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： 郑柯凡

学 号： 1950072

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 2](#_Toc22363)

[1.1背景分析 2](#_Toc21744)

[1.2功能分析 2](#_Toc3787)

[2 设计 2](#_Toc30043)

[2.1思路设计 3](#_Toc16759)

[2.2数据结构的选择 3](#_Toc26926)

[2.3类的设计 3](#_Toc27543)

[3 功能实现 4](#_Toc24229)

[3.1 电网系统运行模块功能的实现 4](#_Toc8656)

[3.1.1电网系统运行模块功能流程图 4](#_Toc8640)

[3.1.2电网系统运行模块功能核心代码 5](#_Toc10826)

[3.1.3电网系统运行模块功能截屏示例 5](#_Toc2441)

[3.2 添加顶点功能的实现 5](#_Toc2919)

[3.2.1添加顶点功能流程图 5](#_Toc18544)

[3.2.2添加顶点功能核心代码 5](#_Toc9499)

[3.2.3添加顶点功能截屏示例 6](#_Toc181)

[3.3 添加边功能的实现 6](#_Toc21059)

[3.3.1添加边功能流程图 6](#_Toc17656)

[3.3.2添加边功能核心代码  7](#_Toc14695)

[3.3.3添加边功能截屏示例 7](#_Toc11876)

[3.4 构造最小生成树功能的实现 8](#_Toc28868)

[3.4.1构造最小生成树功能流程图 8](#_Toc15629)

[3.4.2构造最小生成树功能核心代码 8](#_Toc30627)

[3.4.3构造最小生成树功能截屏示例 9](#_Toc7298)

[4 测试 10](#_Toc22325)

[4.1功能测试 10](#_Toc30108)

[4.1.1添加顶点功能测试 10](#_Toc3988)

[4.1.2添加边功能测试 11](#_Toc16)

[4.1.3构造最小生成树功能测试 11](#_Toc22944)

**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。

**2 设计**

**2.1思路设计**

每个小区之间的电网是非线性的结构，可以用图来进行储存，用顶点表示每个城市，用边表示某两个小区之间的电网能互相接通，边上的权值表示电网相应的造价，为了构造能连接所有小区的成本最低的电网系统，就是要找出这个网络的一棵最小生成树。

**2.2数据结构的选择**

本题采用Prim算法来构造最小生成树，即选点法。为了方便算法中相应的操作，采用用邻接矩阵存储边信息的图存储结构。

**2.3类的设计**

最小生成树中的边结点，存储最小生成树中一条边的信息，因此其数据成员包括边的起点，边的终点以及边的权值。

1. //边结点，存储最小生成树中的边信息
2. **class** EdgeNode {
3. **friend** **class** Graph;
4. **private**:
5. string head;
6. string tail;
7. **int** cost;
8. **public**:
9. EdgeNode() { head = tail = "0"; cost = 0; };
10. ~EdgeNode() {};
11. };

图类，存储每个小区间的电网网络。其私有成员包括用动态数组实现的顶点表和邻接矩阵，存储最小生成树的顺序表（使用了MySTL.h中的MyVector类实现）以及顶点的数量。

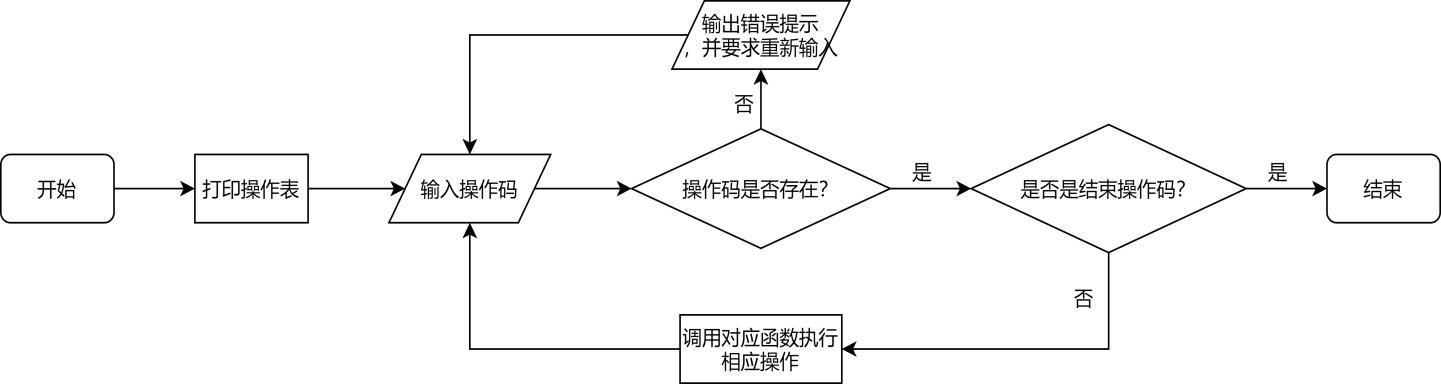
其公有成员包括设置顶点数量，通过顶点名字寻找顶点编号，获取两个顶点之间的边权值，创建顶点，添加边，构造最小生成树等功能函数。

