计算机领域本科教育教学改革试点
工作计划（“101计划”）研究成果

数据结构

俞勇、张铭、陈越、韩文弢

上海交通大学、北京大学、浙江大学、清华大学

第 11 章 查找

11.2 二叉查找树

林劼

电子科技大学

提纲

11.2.1　二叉查找树

11.2.2　二叉查找树-插入

11.2.3　二叉查找树-删除

11.2.4　查找性能分析

11.2.5　作业

11.2 二叉查找树

树的分类：

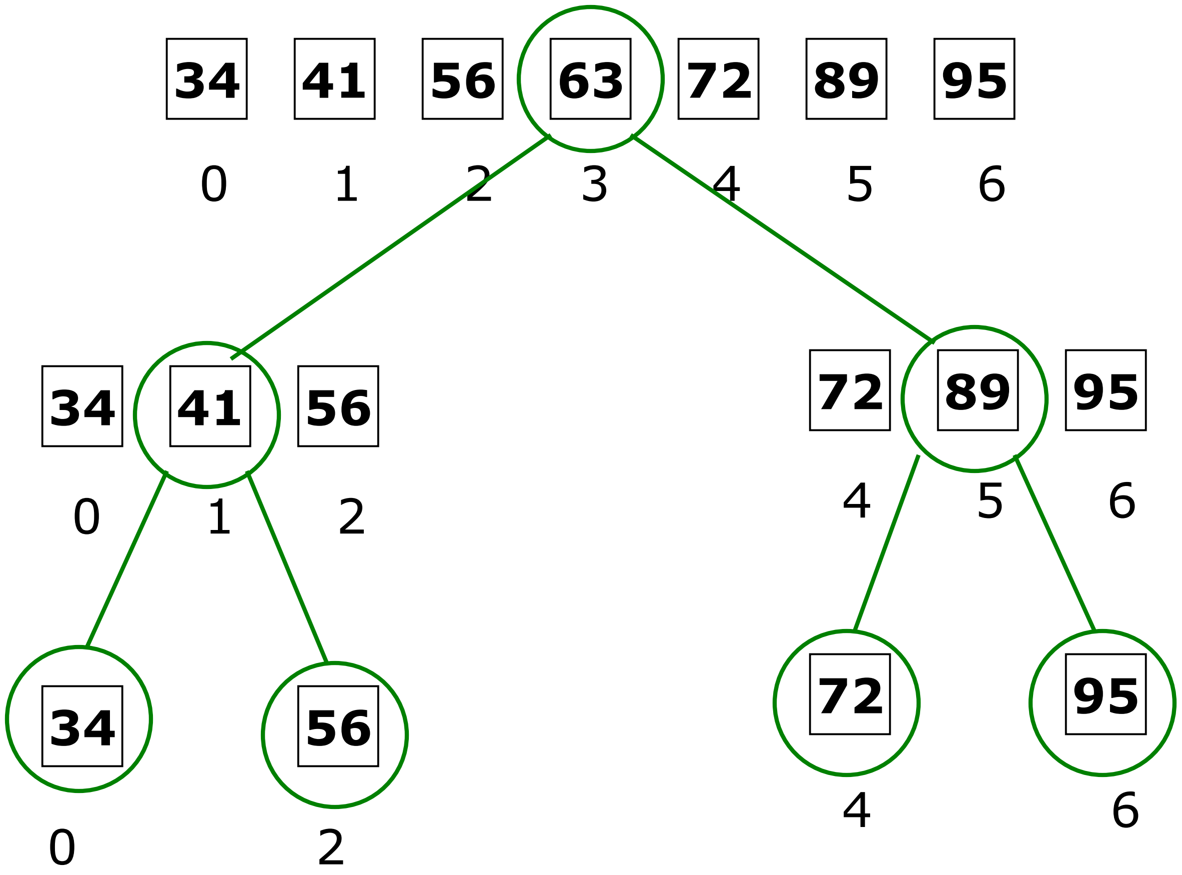
- **一般的树**：各结点可以有多个子结点，甚至子结点数量可以没有上限

- **二叉树**：各结点最多只有两个子结点

  ---**Huffman树：**带权路径长度最小，叶结点权重越大离根越近
  ---**堆**：各结点比其所有子结点大（最大堆）或小（最小堆）
  ---**二叉查找树**

11.2 二叉查找树

二分查找：在有序序列上查找数据，每次比较可以减少一半搜索范围！



可以用二叉树表达有序序列的逻辑结构：

- 如果结点的左子树非空，则左子树中所有结点的数据小于该结点数据
- 同样，如果结点的右子树非空，则右子树中所有结点的数据大于该结点

查找从根结点开始比较，执行以下操作：

- 如果结点存放的数据与查找数据相同，返回结点
- 如果查找的数据比结点小，若左子树是空树，查找失败；否则查找结点的左子树
- 相反，如果查找数据比当前结点大，走向右子树继续查找

11.2 二叉查找树

二分查找：在有序序列上查找数据，每次比较可以减少一半搜索范围！

- **查找元素**：时间复杂度O(log(n)) 效率高！
- **插入新元素**：时间O(n)
- **删除元素**：时间O(n)

---在有序序列中插入新元素或者删除元素，可能需要移动大量数据！

思考：如何存储一组数据，使得插入新元素以及删除元素的操作都能高效完成？

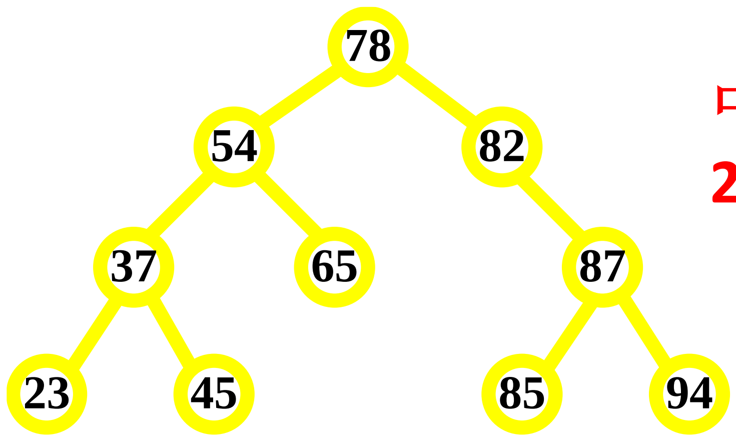
11.2 二叉查找树

二叉查找树（BST）

1. 定义

二叉查找树或者是一棵空树；或者是具有如下特性的二叉树：

- 若根结点的左子树不空，则左子树上所有结点的值均小于根结点的值；
- 若根结点的右子树不空，则右子树上所有结点的值均大于根结点的值；
- 左、右子树本身也是一棵二叉查找树。

