二叉排序树

1. 属动态查找表 → 边查边插

i) 将给定的关键字在二叉排序上查找

ii) 找到 → 返回

未找到 → 将该关键字插入二叉排序树

特点：在插入和查找时，经中序遍历可得有序序列。

☆ 二分查找在查找时数据已经处于有序.

区别、二叉排序树面的数据集合无序，在查找和插入过程中建立有序

考点：

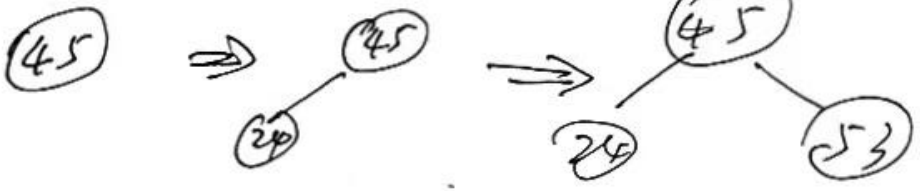
i> 生成二叉排序树

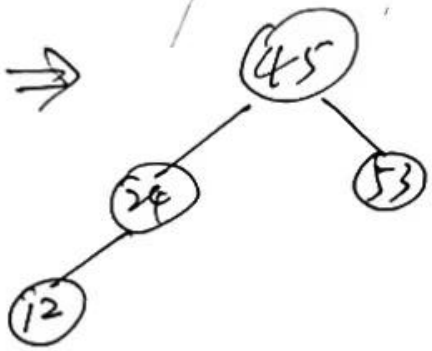
ii> 插入、删除.

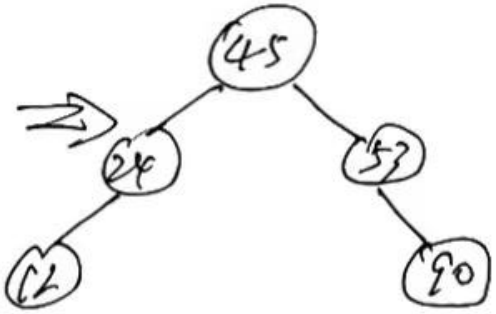
iii> 中序遍历可得有序序列.

iv> 特点.

