# Homework12实验报告

建立Map\_Solution类，含：Node类，两个顶点x，y，边的权重weight（邻接表只需要一个顶点，但这里还要将元素取出，对边的权重做排序，所以直接对另外一个节点也做了标记，一般用邻接表存储只需要y和weight即可表示图）。

构造函数：有6个顶点，所以count置为6，构建的vector<Node>数组，长度也为6，因为根据邻接表的定义，一共六个顶点，所以长度为6。

Create函数，建立一个Node类型变量a，用大括号的方法赋值，一个一个push\_back到对应的adj中，即一个可变长度的二维数组，这样更加有效的利用空间。由于数据要根据题目给定的图来，所以都要一个一个手动输入。

Print函数，二重循环遍历adj，将邻接表中存储的数据打印出来。

Kruskal函数，遍历邻接表，将里面存的数据导入到另一个一维vector<Node> all中，因为邻接表存储时x，y和y，x都会存储到邻接表中，所以在push\_back的时候要检查是否重复，重复的就不再放入，遍历all检查是否重复，同时使用了s做标记，s为-1时就不再放入all中，打印一遍all看看是否导入正确。然后就是克鲁斯卡尔算法的关键了，用到了并查集的思想，先调用Make\_set函数把对应的fa置为本身，即一开始点和点之间没有和任何关联，然后调用C++库中的sort函数对all做排序，方法为Cmp，即边a和b的权重不一样时返回a的权重是否小于b，一样时返回a的起始顶点序号是否小于b，这样就对all根据边的权重和起始顶点做了一个排序。然后从0到m（不包括m，m即vector的大小）遍历all，寻找x和y的根节点，如果他们根节点不一样，即他们不连通，就调用Union\_set函数，这个函数参考了并查集的核心，fa[fa[x]]=fa[y]，把x关联到y，然后输出x和y以及它们边长的权重，总的权重加w，循环的结束条件是k等于n（即k加了n-1次，得到了n-1条边），输出总的权重。fa[fa[x]]=fa[y]着重介绍一下，要和Find函数一起看，假设输入1 2和1 3和1 4，fa[1]=2，fa[2]=3，然后输入1 4时Find函数会把fa[1]更新为3，即它的祖先，这样可以提高检索效率，然后fa[3]=4，即4变成了祖先，以后在输入和1或2有关的数时它们就会被更新为祖先4。

印张悦

10174503110