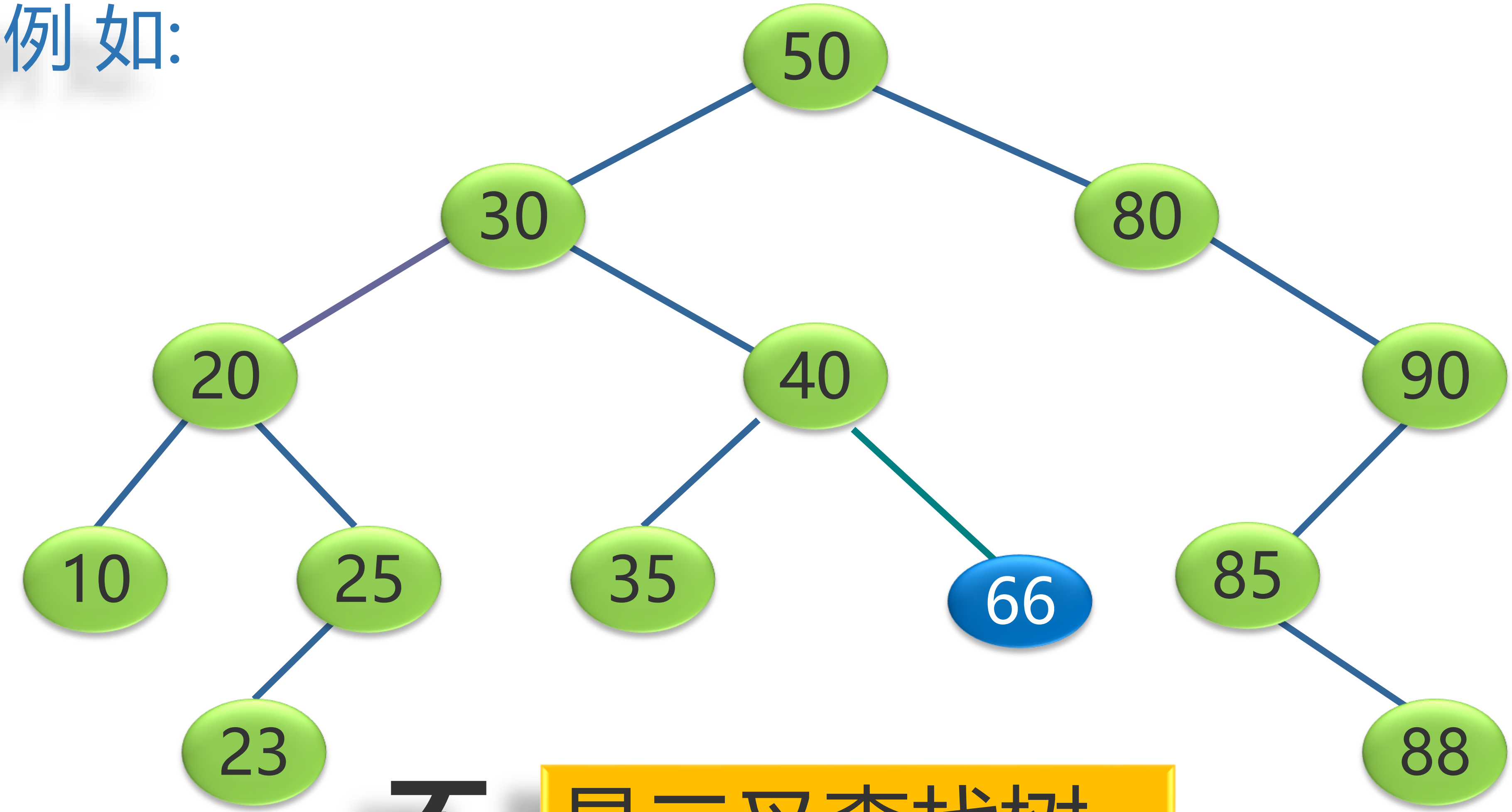


中序遍历序列：

23, 37, 45, 54, 65, 78, 82, 85, 87, 94

对BST中序遍历，输出的结点数据按升序排列！？

例如:



不是二叉查找树。

多选题 1分

设置

重构二叉查找树，只需要下面那种序列即可

A 前序遍历序列

B 中序遍历序列

C 后序遍历序列

D 层序遍历序列

提交

单选题 1分

设置

下面的序列是BTS的前序遍历结果，哪个正确？

A 6 5 3 4 2 1 9 8 7

B 6 5 4 3 2 1 8 9 7

C 6 4 3 2 1 5 7 9 8

D 6 4 3 5 1 2 7 8 9

提交

11.2 二叉查找树

二叉查找树 | 查找 --递归算法

算法：Search(bst, key)
输入：二叉查找树bst，数据key
输出：如果key在树中，返回结点；否则返回NIL

```
if bst=NIL 或 bst.data = key then
|   return bst    //空结点或者与结点数据相同，返回结点
end
if key < bst.data then
|   return Search(bst.left, key)  //递归查找左子树
else   //key > bst.data
|   return Search(bst.right, key)  //递归查找右子树
end
```

11.2 二叉查找树

二叉查找树 | 查找 --非递归算法（二分）

算法：Search(bst, key)
输入：二叉查找树bst，数据key
输出：如果key在树中，返回结点；否则返回NIL

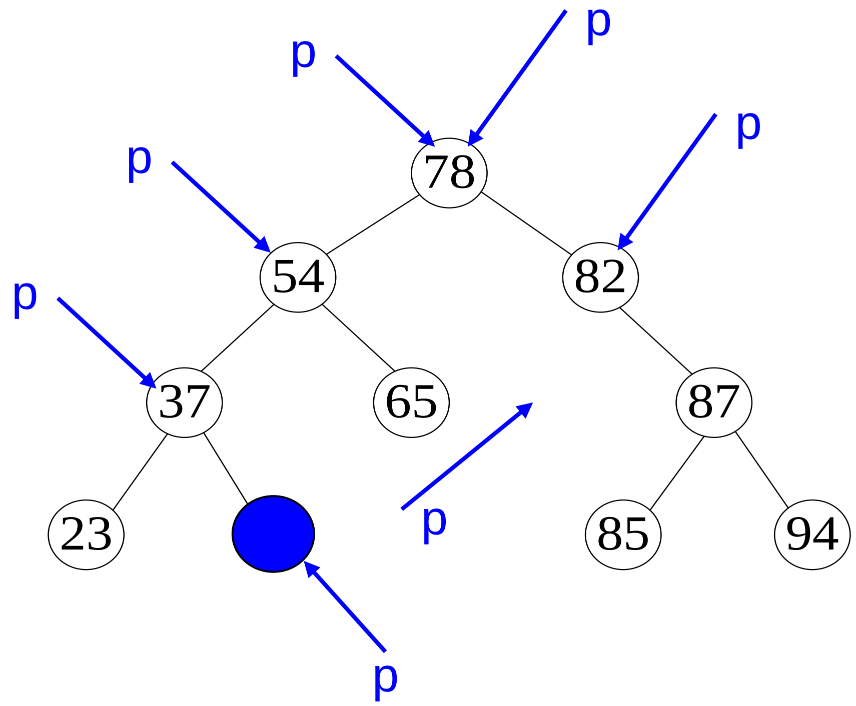
```
1. node_ptr ← bst
2. while node_ptr ≠ NIL 且 node_ptr.data ≠ key do
3. |   if key < node_ptr.data then
4. |   |   node_ptr ← node_ptr.left    //查找左子树
5. |   else    //key > node_ptr.data
6. |   |   node_ptr ← node_ptr.right   //查找右子树
7. |   end
8. end
9. return node_ptr
```