例：有数据集合 {45, 24, 53, 45, 12, 24, 90}

初始：∅ ⇒ 

⇒ 

⇒ 

☆ 1> 每棵一个数据元素，即二叉排序树中每一个叶子结点

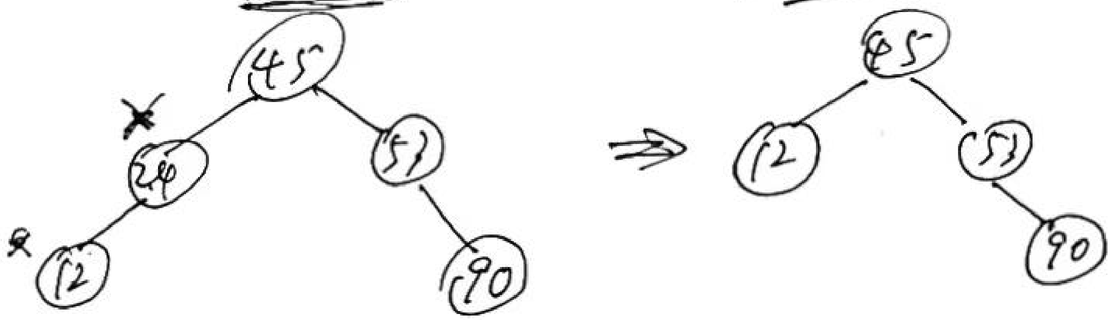
2> 中序遍历一定可以得到一个有序序列。

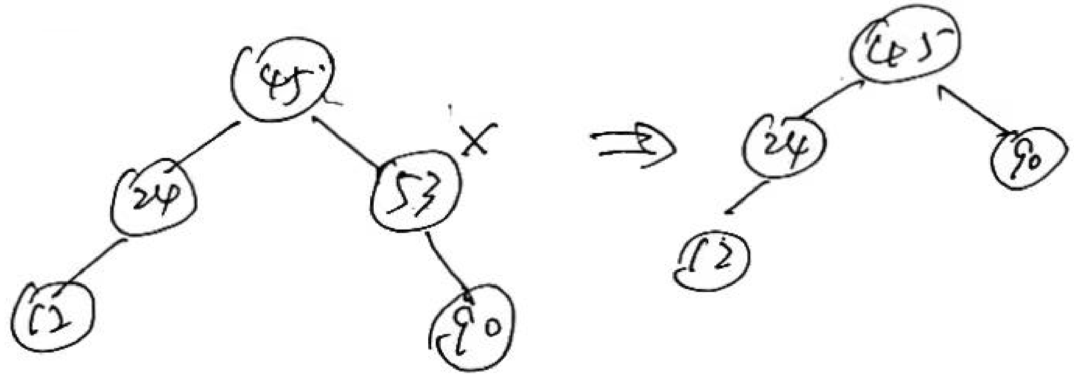
3> 每一个无序序列通过构造二叉排序树，变成一个有序序列。

二叉排序树的结点删除。（对BST删除后生成的新树依然是一棵二叉排序树）

① 删除叶子结点：直接删除（不影响二叉排序性）。

② 删除分支结点，
- 有左子女，左子女挂向父亲（删左挂左）
- 有右子女，右子女挂向父亲，（删右挂右）

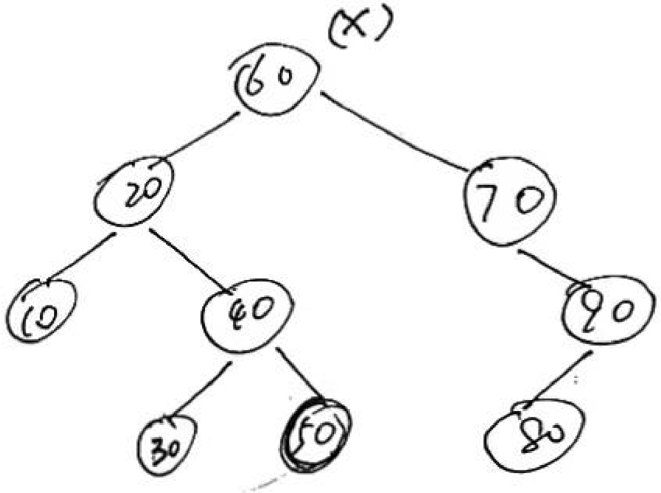




③ 删除左右子女双全的结点

Ⅰ 用其左子树中的直接前驱替代待删结点

