二叉查找树

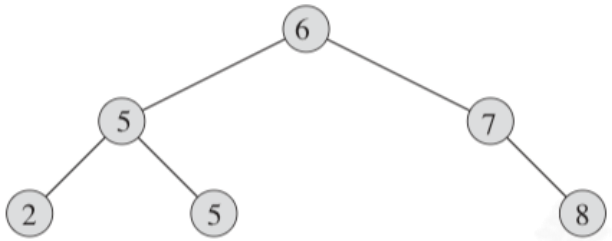
主讲人：邓哲也

查找树

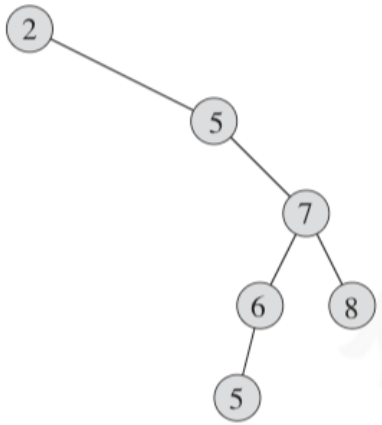
- 查找树（Search Tree）是一种数据结构，它支持多种动态集合操作，包括查找、返回最小值、返回最大值、返回前驱和后继节点、插入和删除。
- 它既可以用作字典，也可以用做优先队列。

二叉查找树

- 如图所示，一颗二叉查找树是按二叉树结构来组织的。
- 这样的树可以用链表结构表示，其中每一个节点都是一个对象。
- 节点中包含 key，left，right 和 parent。
- 如果某个儿子节点或父节点不存在，则相应域中的值即为 NULL。



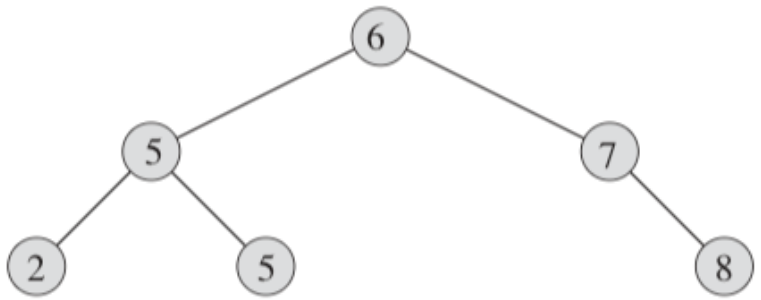
(a)



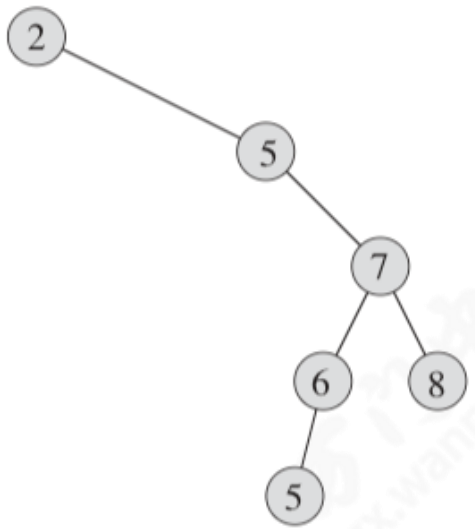
(b)

二叉查找树的性质

- 设 x 为二叉查找树中的一个节点。
- 如果 y 是 x 的左子树的一个节点，则 key[y] <= key[x].
- 如果 y 是 x 的右子树的一个节点，则 key[x] <= key[y].



