**项目说明文档**

**数据结构课程设计**

**——考试报名系统**

作者姓名：香宁雨

学号：1954098

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.题目要求及功能分析 2](#_Toc58787756)

[2.设计 2](#_Toc58787757)

[2.1数据结构设计 2](#_Toc58787758)

[2.2算法设计 2](#_Toc58787759)

[2.3结构体的成员与方法 3](#_Toc58787760)

[2.4测试函数 3](#_Toc58787761)

[3.实现 5](#_Toc58787762)

[3.1Edge结构体 5](#_Toc58787763)

[3.2 unionFind(map<string, string> m, string x)函数 6](#_Toc58787764)

[3.3 checkUnionFind(map<string, string>m)函数 6](#_Toc58787765)

[3.3.1流程图 7](#_Toc58787766)

[3.3.2代码实现 7](#_Toc58787767)

[3.4 Kruskal(vector<Edge> edge, map<string, string> m, vector<string> vertex)函数 8](#_Toc58787768)

[3.4.1流程图 9](#_Toc58787769)

[3.4.2代码实现 10](#_Toc58787770)

[3.5 minKey(map<string,int>key,vector<Edge>& ans,map<string,bool> is\_visited)函数 10](#_Toc58787771)

[3.6 Prim(vector<Edge> edge, vector<string> vertices, string x) 11](#_Toc58787772)

[3.6.1流程图 12](#_Toc58787773)

[3.6.2代码实现 14](#_Toc58787774)

[3.7输出函数 16](#_Toc58787775)

[3.8边是否合法判断 16](#_Toc58787776)

[4.输出测试 17](#_Toc58787777)

[4.1样例输入输出 17](#_Toc58787778)

[4.2不存在最小生成树输入输出 18](#_Toc58787779)

[4.3输入不存在点的边 18](#_Toc58787780)

[4.4错误输入 19](#_Toc58787781)

# 1.题目要求及功能分析

**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。

**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。

**功能分析**：为联通各点并得到最小的树，即为找整张图中的最小生成树。根据题意，本图为无向图，我们需要在此无向图中找到本图的最小生成树。

# 2.设计

## 2.1数据结构设计

本次需处理数据为一张图，所以我们采用结构体来表示我们所需要的边，并用数组这种线性结构进行存储。

2.2算法设计

最小生成树有两种生成方法，一种为Kruskal，一种为Prim，本项目中两种方法都进行了编写，测试者可用不同方法进行代码的测试。

Kruskal算法是基于贪心的思想得到的。首先我们把所有的边按照权值先从小到大排列，接着按照顺序选取每条边，如果这条边的两个端点不属于同一集合，那么就将它们合并，直到所有的点都属于同一个集合为止。

这里的合并集合我们采用了并查集的算法。在计算机科学中，并查集是一种树型的数据结构，用于处理一些不交集（Disjoint Sets）的合并及查询问题。有一个联合-查找算法（union-find algorithm）定义了两个用于此数据结构的操作：Union 以及Find，Union为将两个集合合并，Find是对合并过后的集合进行查找。

而Prim算法更偏向于具有点权的最小生成树，在具有边权的图中生成最小生成树时会将这个该结点所连接的最小边（不包括已经存在于最小生成树中的边）作为点权然后进行当前最小生成树周围点的搜索，将点权最小的结点放入当前最小生成树之中，最终生成最小生成树。

## 2.3结构体的成员与方法



## 2.4测试函数

测试函数采用了用户友好的提示语句，以及更好的可视化界面，为了更好的用户体验，还进行了输入错误的错误提示，同时，因为同时有两种算法实现，本程序中也增加了另一种代码实现的生成最小生成树的代码在注释之中，可由测试者自行选择采用哪种方法。





# 3.实现

## 3.1Edge结构体

我们采用结构体来存储边的数据，包括边的起点、边的终点以及边的权值，为了便于排序，本项目还采用了重载运算符进行小于号的重载，便于排序。

## 3.2 unionFind(map<string, string> m, string x)函数

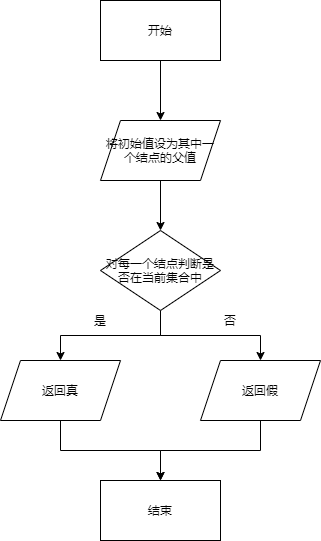
unionFind()函数为并查集的实现函数，通过并查集的查找返回最顶的父亲节点。



## 3.3 checkUnionFind(map<string, string>m)函数

通过checkUnionFind()函数对Kruskal函数生成的最小生成树进行判断，看它是否确实为最小生成树，Kruskal生成的最小生成树具有所有结点的父节点都是同一值的特点，所以本函数通过此特点进行判断。

