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： 伊啸

学 号： 1951220

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](#_Toc60166854)

[1.1 项目简介 1](#_Toc60166855)

[1.2 功能分析 1](#_Toc60166856)

[2 设计 1](#_Toc60166857)

[2.1 数据结构设计 1](#_Toc60166858)

[2.2 类结构设计 1](#_Toc60166859)

[2.3 成员与操作设计 1](#_Toc60166860)

[2.4 系统设计 3](#_Toc60166861)

[3 Prim算法的实现 4](#_Toc60166862)

[3.1 Prim算法的功能流程图 4](#_Toc60166863)

[3.2 Prim算法的核心代码 6](#_Toc60166864)

[3.3 Prim算法截屏示例 6](#_Toc60166865)

[4 测试 7](#_Toc60166866)

[4.1 功能测试 7](#_Toc60166867)

[4.2 边界测试 8](#_Toc60166868)

[4.2.1 初始化无输入数据 8](#_Toc60166869)

[4.3 出错测试 8](#_Toc60166870)

[4.3.1 电网顶点错误 8](#_Toc60166871)

[4.3.2 操作码错误 9](#_Toc60166872)

[4.3.3 插入边的两个顶点不存在 9](#_Toc60166873)

[4.3.4 插入边的权值不为正整数 9](#_Toc60166874)

[4.3.5 顶点指向自身 10](#_Toc60166875)

[4.3.6 输入的边数过多 10](#_Toc60166876)

# 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。设计一个能够满足要求的造价方案。

## 1.2 功能分析

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。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统需要将各个小区直接连接起来，并和相应的电网线路耗费的经济代价联系起来。显然用图容易实现，小区即为顶点，电网线路为小区之间的边，相应的经济代价即为边的权值。

## 2.2 类结构设计

如上所述，首先肯定是图结构，其次是最小生成树及其节点，最后构建了一个最小堆辅助实现功能。

## 2.3 成员与操作设计

**链表结点类（LNode）**

**私有成员：**

int maxVertices;*//图中的最大顶点数量*

int numEdges;*//图中边的数量*

 int numVertices;*//图中的顶点数*

char\* VerticesList;*//顶点链表*

int\*\* Edge;*//邻接矩阵*

**公有操作：**

const int maxWeight = INT\_MAX;*//最大权值*

    Graphmtx(int sz=DefaultVertices);*//构造函数*

    ~Graphmtx() {*//析构函数*

        delete[]VerticesList;

        delete[]Edge;

    }

bool GraphEmpty()const {*//确定图是否为空*

*if* (numEdges == 0)

*return* true;

*else*

*return* false;

    }

bool GraphFull()const {*//确定图是否已满*

*if* (numVertices == maxVertices || numEdges == maxVertices \* (maxVertices - 1) / 2)

*return* true;

*else*

*return* false;

    }

int getVertexPos(char vertex) {*//给出顶点的位置*

*for* (int i = 0; i < numVertices; i++)

*if* (VerticesList[i] == vertex)

*return* i;

*return* -1;

    }

int NumberOfVertices() { *return* numVertices; }*//给出顶点数*

   int NumberOfEdgds() { *return* numEdges; }*//给出边的数量*

   int getWeight(int v1,int v2）{//得到边（v1，v2）权重

*return* v1 != -1 && v2 != -1 ? Edge[v1][v2] : 0;

    }

int getFirstNeighbor(int v）; //给出顶点v的邻接顶点

    int getNextNeighbor(int v, int w);

*//给出顶点v的邻接点w的下一个邻接点*

    bool InsertVertex(const char& vertex);*//插入顶点*

bool InsertEdge(int v1, int v2, int cost);

*//插入一条权重为cost，介于v1和v2之间的边*

    bool removeVertex(int v);*//删除顶点*

    bool removeEdge(int v1, int v2);*//删除边*

    friend class MinSpanTree;

    friend istream& operator>>(istream& input, Graphmtx& graph）;

//重载输入功能

## 2.4 系统设计

系统首先实现对屏幕的初始化，完成对图的创建和输入数据工作，然后根据用户所输入的操作码（option）执行图对应的成员函数。

# 3 Prim算法的实现

## 3.1 Prim算法的功能流程图



## 3.2 Prim算法的核心代码

int\* Vmst = new int[n];

for (int i = 0; i < n; i++)

Vmst[i] = 0;

Vmst[u] = 1;

do {

v = G.getFirstNeighbor(u);

while (v != -1) { //重复检测u的所有相邻顶点

if (Vmst[v] == 0) {

//如果v不在生成树中，则将（u，v）添加到堆中

ed.tail = u;

ed.head = v;

ed.key = G.getWeight(u, v);

//尾结点在生成树中，头节点不在

H.Insert(ed);

}

v = G.getNextNeighbor(u, v); //查找u的下一个相邻顶点v }

while (!H.IsEmpty() && count < n) {

H.RemoveMin(ed); //从堆中弹出边缘权值最小的ed

if (Vmst[ed.head] == 0) {

MST.Insert(ed); //将ed添加到最小生成树

u = ed.head;

Vmst[u] = 1;

count++;