1. **class** Graph {
2. **friend** **class** PowerSystem;
3. **private**:
4. //动态数组，用于存放顶点
5. string\* Vertices;
6. //邻接矩阵存放边
7. **int** \*\*Edge;
8. //顺序表存放最小生成树
9. MyVector<EdgeNode> MinSpanT;
10. //图顶点的数量
11. **int** Verticesnum;
12. **public**:
13. Graph()
14. {
15. Vertices = **new** string[MaxVertices];
16. Edge = **new** **int**\* [MaxVertices];
17. **for** (**int** i = 0; i < MaxVertices; i++)
18. {
19. Edge[i] = **new** **int**[MaxVertices];
20. **for** (**int** j = 0; j < MaxVertices; j++)
21. {
22. Edge[i][j] = Maxn;
23. }
24. }
25. Verticesnum = MaxVertices;
26. };
27. ~Graph() {};
28. //设置顶点数量
29. **void** SetVerticesnum(**int** num){ Verticesnum = num; }
30. //通过名字找顶点编号
31. **int** FindV(string x);
32. //获得边权值
33. **int** GetEdge(string start, string end);
34. //电网系统运行
35. **void** Run();
36. //创建顶点
37. **void** CreateVertices();
38. //添加边
39. **void** AddEdges();
40. //构造最小生成树
41. **void** CreateMinSpanT();
42. //打印最小生成树
43. **void** PrintMinSpanT();
44. };

**3 功能实现**

**3.1 电网系统运行模块功能的实现**

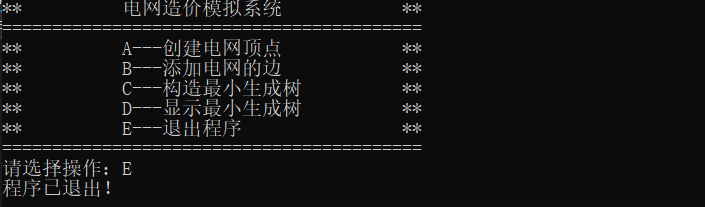
**3.1.1电网系统运行模块功能流程图**



**3.1.2电网系统运行模块功能核心代码**

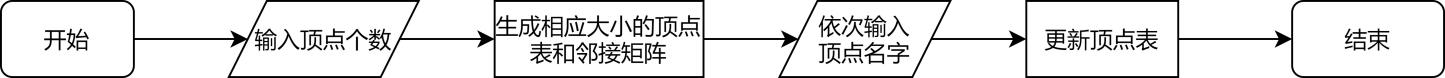
1. //电网系统运行
2. **void** Graph::Run()
3. {
4. **char** option = 'A';
5. cout << "\*\*          电网造价模拟系统            \*\*" << endl
6. << "==========================================" << endl
7. << "\*\*          A---创建电网顶点            \*\*" << endl
8. << "\*\*          B---添加电网的边            \*\*" << endl
9. << "\*\*          C---构造最小生成树          \*\*" << endl
10. << "\*\*          D---显示最小生成树          \*\*" << endl
11. << "\*\*          E---退出程序                \*\*" << endl
12. << "==========================================" << endl;
13. **while** (option != 'E')
14. {
15. cout << "请选择操作：";
16. cin >> option;
17. **switch** (option)
18. {
19. **case**'A':CreateVertices(); **break**;
20. **case**'B':AddEdges(); **break**;
21. **case**'C':CreateMinSpanT(); **break**;
22. **case**'D':PrintMinSpanT(); **break**;
23. **case**'E':**break**;
24. **default**:option = 'A'; cout << "未知操作码，请重新选择！" << endl;
25. }
26. }
27. cout << "程序已退出！" << endl;
28. }