### 3.3.1流程图



### 3.3.2代码实现

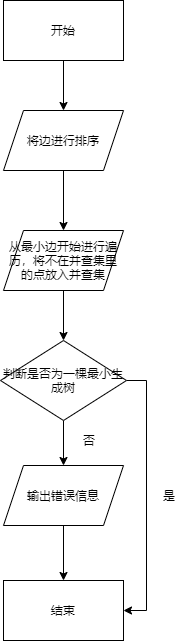


## 3.4 Kruskal(vector<Edge> edge, map<string, string> m, vector<string> vertex)函数

Kruskal首先将所有边进行排序，然后将最小的边挑出来进行并查集的构建，最终形成一棵最小生成树。

当不是所有的点都在当前并查集之中时，证明此最小生成树并没有包括所有的点，所以不是最小生成树，输出错误信息。

### 3.4.1流程图



### 3.4.2代码实现



## 3.5 minKey(map<string,int>key,vector<Edge>& ans,map<string,bool> is\_visited)函数

寻找当前图之中拥有最小点权值的结点。

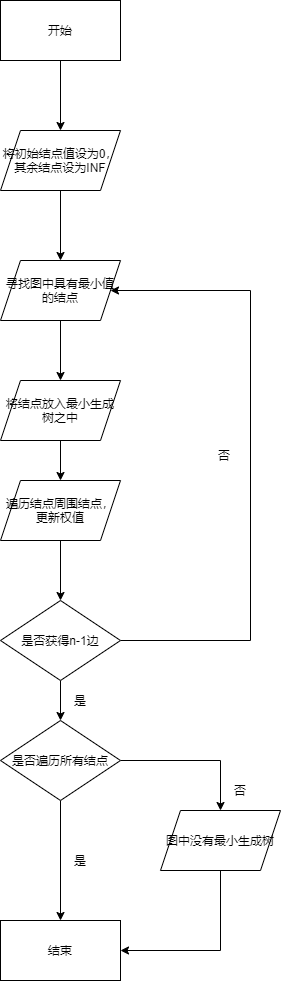


## 3.6 Prim(vector<Edge> edge, vector<string> vertices, string x)

先将所有结点值设为INF，以初始节点开始对周围结点权值进行更新，再从所有结点之中找出权值最小的结点进行遍历，直到所有结点都遍历过，证明已经生成最小生成树。

is\_visited数组中如果存在结点未被遍历，证明当前图中无最小生成树。

### 3.6.1流程图



### 3.6.2代码实现





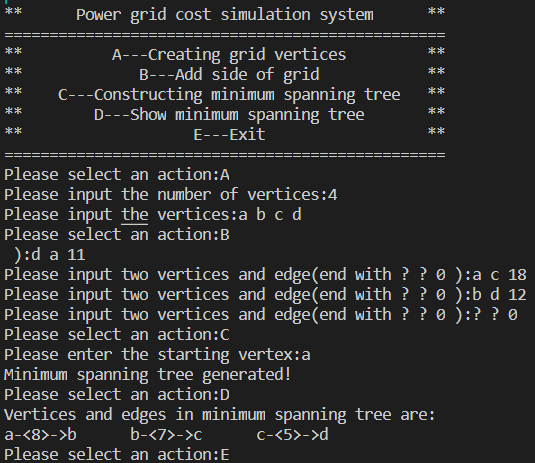
## 3.7输出函数

进行最小生成树的输出。

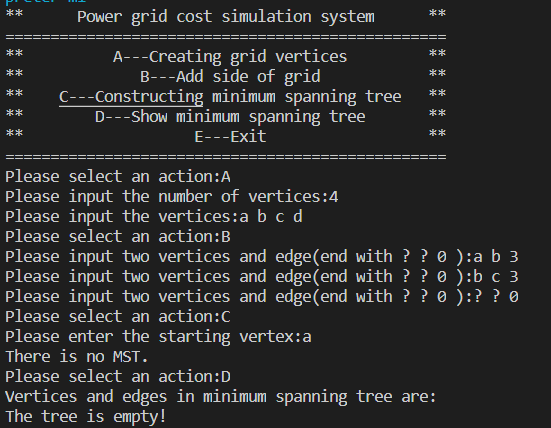
3.8边是否合法判断

# 4.输出测试

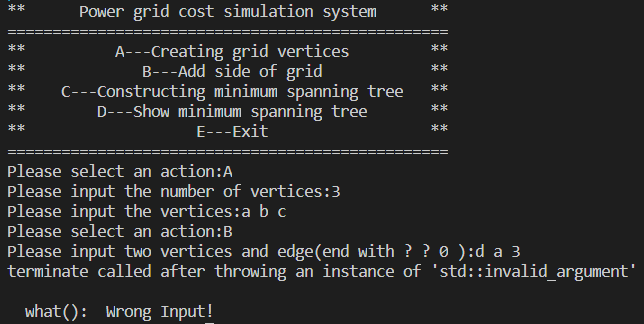
## 4.1样例输入输出



## 4.2不存在最小生成树输入输出



## 4.3输入不存在点的边



## 4.4错误输入

