数据结构课程设计

项目说明文档

**电网建设造价模拟系统**



同济大学

Tongji University

姓名： 林觉凯

学号： 2253744

指导老师： 张颖

学院专业： 软件学院 软件工程

**目录**

**1.项目分析-------------------------------------------------------------------------3**

**1.1 项目背景分析-------------------------------------------------------------------------3**

**1.2项目功能分析--------------------------------------------------------------------------3**

**1.2.1项目功能要求--------------------------------------------------------------------------------3**

**1.2.2项目输入要求--------------------------------------------------------------------------------3**

**1.2.3项目输出要求--------------------------------------------------------------------------------3**

**1.2.4项目示例--------------------------------------------------------------------------------------3**

**2.项目设计-------------------------------------------------------------------------4**

**2.1数据结构设计--------------------------------------------------------------------------4**

**2.2类设计-----------------------------------------------------------------------------------4**

**2.2.1 边结点类(Edgenode)----------------------------------------------------------------------4**

**2.2.2 图类(Graph)--------------------------------------------------------------------------------4**

**2.3成员与操作设计-----------------------------------------------------------------------5**

**2.4系统流程设计--------------------------------------------------------------------------5**

**3.项目实现-------------------------------------------------------------------------6**

**3.1 创建电网结点的实现----------------------------------------------------------------6**

**3.2 添加电网的边的实现----------------------------------------------------------------7**

**3.3 构造最小生成树的实现-------------------------------------------------------------8**

**3.4 显示最小生成树的实现 ----------------------------------------------------------11**

**4.项目测试------------------------------------------------------------------------12**

**4.1项目功能测试-------------------------------------------------------------------------12**

**4.2代码健壮性测试----------------------------------------------------------------------13**

**4.2.1 输入非法的健壮性测试------------------------------------------------------------------13**

**4.2.2 输入不存在的健壮性测试---------------------------------------------------------------13**

**4.2.3 输入重复等健壮性测试------------------------------------------------------------------14**

**5.项目的心得与体会------------------------------------------------------------14**

**1.项目分析**

**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。请设计一个能够满足要求的造价方案。

**1.2 项目功能分析**

**1.2.1项目功能要求**

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。