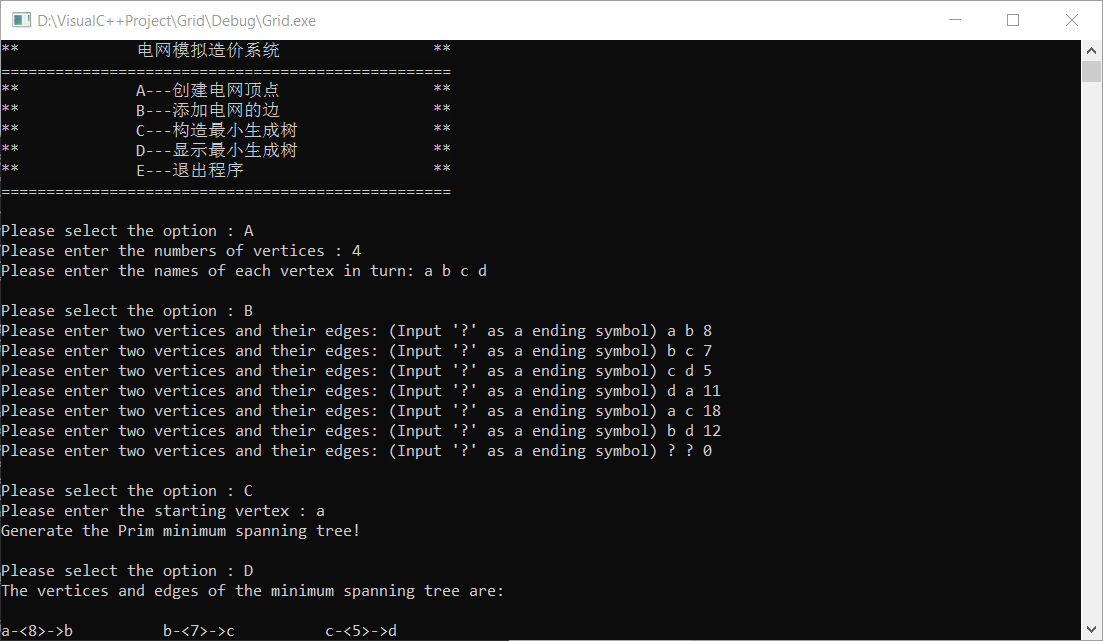
break;

}

}

} while (count < n);

## 3.3 Prim算法截屏示例



# 4 测试

## 4.1 功能测试

**测试用例**：A B 28

B C 16

C D 12

D E 22

E F 25

F A 10

B G 14

D G 18

E G 24

**预期结果**：

A->F->E->D->C->B->G

**实验结果**

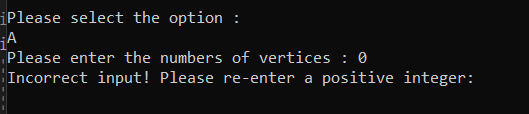
## 4.2 边界测试

### 4.2.1 初始化无输入数据

**测试用例：**初始无输入数据

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**



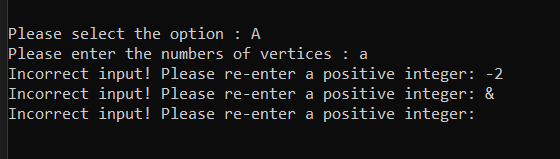
## 4.3 出错测试

### 4.3.1 电网顶点错误

**测试用例：**输入考生人数为负数，abc&%等其他字符

**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：**

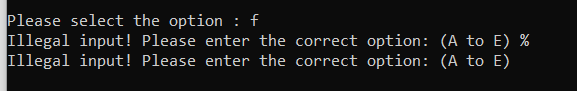


### 4.3.2 操作码错误

**测试用例：**输入操作码错误

**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：**

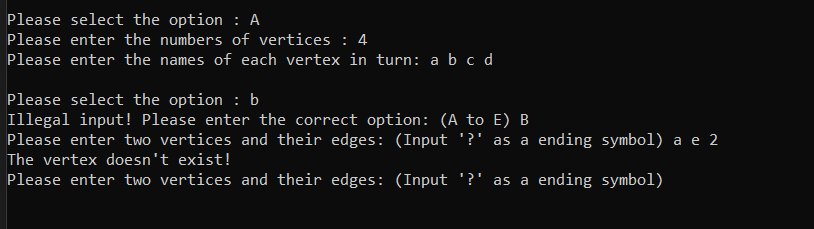


### 4.3.3 插入边的两个顶点不存在

**测试用例：**在不存在的两个顶点间插入边

**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：**

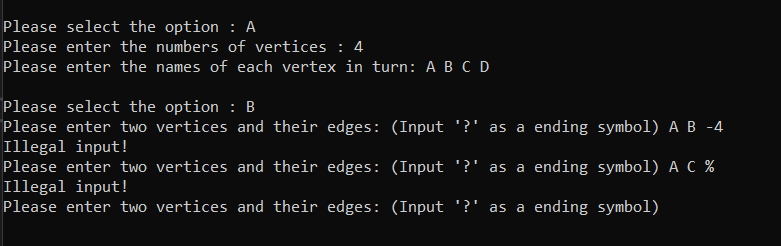


### 4.3.4 插入边的权值不为正整数

**测试用例：插入边的值为负数或其他字符**

**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：**

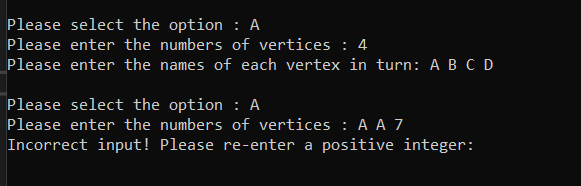


### 4.3.5 顶点指向自身

**测试用例：**顶点指向自身

**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：**

****

### 4.3.6 输入的边数过多

**测试用例：输入过多边数**

**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：**

