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： 何慧琳

学 号： 2152343

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 分析

## 1.1 背景分析

电网建设造价模拟系统是建设电网时不可或缺的一个系统。城市有多个小区，每个小区都要通上电，这就要求每个小区之间的电网都能够互相联通，而每两个小区之间距离等因素的不同，小区间电网工程造价也不相同，为了节约成本，使总工程造价最小，需要计算出满足要求的造价方案。因此开发一个电网建设造价模拟系统对城市的智能化管理、资源的高效利用等都具有十分重要的意义。

## 1.2 功能分析

作为一个简易的电网建设造价模拟系统，首先应该有的功能就是输入各小区的名称和各小区之间的电网造价。其次，电网建设造价模拟系统应该根据输入的数据输出总代价最小的通路。最后，电网建设造价模拟系统还应该确保系统可以正常关闭。

综上所述，一个电网建设造价模拟系统至少应该具有输入、建立一张带权图、构造最小生成树、显示最小生成树的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要将输入的小区作为一个个结点，将各小区之间电网造价作为结点之间边的权值建立一张图，所以定义了一个图类（Graph）来保存初始完整版电网数据。该系统最终是要遍历这张图找到最小生成树，因此还定义了一个最小生成树类（MinSpanTree）以保存找到的最小生成树的边并进行显示，为了最小生成树中的边还定义了一个生成树边结点类（MSTEdgeNode）。

## 2.2 类结构设计

**图类（Graph）**

一张图的要素有构成图的结点、结点之间的边的权值，所以需要一个保存结点的结构，为了避免空间的浪费因为每个城市大小各异小区数量差距较大不确定开设多大节点数组合适，所以采用链表结构存储结点。所以在Graph类中为了表示一个顶点信息定义了顶点结构体，为了保存所有结点定义了定点链表类（不是单独开辟一个链表类考虑到若在类外单独开还需使用参数模板，调用较麻烦，该系统中只在此处用到了链表结构，所以干脆在Graph类中定义一个专门保存结点的链表，方便使用）。至于保存各节点间边的权值，考虑到边可能较稠密故采用prim法构造最小生成树，每次要查找、更新最小生成树集合内顶点到外顶点之间的最短距离，故采用邻接矩阵来保存各边的权值，因此Graph类中定义了成员邻接矩阵。图具有插入顶点、插入边、找最小生成树的功能，有时还需要判断一个点是否在图中，故有如下成员和操作设计。

**公有成员及操作：**

Graph(int sz = MaxVertices);

void InsertVertex(NameType& vertex);//插入顶点

void InsertEdge(NameType v1, NameType v2, DistType weight);//插入以v1为起点，v2为终点，weight为权值的边

void Prim(MinSpanTree& T,NameType start\_name);

int find(NameType name) { return VL.get\_seq(name)>=0; }//图中是否存在name点

**私有成员：**

struct Vertex {//顶点结构体

NameType name;//顶点名

int seq;//顶点在顶点表中的序号

Vertex\* next;//链接指针，指向下一个结点

Vertex(NameType N, int s, Vertex\* n = NULL) :name(N), seq(s), next(n) {};

};

class VertexList {//顶点链表

friend class MinSpanTree;

friend class Graph;

Vertex\* first, \* current;//结点指针

public:

VertexList(Vertex\* f = NULL, Vertex\* c = NULL) :first(f), current(c) {};//构造函数

int get\_seq(NameType name);//返回名为name结点的序号

NameType operator [](int dex);//重载[]运算符取第i个元素名字

};

VertexList VL;//节点链表

int VertexNum = 0;//结点个数

DistType Edge[MaxVertices][MaxVertices];//邻接矩阵

int CurrentEdges=0;//当前边数

**最小生成树边结点类（MSTEdgeNode）**

描述一条边应当包含边两段的结点名，边的权值，故有如下成员设计.

