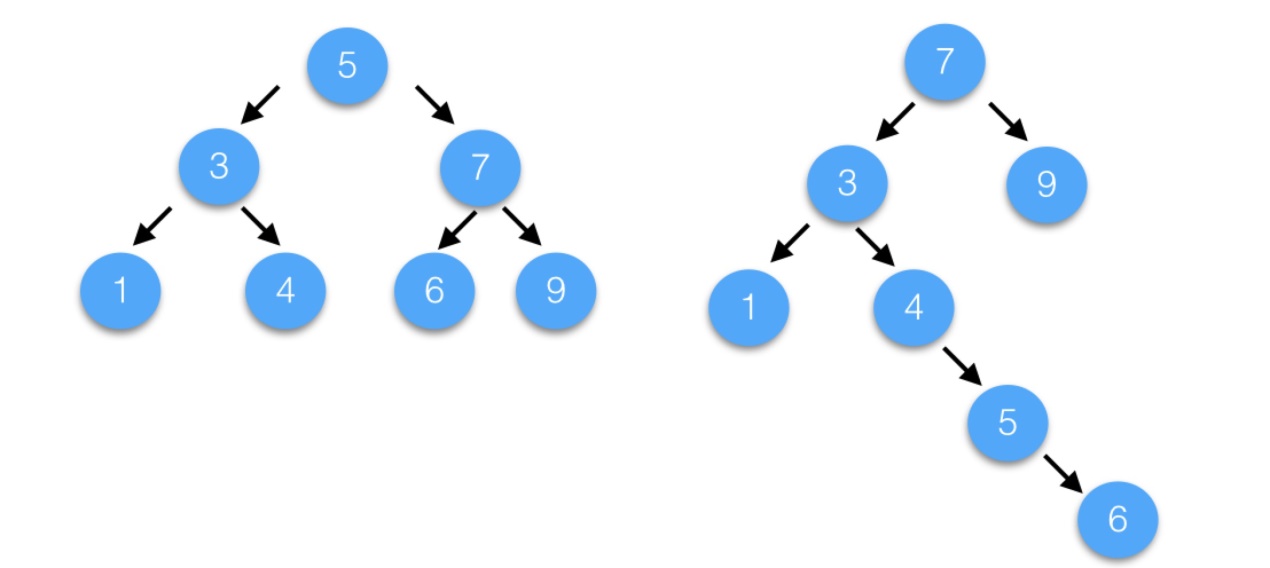
一、基础概念

1. 二叉树

(1) 每个结点最多只有两个子节点的树称为二叉树，有一个或者两个子结点的称为根节点，

没有子结点的称为叶子节点，没有父结点的那个结点是起始结点。

(2) 树有一个高度或深度的概念，是从根到叶子节点经过的节点个数的最大值，左边树的高度为3，右边的为5。

2. 排序二叉树

(1) 排序二叉树也是二叉树，但，它没有重复元素，而且是有序的二叉树。

① 如果左子树不为空，则左子树上的所有节点都小于该节点

② 如果右子树不为空，则右子树上的所有节点都大于该节点

(2) 比如上图的两个二叉树都是排序二叉树。

起始结点为5，左边的子树为3和1依次递减，右边的子树为7和9依次递增。

二、排序二叉树的基础算法

1. 查找

(1) 基本步骤：

① 首先与根节点比较，如果相同，就找到了。

② 如果小于根节点，则到左子树中递归查找。

③ 如果大于根节点，则到右子树中递归查找。

(2) 此外，在排序二叉树中，可以方便的查找最小最大值，最小值即为最左边的节点，从根节点一路查找左孩子即可，最大值即为最右边的节点，从根节点一路查找右孩子即可。

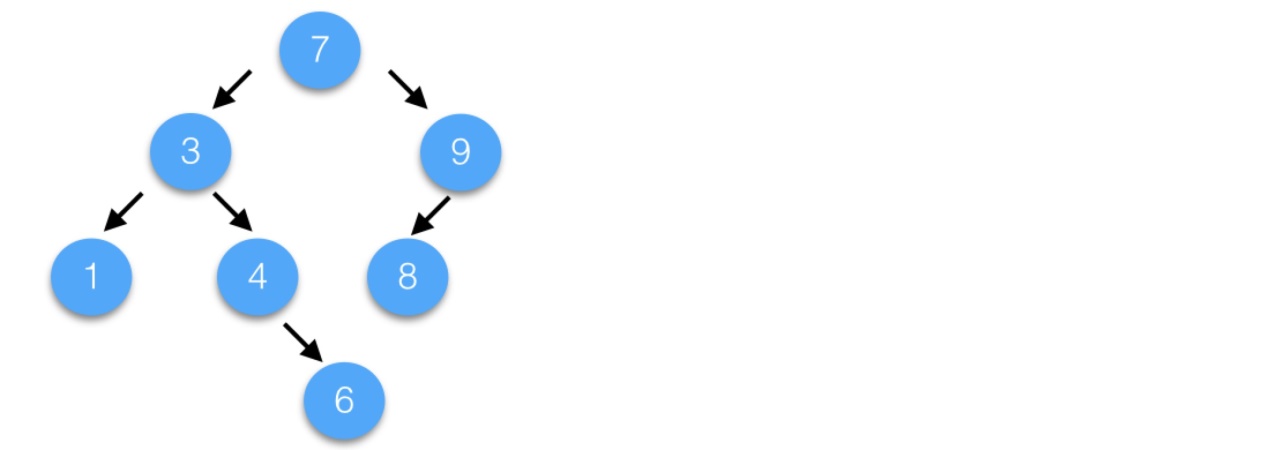
2. 遍历

2.1 按序遍历的步骤：（递归）

① 访问左子树

② 访问当前节点

③ 访问右子树



说明：先访问1，然后是3，因为4没有左子树所以访问4，访问4的右子树6。

2.2 非递归方式，按序遍历排序二叉树

(1) 基本步骤：

① **第一个节点为最左边的节点**，从第一个节点开始，依次找后继节点。给定一个节点，找其后继节点的算法为：

② 如果该节点有右孩子，则后继为右子树中最小的节点。

③ 如果该节点没有右孩子，则后继为父节点或某个祖先节点，从当前节点往上找，如果它是父亲节点的右孩子，则继续找父节点，直到它不是右孩子或父节点为空，第一个非右孩子节点的父亲节点就是后继节点，如果找不到这样的祖先节点，则后继为空，遍历结束

