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： 曹峰源

学 号： 1951328

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 3](#_Toc59042866)

[1.1 背景分析 3](#_Toc59042867)

[1.2 功能分析 3](#_Toc59042868)

[2. 设计 3](#_Toc59042869)

[2.1 数据结构设计 3](#_Toc59042870)

[2.2 类成员 3](#_Toc59042871)

[2.3 系统设计 4](#_Toc59042872)

[3. 功能实现 4](#_Toc59042873)

[3.1 创建电网顶点功能实现 4](#_Toc59042874)

[3.1.1 流程图 4](#_Toc59042875)

[3.1.2 代码实现 4](#_Toc59042876)

[3.1.3 结果显示 5](#_Toc59042877)

[3.2 添加电网边功能的实现 6](#_Toc59042878)

[3.2.1 流程图 6](#_Toc59042879)

[3.2.2 代码实现 6](#_Toc59042880)

[3.2.3 结果显示 7](#_Toc59042881)

[3.3 构造最小生成树功能的实现 8](#_Toc59042882)

[3.3.1 流程图 8](#_Toc59042883)

[3.3.2 代码实现 8](#_Toc59042884)

[3.3.3 结果显示 10](#_Toc59042885)

[3.4 显示最小生成树功能的实现 11](#_Toc59042886)

[3.4.1 代码实现 11](#_Toc59042887)

[3.4.2 结果显示 11](#_Toc59042888)

[3.5 主函数功能实现 12](#_Toc59042889)

[3.5.1 流程图 12](#_Toc59042890)

[3.5.2 代码实现 12](#_Toc59042891)

[3.5.3 结果显示 14](#_Toc59042892)

[4. 错误测试 14](#_Toc59042893)

[4.1 顶点个数错误 14](#_Toc59042894)

[4.2 操作码错误 15](#_Toc59042895)

[4.3 输入的点已经存在 15](#_Toc59042896)

[4.4 插入边的起点或终点不存在 15](#_Toc59042897)

[4.5 最小生成树的起点不存在 16](#_Toc59042898)

1. 分析
   1. 背景分析

现在城市的生活已经离不开电，这就涉及到供电的问题，从最简单的情况考虑，用电线将n个小区连接在一起，这就涉及到了怎么样接线是最短的，是收益最高的？这就需要程序来帮助计算最小的生成线路。

* 1. 功能分析

这段程序中，首先要能满足小区信息的输入，然后能满足小区之间权值（cost）的输入。核心功能是根据输入的小区信息以及权值将最小耗值的线路找出来，并将其输出。此外，我们认为小区之间权值是唯一的，取最小的权值，如果多次输入这两个小区间的权值，认为是违法的。

1. 设计
   1. 数据结构设计

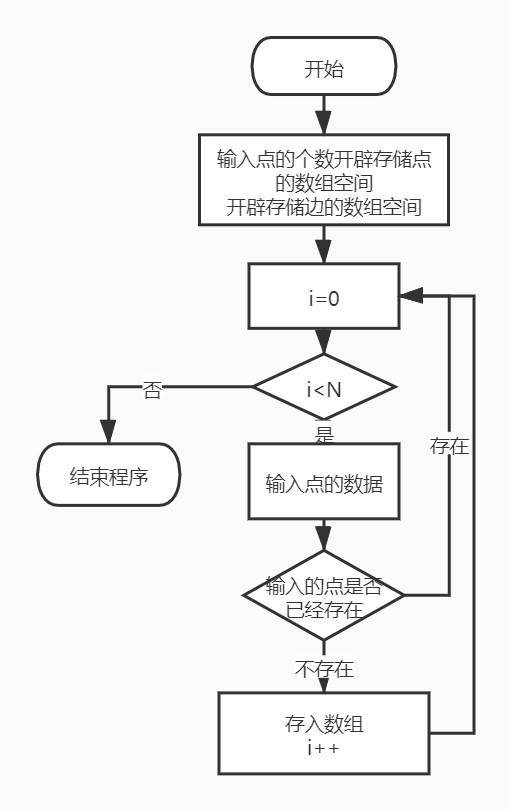
因为不同点代表的不同小区之间连线较为复杂，所以采取图这样的存储方式。每个小区采用char数组来存储点，小区间的权值采取邻接矩阵来存储。

