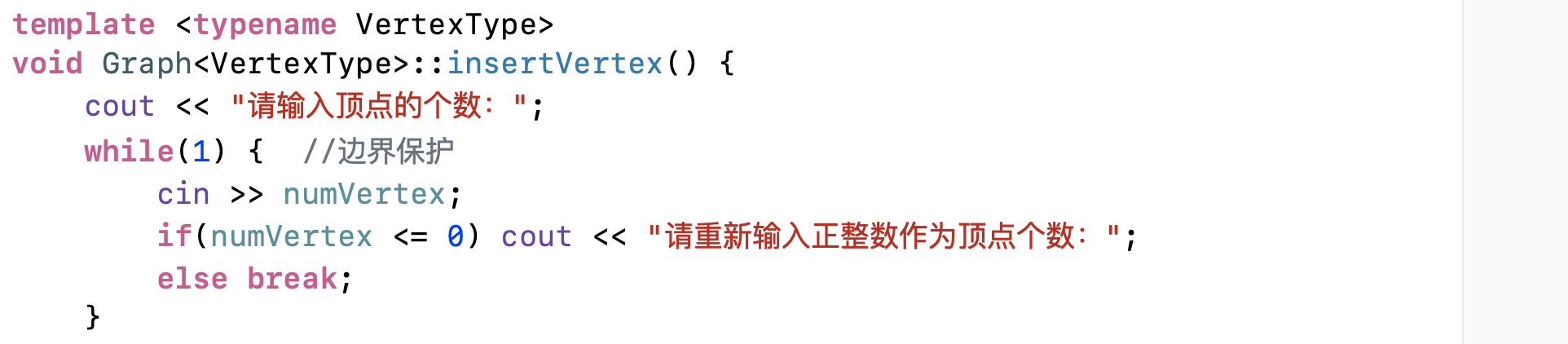
### 3.1.2 run()、initialize()函数实现时需要注意的细节

如果输入了A～D之外的操作，利用switch的default向用户报错、并重新输入；循环在操作码为D时停止，系统运行结束。

## 3.2 初始化顶点功能的实现

### 3.2.1 初始化顶点功能流程图

### 3.2.2 初始化顶点功能核心代码



## 3.3 初始化边功能的实现

### 3.3.1 初始化边功能流程图

### 3.3.2 初始化边功能核心代码

### 3.3.3 insertEdge()函数实现时需要注意的细节

因为不知道具体边的数量，所以规定以“？ ？ 0”来结束边的输入。先统计边的条数，再将边的信息存入邻接矩阵，这就要求用队列来暂存输入的信息。需要注意的是，因为数据结构中用到的多是简单图，所以对于这样一个无向图其边数的上限是numVertex\*(numVertex-1)/2，所以在边输入时需要做一个边界保护，边数达到上限后提示用户已经构成完全图、强行结束边的输入。

## 3.4生成最小生成树功能的实现

### 3.4.1 生成最小生成树功能流程图

### 3.4.2 生成最小生成树功能核心代码



### 3.4.3 Prim()函数实现时需要注意的细节

要及时把加入了MST的顶点对应序号的nearVex数组置为-1。更新lowCost数组时考虑到变化之可能是因为刚刚加入MST的minVex引起顶点进入MST的开销变小，所以只需将原lowCost数组的值与该顶点与minVex间边的大小（邻接矩阵对应的值）做比较，看是否需要更新即可。