(a)



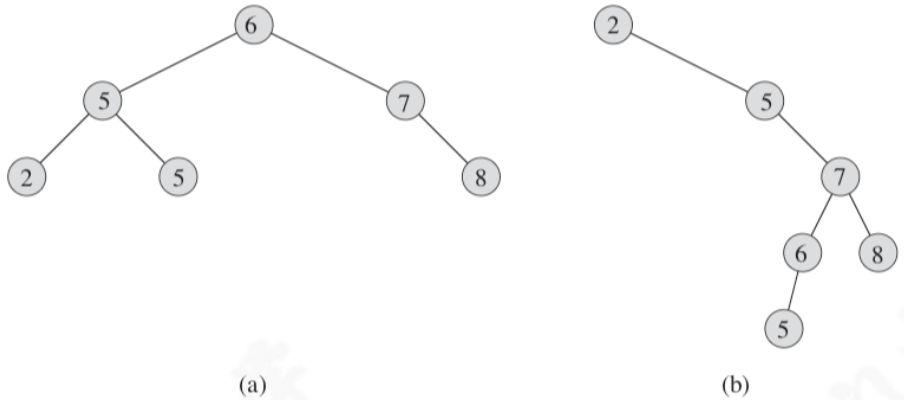
(b)

二叉查找树的中序遍历

根据二叉查找树的性质，可以用一个递归算法按排列顺序输出树中的所有关键字。这种算法称为中序遍历算法。

因为一子树根的关键字在输出时介于左子树和右子树的关键值之间。

```
inorder(x):
    if x != NULL:
        inorder(left[x])
        print key[x]
        inorder(right[x])
```



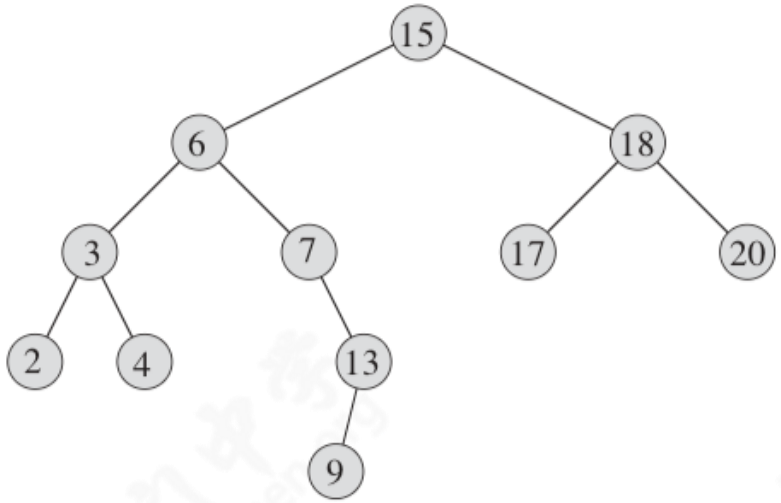
只要调用inorder(root[T])，就可以输出一颗二叉查找树 T 中的全部元素。

二叉查找树的查询操作

给定指向树根的指针和关键字 k，过程 Tree-Search 返回指向包含关键字 k 的节点（如果存在的话）的指针；否则，返回 NULL

```
Tree-Search(x, k)
    if x == NULL or k == key[x]:
        return x
    if k < key[x]:
        return Tree-Search(left[x], k)
    else
        return Tree-Search(right[x], k)
```

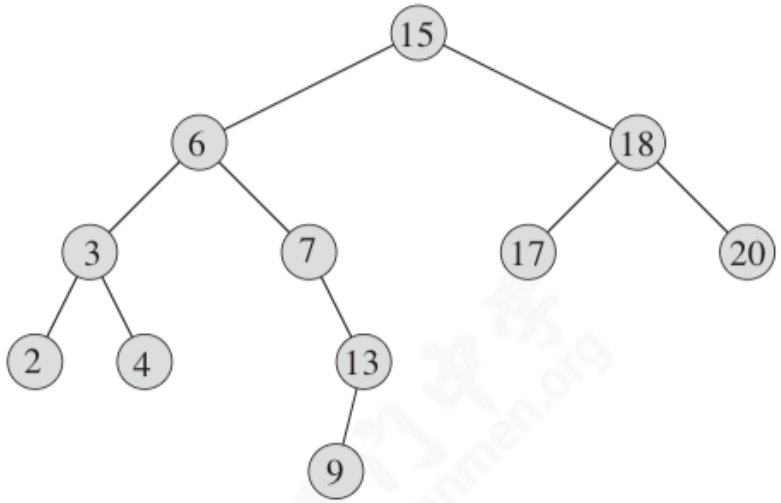
试试模拟一下 Tree-Search(root, 13)