* 1. 类成员

1. **class** graph
2. {
3. **private**:
4. **int** numofvertex;//图中点的个数
5. **int** numofedge;//图中边的个数
6. **char**\* avertex;//存放点的数组
7. **int**\*\* aedge;//存放边的二维数组
8. **int**\* primedge;//存放最小生成树边的数据
9. **char**\* primu;//存放最小生成树起点
10. **char**\* primv;//存放最小生成树终点
11. **public**:
12. graph() { numofvertex = 0; numofedge = 0; }
13. ~graph()
14. {
15. **delete**[]avertex; //释放点数组
16. **for** (**int** i = 0; i < numofvertex; i++)//释放边数组
17. {
18. **delete**[]aedge[i];
19. }
20. **delete**[]aedge;
21. }
22. **int** getnumofvertex() { **return** numofvertex; }//得到点的个数
23. **int** getnumofedge() { **return** numofedge; }//得到边的个数
24. **void** create(**int** n);//用于开辟一个大小为n的数组来存储点的数据
25. **void** insertedge(**char** u, **char** v, **int** cost);//向图中插入边
26. **void** prim(**char** u);//prim算法实现最小生成树
27. **void** printprim();//将最小生成树打印出来
28. };
    1. 系统设计

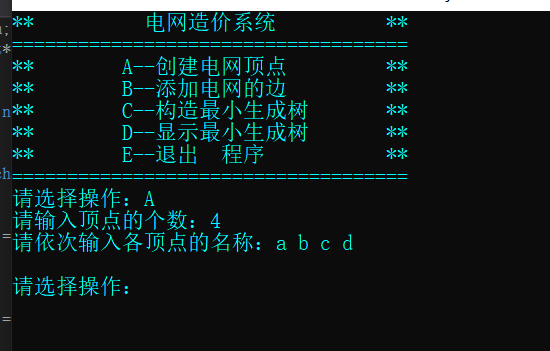
在操作台控制界面中，提供可选择的信息，选择对应的操作码后进行相关的操作，满足手动退出程序。

1. 功能实现
   1. 创建电网顶点功能实现
      1. 流程图

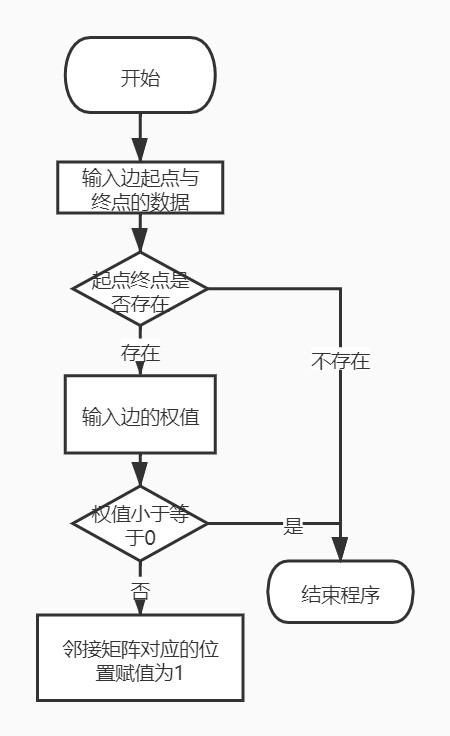


* + 1. 代码实现

1. **void** graph::create(**int** n)
2. {
3. numofvertex = n;
4. aedge = **new** **int**\* [numofvertex];
5. **for** (**int** i = 0; i < numofvertex; i++)
6. {
7. aedge[i] = **new** **int**[numofvertex];
8. }
9. **char** u;
10. avertex = **new** **char**[numofvertex];
11. **for** (**int** i = 0; i < numofvertex; i++)
12. {
13. avertex[i] = '0';
14. }
15. **for** (**int** i = 0; i < numofvertex; i++)
16. {
17. **for** (**int** j = 0; j < numofvertex; j++)
18. {
19. aedge[i][j] = 0;
20. }
21. }
22. **for** (**int** i = 0; i < numofvertex; i++)
23. {
24. cin >> u;
25. **int** j = 0;
26. **for** (j = 0; j < numofvertex; j++)
27. {
28. **if** (avertex[j] == '0') **break**;
29. **if** (avertex[j] == u) **break**;
30. }
31. **if** (avertex[j] == '0')
32. {
33. avertex[i] = u;
34. }
35. **else**
36. {
37. cout << "输入的点已经存在" << endl;
38. i--;
39. }
40. }
41. }
    * 1. 结果显示