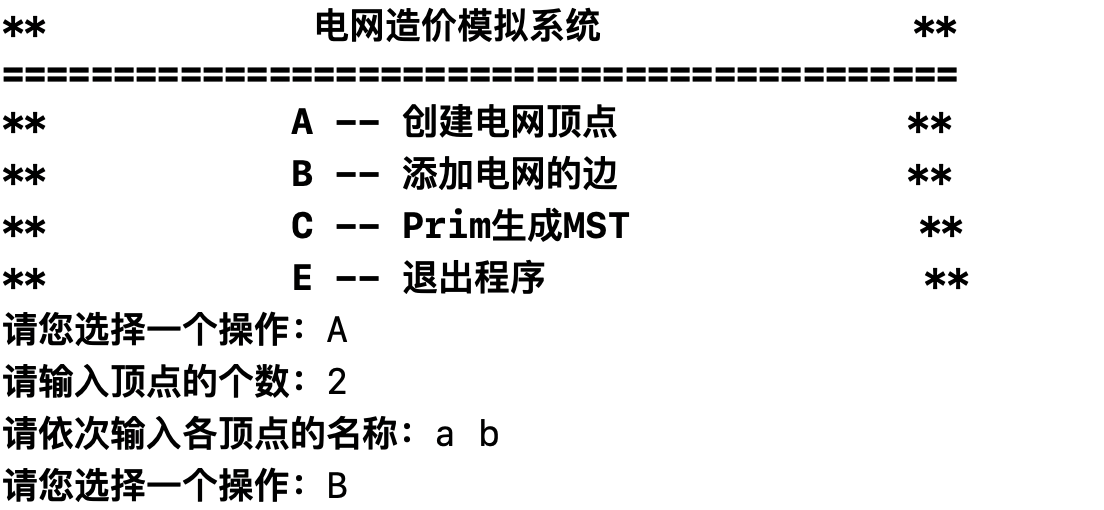
## 3.5 打印最小生成树功能的实现

### 3.5.1 打印最小生成树功能的代码

## 3.6 主函数的实现

### 3.6.1 主函数代码

### 3.6.2 总体系统截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 初始化顶点功能测试

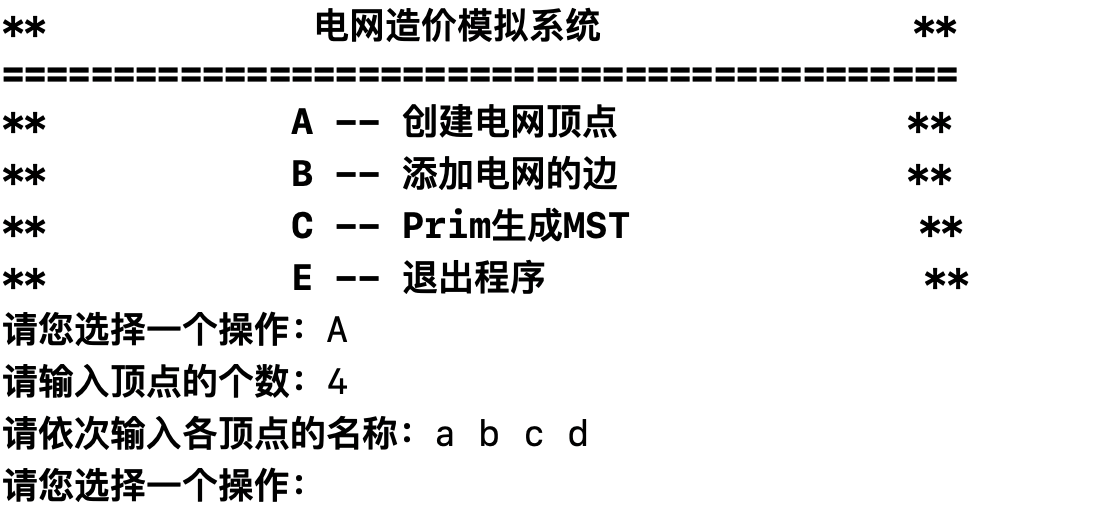
**测试用例**：

A

a b c d

**预期结果**：

成功开启插入顶点功能，a b c d四个顶点加入图中

**实验结果：**

### 4.1.2初始化边功能测试

**测试用例：**

a b 8

b c 7

c d 5

d a 11

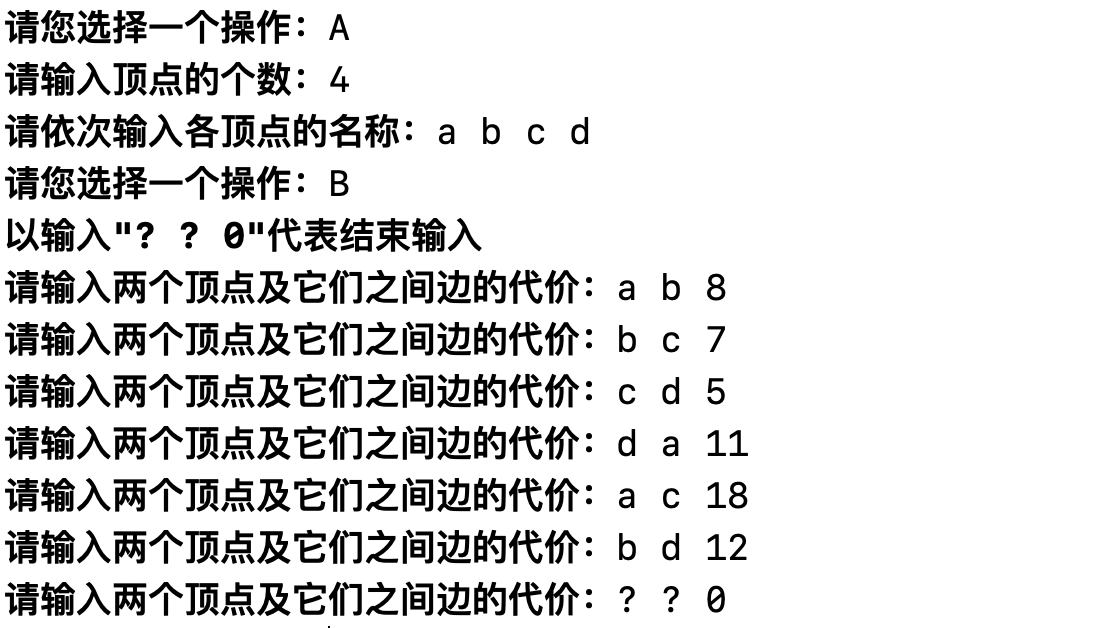
a c 18

b d 12

? ? 0

**预期结果：**

成功输入6条边。

**实验结果：**

### 4.1.3 生成最小生成树功能测试

**测试用例：**

A