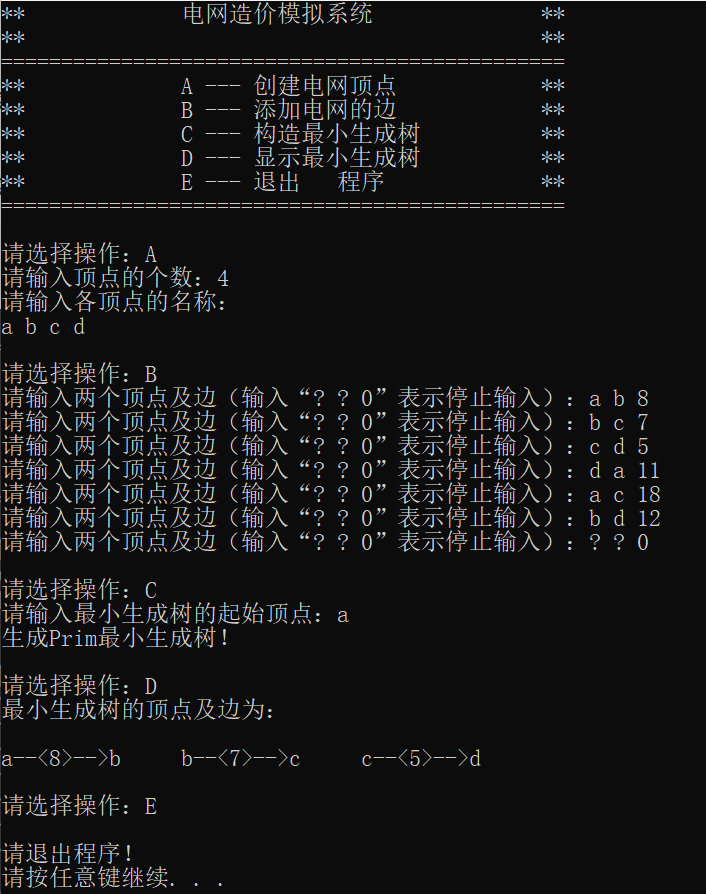
**1.2.2项目输入要求**

用户输入菜单选项并执行相应的操作，这些操作需要用户输入相应的数据。

**1.2.3项目输出要求**

按照用户的要求输出指定的操作提示语句，并打印出最小生成树。

**1.2.4项目示例**

****

**2.项目设计**

**2.1 数据结构设计**

此次项目需要实现最小生成树的生成并打印。

图是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。本次项目中的图为无向图。

此外，图的存储结构有两种：一种是邻接矩阵法，另一种则是邻接表法。本程序使用的是邻接矩阵法。

**2.2类设计**

为了实现最小生成树的生成和打印，我设计了两个类：边结点类和图类，两个类之间的耦合关系可以采用嵌套、继承等多种关系。本程序也构造了这两个基本的边结点类和图类来进行相应的操作。

**2.2.1边结点类**(Edgenode)

//边结点类，每一个边结点都记录着该边的起始顶点、终止顶点和边的权重

class Edgenode

{

public:

char start; //边的起始顶点

char end; //边的终止顶点

int weight; //边的权重

Edgenode() { start = end = ' '; weight = 0; }

};

**2.2.2图类**(Graph)

//图类

class Graph

{

public:

int Vertices\_num = 0; //图的顶点数

int Edges\_num = 0; //图的边数

char\* Vertices\_list = NULL; //顶点数组指针

int\*\* Edge\_matrix = NULL; //邻接矩阵指针

Edgenode\* Minispantree = NULL; //最小生成树指针

int Transform\_index(char Nodename);

void Add\_node();

void Add\_edge();

void Create\_tree();

void Print\_tree();

};

**2.3成员与操作设计**

边结点类的公有成员边的起始顶点、终止顶点和边的权重；图类的公有成员为图的顶点数、图的边数、顶点数组指针、邻接矩阵指针和最小生成树指针；图类的公有操作为将图中顶点的名称转化为该边在顶点在顶点数组中的位置下标、顶点的添加、边的添加、最小生成树的生成和最小生成树的打印。

**2.4系统流程设计**

系统操作的整体流程大致如下：

**程序开始运行**

**选择菜单选项**

**创建结点、添加边、构造最小生成树、显示最小生成树、退出**

**输出相应结果**

**3.项目实现**

**3.1 创建电网结点的实现**

//添加结点函数

void Graph::Add\_node()

{

cout << "请输入顶点的个数：";

int Node\_num;

//输入的错误处理（非法字符和顶点个数限制）

while (1)

{

cin >> Node\_num;

if (cin.good() == 0)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入错误！请重新输入！" << endl; //非法字符

continue;

}

if (Node\_num < 2)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "顶点数应该是大于等于2的正整数，请重新输入！" << endl;

//顶点个数大于等于2

continue;

}

break;

}

Vertices\_num = Node\_num; //顶点个数

Vertices\_list = new char[Vertices\_num]; //顶点数组

cout << "请输入各顶点的名称：" << endl;

for (int i = 0; i < Vertices\_num; i++) //创建顶点数组

cin >> Vertices\_list[i];

}

创建电网结点首先需要对要创建电网结点的个数进行健壮性测试，（不能输入非法数据或是结点数小于2）；创建结点实际上是将顶点的名称和对应邻接矩阵的下标位置（也即在一维顶点数组的下标）联系起来。所以我们需要为Graph类的公有成员Vertices\_list分配Vertices\_num个空间，之后将对应位置的顶点名称逐一对应输入进去即可。

**3.2 添加电网的边的实现**

//添加边函数

void Graph::Add\_edge()

{

//前提判断（是否已经添加顶点）

if (Vertices\_num == 0)

{

cout << "无顶点！请先添加顶点！" << endl;

return;

}

//为邻接矩阵分配空间

Edge\_matrix = new int\* [Vertices\_num];

for (int i = 0; i < Vertices\_num; i++)

Edge\_matrix[i] = new int[Vertices\_num];

for (int i = 0; i < Vertices\_num; i++) //初始化邻接矩阵

{

for (int j = 0; j < Vertices\_num; j++)

{

if (i == j)