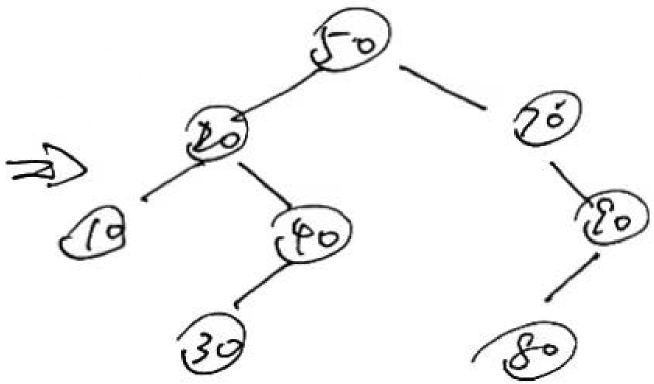
Ⅱ: 用……直接后继……



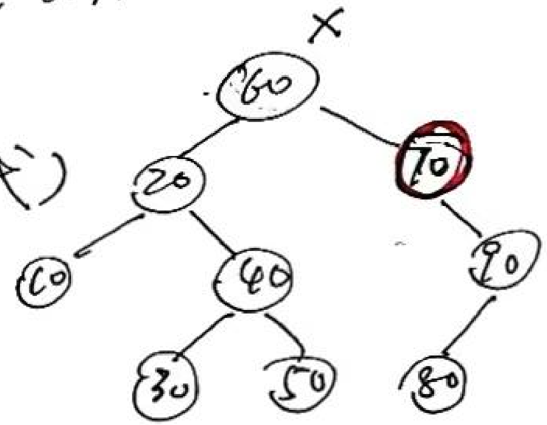
Ⅰ: 10 20 30 40 50 [60] 70 80 90

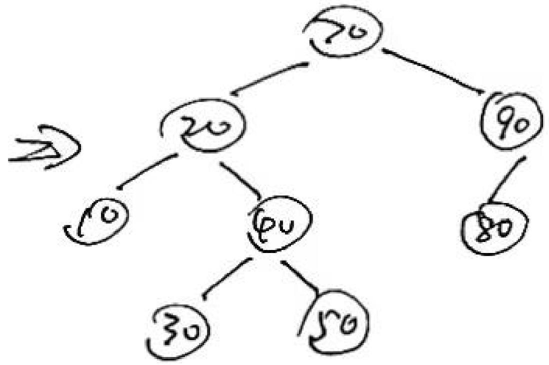
直接前驱　直接后继

（用直接前驱替代）

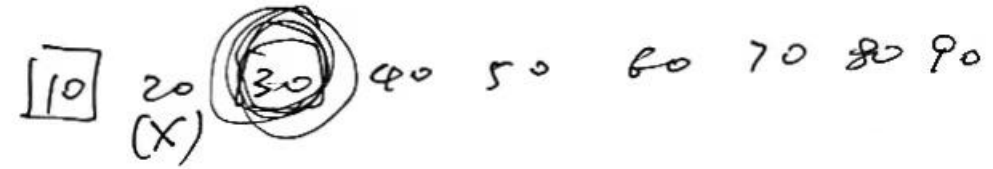


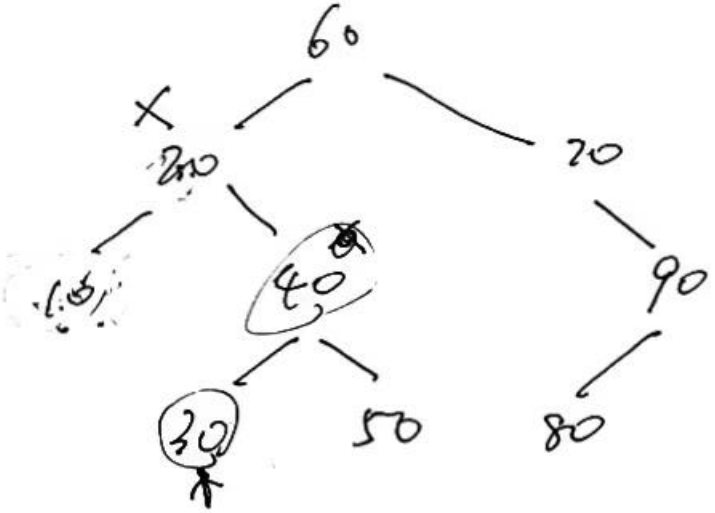
Ⅱ: （用直接后继替代）



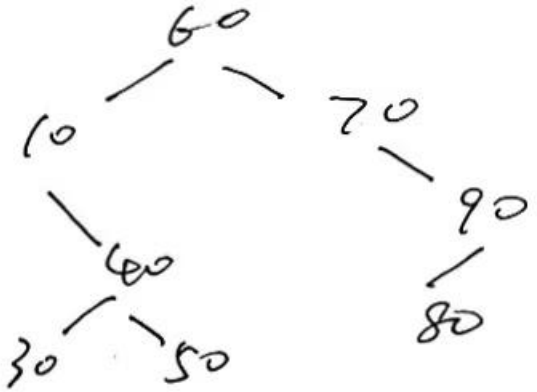


⑥

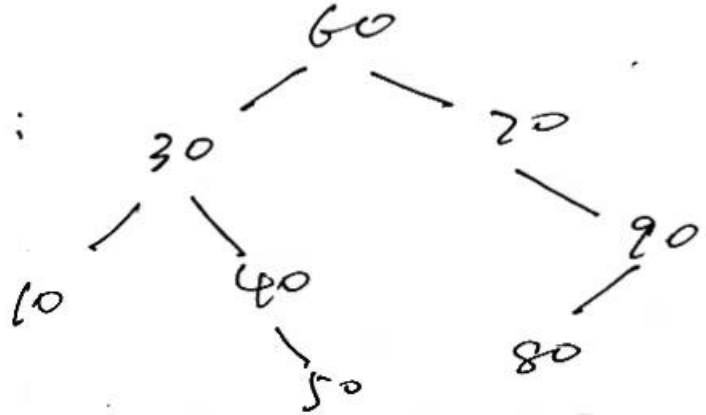

10
20
30
40
50
60
70
80
90
(X)