11.2 二叉查找树

二叉查找树 | 查找

key=45 ✔

key=81 ✘



11.2 二叉查找树

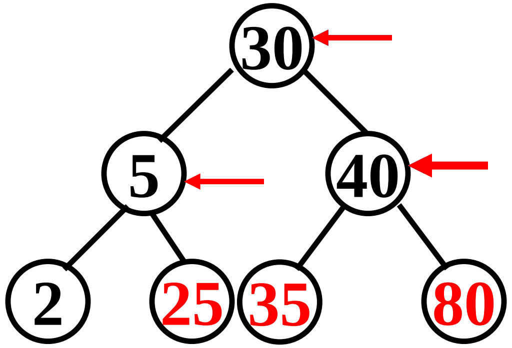
二叉查找树 | 插入

“插入”操作在**查找不成功**时才进行；

“插入”的数据存放在二叉查找树的新叶结点中；

11.2 二叉查找树

二叉查找树 | 插入



插入 80

① 检查 80 是否在树中

② 80 > 40, 所以必定应该是 40的右子树

插入 35

①检查35是否在树中

② 35 < 40，所以必定应该是 40的左子树

插入 25

①检查25是否在树中

② 25 > 5,所以必定应该是 5的右子树

11.2 二叉查找树

二叉查找树 | 插入 --递归算法

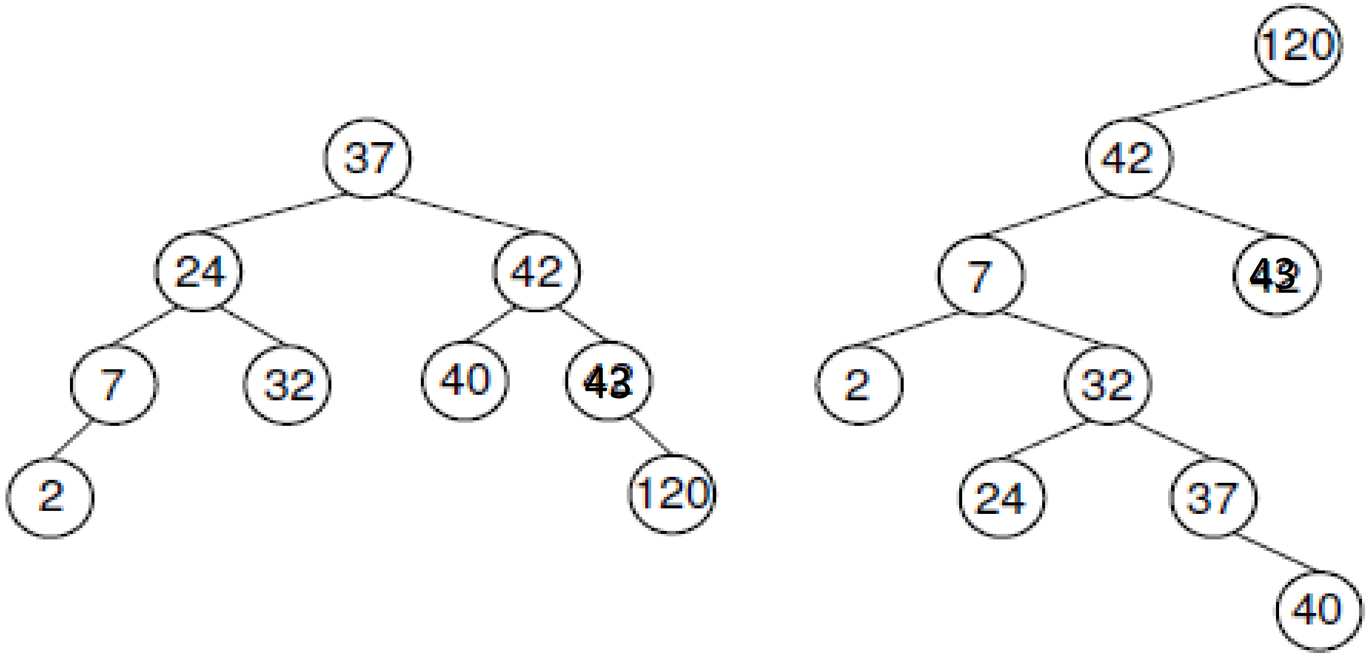
算法：Insert(bst, key)
输入：二叉查找树bst，数据key
输出：返回插入新结点后的树

```
if bst = NIL then  //空树
|   bst ← new BinaryLeafNode(key) //创建新叶结点，数据为key
else if key < bst.data then
|   bst.left ← Insert(bst.left, key)  //插入至左子树
else   //key > bst.data
|   bst.right ← Insert(bst.right, key)  //插入至右子树
end
return bst
```

```
node ← bst
father ← NIL   //结点node的父结点
while node ≠ NIL 且 node.data ≠ key do  //二分查找
|   father ← node      //node下移
|   if key < node_ptr.data then
|   |   node ← node.left   //查找左子树
|   else   //key > node_ptr.data
|   |   node ← node.right //查找右子树
|   end
end
if node = NIL then //key不在树中
|   node ← new BinaryLeafNode(key) //创建新叶结点，数据为key
|   if father = NIL then   //bst是空树
|   |   bst ← node            //新建结点成为树根
|   else if key < father.data then  //father.left一定NIL (?)
|   |   father.left ← node //新叶结点成为father的左子结点
|   else  // key > father.data 且 father.right = NIL
|   |   father.right ← node //新叶结点成为father的右子结点
|   end
end
return bst
```

11.2 二叉查找树

二叉查找树 | 插入



(a) (b)

BST的结构取决于元素插入的顺序！

单选题 1分

设置

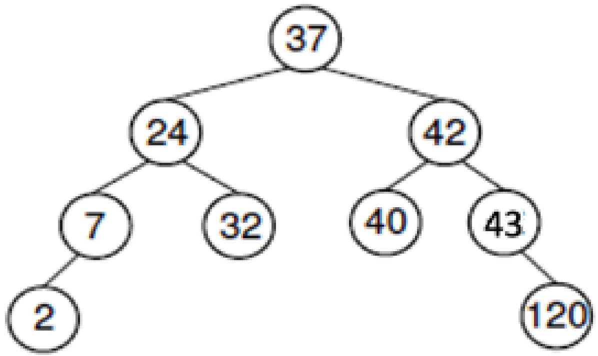
将序列中的元素依次插入空BTS中，右边的二叉查找树可由下面哪个序列生成?