Edge\_matrix[i][j] = 0; //自己到自己的边权重为0

else

Edge\_matrix[i][j] = MAXWEIGHT; //其它的边初始化为MAXWEIGHT

}

}

char start, end;

int index\_1, index\_2;

int weight;

while (1)

{

cout << "请输入两个顶点及边（输入“? ? 0”表示停止输入）：";

cin >> start >> end >> weight;

if (start == '?' || end == '?' || weight == 0) //依据题目要求设定输入中止条件

break;

index\_1 = Transform\_index(start); //将start顶点转化为其在结点数组的下标

index\_2 = Transform\_index(end); //将end顶点转化为其在结点数组的下标

if (index\_1 == -1 || index\_2 == -1) //判断边是否存在

{

cout << "你输入的边不存在！" << endl;

continue;

}

else if (Edge\_matrix[index\_1][index\_2] == 0) //判断输入的两顶点是否重复

{

cout << "你输入的两个顶点相同！请重新输入！" << endl;

continue;

}

else if (Edge\_matrix[index\_1][index\_2] > 0 && Edge\_matrix[index\_1][index\_2] != MAXWEIGHT) //判断是否重复输入两顶点和其边权重

{

cout << "你重复输入两顶点的边关系！输入无效！请重新输入！" << endl;

continue;

}

else if (index\_1 > -1 && index\_1 < Vertices\_num && index\_2 > -1 && index\_2 < Vertices\_num && Edge\_matrix[index\_1][index\_2] == MAXWEIGHT)

{

Edge\_matrix[index\_1][index\_2] = Edge\_matrix[index\_2][index\_1] = weight; //输入起始顶点、终止顶点和对应权重并赋值完成

Edges\_num++; //边数加一

}

}

}

在添加电网边的前提是需要有结点，所以一开始我们做了一个前提判断。为邻接矩阵分配空间的过程类似于动态分配数组（只不过需要分配两次）。下一步边需要将顶点的名称转化为其在顶点数组的相应下标。初始邻接矩阵是将自身指向自身的路径权重设为0，其它在初始化时都没有连接，故为MAXWEIGHT。之后通过输入顶点的信息和对应边的权重为邻接矩阵赋值（有许多错误输入的判断比如顶点是否存在、是否重复等等）。

**3.3 构造最小生成树的实现**

//创建最小生成树函数

void Graph::Create\_tree()

{

//前提判断（是否已经输入顶点）

if (Vertices\_num == 0)

{

cout << "没有顶点！无法创建最小生成树！请首先创建顶点和边的关系！" << endl;

return;

}

//前提判断（是否已经创建顶点之间边的关系）

if (Edge\_matrix == NULL)

{

cout << "没有顶点之间的边关系！无法创建最小生成树！请先创建顶点之间边的关系！" << endl;

return;

}

//lowcost数组，存放生成数顶点集合内各顶点到生成树外各顶点的个边上的最小权值

int\* lowcost = new int[Vertices\_num];

//nearvex数组，记录生成树顶点集合外各顶点距离集合内哪个顶点最近（权重最小）

int\* nearvex = new int[Vertices\_num];

char Start\_vertice;

cout << "请输入最小生成树的起始顶点：";

cin >> Start\_vertice;

while (Transform\_index(Start\_vertice) == -1)

{

cout << "你输入的最小生成树的起始顶点不存在！请重新输入！";

cin >> Start\_vertice;

}

int Start\_index = Transform\_index(Start\_vertice);

//输入起始顶点并转化为该顶点在顶点数组内的下标

lowcost[Start\_index] = 0;

nearvex[Start\_index] = -1;

for (int i = 0; i < Vertices\_num; i++)

{

if (i != Start\_index)

{

lowcost[i] = Edge\_matrix[Start\_index][i];

nearvex[i] = Start\_index;

}

} //初始化两个数组

Minispantree = new Edgenode[Vertices\_num - 1]; //最小生成树内应该有顶点数减一个边结点

int count = 0;

for (int i = 1; i < Vertices\_num; i++)

{

int min = MAXWEIGHT;

int v = Start\_index;

//求生成树外顶点到生成树内顶点具有最小权值的边，v是当前具有最小权值的边的位置

for (int j = 0; j < Vertices\_num; j++)

{

if (nearvex[j] != -1 && lowcost[j] < min)

{

v = j;

min = lowcost[j];

