****

**Tongji University**

**数据结构课程设计实验报告**

**008-电网建设造价模拟系统**

专 业： 软 件 工 程

指导教师： 张 颖

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **项目概览**
   1. **项目背景**

电网的建造是关键的民生问题之一。电网建设需要将保证电力能够运输到所有村庄聚落，而在此基础之上，需要考虑电网的建设造价问题。整个输电网络可以看做一张图，而求电网最低造价即是求加权边长的图的权最小构造方法，即最小生成树。

* 1. **项目要求**
     1. **项目整体要求**

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。

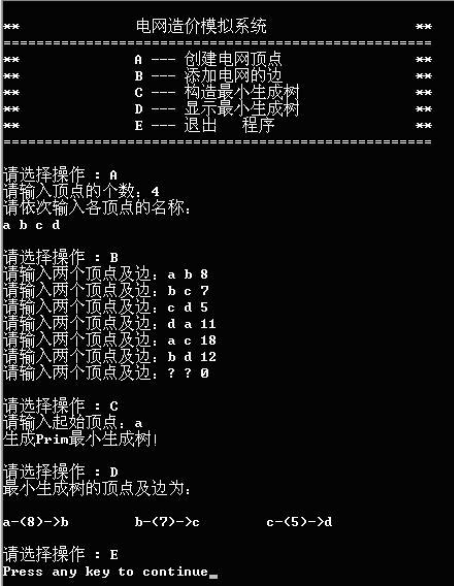
* + 1. **项目输入要求**

项目输入要求实现客户手动键入想进行的操作（A-E，不区分大小写），程序得到命令后自动跳转到各个选项，并根据选项功能进一步提示用户输入。

* + 1. **项目输出要求**

要求项目按照不同功能下进行个性化输出。

* + 1. **项目示例**



* 1. **项目需求分析**

根据项目要求可以得到，本次项目的实质是将电网的各个站点视作图的各个顶点，将电网输电线视作图的边。求电网造价最小即是求最小生成树的过程。因此本项目需要实现图的数据结构，并在此基础上利用Prim算法实现求最小生成树。

* + 1. **项目功能完善**

由于电网是关键的民生问题之一，需要程序功能完善，保证用户能够创建电网顶点，创建电网输电线，生成最小生成树以及打印最小生成树几个功能。在用户输入时要进行输入错误处理并给出合理提示。

* + 1. **执行效率优化**

为了提高用户的使用体验，需要优化程序的执行效率。这一点可以从程序的空间开销和时间开销入手，尽量避免不必要的递归、循环等。

* + 1. **代码的健壮性**

由于程序在用户生活中起到重要作用，故需要保证代码的健壮性。当用户进行不正确使用时，代码需要进行报错提示，避免崩溃退出。同时也要保证存储结构封装安全，数据存储严谨无遗漏。

* + 1. **最小生成树可视化**

为了方便用户随时核对已生成的电网成果，仅数字提示并不能满足直观性。因此实现最小生成树的可视化，将各个顶点及其夹边打印出来，以供客户随时查看。

1. **数据结构设计与算法使用**
   1. **数据结构的使用与设计**

本次项目问题较为复杂，涉及数据结构较多。主要使用了双向链表、向量、成对和图的数据结构。

图作为主题框架支撑起实现程序功能，构造最小生成树的任务，而向量作为图的低一层数据结构负责存储图的顶点和边以及最小生成树。成对类在程序实现过程中将两个有关系的量联系起来，起到辅助作用。而双向链表则是作为程序的底层数据结构，负责数据的底层存储。

* + 1. **双向链表的设计**

本次项目为了避免占用内存过大，同时考虑到项目存在很多的数据插入和数据删除操作，故使用链表作为底层数据结构。而考虑到程序的执行效率问题，为了避免链表的多次遍历提高时间开销，故选择双向链表，通过牺牲空间复杂度来提高时间复杂度，提高程序运行效率。

本次项目链表类设计与之前项目相同，故此处不再赘述。

* + 1. **向量类的设计**

本次项目需要实现数据结构来方便实现图的顶点和边的存储。因此选择向量作为存储各个数据的数据结构。其以链表作为底层数据结构，承担存储电网邻接表，生成的最小生成树等的作用。

本次的向量类设计与之前相同，故此处不再赘述。

* + 1. **图类的设计**

图（graph）是一种灵活的非线性结构。图一般由顶点集合（vertex）及顶点间的关系集合组成的一众数据结构：Graph = ( V , E )。其中顶点集合是有穷非空集合，边（edge）集合是顶点间关系的有穷集合。