二叉查找树的最小元素

要查询二叉树中具有最小关键字的元素，只要从根节点开始，沿着各节点的 left 指针查找下去，直到遇到 NULL 为止。

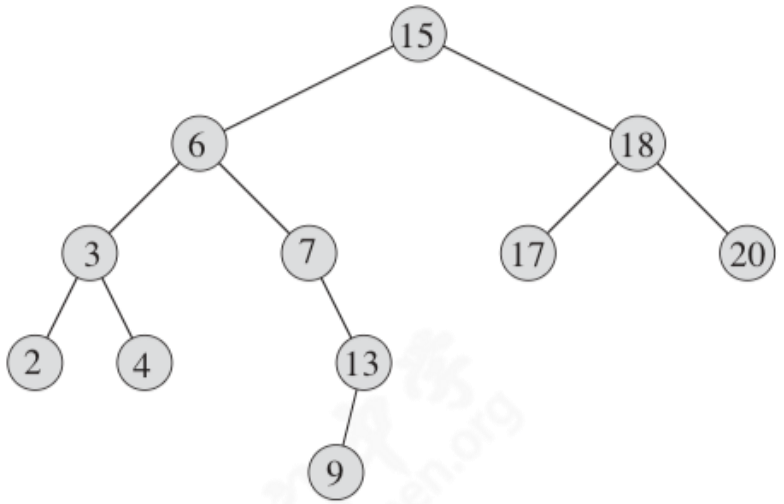
```
Tree-Minimum(x):
    while left[x] != NULL:
        x = left[x]
    return x
```



二叉查找树的最大元素

要查询二叉树中具有最小关键字的元素，只要从根节点开始，沿着各节点的 right 指针查找下去，直到遇到 NULL 为止。

```
Tree-Maximum(x):
    while right[x] != NULL:
        x = right[x]
    return x
```