}

}

if (v != Start\_index) //v==Start\_index表示再也找不到要求的顶点了

{

Edgenode e; //最小生成树结点的辅助单元

e.start = Vertices\_list[nearvex[v]];

e.end = Vertices\_list[v];

e.weight = lowcost[v];

Minispantree[count++] = e; //选出的边结点计入生成树

nearvex[v] = -1; //作该边已加入生成树的标记

for (int j = 1; j < Vertices\_num; j++) //j不在生成树中

{

if (nearvex[j] != -1 && Edge\_matrix[v][j] < lowcost[j]) //需要修改

{

lowcost[j] = Edge\_matrix[v][j];

nearvex[j] = v;

}

}

}

}

cout << "生成Prim最小生成树！" << endl;

}

构造最小生成树有许多方法，这里选择Prim算法来构造最小生成树。一开始我们同样做了条件判断，即是否存在顶点和对应边的关系。下面便是Prim算法的具体实现过程。我们需要构造两个辅助数组lowcost和nearvex,前者用于存放生成数顶点集合内各顶点到生成树外各顶点的个边上的最小权值，后者用于记录生成树顶点集合外各顶点距离集合内哪个顶点最近（权重最小）。从初始状态出发，反复在lowcost中选择不在最小生成树中的边（即nearvex[i]!=-1&&lowcost[i]最小），然后将其加入最小生成树集合（nearvex[i]置为-1），此时需要遇到一个辅助的最小生成树结点单元。之后再用生成树顶点集合外各顶点i到刚加入该集合的新顶点v的距离和他原来到生成树顶点集合中顶点的最短距离作比较，取距离近的作为这些集合外顶点到生成树顶点集合内顶点的最短距离。

**3.4 显示最小生成树的实现**

//打印最小生成树函数

void Graph::Print\_tree()

{

//前提判断（是否已经输入顶点）

if (Vertices\_num == 0)

{

cout << "没有顶点！无法显示最小生成树！请首先创建顶点和边的关系！" << endl;

return;

}

//前提判断（是否已经创建顶点之间边的关系）

if (Edge\_matrix == NULL)

{

cout << "没有顶点之间的边关系！无法打印最小生成树！请先创建顶点之间边的关系！" << endl;

return;

}

//前提判断（是否已经生成最小生成树）

if(Minispantree == NULL)

{

cout << "未创建最小生成树，无内容可显示！请创建最小生成树：" << endl;

return;

}

cout << "最小生成树的顶点及边为：" << endl << endl;

for (int i = 0; i < Vertices\_num - 1; i++) //遍历最小生成树内边结点并输出

cout << Minispantree[i].start << "--<" << Minispantree[i].weight << ">-->" << Minispantree[i].end << " ";

cout << endl;