由于本次项目中电网各个基站之间是由无向输电线相连，因此可以设计成为无向图。

对于图存储结构的选用，包括邻接矩阵法和邻接表法。考虑到两种存储方法在执行遍历操作和Prim算法时，都是邻接表法时间复杂度较低，因此为了降低时间开销提高程序运行速率，本次项目选择邻接表法对图进行存储。

具体图类实现如下：

1. 图类的数据存储：

需要声明图的顶点变量，邻接表变量和最小生成树变量进行数据存储。顶点和最小生成树均可利用向量进行直接存储，而邻接表需要把向量类中的每一个对象添加链表特征实现。

1. //顶点

Vector<string> vertex;

1. //邻接表

Vector<linkList<Pair<int, int>>> adList;

1. //prim算法生成的最小生成数

Vector<Pair<int, int>> mstTree;

1. 图类功能函数的实现

图类主要需要实现初始化，清空，判断顶点和构造最小生成树几个函数。为了保证程序完整还需要实现顶点的建立加入，边的建立（隐含邻接表的建立思想），构造与打印最小生成树的功能函数。

1. //清空图

void clear();

1. //Prim算法构造最小生成树

void prim(int pos);

1. //判断顶点是否已经在图中

int IsVertex(const string& vertex);

1. //比较两个字符串的辅助函数

bool strcmp(const string& str1, const string& str2);

1. //创建顶点

void setVertex();

1. //创建边

void setEdge();

1. //构造最小生成数

void buildMST();

1. //打印最小生成树

void printMST();

* 1. **算法功能的实现**
     1. **Prim算法的实现**

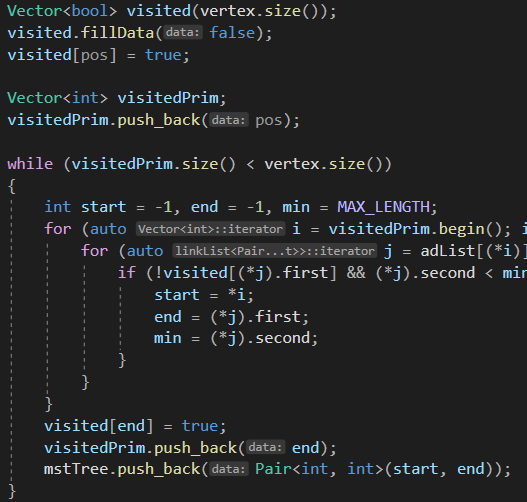
连通图中的每一棵生成树，都是原图的一个极大无环子图。因此从中删去任何一条边，生成树就不再连通；反之，在其中引入任何一条新边，都会形成（恰好）一个回路。

若连通网络由n个顶点组成，则其生成树必含n个顶点，n-1条边。因此，构造最小生成数的准则有三条：

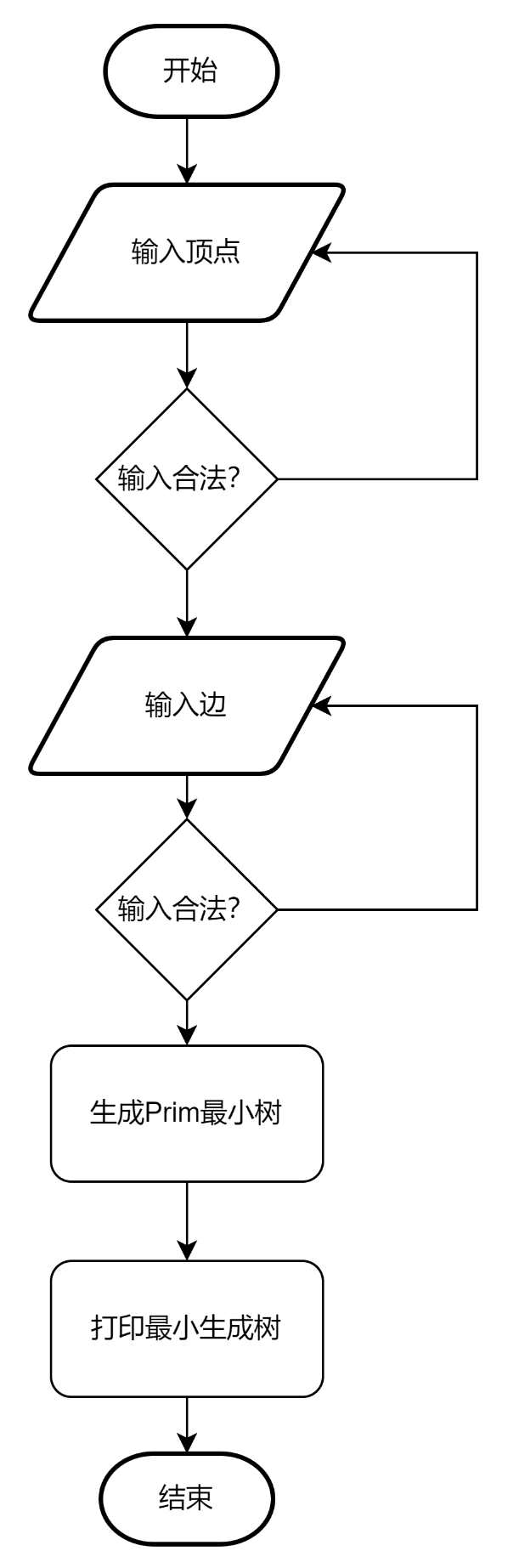
1. 只能使用该网络中的边来构造最小生成树；
2. 只能使用恰好n-1条边来联络网络中的n个顶点；
3. 选用的n-1条边不能构成回路。

