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： 曹晓慈

学 号： 2052844

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

## 分析

### 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。

### 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。

## 设计

### 数据结构设计

由题意可知该题目是一个多结点同时根据权重生成总的权重和最小的图，考虑到该题目的顶点，边，以及边的权重，故该题目采用图的数据结构。

### 类结构设计

该题目选用图（MST）类来进行实现。

### 成员与操作设计

const int MaxVertices = 10;

const int MaxEdges = 50;

class MST

{

public:

MST();

void creatVertice(); //A选项创建电网顶点

void addEdge(); //B选项添加电路的网

void creatTree(); //C选项构造最小生成树

void DisplayTree(int begin); //D选项显示最小生成树

int getFirstNeighbor(int v); //给出顶点位置为v的第一个邻接顶点的位置，如果找不到则返回-1

int getNextNeighbor(int v, int w); //给出顶点v的某邻接顶点w的下一个邻接点，找不到则返回-1

int getStart\_num() { return start\_num; };

private:

int numVertices; //当前顶点数

int numEdges; //当前边数

int count = 0; //初始时最小生成树上的节点个数

int Edge[MaxVertices][MaxVertices]; //记录边的权重

int store[MaxVertices][MaxVertices] = { 0 };

string start; //开始结点

int start\_num; //开始结点的序号

vector <string> Vertices\_Name; //储存结点序号对应的名字

};

## 实现

### 创建电网顶点的实现

该操作较简单，获得顶点个数以及将各顶点的名称存储到Vertices\_Name中，同时各个顶点有相应的顺序序号。

核心代码：

cout << "请输入顶点的个数:";

cin >> numVertices;

cout << "请依次输入各顶点的名称:" << endl;

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

{

cin >> name;

Vertices\_Name.push\_back(name);

}

### 添加电网的边

该操作调用addEdge（）函数，输入两个顶点和对应边的权重之后对Vertices\_Name进行遍历找到两个顶点，并将两个顶点连线的权重记录到Edge的二维数组中去。这里对于边的输入要以？ ？ 0作为结尾，表示输入完成。

核心代码：

cin >> name\_a;

cin >> name\_b;

cin >> weight;

while (name\_a != "?" && name\_b != "?" && weight != 0)

{

int i = 0, j = 0;

while (name\_a != Vertices\_Name[i])

i++;

while (name\_b != Vertices\_Name[j])

j++;

Edge[i][j] = weight;

Edge[j][i] = weight;

cout << "请输入两个顶点及边:";

cin >> name\_a;

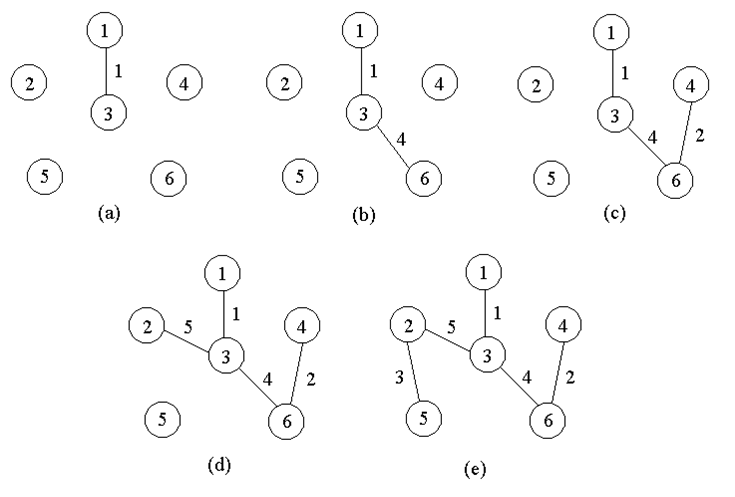
cin >> name\_b;

cin >> weight;

}

### 构造最小生成树的实现

这里用到了Prim算法，设置一个visited数组来记录顶点是否加入树中，从起始顶点开始，利用双层循环，寻找已加入树中的顶点与未加入树中的顶点的连线的权重进行比较，两层循环完成后最小值对应的边的终点加入到树中，再继续进行循环直到最小生成树中的顶点数目等于图中的顶点数目为止。将最小生成树以一个矩阵储存，行指向列，数值代表权重，为零则表示没有边。



核心代码：

while (count != numVertices)

{

mark = 0;

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

{

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

{

if (visited[i] == 1 && visited[j] == 0 && i != j && Edge[i][j] != 0)

{

if (mark == 0)

{

min = Edge[j][i];

mark = 1;

newVertice\_x = j;

newVertice\_y = i;

}

else

{

tmp = Edge[j][i];

if (tmp < min)

{

min = tmp;

newVertice\_x = j;

newVertice\_y = i;

}

}

}

}

}

visited[newVertice\_x] = 1;

store[newVertice\_y][newVertice\_x] = Edge[newVertice\_y][newVertice\_x];

count++;