**3.1.3电网系统运行模块功能截屏示例**



**3.2 添加顶点功能**的实现

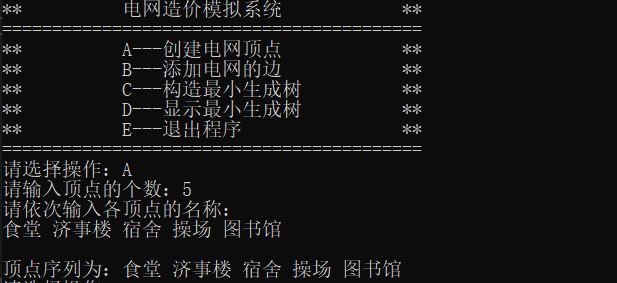
**3.2.**1添加顶点功能流程图



**3.2.2添加顶点功能核心代码**

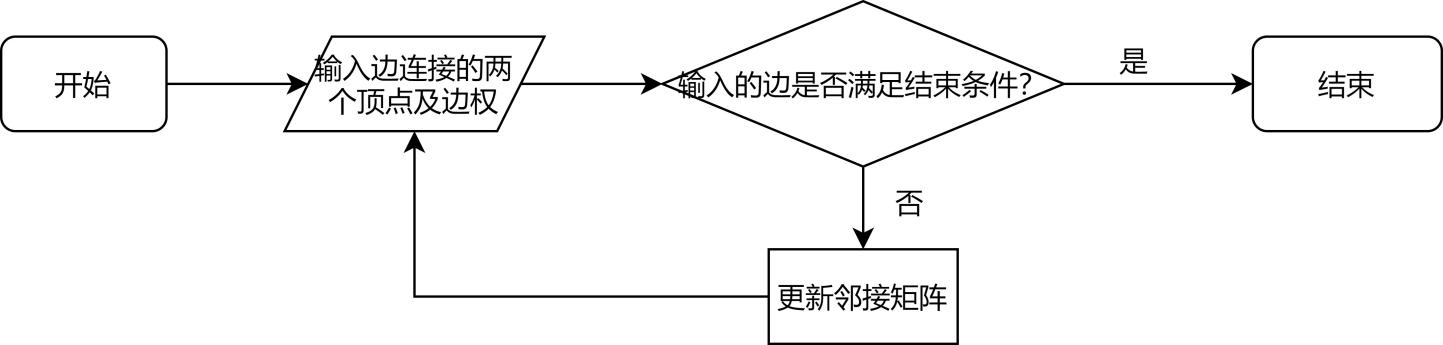
1. //创建顶点
2. **void** Graph::CreateVertices()
3. {
4. **int** verticesnum; string verticesname;
5. cout << "请输入顶点的个数：";
6. cin >> verticesnum;
7. Vertices = **new** string[verticesnum];
8. Edge = **new** **int**\*[verticesnum];
9. **for** (**int** i = 0; i < verticesnum; i++)
10. {
11. Edge[i] = **new** **int**[verticesnum];
12. **for** (**int** j = 0; j < verticesnum; j++)
13. {
14. Edge[i][j] = Maxn;
15. }
16. }
17. SetVerticesnum(verticesnum);
18. cout << "请依次输入各顶点的名称：" << endl;
19. **for** (**int** i = 0; i < verticesnum; i++)
20. {
21. cin >> verticesname;
22. Vertices[i] = verticesname;
23. }
24. cout << endl;
25. }