**私有成员：**

int tail, head;//生成树各边的两顶点

float cost;//生成树各边的代价

MSTEdgeNode() :tail(0),head(0),cost(0) {};

MSTEdgeNode(MSTEdgeNode \*mstedge):tail(mstedge->tail),head(mstedge->head),cost(mstedge->cost) {};

**最小生成树类（MinSpanTree）**

最小生成树由其中的边组成，于是定义了一个边数组，此外它还应具备插入边、打印最小生成树中的边的共能，故有如下设计。

**私有成员：**

MSTEdgeNode\* edgevalue;//边值数组

int MaxSize, CurrentSize;//最大边数，当前边数

**公有操作：**

MinSpanTree(int sz= MaxEdges) :MaxSize(sz), CurrentSize(0) { edgevalue = new MSTEdgeNode[MaxSize]; }

void Insert(MSTEdgeNode &e);//插入一条边

void print(Graph<char\*,float> graph);//打印最小生成树

## 2.4 系统设计

系统首先调用main()函数显示菜单界面，提示用户按顺序选择操作吗，并根据输入的操作码完成顶点名称、边、起始顶点的输入工作，调用Graph类的成员函数完成图中顶点和边的插入，根据用户选择的起始顶点构造最小生成树并打印。

# 3 实现

## 3.1 图的初始化功能的实现

### 3.1.1图的初始化功能核心代码

void Graph<NameType, DistType>::InsertVertex(NameType& vertex)

{//插入顶点

if (!VertexNum) {//结点链表为空

VL.first=VL.current = new Vertex(vertex, VertexNum);

}

else {

VL.current->next = new Vertex(vertex, VertexNum);//在结点链表末尾加入新结点

VL.current = VL.current->next;

}

VertexNum++;//结点个数+1

}

void Graph<NameType, DistType>::InsertEdge(NameType v1, NameType v2, DistType weight)

{//插入以v1为起点，v2为终点，weight为权值的边

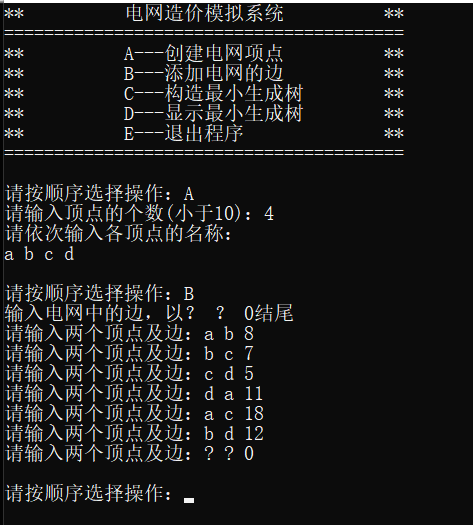
int i = VL.get\_seq(v1);

int j = VL.get\_seq(v2);

Edge[i][j] = weight;//在邻接表中更改权值

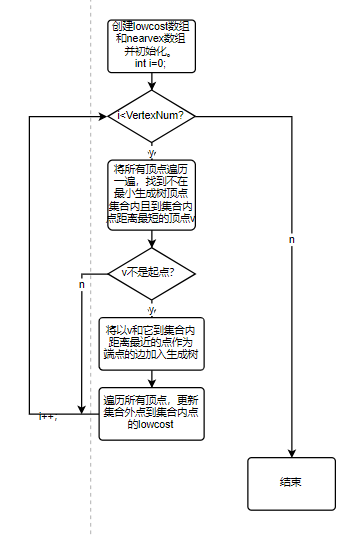
}

### 3.1.2 图的初始化功能截屏示例



## 3.2 构造最小生成树功能的实现

### 3.2.1 构造最小生成树功能流程图



### 3.2.2 构造最小生成树功能核心代码

//以start\_name为起点用prim算法找到最小生成树放入T

DistType \*lowcost = new DistType[VertexNum];

DistType\* nearvex = new DistType[VertexNum];

int start = VL.get\_seq(start\_name);

for (int i = 0; i < VertexNum; i++) {

lowcost[i] = Edge[start][i];//顶点到各边的代价

nearvex[i] = start;//最短带权路径

}

nearvex[start] = -1;//顶点0加到生成树顶点集合

MSTEdgeNode e;//最小生成树结点辅助单元

for (int i = 0; i < VertexNum; i++) {//循环n-1次，加入n-1条边

if (i == start)

continue;

DistType min = INT\_MAX;//最短边长

int v = start;

for (int j = 0; j < VertexNum; j++) {

//遍历更新，找到不在最小生成树顶点集合的到集合内点距离最短的顶点v及其距离min

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

v = j;

min = lowcost[j];

}

}

if (v != start) {//v==start表示再也找不到要求顶点了

e.head = nearvex[v];

e.tail = v;

e.cost = lowcost[v];