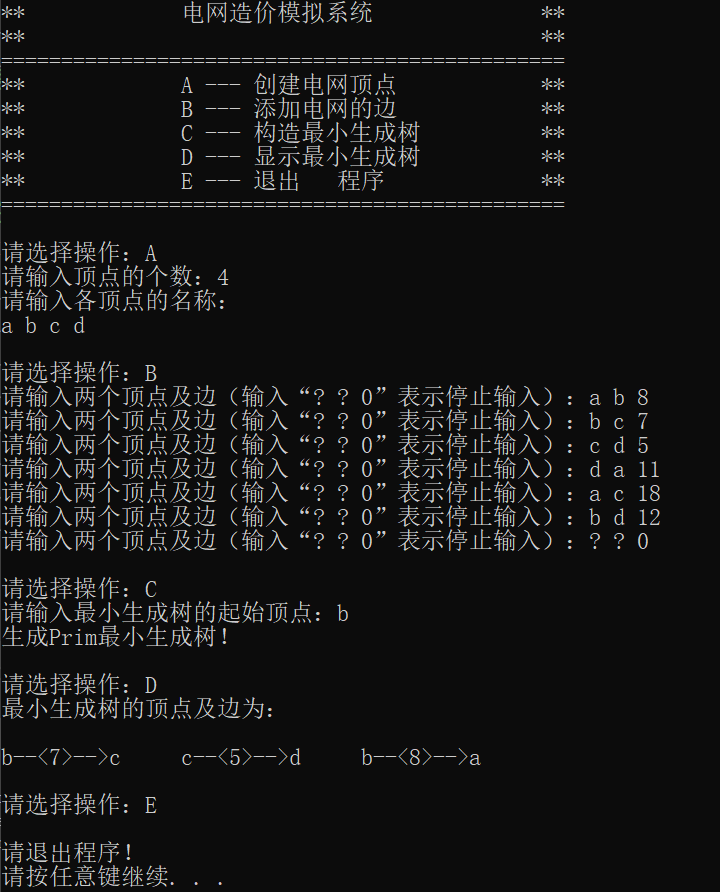
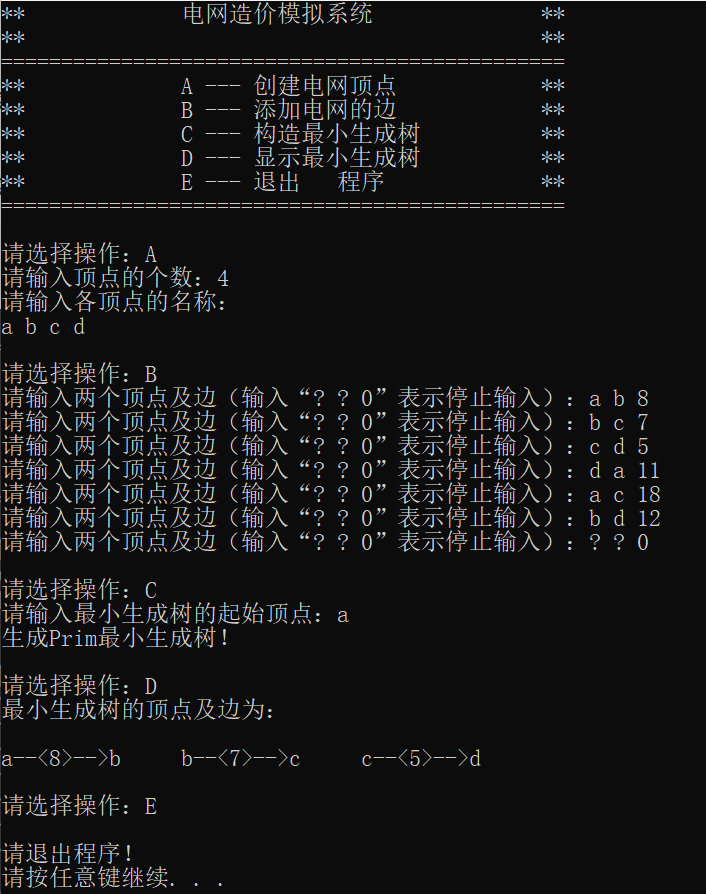
}