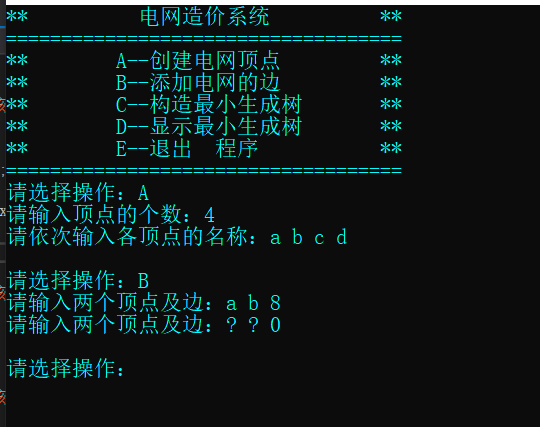


* 1. 添加电网边功能的实现
     1. 流程图

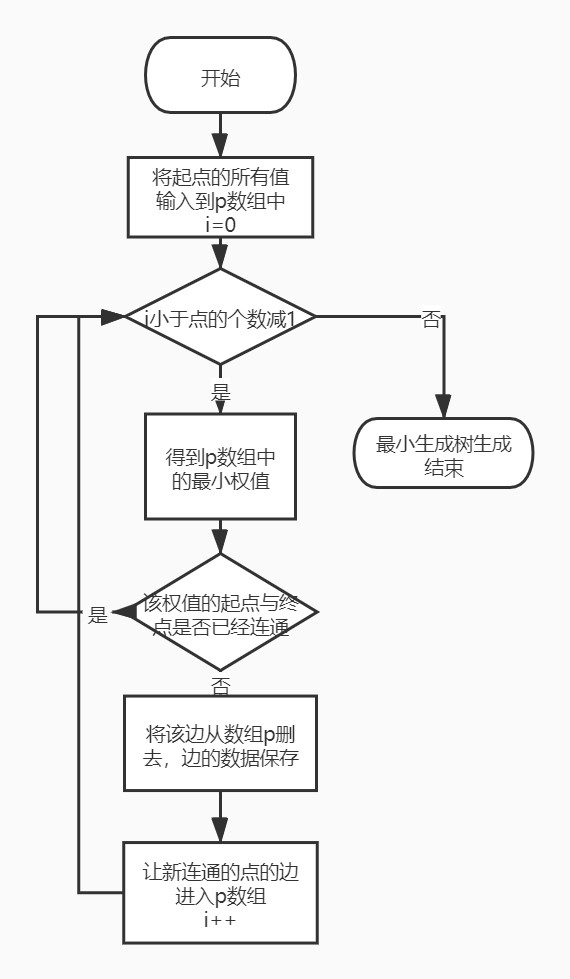


* + 1. 代码实现

1. **void** graph::insertedge(**char** u, **char** v, **int** cost)
2. {
3. **int** x = -1, y = -1;
4. **for** (**int** i = 0; i < numofvertex; i++)
5. {
6. **if** (avertex[i] == u) { x = i; **break**; }
7. }
8. **if** (x == -1)
9. {
10. cout << "该图中没有点"<< u << endl;
11. **return**;
12. }
13. **for** (**int** i = 0; i < numofvertex; i++)
14. {
15. **if** (avertex[i] == v) { y = i; **break**; }
16. }
17. **if** (y == -1)
18. {
19. cout << "该图中没有点"<< v << endl;
20. **return**;
21. }
22. **if** (cost<=0)
23. {
24. cout << "该边的权值违法！";
25. **return**;
26. }
27. aedge[x][y] = cost;
28. aedge[y][x] = cost;
29. numofedge++;
30. }
    * 1. 结果显示

