四、树Tree（下）

1.二叉搜索(排序/查找)树Binary Search Tree
    1.1.特点：
        左子树小于根节点
        右子树大于根节点
        左右子树都是二叉搜索树
    1.2.操作：
        查找：元素X/最小元素/最大元素，返回其地址：
            与根节点比大小/最左结点/最右结点
        插入：不断比较，找到空位置则插入（注意记住上一个节点的位置，用来插入（return来实现））
        删除：
            a.有一个孩子节点：放在被删除的节点位置
            b.有两个儿子节点：右子树最小或左子树最大来替代被删除元素的位置（这两个元素一定只有一个孩子节点，于是再把其孩子节点放在他的位置上）
2.平衡二叉树Balanced Binary Tree：使ASL尽量小
    2.1.指标：
        平衡因子Balance Factor：BF(T)=hl-hr：
        任一节点左右高度差不超过一：lBF(T)l<=1
    2.2.高度为h的最小结点数nh=n（h-1）+n(h-2)+1
                    =F(h+2)–1   Fi为斐波那契数列（指数函数）
    2.3.查找效率：O（logn）？？？？？
    2.4.平衡二叉树的调整：
        a.RR旋转：
            麻烦结点（从下往上找，最下面的被破坏平衡的结点）在发现者右子树的右子树：RR插入：RR旋转（右单旋）：麻烦结点变成左子树，原来的右子树提上来，右子树的左子树变成麻烦结点的右子树
        b.LL旋转：
        c.LR旋转：：最下面的放到中间（只调整相关的中间三个结点
        d.RL旋转
3.堆heap
    3.1.优先队列Priority Queue:根据优先权而不是时间先后
        3.1.1数组实现：
            插入：尾部O(1)
            删除：查找最大（小）关键字O（n）+从数组中删除O（n），因为删除后要挪
        3.1.2.链表实现：
            插入：头部O（1）
            删除：查找O（n）+删除O（1）
        3.1.3.有序数组实现
            插入：找到合适位置O（n）或者O(logn)
                移动元素并插入O(n)
            删除：删除最后一个元素 O（1）（最后一个最大）
        3.1.4.有序链表
            插入：查找O（n）+插入O（1）
            删除：删除首元素或尾元素O（1）
        3.1.5.用二叉搜索树？：每次删除最大越来越歪
        3.1.6.完全二叉树表示：
    3.2.堆的两个特性：
        a.结构性：用数组表示的完全二叉树（必要条件）
        b.有序性：最大堆MaxHeap，大顶堆：每个子树root最大
                最小堆MinHeap，小顶堆：

从根节点向下划任意路径都有序：递减/递增

3.3.最大堆的操作：

创建：

N个元素一个个比较并放入初始为空的堆中O(NlogN)

先顺序放入成完全二叉树→交换排序O(logN)

排序方法：从下往上，让每个子堆都有序

插入：item与父节点比（E[i/2]<item?），

item大则父节点下移

[0]插入一个Max哨兵，for内少写一个判断

删除：

删除根：用最后一个的子节点tmp替代根节点

从根向下比，破坏有序性则儿子向上换

直到找到一个位置放入tmp

O(logn)　（向左或右，每次减半？）

4.哈夫曼树与哈夫曼编码：HuffmanTree

4.1.带权路径长度WPL：权重与路径长度相乘求和

最优二叉树或哈夫曼树：WPL最小

4.2.哈夫曼树构造：

每次把权值最小的两棵二叉树合并（权值相加为根节点）

把根节点和剩下结点继续对比，找出两个最小的合并

O(NlogN) ？？

4.3.哈夫曼树特点：

没有度为1的结点

n个叶子节点的哈夫曼树共有2n-1个节点

任意左右子树交换后仍是哈夫曼树

同一组权值可以有不同构的两棵哈夫曼树（WPL一致）

4.4.哈夫曼编码（不等长编码）

频率高的码长短

前缀码prefix code：

任何字符的编码都不是其他字符编码的前缀

有二义性

避免二义性：二叉树编码：字符只在叶节点上（左0右1）

5.集合

5.1.表示：

集合运算：交、并、补、差、判定一个元素是否属于集合

并查集：集合并+查某元素属于什么集合

5.2.判断连通性：

元素相连则把两个元素合并到一个集合中

判断是否相连则是判断是否属于一个集合

5.3.树结构表示：

双亲表示法，孩子指向双亲（存data和parent）

树的合并：一个树的根节点指向另一个树的根节点

根节点的父节点表示成-n，n代表树一共有多少个结点

小树挂到大树上面