a b c d

B

a b 8

b c 7

c d 5

d a 11

a c 18

b d 12

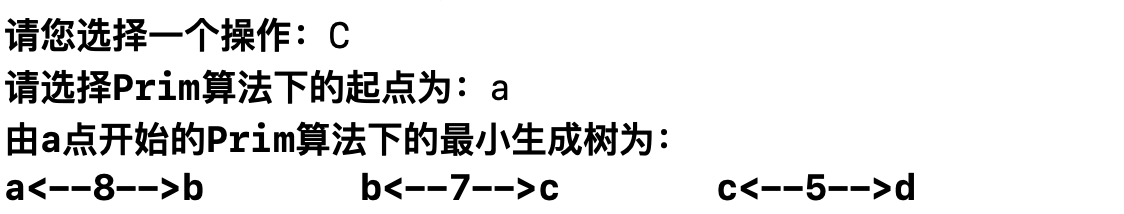
? ? 0

C

a //Prim算法的起点

**预期结果：**

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

**实验结果：**

## 4.2 边界测试

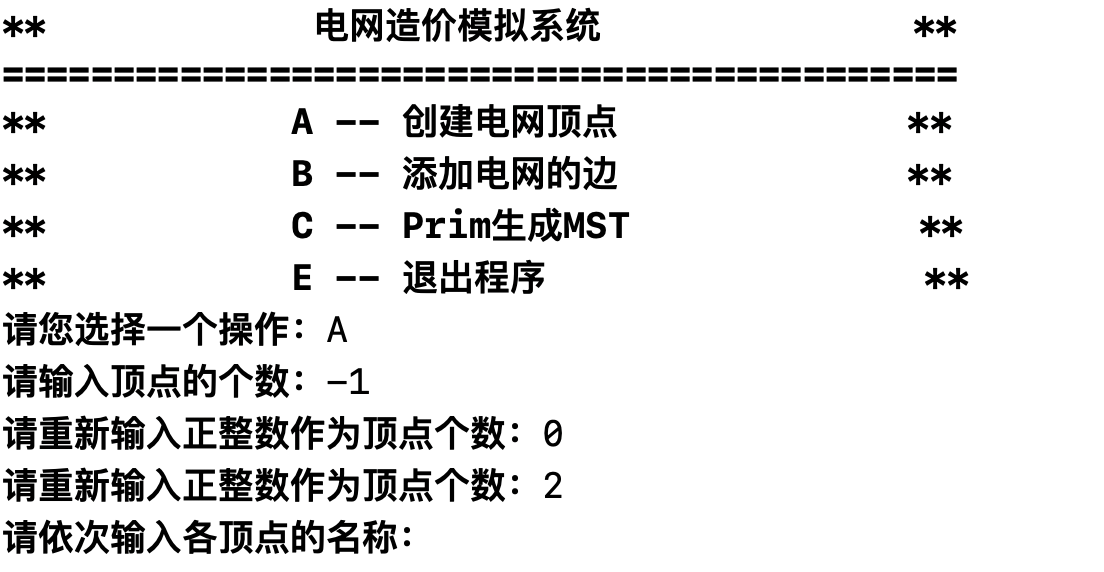
### 4.2.1 输入顶点个数不合法

**测试用例：**

尝试输入若干非正整数，如 -1 0

**预期结果：**

立即给出错误提示，并要求用户重新输入一个正整数，程序运行正常不崩溃。

**实验结果：**

### 4.2.2输入边的代价不合法

**测试用例：**

尝试输入负数作为边的代价。

A

2

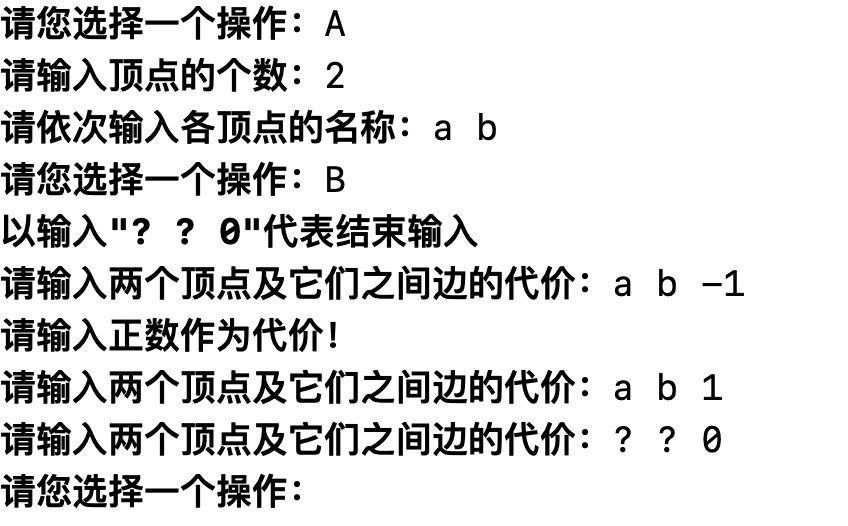
a b