3. 插入

3.1 基本步骤：

在排序二叉树中，插入元素首先要找插入位置，即新节点的父节点。

① 与当前节点比较，如果相同，表示已经存在了，不能再插入

② 如果小于当前节点，则到左子树中寻找，如果左子树为空，则当前节点即为要找的父节点。

③ 如果大于当前节点，则到右子树中寻找，如果右子树为空，则当前节点即为要找的父节点。

④ 找到父节点后，即可插入，如果插入元素小于父节点，则作为左孩子插入，否则作为右孩子插入。

4. 删除

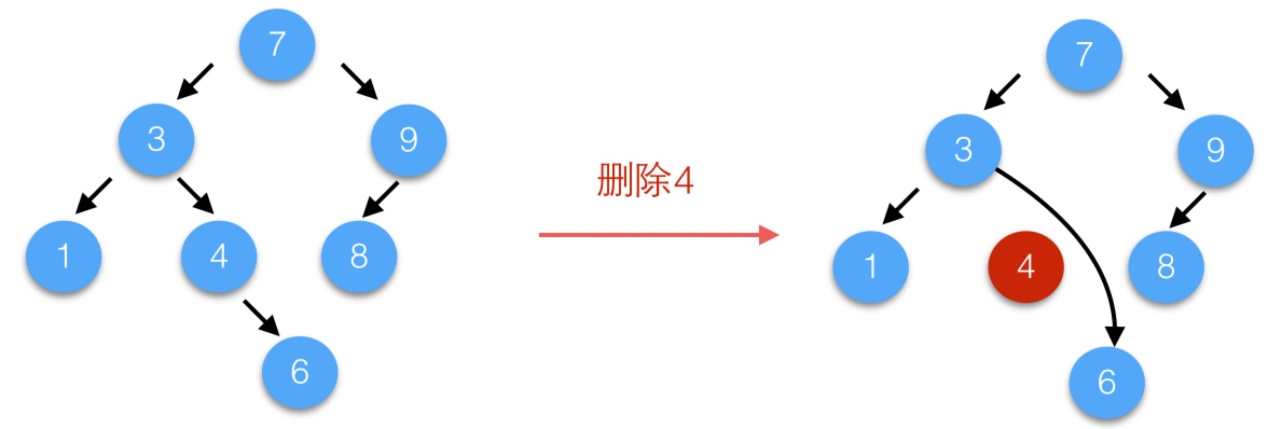
4.1 如果为叶子节点

直接修改对应节点为空就行。

4.2 如果结点有一个子结点

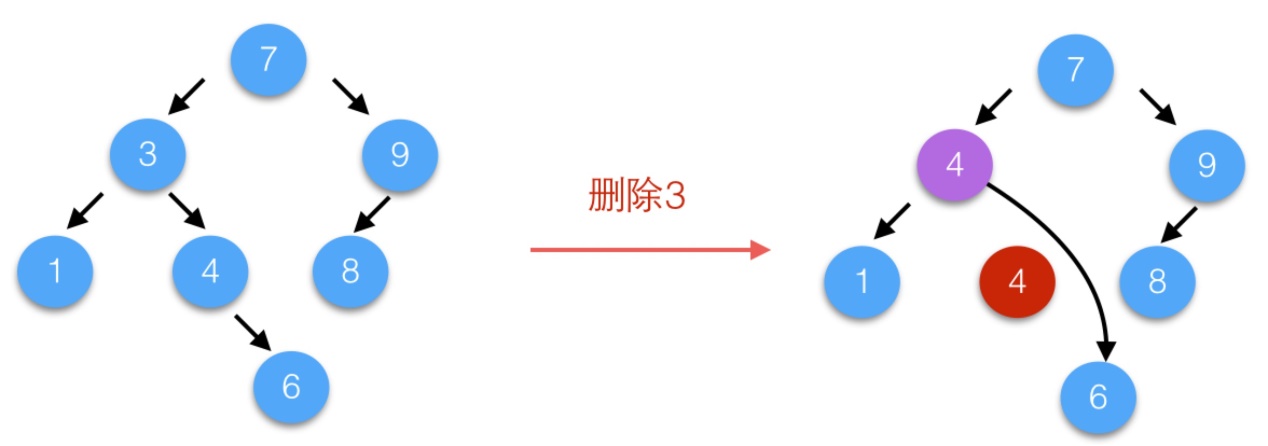
替换待删节点为孩子节点，或者说，在孩子节点和父节点之间直接建立链接。

(1) 如果要删除4，则将4的父结点3直接链接到4的子结点6。



4.3 如果结点有两个子结点

首先找该节点的后继（根据之前介绍的后继算法，**后继为右子树中最小的节点，这个后继一定没有左孩子**），找到后继后，替换待删节点为后继的内容，然后再删除后继节点。后继节点没有左孩子，这就将两个孩子的情况转换为了叶子节点或只有一个孩子的情况。

(1) 

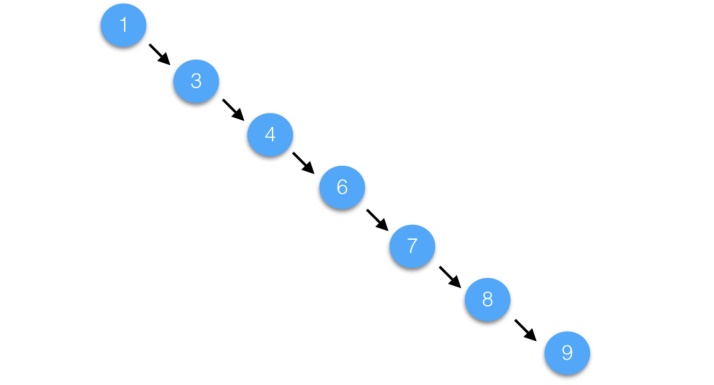
说明：

从左边二叉树中删除节点3，3有两个孩子，后继为4，首先替换3的内容为4，然后再删除节点4。

三、平衡的排序二叉树

1. 背景

极端情况下，排序二叉树可能退化为一个链表，如下：



退化为链表后，排序二叉树的优点就都没有了，即使没有退化为链表，如果排序二叉树高度不平衡，效率也会变的很低。

