离散数学课程设计

项目说明文档

最小生成树

|  |  |
| --- | --- |
| 作者姓名： | 杨烜赫 |
| 学 号： | 2252709 |
| 指导教师： | 李 冰 |
| 学院专业： | 软件学院 软件工程 |



同济大学

Tongji University

1. **题目简介**
2. **实验目的**

如下图所示的赋权图表示某七个城市，预先计算出它们之间的一些直接通信道路造价（单位：万元），试给出一个设计方案，使得各城市之间既能够保持通信，又使得总造价最小，并计算其最小值。

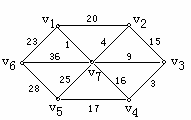
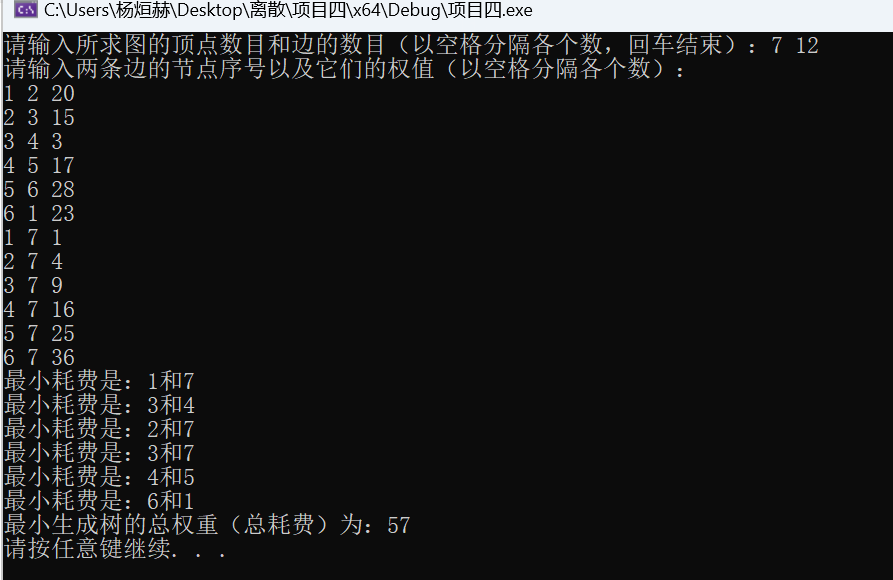


图1 七个城市赋权图

**2.实验内容**

用 Kruskal 算法求解给出图的最小生成树。

1. **项目示例**
2. **项目实现**

## 实验原理

Kruskal是一种典型的贪心策略算法，其基本思想是：按照权值从小到大的顺序选择n-1条边，并保证这 n-1 条边不构成回路。具体做法为：首先构造一个只含n个顶点的森林，然后依权值从小到大从连通网中选择边加入到森林中，并使森林中不产生回路，直至森林变成一棵树为止。

Kruskal 和 Prim 算法的对比：

Prim 算法是从图中某一个顶点出发，寻找它相连的所有结点，比较这些结点的权值大小，然后连接权值最小的那个结点。然后将寻找这两个结点相连的所有结点，找到权值最小的连接。重复上一步，直到所有结点都连接上。