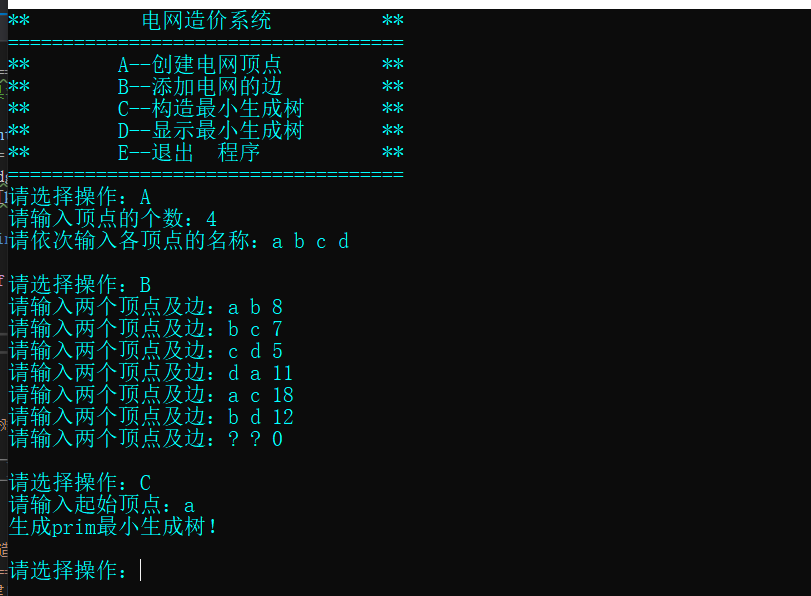


* 1. 构造最小生成树功能的实现
     1. 流程图



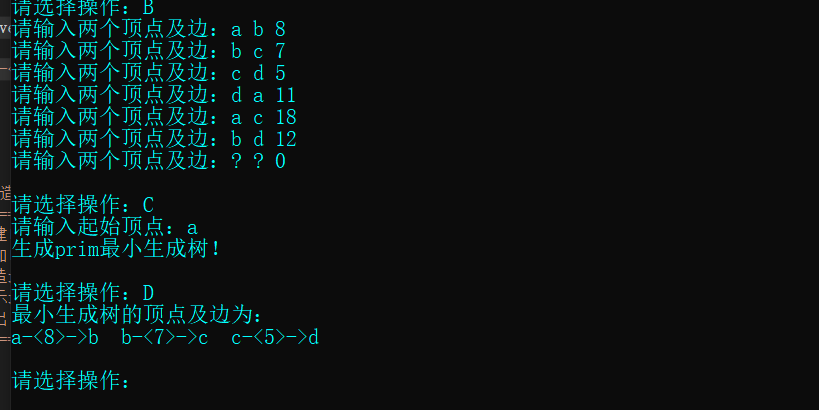
* + 1. 代码实现

1. **void** graph::prim(**char** u)
2. {
3. **int** num1 = getnumofedge();//num1等于图中边的个数
4. **int** num2 = getnumofvertex();//num2等于图中点的个数
5. primedge = **new** **int**[num2 - 1];//primedge用于存放最小生成树边的数据
6. primu = **new** **char**[num2 - 1];//用于存放最小生成树起点的数据
7. primv = **new** **char**[num2 - 1];//用于存放最小生成树终点的数据
8. **int**\* p = **new** **int**[num1];//开辟一个大小为边个数的数组
9. **int**\* q = **new** **int**[num2];//开辟一个大小为点个数的数组
10. **int** i = 0, j = 0, k = 0;
11. **for** (i = 0; i < numofvertex; i++)
12. {
13. **if** (avertex[i] == u)**break**;
14. }
15. **if** (i == numofvertex)
16. {
17. cout << "输入的起始顶点有误！" << endl;
18. **return**;
19. }
20. i = 0;
21. **for** (k = 0; k < num1; k++)//将p数组中每个值都先赋为0
22. {
23. p[k] = 0;
24. }
25. k = 0;
26. **for** (i = 0; i < numofvertex; i++)//现将起点u的所有边进入p数组
27. {
28. **if** (avertex[i] == u) **break**;
29. }
30. **for** (j = i; j < numofvertex; j++)
31. {
32. **if** (aedge[i][j] != 0)
33. {
34. p[k] = aedge[i][j]; k++;
35. }
36. }
37. k = 0;
38. **for** (i = 0; i < numofvertex - 1; i++)
39. {
40. **int** min = 0, count = 0;
41. j = 0;
42. **while** (j < num1)//统计现在p数组中有多少条边
43. {
44. **if** (p[j] == 0) **break**;
