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： 祝新元

学 号： 1751629

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](#_Toc532926548)

[1.1 背景分析 1](#_Toc532926549)

[1.2 功能分析 1](#_Toc532926550)

[2 设计 2](#_Toc532926551)

[2.1 数据结构设计 2](#_Toc532926552)

[2.2 类结构设计 2](#_Toc532926553)

[2.3 成员与操作设计 2](#_Toc532926554)

[2.4 系统设计 3](#_Toc532926555)

[3 实现 4](#_Toc532926556)

[3.1 创建电网结点功能的实现 4](#_Toc532926557)

[3.1.1 创建电网结点功能核心代码 4](#_Toc532926558)

[3.1.2 创建电网结点功能截屏示例 4](#_Toc532926559)

[3.2 添加电网的边功能的实现 5](#_Toc532926560)

[3.2.1 添加电网的边功能核心代码 5](#_Toc532926561)

[3.2.2 添加电网的边功能截屏示例 6](#_Toc532926562)

[3.3 Prim算法的实现 6](#_Toc532926563)

[3.3.1 Prim算法核心代码 6](#_Toc532926564)

[3.3.2 Prim算法截图示例 8](#_Toc532926565)

[3.4 总体系统的实现 8](#_Toc532926566)

[3.4.2 总体系统核心代码 8](#_Toc532926567)

[3.4.3 总体系统截屏示例 11](#_Toc532926568)

[4 测试 11](#_Toc532926569)

[4.1 功能测试 11](#_Toc532926570)

[4.1.1 添加电网顶点功能测试 11](#_Toc532926571)

# 1 分析

## 1.1 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 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。

综上所述，电网建设造价模拟系统实际上是要在一个无向图中建造一棵最小生成树。

# 2 设计

## 2.1 数据结构设计

采用邻接表的表示方法来存储加权联通图，用了Prim算法来生成最小生成树。算法描述如下：

1).输入：一个加权连通图，其中顶点集合为V，边集合为E；

2).初始化：Vnew = {x}，其中x为集合V中的任一节点（起始点），Enew = {},为空；

3).重复下列操作，直到Vnew = V：

a.在集合E中选取权值最小的边<u, v>，其中u为集合Vnew中的元素，而v不在Vnew[集合](https://baike.baidu.com/item/%E9%9B%86%E5%90%88/2908132)当中，并且v∈V（如果存在有多条满足前述条件即具有相同权值的边，则可任意选取其中之一）；

b.将v加入集合Vnew中，将<u, v>边加入集合Enew中；

4).输出：使用集合Vnew和Enew来描述所得到的[最小生成树](https://baike.baidu.com/item/%E6%9C%80%E5%B0%8F%E7%94%9F%E6%88%90%E6%A0%91)。

## 2.2 类结构设计

加权有向图的结构不在赘述，Prim算法需要包括两个抽象数据类型（ADT）——最小生成树边结点类（MSTEdgeNode）与最小生成树类（MinSpanTree），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统采用struct描述最小生成树边结点类（MSTEdgeNode），这样使得最小生成树类（MinSpanTree）可以访问链表结点。

## 2.3 成员与操作设计

**最小生成树边结点类（MSTEdgeNode）**

**公有成员：**

int tail,head;//两顶点

E key; //边的权值

**公有操作：**

MSTEdgeNode():tail(-1),head(-1),key(0)

{}//具有默认参数的构造函数

**最小生成树类（MinSpanTree）**

**私有成员：**

MSTEdgeNode<T,E> \*edgevalue; //边数组

int maxSize,n; //最大大小和当前大小

**公有操作：**

MinSpanTree<T,E>::MinSpanTree(int sz=DefaultSize-1):maxSize(sz),n(0)

// MinSpanTree的构造函数，构造一个大小为sz的数组

void MinSpanTree<T,E>::Insert(MSTEdgeNode<T,E> &item)

//插入边

bool MinSpanTree<T,E>::isEmpty()