A 37, 42, 120, 43, 7, 32, 24, 2, 40

B 37, 42, 40, 24, 2, 43, 120, 7, 32

C 37, 24, 42, 32, 7, 2, 120, 43, 40

D 37, 42, 43, 24, 7, 120, 2, 40, 32



提交

11.2 二叉查找树

二叉查找树 | 删除

删除可分**三种情况**讨论：

（1）被删除的结点是叶子

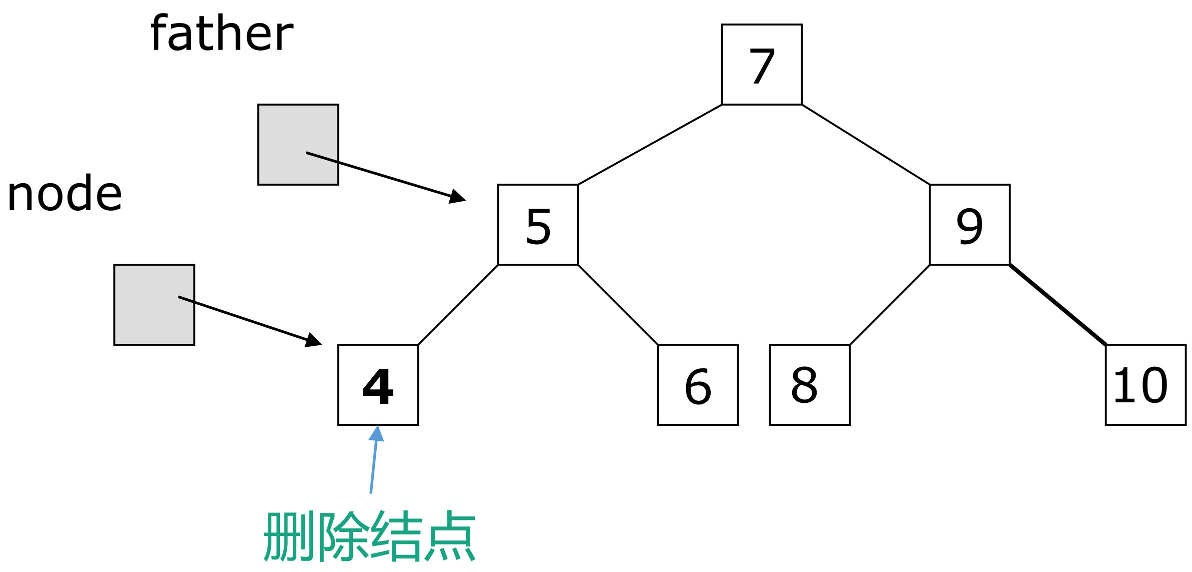
（2）被删除的结点只有左子结点或者只有右子结点

（3）被删除的结点既有左子结点，也有右子结点

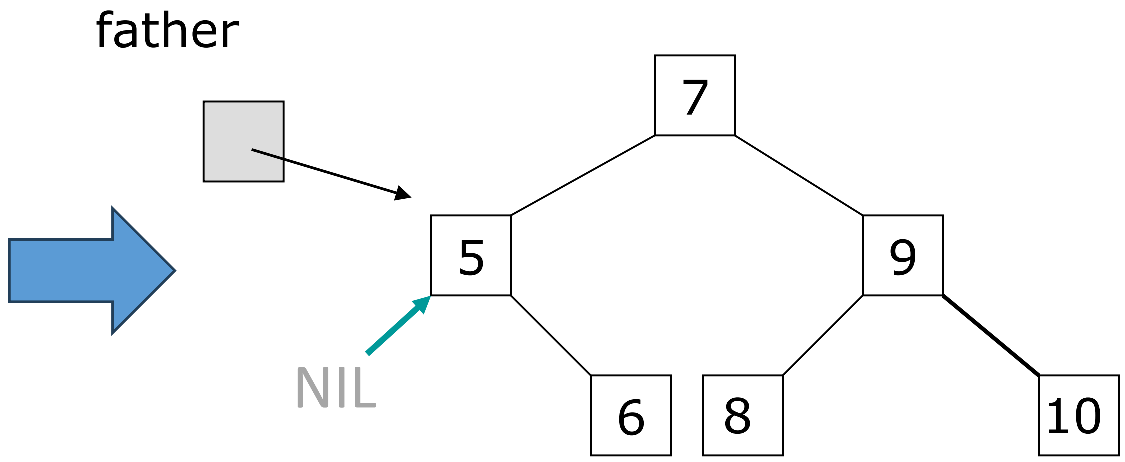
11.2 二叉查找树

二叉查找树 | 删除

情形一：被删除的结点是叶子

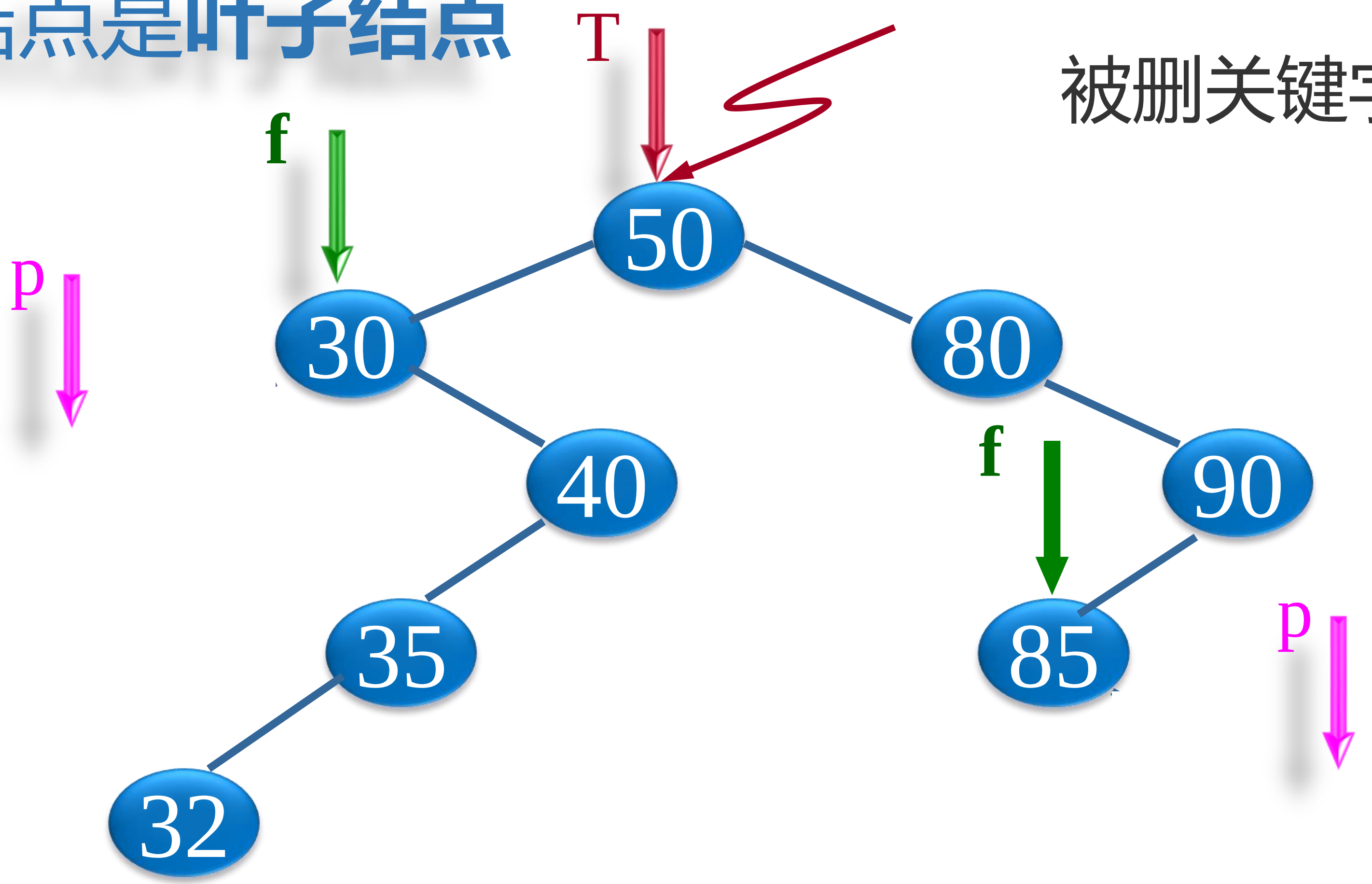


通过二分查找找到删除结点



去掉结点4，父结点的对应指针赋空

（1）被删除的结点是**叶子结点**



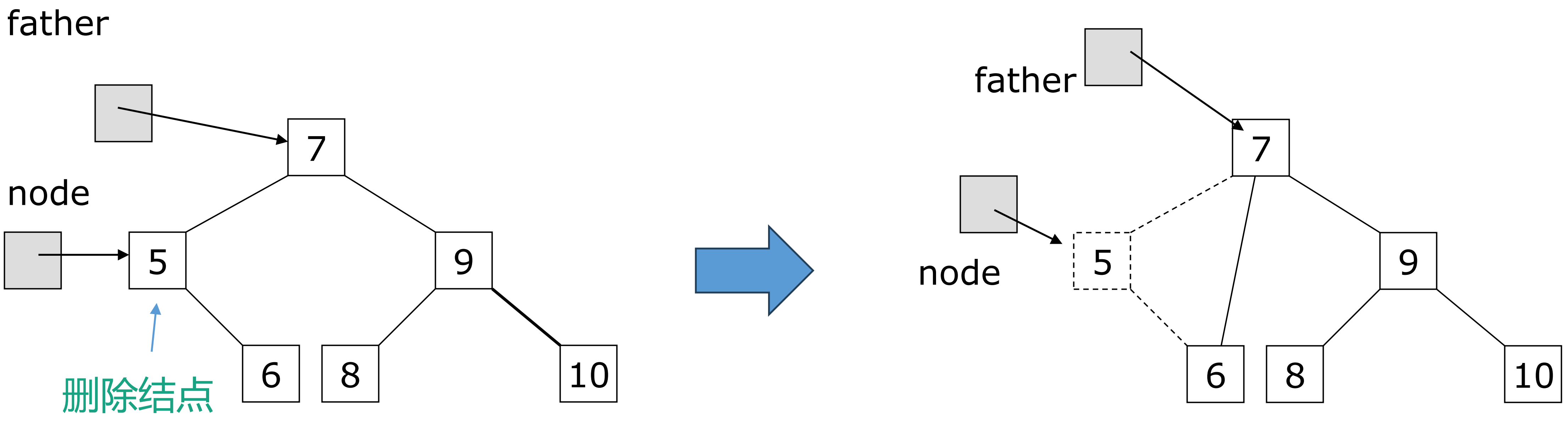
被删关键字 = **88**

其父结点中相应指针域的值改为“空”