T.Insert(e);//选中的边加入生成树

nearvex[v] = -1;//作该边已加入生成树标记

for (int j = 0; j < VertexNum; j++) {

//每新加入一个结点，更新一遍集合外结点到集合内结点的lowcost

if (nearvex[j] != -1 && Edge[v][j] < lowcost[j]) {

//j不在生成树中且原lowcost比j到新加入的点距离大，需要修改

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

nearvex[j] = v;

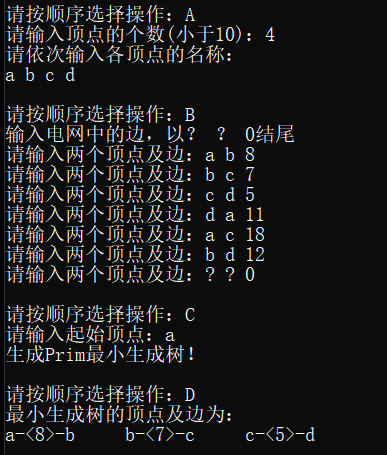
}

}

### }

### }

### 3.2.3 构造最小生成树功能截屏示例



# 4 测试

## 4.1 功能测试

**测试用例**：

顶点数：4

顶点名：a b c d

边：

a b 8

b c 7

c d 5

d a 11

a c 18

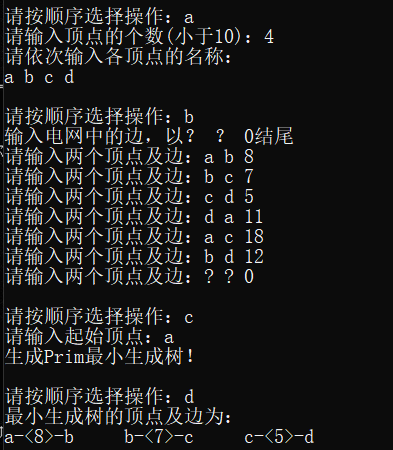
b d 12

? ? 0

**预期结果**：

a-<8>-b b-<7>-c c-<5>-d

**实验结果**



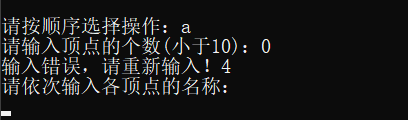
## 4.2 边界测试

### 4.2.1 初始化无输入数据

**测试用例：**输入顶点个数为0

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

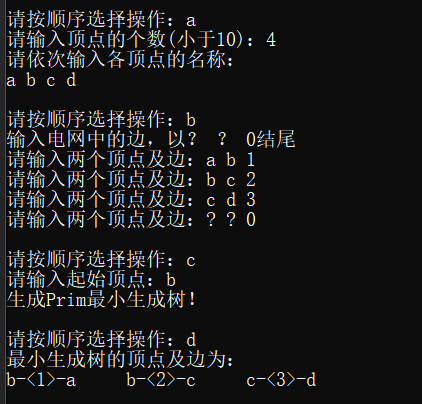


### 4.2.2 输入最少可能边数（n-1）

**测试用例：**顶点数4个，输入3条边

**预期结果：**程序正常运行。

**实验结果：**

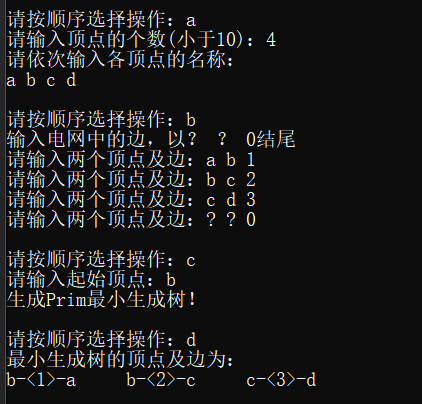


### 4.2.3 输入起点不是顶点序列中的第一个顶点

**测试用例：**顶点序列a b c d,起始顶点b

**预期结果：**程序结果正确。

**实验结果：**



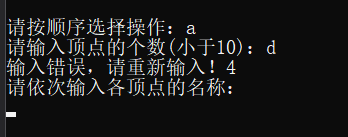
## 4.3 出错测试

### 4.3.1 输入顶点数错误

**测试用例：**输入顶点个数为非数

**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：**

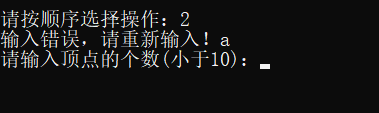
****

### 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 输入边数太少不可能构成通路

**测试用例：**顶点数4个，输入2条边

**预期结果：**程序给出提示信息，程序正常运行。

**实验结果：**

