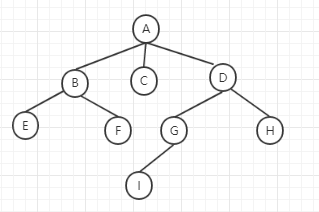
# 二叉搜索树

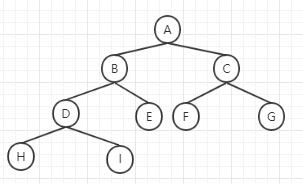
# 1.树

是由n（n>=1）个有限节点组成一个具有层次关系的集合。



# 二叉树

二叉树是一种特殊的树，每个节点最多只有两个子节点。



二叉树有以下几个特征：

1. 深度为k的二叉树最多有2k-1个节点。
2. 如果一个二叉树有n个节点，那么它最少有log2(n+1)层。
3. 如果一个二叉树中，度为0的节点有n0个，度为2的节点有n2个，那么n0=n2+1。

证明，总结点数为n，度为0的数为n0，度为1的数为n1，度为2的数为n2。

因为二叉树中只有度为0、1、2的节点，所以n=n0+n1+n2。

因为度为0的有0个子节点，度为1的有1个子节点，度为2的有2个子节点，根节点没有父节点，所以n=n1+2n2+1。

所以n0=n2+1。

1. 第i层上的节点数目最多为2k-1。

# 满二叉树

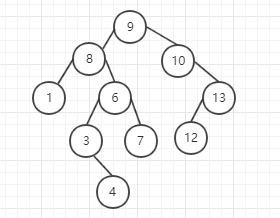
每层节点数都为2k-1。

# 完全二叉树

只有最后两层存在度为1的节点，并且叶子节点集中在靠左的位置上。

# 二叉搜索树

二叉搜索树上，左子树上的节点永远小于该节点，右子树上的节点永远大于该节点。数组适用于查找元素，链表适用于插入删除元素，而二叉搜索树既适用于查找元素也适用于插入删除元素。



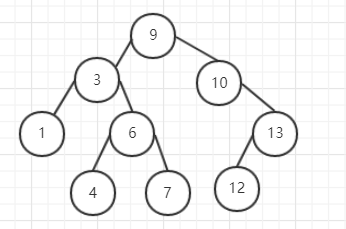
二叉搜索树的插入操作比较简单。主要看一下删除操作。删除的节点主要有下面几种情况：

1. 没有子节点。
2. 有一个子节点。
3. 有两个子节点。

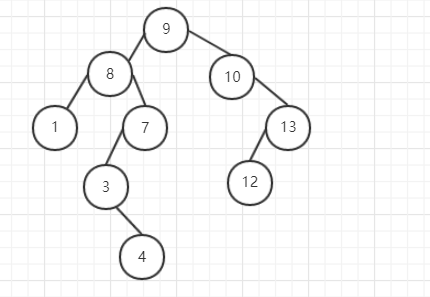
第一种情况比较简单，直接将删除节点的父节点的左子节点或右子节点置空即可。第二种情况也比较简单，直接将删除节点的父节点的左子节点或右子节点指向删除节点的子节点就好了。重点看一下第三种情况。

主要做的操作是：找到删除节点中序遍历的后继（或前驱）节点，然后用后继节点替换掉删除节点，最后删除后继节点。后继节点是删除节点右子树的最左边的节点，所以后继节点不可能再有左子树，删除后继节点就变成了第二种情况。

例如删除8之后：



删除6之后：



下面是代码实现：



这里主要处理了：后继节点的子节点与后继节点的父节点之间的关系，后继节点与删除节点父节点的关系，后继节点与删除节点的关系。

1. 首先找到中序遍历的后继节点afterNode。
2. a后继节点不是删除节点的右子节点：
3. 将后继节点的左子节点指向后继节点的右子节点。
4. 如果后继节点右子节点不为空，则将右子节点的父节点指向后继节点的父节点。
5. 将删除节点的父节点的左子节点或右子节点指向后继节点。
6. 将后继节点的左子节点指向删除节点的左子节点。
7. 将后继节点的右子节点指向删除节点的右子节点。
8. 将后继节点的父节点指向删除节点的父节点。

b后继节点是删除节点的右子节点：

1. 将后继节点的左子节点指向删除节点的左子节点。
2. 将后继节点的父节点指向删除节点的父节点。
3. 将删除节点的父节点的左子节点或右子节点指向后继节点。

完整源码见BTree.java。