45. j++;
46. }
47. count = j;//count为当前p数组中有多少边
48. **for** (j = 0; j < count; j++)
49. {
50. **if** (p[min] > p[j]) { min = j; }
51. }
52. **int** mincost = p[min];//得到最小开销，将其交换到数组p中的靠后位置
53. p[min] = p[count - 1];
54. p[count - 1] = mincost;
55. **int** m = 0, n = 0;
56. **for** (m = 0; m < numofvertex; m++)//得到原边矩阵中的起点和终点的信息，如果起点和终点不满足prim算法，则忽略
57. {
58. **for** (n = m; n < numofvertex; n++)
59. {
60. **if** (aedge[m][n] == mincost)//在aedge数组中找到该最小值的位置
61. {
62. **if** (q[m] == 1 && q[n] == 1) { p[count - 1] = 0; i--; }//如果该最小值的两个点已经连通了，就将该边删掉，i--
63. **else**//如果没有连通
64. {
65. p[count - 1] = 0;//先将该边删掉
66. q[m] = 1; q[n] = 1;//两个点置为连通
67. primedge[k] = mincost;//相关的生成树的值进入特定数组存储
68. primu[k] = avertex[m]; primv[k] = avertex[n];
69. k++;
70. **for** (**int** s = n; s < numofvertex; s++)//将下一个点连通的边进入p数组
71. {
72. **if** (aedge[n][s] != 0) { p[count - 1] = aedge[n][s]; count++; }
73. }
74. }
75. }
76. }
77. }
78. }
79. cout << "生成prim最小生成树！" << endl << endl;
80. }
    * 1. 结果显示

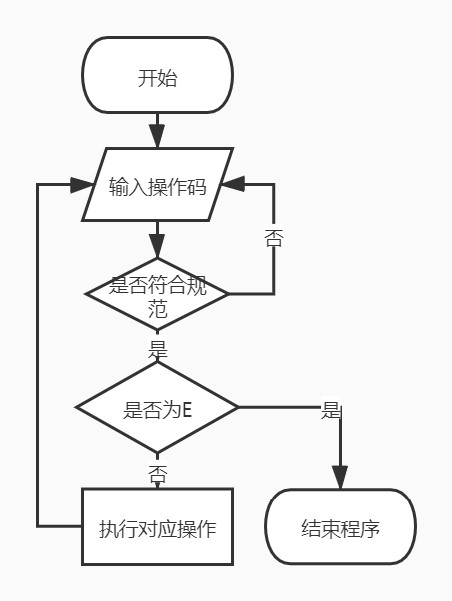


* 1. 显示最小生成树功能的实现
     1. 代码实现

1. **void** graph::printprim()
2. {
3. **for** (**int** i = 0; i < numofvertex - 1; i++)
4. {
5. cout << primu[i] << "-<" << primedge[i] << ">->" << primv[i] << "  ";
6. }
7. }
   * 1. 结果显示