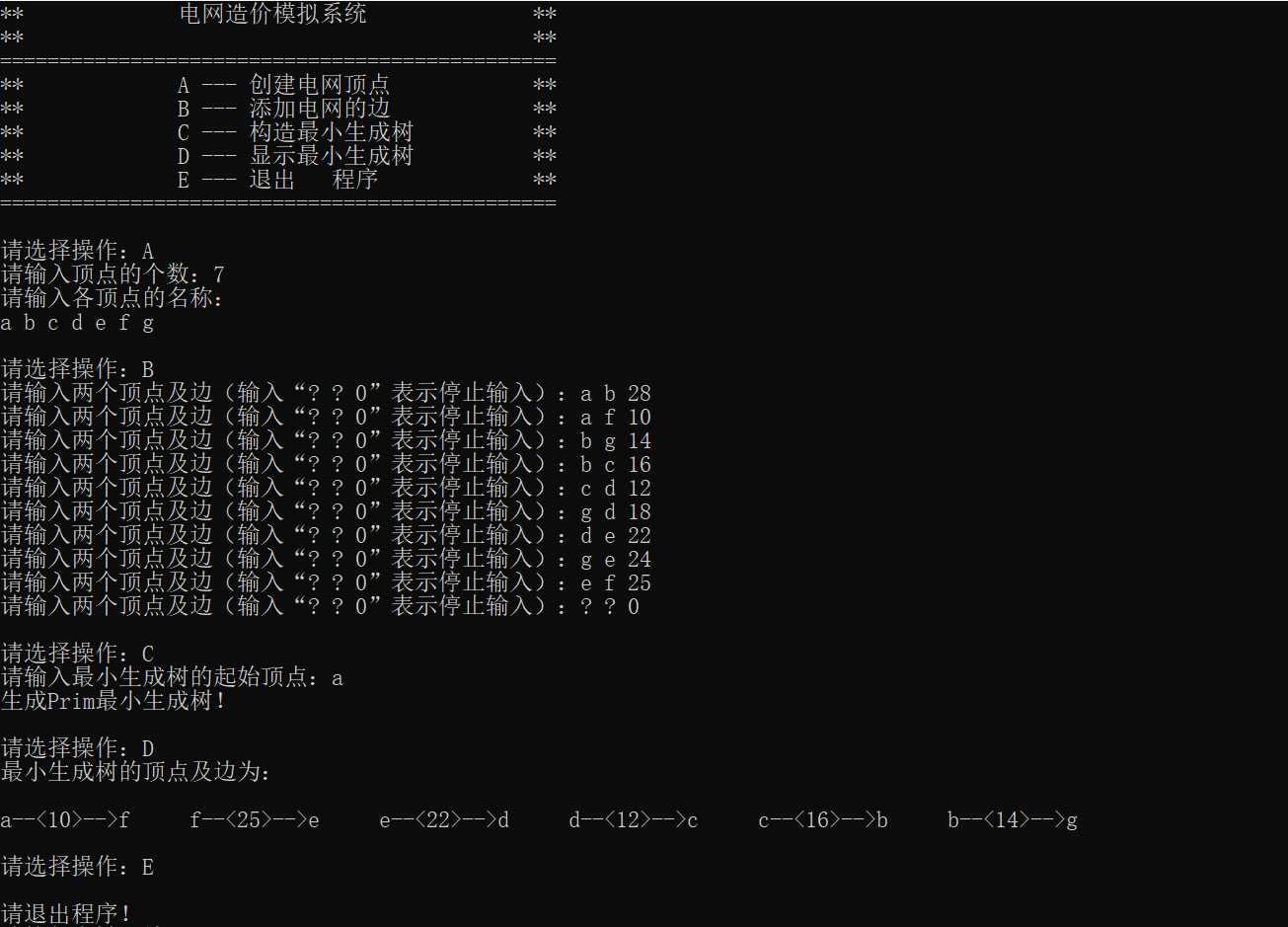
打印出最小生成树的操作较为简单，首先我们还需要进行条件判断（是否添加顶点、是否添加顶点之间边与边的关系和是否构造完成最小生成树）。之后在遍历Graph类的公有成员Minispantree数组将每一个边结点输出（注意，在顶点个数为Vertices\_num的图中最小生成树的边有Vertices\_num-1条）。

**4.项目测试**

**4.1 项目功能测试**

测试用例1 测试用例2

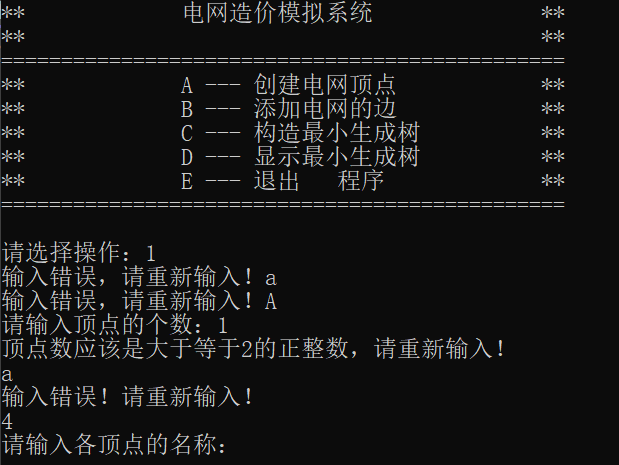


 测试用例3(为老师上课PPT内容，输出正确)