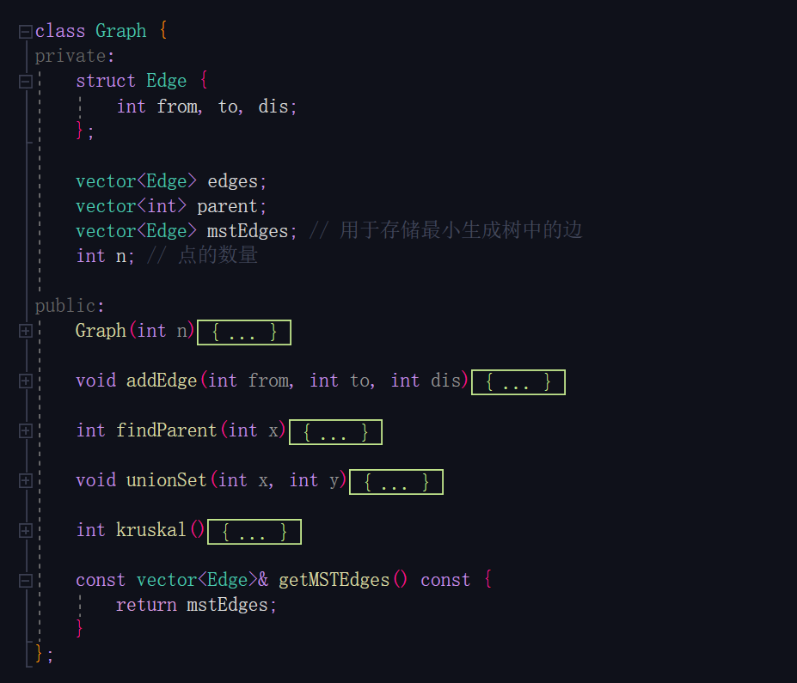
设图有 n 个点，m 条边。

易分析得到，Prim 算法对于每一个点都需要将全部点遍历一遍，以找到最近的点加入集合。故时间复杂度为 O(𝑜 2 ).

对 Kruskal 算法分析，对 m 条边的排序所花时间为 O（𝑛𝑚𝑝𝑔𝑛）。在建立最小生成树时，前面通过路径压缩的方法，已经将每次并查集操作可看为 O(1)，故建立最小生成树的时间复杂度为 O(𝑜)。总时间复杂度为 O（𝑛𝑚𝑝𝑔𝑛 + 𝑜）.

可以分析得到，当图为稠密图，边数接近𝑜 2 时，Prim 的复杂度为 O(𝑜 2 )要优秀于 Kruskal 的（𝑜 2 𝑚𝑝𝑔𝑜），所以 Prim 比 Kruskal 更适合于稠密图的求解。