**3.2.3添加顶点功能截屏示例**



**3.3 添加边功能的实现**

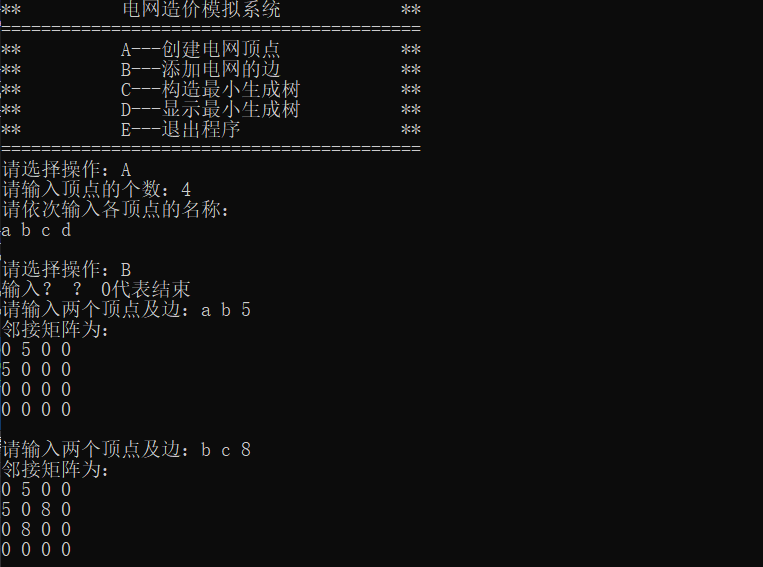
**3.3.1添加边功能流程图**

****

**3.3.2添加边功能核心代码**

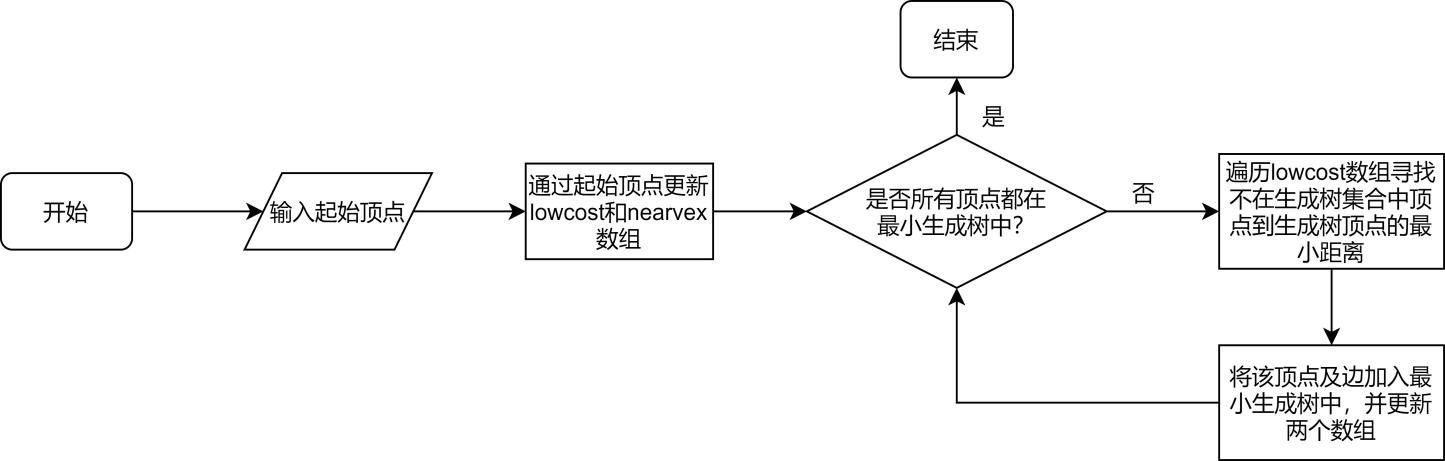
1. //添加边
2. **void** Graph::AddEdges()
3. {
4. cout << "输入？ ？ 0代表结束" << endl;
5. string start, end; **int** value;
6. cout << "请输入两个顶点及边：";
7. cin >> start >> end >> value;
8. **while** (start != "?" && end != "?")
9. {
10. **int** u = FindV(start);
11. **int** v = FindV(end);
12. **if** (u != -1 && v != -1)
13. {
14. Edge[u][v] = value;
15. Edge[v][u] = value;
16. }
17. **else**
18. {
19. cout << "您输入了无效的顶点，请重新输入" << endl;
20. }
21. cout << "请输入两个顶点及边：";
22. cin >> start >> end >> value;
23. }
24. cout << endl;
25. }