**4.2 代码健壮性测试**

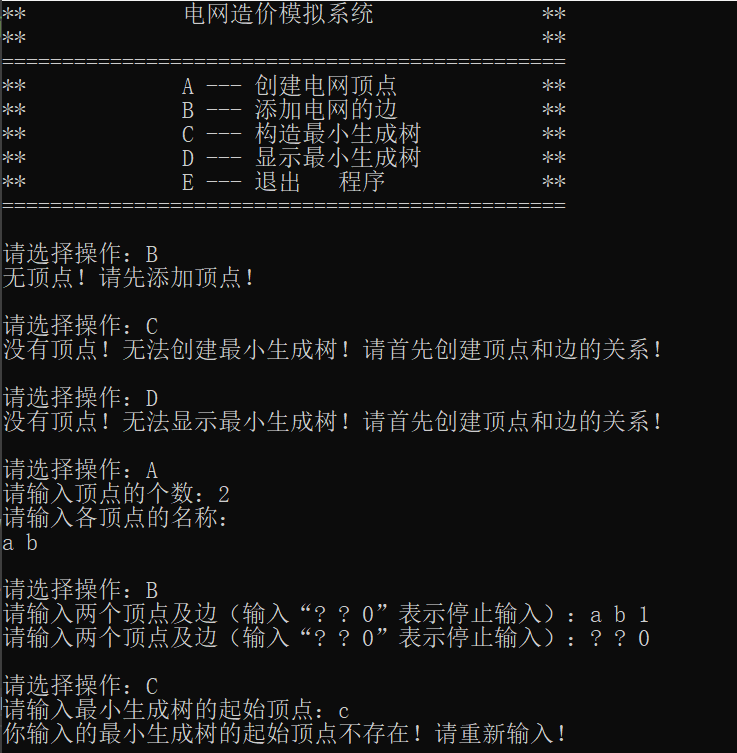
**4.2.1 输入非法的健壮性测试**

在输入菜单选项操作数、输入顶点数、顶点的名称等等部分该程序都做了相应的代码健壮性完善。

****

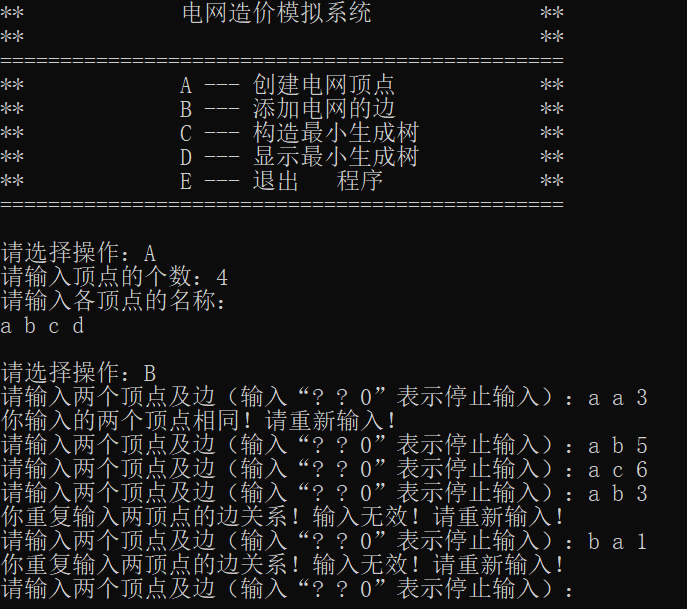
**4.2.2 输入不存在的健壮性测试**

在输入最小生成树的起点时不存在、或是在执行操作B、C或D先前的准备操作没有完成，程序都会给出相应的提醒错误。



**4.2.3 输入重复等健壮性测试**

在输入顶点之间边的关系时，当输入重复的顶点（相同或是相反（因为时无向图所以a-b和b-a相同）），都会给出相应的提示。

****

**5.项目心得与体会**

此次项目利用图的性质来模拟电网造价系统。图经常表达两个事物中是否具有某些关系。在此次项目中，我首先加深了用邻接矩阵来表示图的方法的影响，邻接矩阵是表示图的一种方便又简单的方法；其次，我学会了用Prim算法来构造最小生成树，Prim算法的主要思想是“寻找点”，过程比较复杂，但是在编写的过程一步步实现就比较易懂。本次项目还有许多错误处理的机制，需要我们注意每一个选项实现的顺序和每一种情况实现的条件（尤其是在Prim算法中，又很多判断条件），这需要我们细心一点。在之后的学习中还需要多做一些相应的练习来巩固对图和关于图的各类算法的理解。