11.2 二叉查找树

二叉查找树 | 删除

情形二：被删除的结点只有左子结点或者只有右子结点



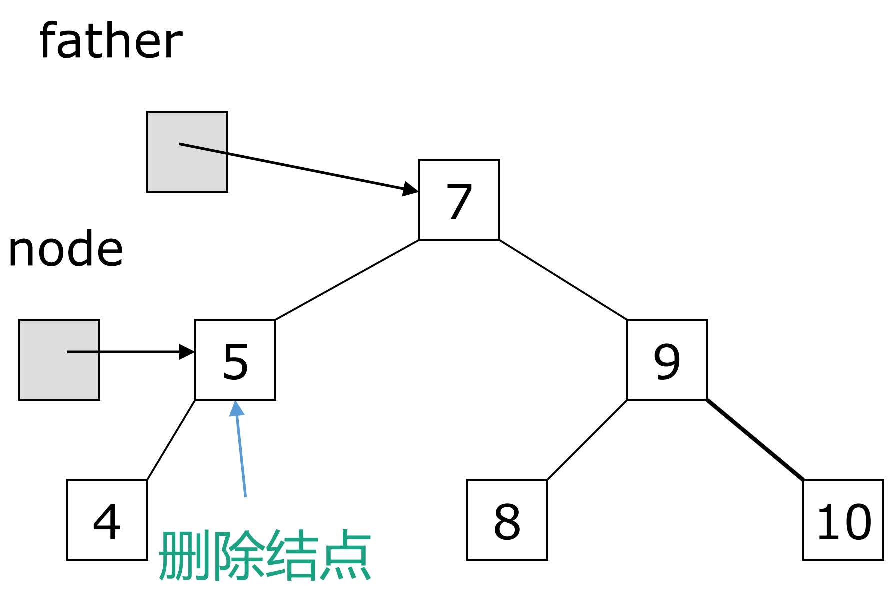
通过二分查找找到删除结点

去掉结点5，子结点成为其父结点的孩子

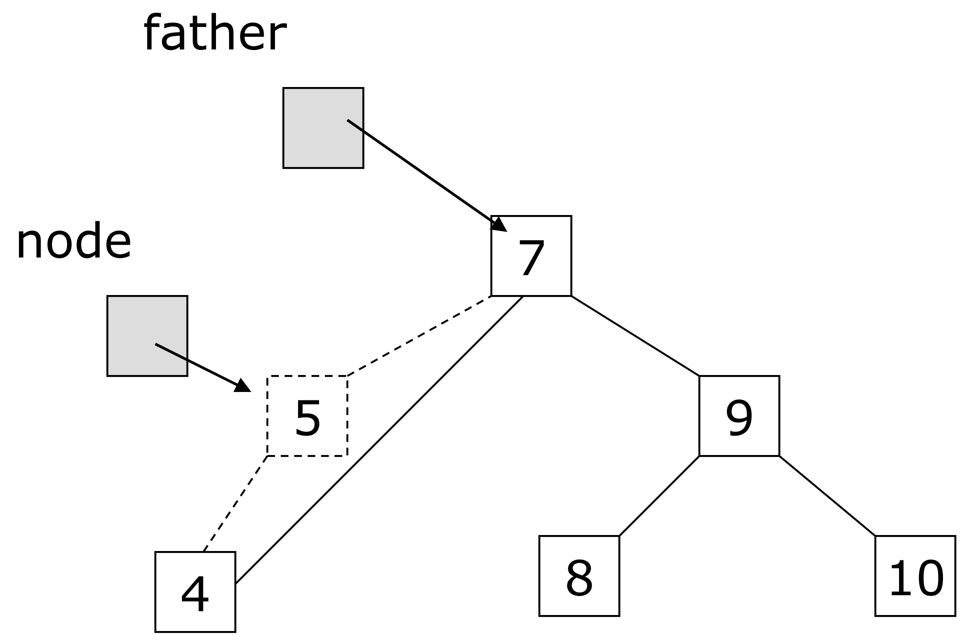
11.2 二叉查找树

二叉查找树 | 删除

情形二： 被删除的结点只有左子结点或者只有右子结点



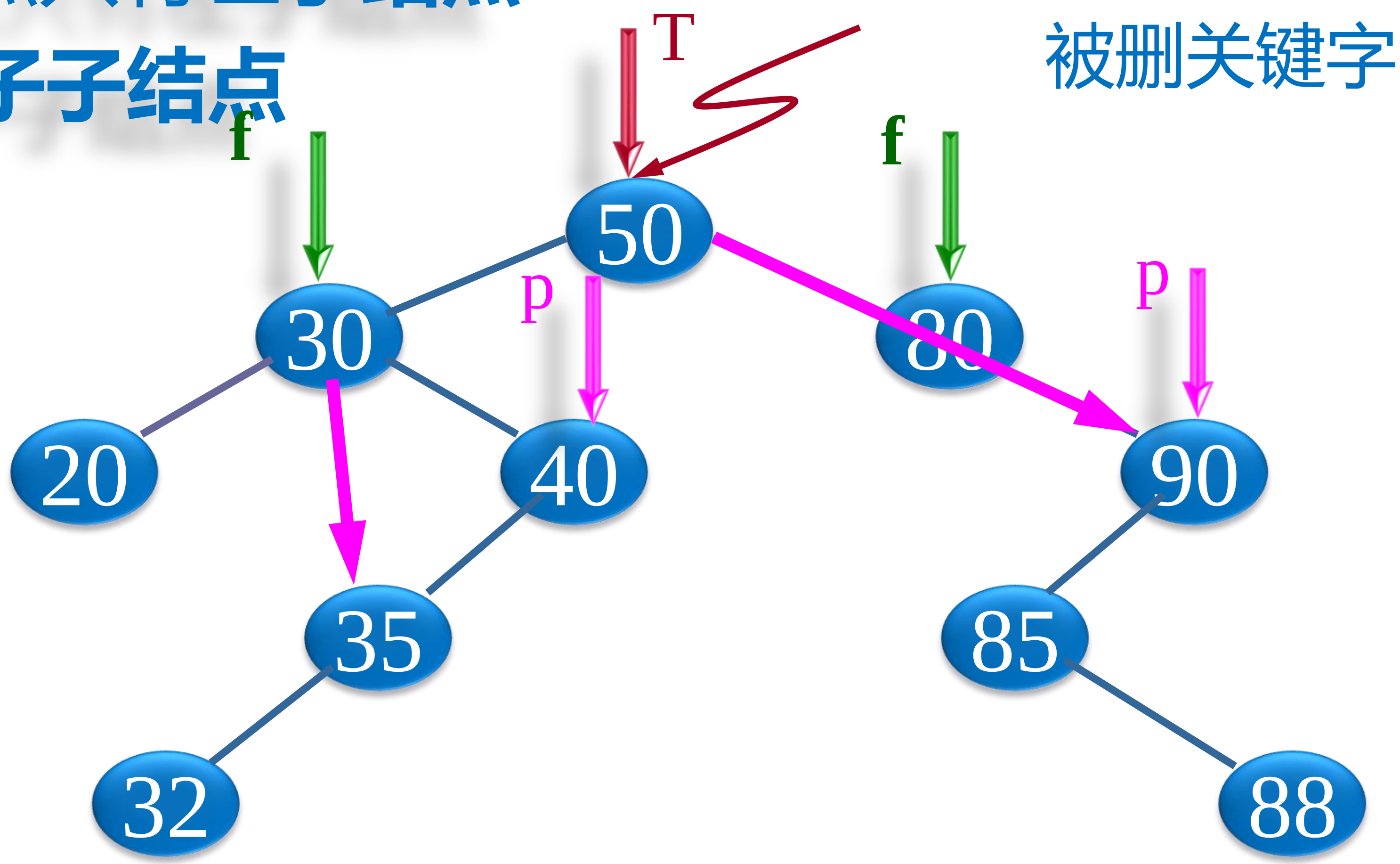
通过二分查找找到删除结点



去掉结点5，子结点成为其父结点的孩子

（2）被删除的结点只有**左子结点**或者只有**右子子结点**

被删关键字 = **80**



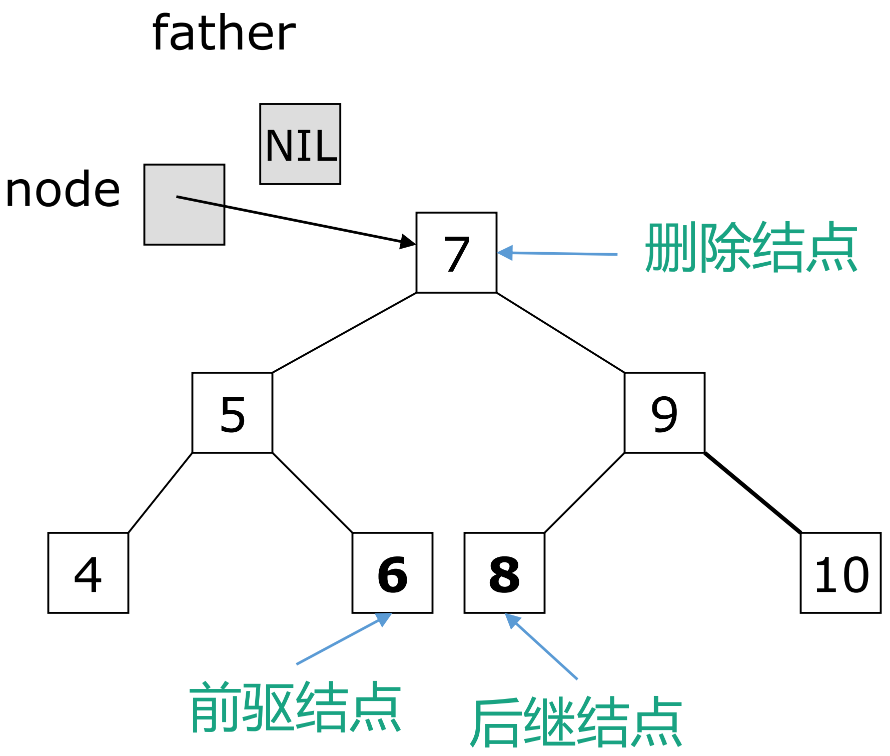
其父结点的相应指针域的值改为 “指向被删除结点的左子树或右子树” 。

11.2 二叉查找树

二叉查找树 | 删除

思考：
1. 替换结点有什么特征?
2. 如何找到替换结点

情形三：被删除的结点既有左子结点，也有右子结点

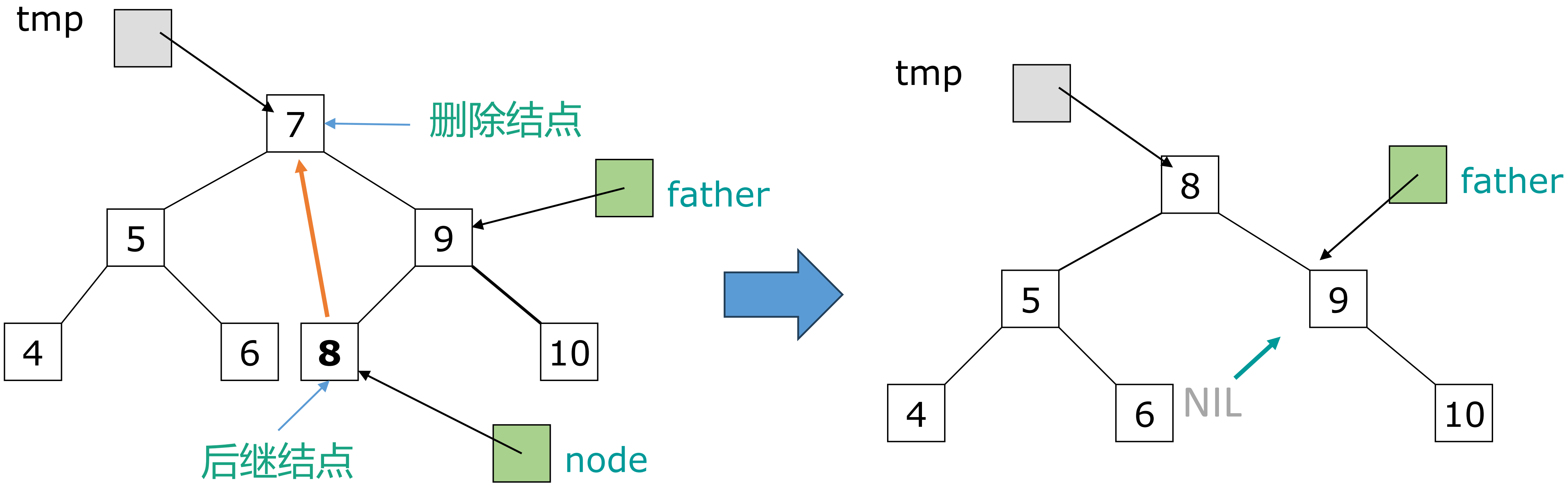


原则：
- 不直接删除有双子结点的结点
- 找到替换结点：左子树中的最大值（前驱）或 右子树中的最小值（后继）
- 将替换结点的数据代入删除结点，然后删除替换结点

11.2 二叉查找树

二叉查找树 | 删除

情形三：被删除的结点既有左子结点，也有右子结点



找到后继结点，将数据代入删除结点