**3.3.3添加边功能截屏示例**



**3.4 构造最小生成树功能的实现**

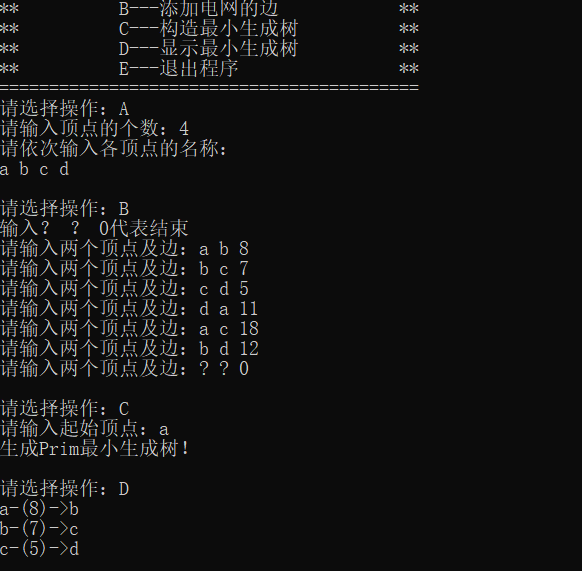
**3.4.1构造最小生成树功能流程图**

****

**3.4.2构造最小生成树功能核心代码**

1. //构造最小生成树
2. **void** Graph::CreateMinSpanT()
3. {
4. string point;
5. **int** startpoint = 0;
6. cout << "请输入起始顶点：";
7. cin >> point;
8. startpoint = FindV(point);
9. **int** verticesnum = Verticesnum;
10. **int**\* lowcost = **new** **int**[verticesnum];
11. **int**\* nearvex = **new** **int**[verticesnum];
12. **for** (**int** i = 0; i < verticesnum; i++)
13. {
14. **if** (i != startpoint)
15. {
16. lowcost[i] = Edge[startpoint][i];
17. nearvex[i] = startpoint;
18. }
19. }
20. nearvex[startpoint] = -1;
21. EdgeNode temp;//最小生成树结点辅助单元
22. **for** (**int** i = 1; i < verticesnum; i++)
23. {
24. **int** min = Maxn; **int** v = -1;
25. **for** (**int** j = 0; j < verticesnum; j++)
26. {
27. **if** (nearvex[j] != -1 && lowcost[j] < min)
28. {
29. v = j; min = lowcost[j];
30. }
31. }
32. **if** (v>=0)
33. {
34. temp.head = Vertices[nearvex[v]];
35. temp.tail = Vertices[v];
36. temp.cost = lowcost[v];
37. MinSpanT.push\_back(temp);
38. nearvex[v] = -1;
39. **for** (**int** j = 0; j < verticesnum; j++)
40. {
41. **if** (nearvex[j] != -1 && Edge[v][j] < lowcost[j])
42. {
43. lowcost[j] = Edge[v][j];
44. nearvex[j] = v;
45. }
46. }
47. }
48. }
49. cout << "生成Prim最小生成树！" << endl << endl;;
50. }

**3.4.3构造最小生成树功能截屏示例**



**4 测试**

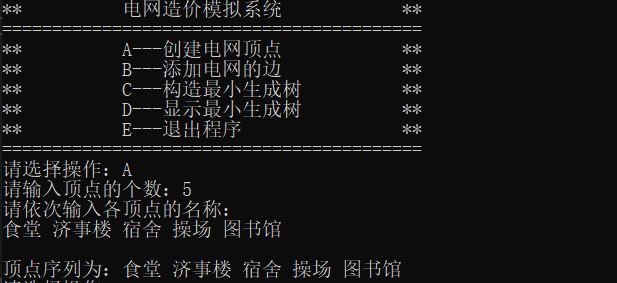
**4.1功能测试**

**4.1.1添加顶点功能测试**

**测试用例：**食堂 济事楼 宿舍 操场 图书馆

**预期结果：**食堂 济事楼 宿舍 操场 图书馆

**实验结果：**



**4.1.2添加边功能测试**

**测试用例：**(顶点：a b c d)

a b 5

b c 8

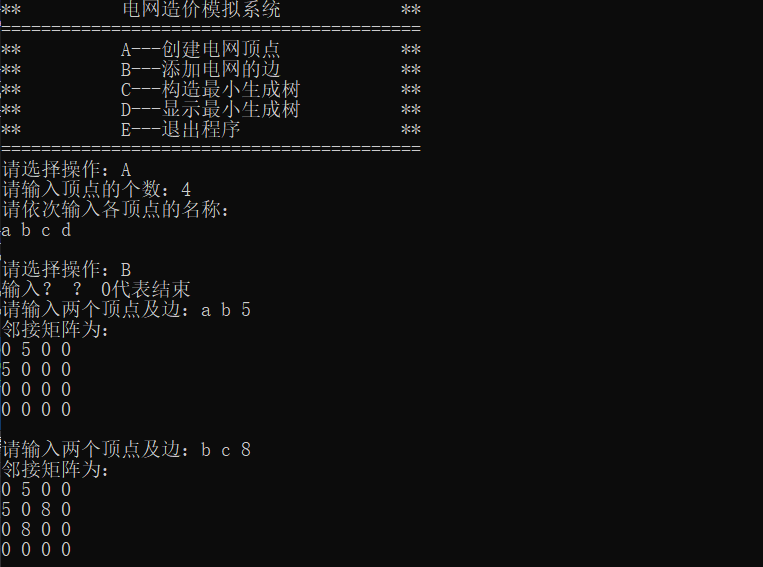
**预期结果：**0 5 0 0

5 0 8 0

0 8 0 0

0 0 0 0

**实验结果：**



**4.1.3构造最小生成树功能测试**

**测试用例：(顶点：a b c d)**

a b 8

b c 7

c d 5

d a 11

a c 18

b d 12

**预期结果：**a-(8)->b

b-(7)->c

c-(5)->d

**实验结果：**