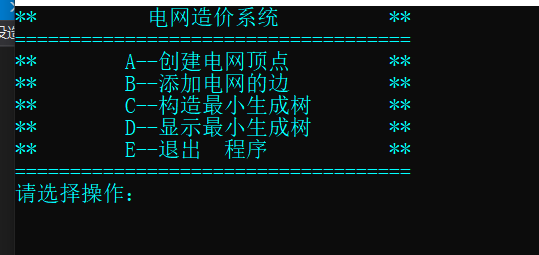


* 1. 主函数功能实现
     1. 流程图



* + 1. 代码实现

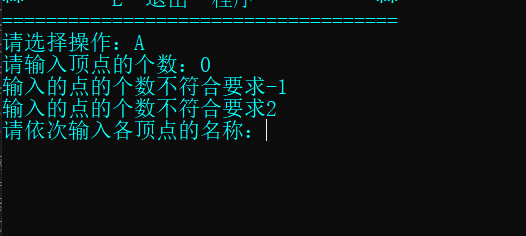
1. **int** main()
2. {
3. cout << "\*\*          电网造价系统          \*\*" << endl;
4. cout << "====================================" << endl;
5. cout << "\*\*        A--创建电网顶点         \*\*" << endl;
6. cout << "\*\*        B--添加电网的边         \*\*" << endl;
7. cout << "\*\*        C--构造最小生成树       \*\*" << endl;
8. cout << "\*\*        D--显示最小生成树       \*\*" << endl;
9. cout << "\*\*        E--退出  程序           \*\*" << endl;
10. cout << "====================================" << endl;
11. graph g;
12. cout << "请选择操作：";
13. **char** order;
14. **while** (cin >> order)
15. {
16. **switch** (order)
17. {
18. **case** 'A':
19. {
20. cout << "请输入顶点的个数：";
21. **int** N;
22. **while** (cin >> N)
23. {
24. **if** (N > 0) **break**;
25. **else** cout << "输入的点的个数不符合要求";
26. }
27. cout << "请依次输入各顶点的名称：";
28. g.create(N);
29. cout << endl;
30. **break**;
31. }
32. **case** 'B':
33. {
34. **while** (1)
35. {
36. cout << "请输入两个顶点及边：";
37. **char** u, v;
38. **int** cost;
39. cin >> u >> v >> cost;
40. **if** (u == '?') **break**;
41. g.insertedge(u, v, cost);
42. }
43. cout << endl;
44. **break**;
45. }
46. **case** 'C':
47. {
48. cout << "请输入起始顶点：";
49. **char** u;
50. cin >> u;
51. g.prim(u);
52. **break**;
53. }
54. **case** 'D':
55. {
56. cout << "最小生成树的顶点及边为：" << endl;
57. g.printprim();
58. cout << endl << endl;
59. **break**;
60. }
61. **case** 'E':
62. {
63. system("pause");
64. exit(0);
65. **break**;
66. }
67. **default**:cout << "输入的操作码有误" << endl; **break**;
68. }
69. cout << "请选择操作：";
70. }
71. **return** 0;
72. }
    * 1. 结果显示



1. 错误测试
   1. 顶点个数错误

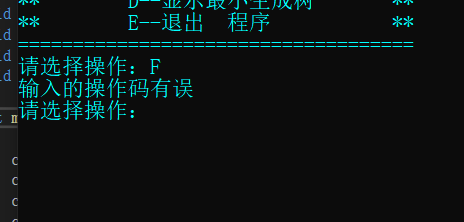
认为顶点数小于等于0时属于违法的输入

希望实现：程序不崩溃，提示错误信息重新输入



* 1. 操作码错误

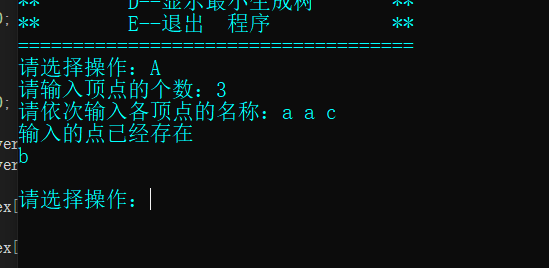
当输入的操作码不存在时，希望实现：程序不停止，显示错误信息，重新输入。



* 1. 输入的点已经存在

当出现有重复出现的点时，对于程序是难以判别区别的，认为属于违法操作

希望实现：程序显示错误信息，重新输入

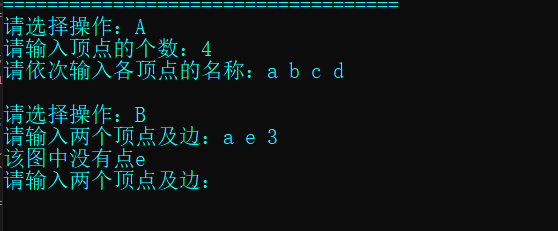


此时点数组中的数据依次为a b c

* 1. 插入边的起点或终点不存在

如果出现插入的边的起点或终点不存在的情况

希望实现：程序不崩溃，出现报错信息，这次输入作废



* 1. 最小生成树的起点不存在

最小生成树的起点是由人手动输入的，所以起点可能会有不存在的情况

希望实现：程序不崩溃，最小生成树停止生成