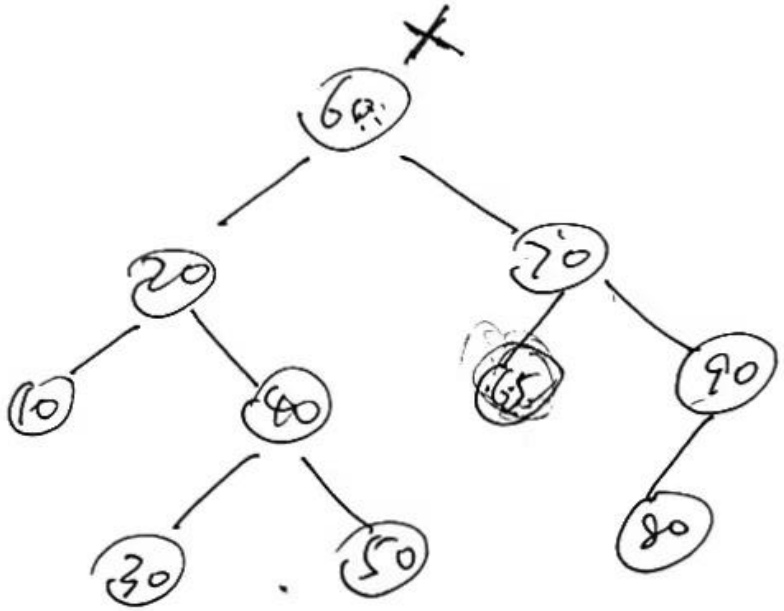

60
X
70
40
90
50
80

I:


60
10
70
90
80
30
50

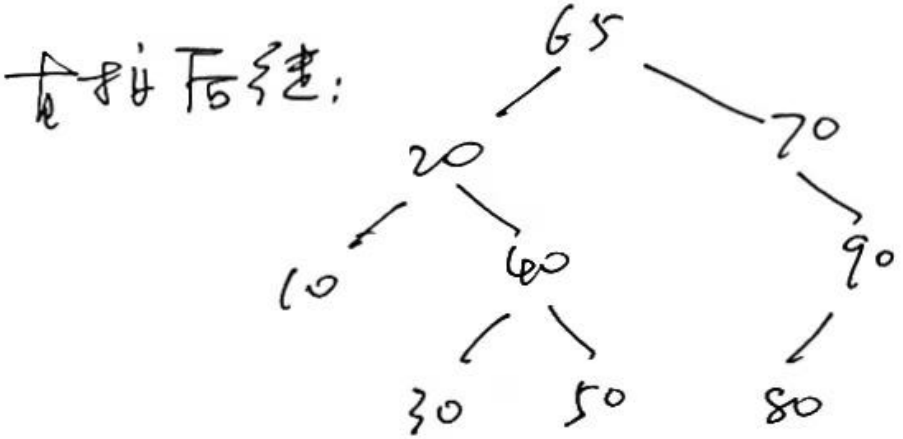
II:


60
30
70
10
40
90
80
50



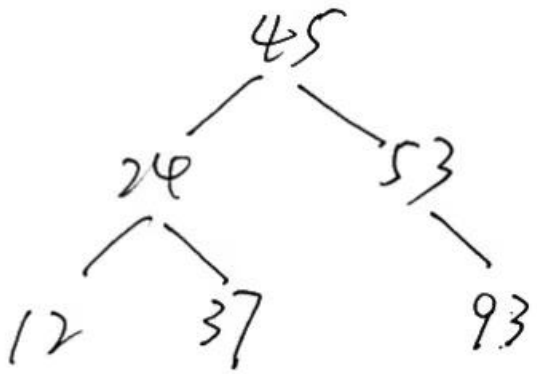
10 20 30 40 50 60 65 70 90 80

↑ (X)



二叉排序树的查找分析

例：{45, 24, 53, 12, 37, 93} ⇒ 最多3次即可找到.



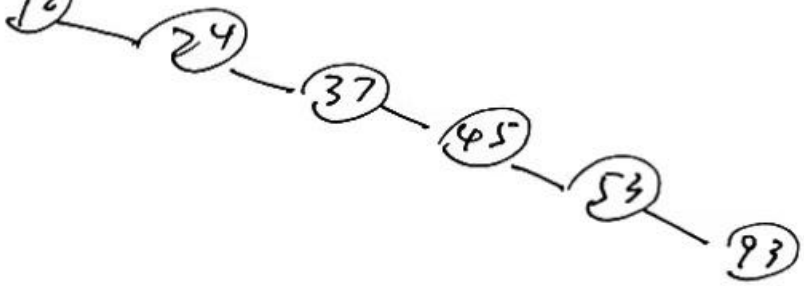
$$ASL = \frac{1\times1+2\times2+3\times3}{6} = \frac{1+4+9}{6} = \frac{7}{3}$$

时间复杂度：$O(\log_2 n)$.

※ 同样的数据元素，依然是序，但次序不同.

{12, 24, 37, 45, 53, 93}

$$ASL = \frac{1+2+3+4+5+6}{6} = \frac{21}{6} = \frac{7}{2}.$$



时间：$O(n)$

☆ 二叉排序树无法控制树的深度

导致查找ASL上升，效率下降。

优点：可以使一个无序序列在 $\log_2 n$ 级别上有序

→ 解决：压缩 BST 的深度

⇒ 无论数据集合中数据之间是什么次序，都可以使深度得到压缩

—— AVL（平衡二叉树）。