B

a b -1

**预期结果：**

立即给出错误提示，并要求用户重新输入一个正数，程序运行正常不崩溃。

**实验结果：**

### 4.2.3输入边的数量不合法

**测试用例：**

尝试输入超过完全图边数量个的边。

A

2

a b

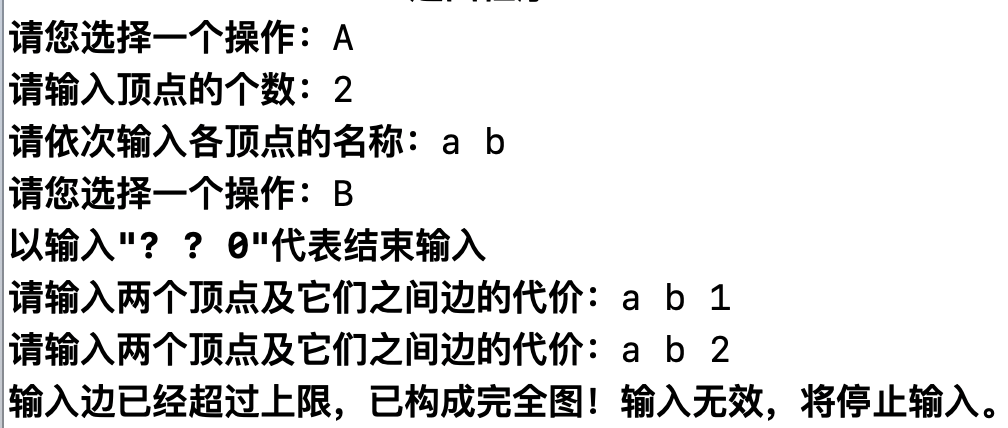
B

a b 1

a b 2

**预期结果：**

立即给出错误提示，表示已构成完成图、停止边的输入。

**实验结果：**

### 4.2.4输入边的顶点不合法

**测试用例：**

尝试输入不存在图中的顶点作为边两侧的顶点。

A

2

a b

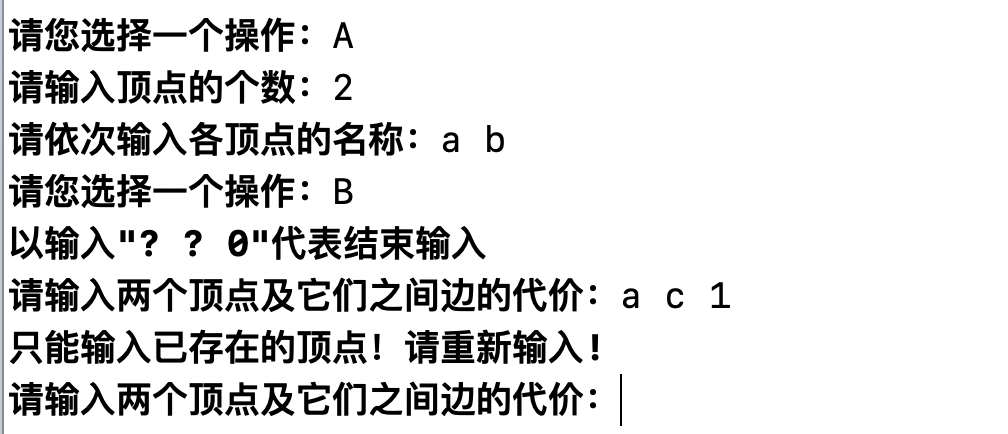
B

a c 1

**预期结果：**

立即给出错误提示，并要求用户重新存在图中的顶点，程序运行正常不崩溃。

**实验结果：**

****

# 5 总结

## 5.1 遇到的错误

### 5.1.1 xx函数中

## 5.2 改进与优化

## 5.3 项目心得