《离散数学》课程实验报告4 最小生成树

**1、题目简介**

题目简介：最小生成树问题是图论中的经典问题之一。给定一个带权重的无向图，任务是找到一个包含图中所有节点的子图，使得该子图是一棵树且边的权重之和最小。换句话说，要找到连接图中所有节点的最小权重的边集合，构成一棵树，这棵树被称为最小生成树。

要求：编写一个算法，从给定的带权重无向图中找到其最小生成树，并输出最小生成树的边集合和总权重。你可以选择使用经典的最小生成树算法，如Prim算法或Kruskal算法，来解决这个问题。定义关系的自反性、对称性和传递性。

**2、解题思路**

设有一个有 n个顶点的连通网络n个顶点的连通网络N = { V, E } , 最初先构造一个只有 最初先构造一个只有 n 个顶点，没有边的非连 个顶点，没有边的非连 通图 T = { V, ∅ }, 图中每个顶点自成一个连通 图中每个顶点自成一个连通 分量。当在 E 中选到一条具有最小权值的边时 中选到一条具有最小权值的边时,若该边的两个顶点落在不同的连通分量上，则 若该边的两个顶点落在不同的连通分量上，则将此边加入到 T 中；否则将此边舍去，重新选 中；否则将此边舍去，重新选 择一条权值最小的边。如此重复下去，直到所 择一条权值最小的边。如此重复下去，直到所 有顶点在同一个连通分量上为止。 有顶点在同一个连通分量上为止。

并且利用并查集的运算检查依附于一条边的两个顶点 tail、head是否在同一个连通 是否在同一个连通 分量 (即并查集的同一个子集合 即并查集的同一个子集合) 上, 是则舍去 这条边；否则将此边加入 这条边；否则将此边加入 T，同时将这两个顶 同时将这两个顶 点放在同一个连通分量上。随着各边逐步加入 点放在同一个连通分量上。随着各边逐步加入 到最小生成树的边集合中，各连通分量也在逐 到最小生成树的边集合中，各连通分量也在逐步合并，直到形成一个连通分量为止。逐步合并，直到形成一个连通分量为止。

**3、数据结构设计**

这个作业的核心数据结构是无向图，同时我才用克鲁斯卡尔算法解题，核心的算法是找边法，所以我采用vector类来存储所有的边，同时设置一个结构体存储边，里面有开始顶点，结束顶点以及边的权值。

**4、核心算法**

**//寻找head顶点在并查集中的祖先**

int find(int head, vector<int>points)

{

while (points[head - 1] != -1)

{

head = points[head - 1] + 1;

}

return head - 1;

}

**//判断是否要选择这条边**

bool inpoints(int head, int tail, vector<int> points)

{

return find(head, points) == find(tail, points);

}

int Kruskal(int v, int n, vector<side\*>& sides)

{

int sum = 0;

vector<int> points;

for (int i = 1; i <= n; i++)//初始化并查集

{

points.push\_back(-1);

}

int pointnum = 0;

int index = 0;

while (pointnum<n-1)

{

int head = sides[index]->head;

int tail = sides[index]->tail;

if (!inpoints(head, tail,points))//判断新加入的边是否在同一个集合里面

{

//输出这条边，然后更新并查集，要将后面点的祖先接到前面的点上

points[find(tail, points)] = head - 1;

cout << "最小耗费是：" << head << "和" << tail<<endl;

pointnum++;

sum += sides[index]->cost;

}

index++;

}

return sum;

}

struct side//定义边的结构体

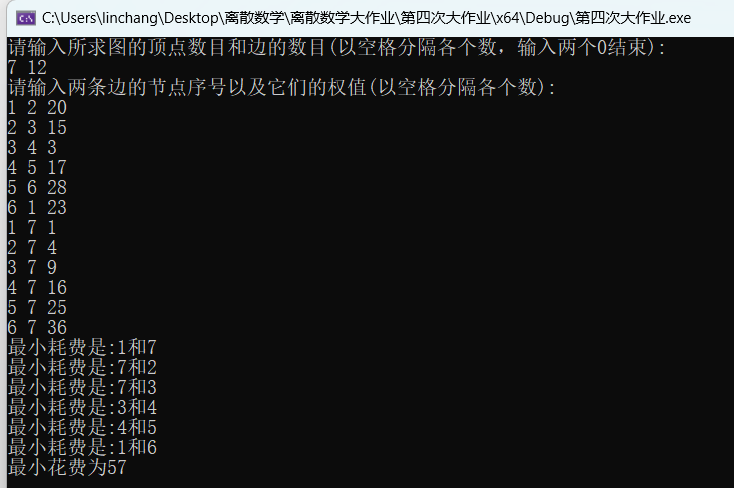
{

int head;

int tail;

int cost;

};}**结果展示：**



**5、心得体会**

这次的大作业主要有一下心得体会：

1、加强了我对于将实际问题转化为一个数学模型的能力，在这次的作业中的问题用例是一个实际的问题：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。然后我将这个问题巧妙的转化成了无向图求最小生成树的问题，加强了我对于将实际问题转化为一个数学模型的能力。

2、提高了我解决实际问题的能力，将一个实际问题，抽象化为一个逻辑表达式后，可以借助计算机优秀的计算能力，设计出一个合理的算法，通过计算机解决实际问题，让我感受到了计算机技术的强大。

3、巩固了在课堂上学到的知识，相当于再次复习了书本中逻辑命题中的知识点，比如如何将一个无向图用邻接矩阵存储起来，以及如何求解一个无向图的最小生成树。同时也拓展了我的知识，prim算法在课上没有提及，我课后自己查找资料自学了prim算法，拓展了我的知识面。注：提交的是Kruskal算法，prim也写了一版。

4、提高了我的代码能力。将一个算法实际用代码写出来还是十分考验我的代码能力的，比如实际写代码的过程中我就出现了数组越界等诸多问题，整个作业完成之后，我感觉到我的代码能力得到了提升。