构造最小生成树的算法，较为典型的是：Kruskal算法和Prim算法，本次项目选择使用Prim算法，这一算法采用了逐步求解（grady）的策略，是一种贪心策略。

算法主要思想是迭代，每次迭代都将顶点分成在最小生成树中和不在最小生成树中两部分，选择两部分间权值最小的边作为最小生成树的新边，随其顶点一起加入最小生成树。因此每一轮迭代都会使最小生成树中新增加一点一边。当其包含n个顶点，n-1条边时，算法即可终止。



1. **项目流程图解**
   1. **项目主体流程图解**

****

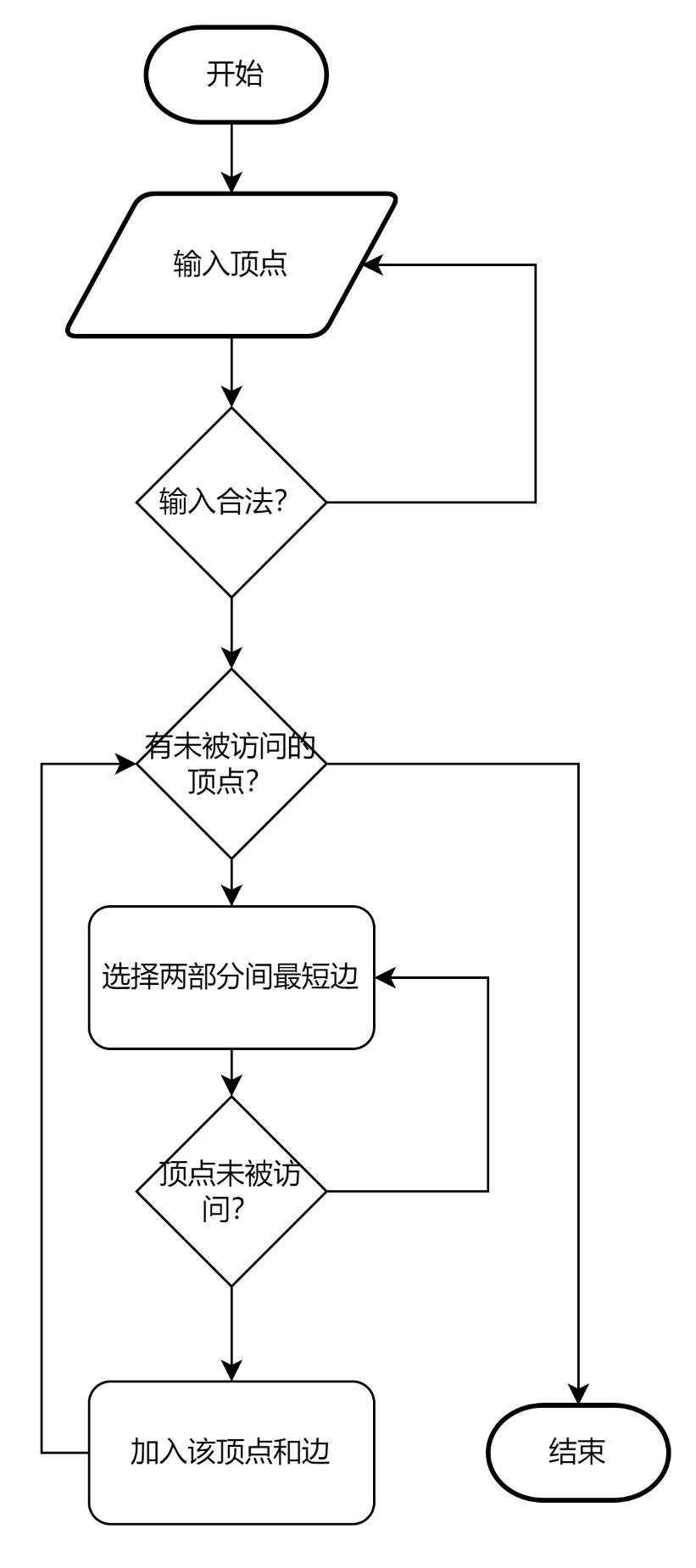
否

是

否

是

* 1. **Prim算法流程图解**

****

是

否

否

否

是

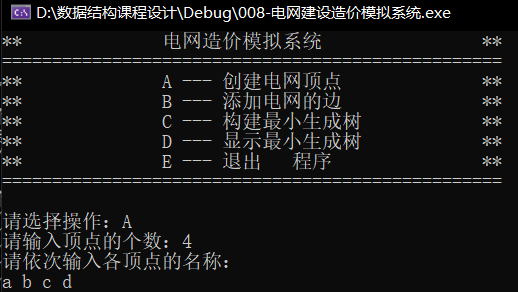
是

1. **程序测试**
   1. **添加顶点功能测试**

测试数据：A

4