删除后继结点

二叉查

算法：Removal(bst, key)

- 输入：二叉查找树bst，数据key
- 输出：删除带有数据key的结点，返回BST

```
node ← bst
father ← NIL   //结点node的父结点
while node ≠ NIL 且 node.data ≠ key do        //二分查找
|   father ← node      //node下移
|   if key < node.data then
|   |   node ← node.left    //查找左子树
|   else   //key > node.data
|   |   node ← node.right //查找右子树
|   end
end
if node = NIL then //key不在树中，直接返回
|   return bst
end
if node.left ≠ NIL 且 node.right ≠ NIL then  //删除结点的左右子树非空
|   tmp ← node                //用tmp指向删除结点
|   father ← node
|   node ← node.right   //走到右子树，查找后继结点
|   while node.left ≠ NIL  do
|   |  father ← node
|   |  node ← node.left     //移到左分支，直到左子树变空 (?)
|   end    //循环结束时，node指向后继结点
end
tmp.data ← node.data    //后继结点的数据代入删除结点
```

11.2 二叉查找树

二叉查找树 | 删除 --非递归算法

算法：Removal(bst, key)

- 输入：二叉查找树bst，数据key
- 输出：删除带有数据key的结点，返回BST

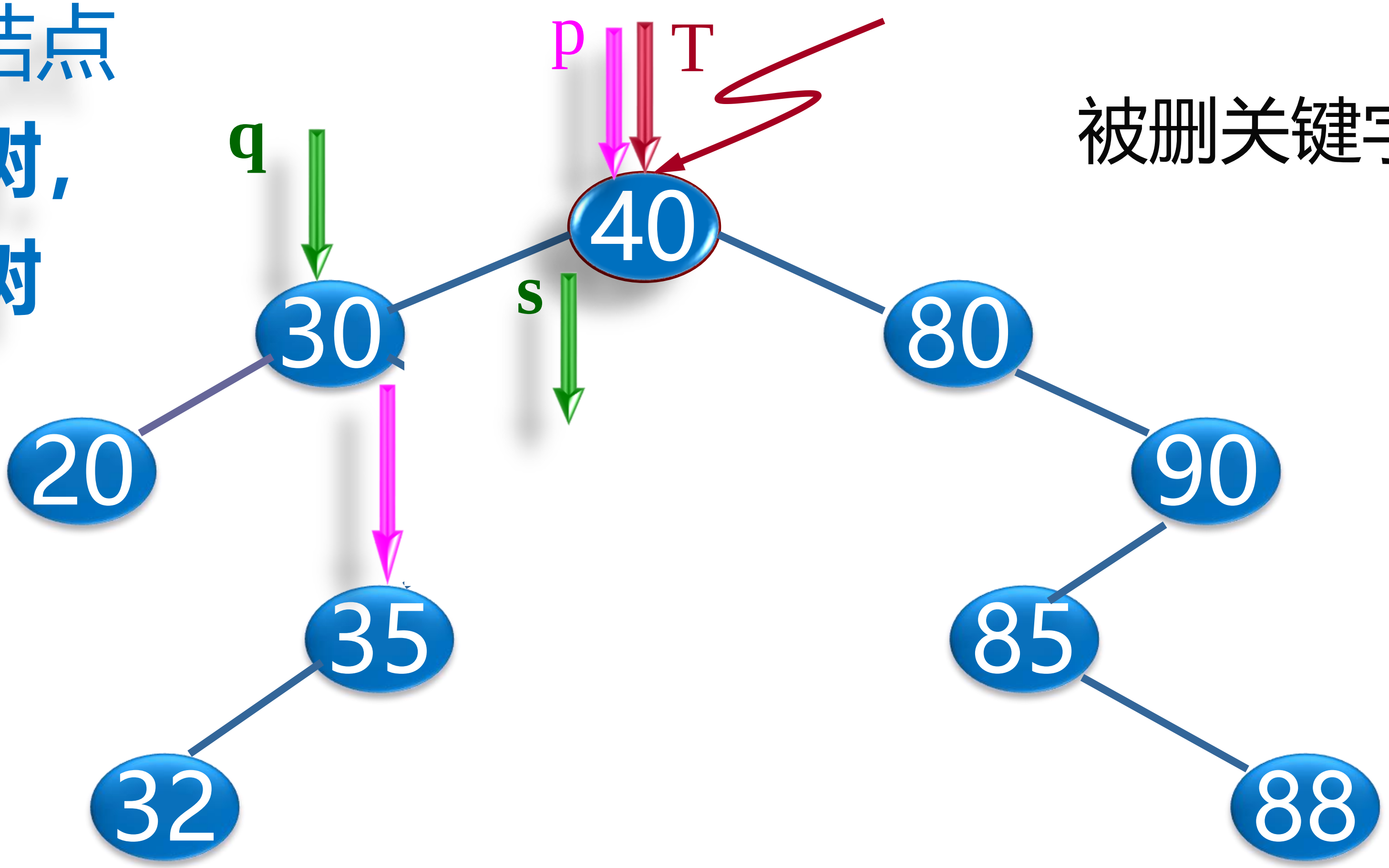
```
            //删除结点node，且该结点最多只有一个非空子树(?)
node_cld ← NIL      //如果node有非空子树，用node_cld指向子结点
if node.left ≠ NIL then      //左子树非空
|  |  node_cld ← node.left   //node_cld指向左子树
|  else
|  |  node_cld ← node.right
|  end
end

if father = NIL  then //删除结点是根结点，即node = bst
|  bst ← node_cld  //node_cld成为新的树根
else if father.left = node then  //删除结点在父结点左边
|  father.left ← node_cld     //node_cld成为父结点的左子结点
else       // father.right = node
|  father.right ← node_cld //node_cld成为父结点的右子结点
end
return bst    //返回删除结点后的查找二叉树
```