//是否是空

void MinSpanTree<T,E>::show();

//输出结果

int MinSpanTree<T,E>::getTail(int i)

//得到第i条边的起点

int MinSpanTree<T,E>::getKey(int i)

//得到第i条边的值

int MinSpanTree<T,E>::getHead(int i)

//得到第i条边的终点

## 2.4 系统设计

系统首先实现对屏幕的初始化，完成对无向图ElecGraph的创建和最小生成树MST的创建，然后根据用户所输入的操作码（operatorCode）执行对应的成员函数。

# 3 实现

## 3.1 创建电网结点功能的实现

### 3.1.1 创建电网结点功能核心代码

cout << "请输入顶点的个数：";

cin >> numVertices;

if (numVertices>50 || numVertices<0)

{

cout << "请输入50以内的正整数：";

cin >> numVertices;

}

cout << "请依次输入各顶点的名称：";

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

{

cin >> nameOfVertices[i];

ElecGraph.insertVertex(nameOfVertices[i]);

}

### 3.1.2 创建电网结点功能截屏示例



## 3.2 添加电网的边功能的实现

### 3.2.1 添加电网的边功能核心代码

cout << "请输入两个顶点及边：";

string u, v;

int value;

cin >> u >> v >> value;

//添加边

int v1 = ElecGraph.getVertexPos(u);

int v2 = ElecGraph.getVertexPos(v);

//如果出现其他情况则输入结束

if (v1 == -1 || v2 == -1)

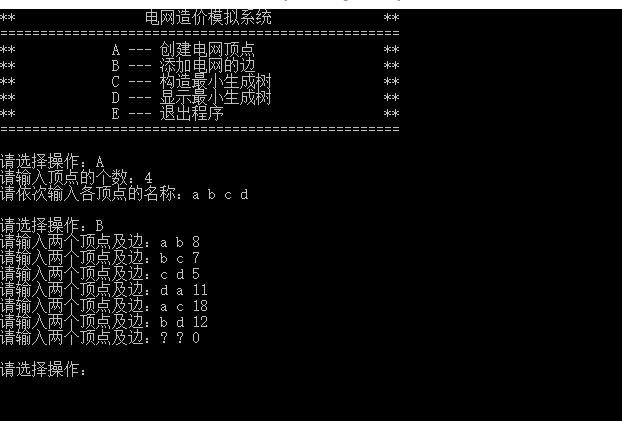
{

break;

}

ElecGraph.insertEdge(v1, v2, value);

### 3.2.2 添加电网的边功能截屏示例



## 3.3 Prim算法的实现

### 3.3.1 Prim算法核心代码

MSTEdgeNode<T,E> ed;

int i,v,count;

int n=G.getNumVertices(); //顶点数

int m=G.getNumEdges(); //边数

int u=G.getVertexPos(u0); //求起始顶点号

MinHeap< MSTEdgeNode<T,E> > H(m); //最小堆，关键码类型为E

bool \*Vmst=new bool[n];

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

{

Vmst[i]=false;

}

Vmst[u]=0;

count=1; //u加入最小生成树

do

{

v=G.getFirstNeighbor(u);

while(v!=-1) //重复检测u的所有邻接节点

{

if(Vmst[v]==false) //若v不在生成树，把（u，v）加入堆

{

ed.tail=u; //tail在树内，head不在

ed.head=v;

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

H.Insert(ed);

}

v=G.getNextNeighbor(u,v); //下一个邻接节点

}

while(H.IsEmpty()==false && count<n)

{

H.RemoveMin(ed); //从堆中退出具有最下权值的边ed

if(Vmst[ed.head]==false)

{

MST.Insert(ed);

u=ed.head;

Vmst[u]=true;

count++;

break;

}

}

}while(count<n);

if(MST.isEmpty())