## 实现过程

类的总实现：

**具体实现如下：**



**Graph 类**

用于表示图的数据结构，包含图的边和顶点。

成员变量：

edges：存储图中所有边的信息。

parent：用于并查集中表示每个顶点的父顶点。

mstEdges：存储最小生成树中的边。

n：图中顶点的数量。

功能如下：

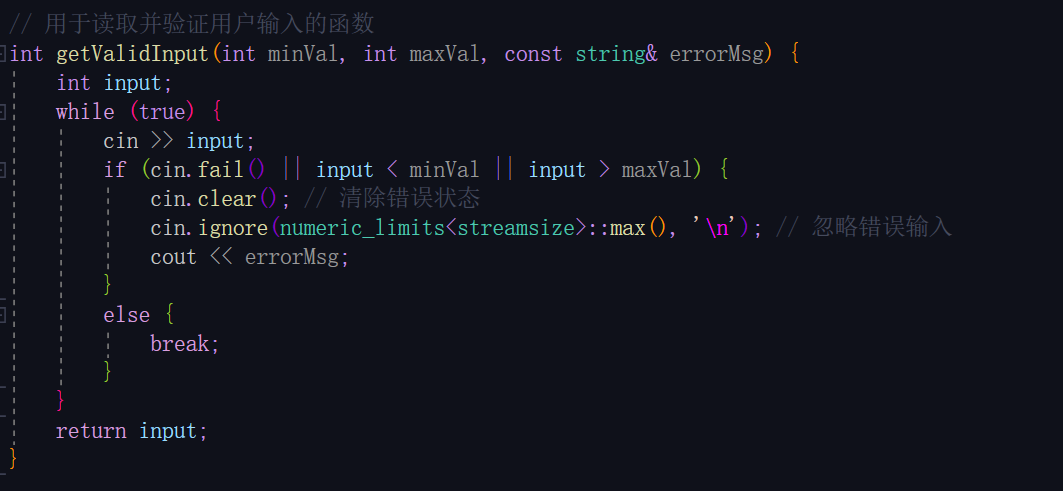
addEdge：向图中添加边。

findParent：并查集中查找顶点的根顶点。

unionSet：并查集中合并两个顶点所在的集合。

kruskal：实现 Kruskal 算法，计算图的最小生成树，并返回树的总权重。

getMSTEdges：获取最小生成树中的边。



**getValidInput 函数**

用于读取并验证用户的输入，确保输入的整数在指定的范围内。

参数包括最小值 minVal、最大值 maxVal 和错误信息 errorMsg。

使用循环和条件判断来验证输入，并在输入不合法时提示用户重新输入。

**main 函数**

主程序入口，用于读取用户输入并展示最小生成树的结果。

流程包括：

读取图的顶点数和边数。

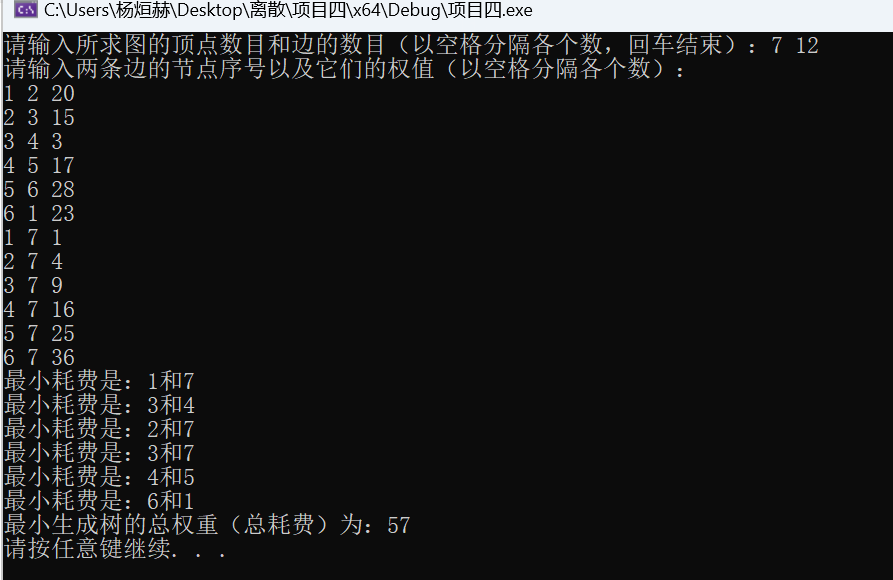
创建 Graph 对象。

循环读取每条边的信息并添加到图中。

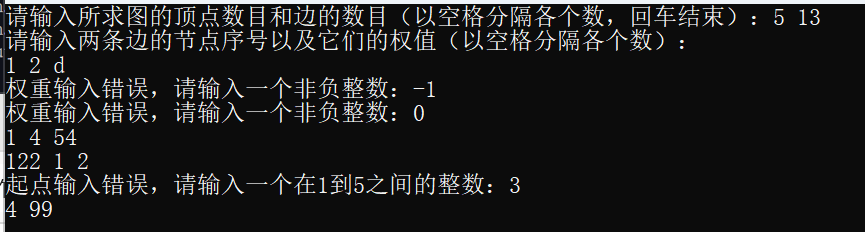
调用 kruskal 方法计算最小生成树。

输出最小生成树的边和总权重。

**三、项目测试**

****结果测试：

**实际值与理论值相同**

健壮性测试：

#### 可看出代码对节点序号和节点权值的输入做到了较好的错误处理，当用户输入错误数据时会提示具体的错误信息并使用户重新输入正确的数据，体现了代码健壮性良好。

**四、心得与总结**

这个实验通过实现 Kruskal 算法来计算图的最小生成树，提供了一个深入学习图论和算法实现的机会。使用并查集数据结构来优化 Kruskal 算法的效率。这不仅提高了我对并查集结构的理解，还体现了数据结构在优化算法中的重要性。Kruskal 算法的实现进一步锻炼了我编写高效算法的能力。通过这次实验，不仅加强了我对图论和算法的理解，还提高了编程技巧，特别是在实现复杂算法和处理用户输入方面。通过实际编写和测试代码，能够更深入地理解理论概念，并学会如何将这些概念应用于解决实际问题。这次实验的经验对日后在更复杂的场景中应用离散数学和数据结构非常有帮助。