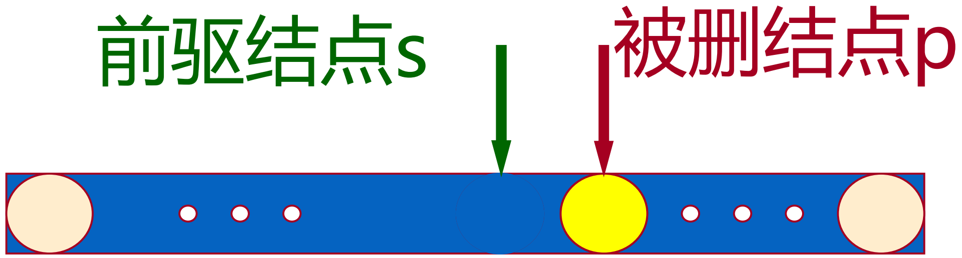
(3) 被删除的结点

既有左子树，

也有右子树

被删关键字 = 50





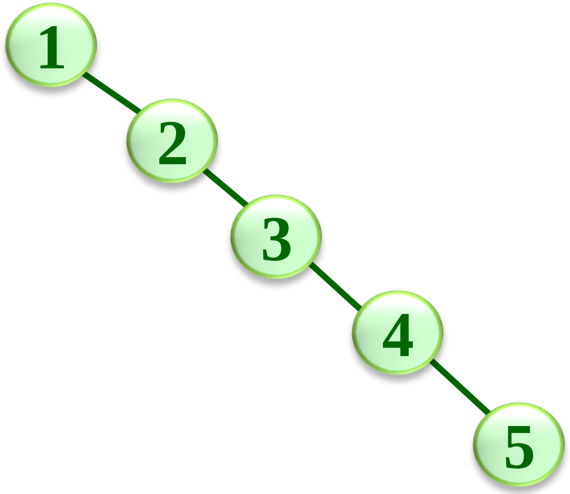
以其前驱替代之，然后再删除该前驱结点

11.2.4查找性能的分析

对于一棵特定的二叉排序树，均可按照平均查找长度的定义来求它的 *ASL* 值，*n* 个关键字，由于查找顺序不同可构造出不同形态的多棵二叉排序树，其平均查找长度的值不同，甚至可能差别很大。

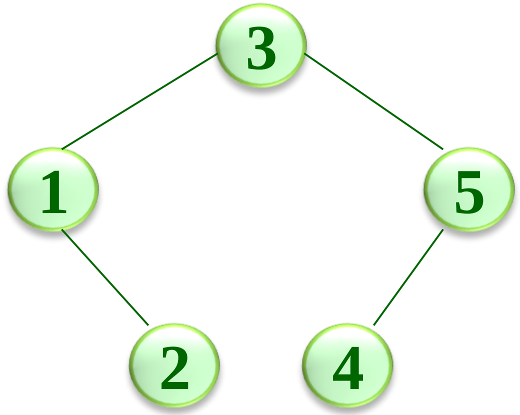
查找性能的分析

例如：由关键字序列 1，2，3，4，5构造而得的二叉排序树，



ASL =（1+2+3+4+5）/ 5
= 3

由关键字序列 3，1，2，5，4构造而得的二叉排序树，



ASL =（1+2+3+2+3）/ 5
= 2.2

二叉查找树常见面试题

1. 给定一个整数数组A[1..n]，按要求返回一个新数组 *counts[1..n]*。数组 *counts* 有该性质： counts[i] 的值是 A[i] 右侧小于 A[i] 的元素的数量。

示例:

输入: [5,2,6,1]

输出: [2,1,1,0]   hint： 从后往前

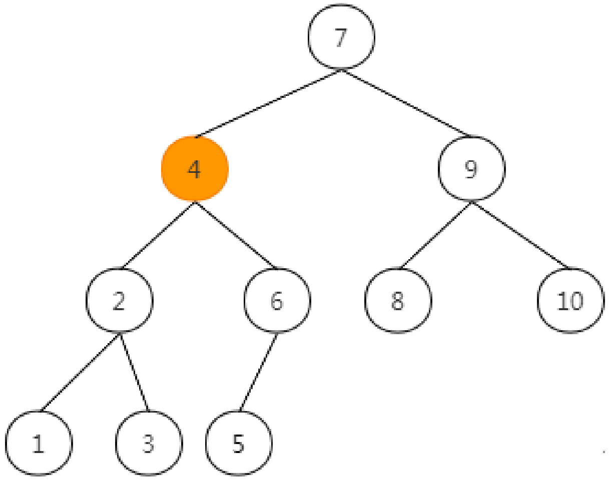
2.给定一个二叉查找树, 找到该树中两个指定节点的最近公共祖先。

3.给定一个二叉树，判断其是否是一个有效的二叉查找树。

4. 查找二叉查找树的第k小元素

11.2.5 作业

1、请画出在下图所示二叉查找树中删除结点4之后的二叉查找树。



谢谢观看