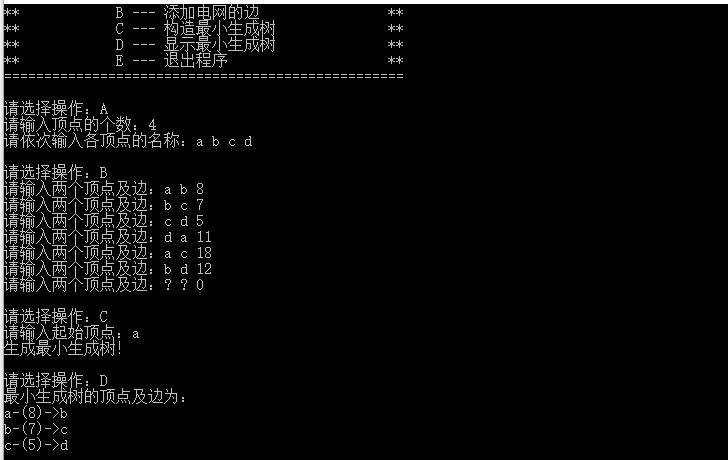
{

return false;

}

return true;

### 3.3.2 Prim算法截图示例



## 3.4 总体系统的实现

### 3.4.2 总体系统核心代码

while (1)

{

cout << endl;

cout << "请选择操作：";

cin >> order;

switch (order)

{

case 'A':

{

cout << "请输入顶点的个数：";

cin >> numVertices;

if (numVertices>50 || numVertices<0)

{

cout << "请输入50以内的正整数：";

cin >> numVertices;

}

cout << "请依次输入各顶点的名称：";

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

{

cin >> nameOfVertices[i];

ElecGraph.insertVertex(nameOfVertices[i]);

}

continue;

}

case 'B':

{

while (1)

{

cout << "请输入两个顶点及边：";

string u, v;

int value;

cin >> u >> v >> value;

//添加边

int v1 = ElecGraph.getVertexPos(u);

int v2 = ElecGraph.getVertexPos(v);

//如果出现其他情况则输入结束

if (v1 == -1 || v2 == -1)

{

break;

}

ElecGraph.insertEdge(v1, v2, value);

}

continue;

}

case 'C':

{

cout << "请输入起始顶点：";

string st;

cin >> st;

if (Prim(ElecGraph, st, MST))

{

cout << "生成最小生成树！" << endl;

}

continue;

}

case 'D':

{

cout << "最小生成树的顶点及边为：" << endl;

for (int i = 0; i < numVertices - 1; i++)

{

cout << ElecGraph.getValue(MST.getTail(i)) << "-("

<< MST.getKey(i) << ")->"

<< ElecGraph.getValue(MST.getHead(i)) << endl;

}

continue;

}

case 'E':

{

return 0;

}

default:

{

cout << "不合法的操作！" << endl;

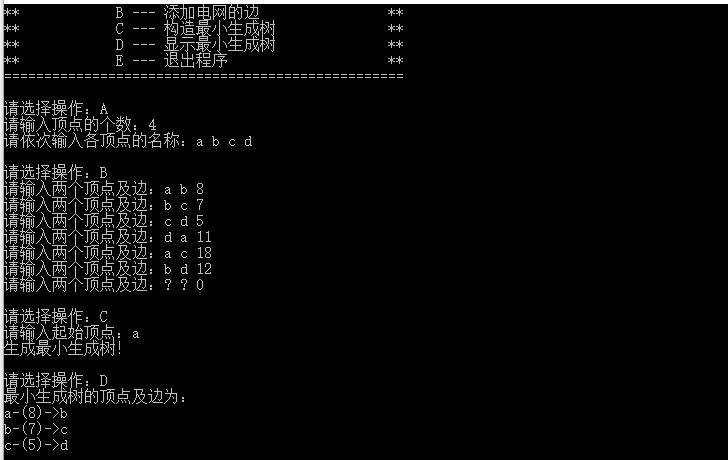
continue;

}

}

}

### 3.4.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 添加电网顶点功能测试

**测试用例**：60

**预期结果**：



**实验结果**