}

### 最小生成树显示的实现

这里调用DisplayTree(int begin)函数，考虑到是最小生成树，所以我在这里用到了递归的结构，这里还要用到了getFirstNeighbor(int v) 和getNextNeighbor(int v, int w)，具体思路是从起始顶点出发，首先获得他的第一个邻接点，并输出起点指向终点，这里用递归继续对第一个邻接点进行getFirstNeighbor(int v)的操作，先将这条路线的所有边输出完之后，再调用getNextNeighbor(int v, int w)继续采用上面的递归，考虑到会有很多分支所以，采用while循环直到无分支结束循环。

核心代码：

void MST::DisplayTree(int begin)

{

int tmp = 0;

if (getFirstNeighbor(begin)!=-1)

{

cout << Vertices\_Name[begin] << "-(" << store[begin][getFirstNeighbor(begin)] << ")->" << Vertices\_Name[getFirstNeighbor(begin)] << " "; //输出第一个邻接点

DisplayTree(getFirstNeighbor(begin)); //沿第一个邻接点进行递归输出

tmp = getFirstNeighbor(begin);

while (getNextNeighbor(begin, tmp) != -1) //若有分支

{

cout << Vertices\_Name[begin] << "-(" << store[begin][getNextNeighbor(begin, tmp)] << ")->" << Vertices\_Name[getNextNeighbor(begin, tmp)] << " ";

DisplayTree(getNextNeighbor(begin,tmp));

tmp = tmp + getNextNeighbor(begin, tmp); //多分支时跳过前面已经调用过的分支避免陷入死循环

}

}

### 整体思路和系统实现

采用Prim算法，构造MST类，本题中心思想是逐个将顶点连通来构造最小生成树，从连通网络中的某一顶点出发，将起点加入生成树的集合中，选择与它关联的权最小的边，将其终点加入到生成树的集合中，之后每一步从一个顶点在生成树的集合中，和另一个不在生成树的集合中的点集中找到权值最小的边加入生成树的边集中，直到所有顶点加入生成树集合中。

主要系统的核心代码：

char order;

MST Electricity\_Grid;

output();

cout << "请选择操作:" ;

cin>>order;

while (order != 'E')

{

switch (order)

{

case 'A':

Electricity\_Grid.creatVertice(); //A功能创建节点

break;

case 'B':

Electricity\_Grid.addEdge(); //B功能创建边

break;

case 'C':

Electricity\_Grid.creatTree(); //C功能创建最小生成树

break;

case 'D':

Electricity\_Grid.DisplayTree(Electricity\_Grid.getStart\_num()); //D功能显示最小生成树

cout << endl;

break;

default:

cout << "请输入正确的命令:";

break;

}

cout << "请选择操作" << endl;

cin >> order;

}

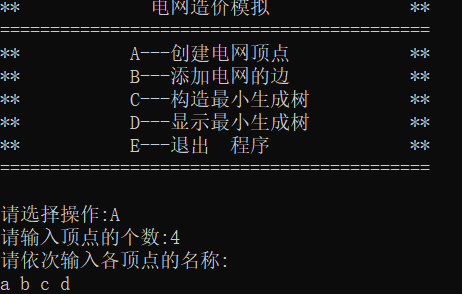
## }四．测试

### 1.创建顶点测试

测试用例：A 4 a b c d

预期结果：程序正常运行

实验结果：

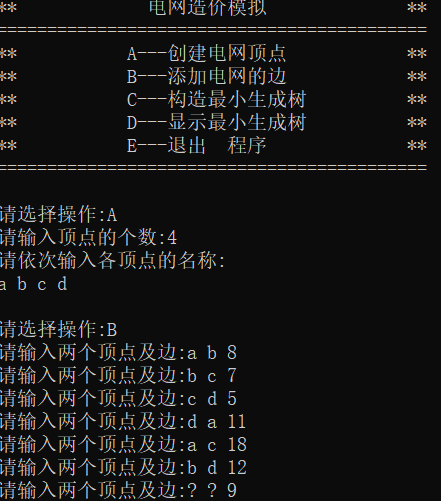


### 2.添加电网的边测试

测试用例：如截图

预期结果：程序正常运行

实验结果：

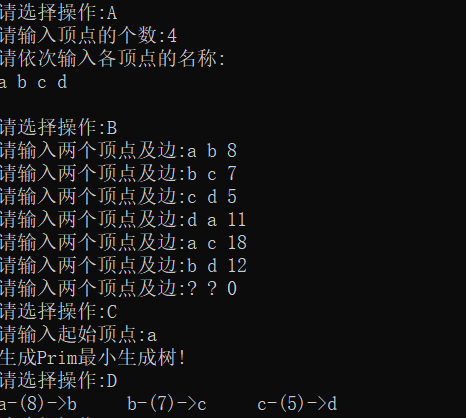


### 最小生成树的生成和显示测试

测试用例：如截图

预期结果：程序正常运行且给出正确答案

实验结果：



测试用例：如截图

预期结果：程序正常运行且给出正确答案

实验结果：

