首先，什么是二叉树？二叉树是一种数据结构，用来存储一组数的一个集合。为了好理解根据它的性质我们把它称为树。它有什么性质呢？首先在二叉树里，一个节点，它可以存放一个数，也有个编号，这就是树的一个存放数据的一个最小单位。一个节点它最多只有两个子节点，没有父亲节点的叫根节点，没有子节点的叫叶子结点。节点编号从1开始。这样的话父节点与子节点的关系就有父节点的编号 \* 2等于其左节点的编号，再加1就是其右节点的编号。所以这就是为什么可以用一个一维数组来实现。

二叉树的其他性质：

**性质1**：二叉树第i层上的结点数目最多为 **2{i-1}**(i≥1)。（树的第几层什么的应该很好理解，跟族谱差不多的）  
**性质2**：深度为k的二叉树至多有**2{k}-1**个结点(k≥1)。（这里的深度就是几层的意思）  
**性质3**：包含n个结点的二叉树的高度至少为**log2 (n+1)**。（高度就是层数）  
**性质4**：在任意一棵二叉树中，若终端结点的个数为**n0**，度为2的结点数为**n2**，则**n0=n2+1**。

什么是二叉查找树呢？

二叉查找树(Binary Search Tree)，又被称为二叉搜索树。它的性质就是左节点存储的数据小于等于根节点的数据，右节点的数据大于等于根节点的数据。

很好实现，先用一个循环用一维数组存好后，从后往前遍历数组，根据二叉树的性质把所有有子节点的节点弄成二叉查找树，这样一来，整个二叉树也就变成了二叉查找树。

输出最小值跟输出最大值就不说了，很简单。

查找某个值就拿这个值跟每个有子节点的节点比较，如果大于就在父节点的右节点方位，小于就在父节点的左节点方位，一直比较直到相等。

说下插入跟删除

直接在数组屁股后面插入，然后重新建立二叉查找树，这是最简单粗暴的一个方法。重新建立也不用多长时间，遍历节点数的一半就好了。因为叶子节点占总结点数的一半（总节点数量为偶数时）。其实也不用重新建立，只要调整插入的节点就好了，插入到最后面，跟父节点比较大小，比父节点大放右边，比父节点小放左边。

不过这种只判断父节点跟子节点大小的方法有个缺陷，就是如果一开始左节点和右节点都比父节点大怎么办呢？那只能比较两个节点的大小，小的那个放在右节点，大的那个放在较小那个的右节点上。这样就导致一个问题，需要判断数组长度是否足够，不够就需要扩容数组了。还有就是我们是根据编号 \* 2这样访问子节点的，这样的话需要大一点的数组长度。会有些浪费空间。

要删除某个数，就需要找到它。查找上面说过了。找到后，若该节点有子节点的话需要向上提升。优先找左节点，没有左节点找右节点。覆盖掉父节点的值。这样递归下去。找到叶子结点就返回。