2. 平衡二叉树概念

(1) 高度平衡

平衡具体定义是什么呢？有一种高度平衡的定义，即任何节点的左右子树的高度差最多为一，满足这个平衡定义的排序二叉树又被称为AVL树。

(2) 大致平衡

① 在TreeMap的实现中，用的并不是AVL树，而是红黑树，与AVL树类似，红黑树也是一种平衡的排序二叉树，**也是在插入和删除节点时通过旋转操作来平衡的。**

② 红黑树是大致平衡的，大致就是确保对于任意一条从根到叶子节点的路径，没有任何一条路径的长度会比其他路径长过两倍。红黑树减弱了对平衡的要求，但降低了保持平衡需要的开销，在实际应用中，统计性能高于AVL树。

③ 红黑树名字由来。因为它对每个节点进行着色，颜色或黑或红，并对节点的着色有一些约束，满足这个约束即可以确保树是大致平衡的。

四、总结

1. 排序二叉树保持了元素的顺序，而且是一种综合效率很高的数据结构，基本的保存、删除、查找的效率都为O(h)，h为树的高度，在树平衡的情况下，h为log2(N)，N为节点数，比如，如果N为1024，则log2(N)为10。

2. 基本的排序二叉树不能保证树的平衡，可能退化为一个链表，有很多保持树平衡的算法，AVL树是第一个，能保证树的高度平衡，但红黑树是实际中使用更为广泛的，虽然只能保证大致平衡，但降低了维持树平衡需要的开销，整体统计效果更好。

3. 与哈希表一样，树也是计算机程序中一种重要的数据结构和思维方式。为了能够快速操作数据，哈希和树是两种基本的思维方式，不需要顺序，优先考虑哈希，需要顺序，考虑树。除了容器类TreeMap/TreeSet，数据库中的索引结构也是基于树的（不过基于B树，而不是二叉树），而索引是能够在大量数据中快速访问数据的关键。