a b c d

****

* 1. **添加边功能测试**

测试数据：B

a b 8

b c 7

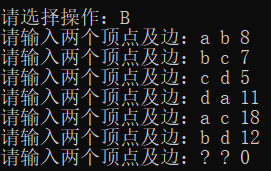
c d 5

d a 11

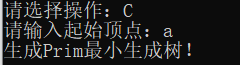
a c 18

b d 12

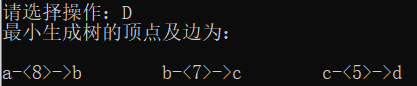
? ? 0

****

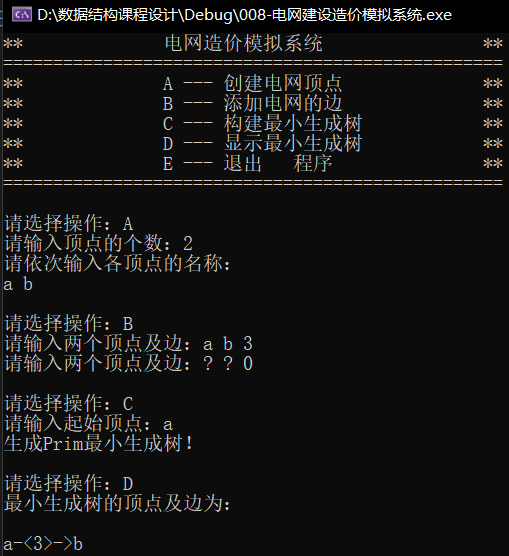
* 1. **构造Prim最小树功能测试**

****

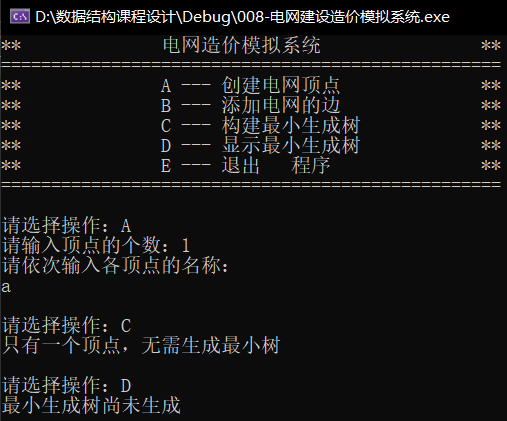
* 1. **打印最小生成树功能测试**

****

* 1. **边界条件测试**
     1. **只有两顶点的最小生成树**

****

* + 1. **只有一顶点的图**

****