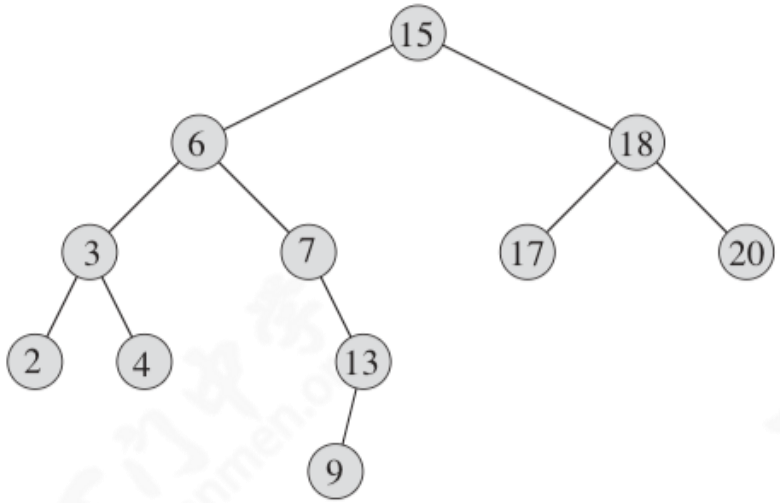
二叉查找树的前驱和后继

给定一个二叉查找树中的节点，有时候要求找出在中序遍历顺序下它的后继。

如果所有的关键字均不相同，则某一结点 x 的后继即具有大于 key[x] 中的关键字中最小者的那个节点。

根据二叉查找树的结构，不用对关键字做任何比较。

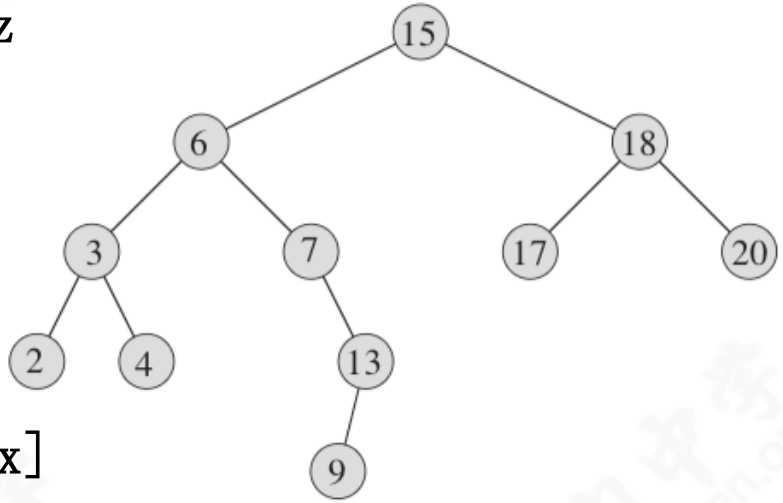
```
Tree-Successor(x):
    if right[x] != NULL:
        return Tree-Minimum(right[x])
    y = parent[x]
    while y != NULL and x = right[y]:
        x = y
        y = parent[y]
    return y
```



二叉查找树的插入

为将一个新值 v 插入到二叉查找树 T 中，可以调用Tree-Insert。

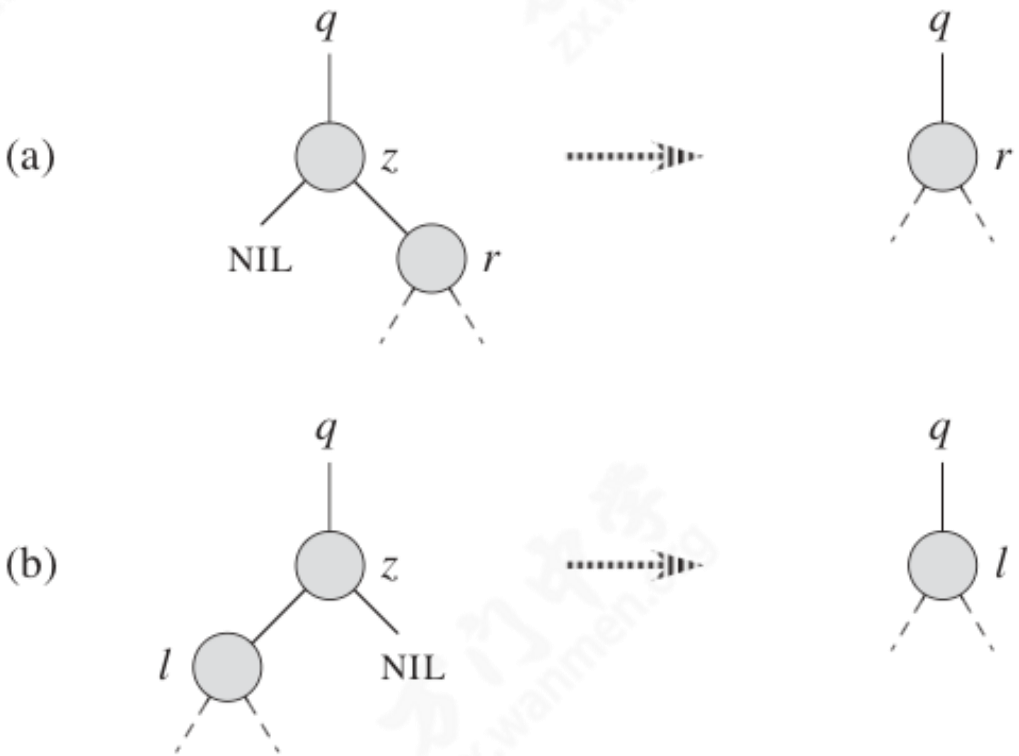
传给该过程的参数是个节点 z，并且有 key[z] = v, left[z] = NULL, right[z] = NULL，该过程修改 T 和 z 的某些域，并把 z 插入到树中合适的位置上。




```
Tree-Insert(T, z):
    y = NULL, x = root[T]
    while x != NULL:
        y = x
        if (key[z] < key[x]) x = left[x]; else x = right[x]
    parent[z] = y
    if y == NULL then root[T] = z
        else if key[z] < key[y] then left[y] = z
        else right[y] = z
```

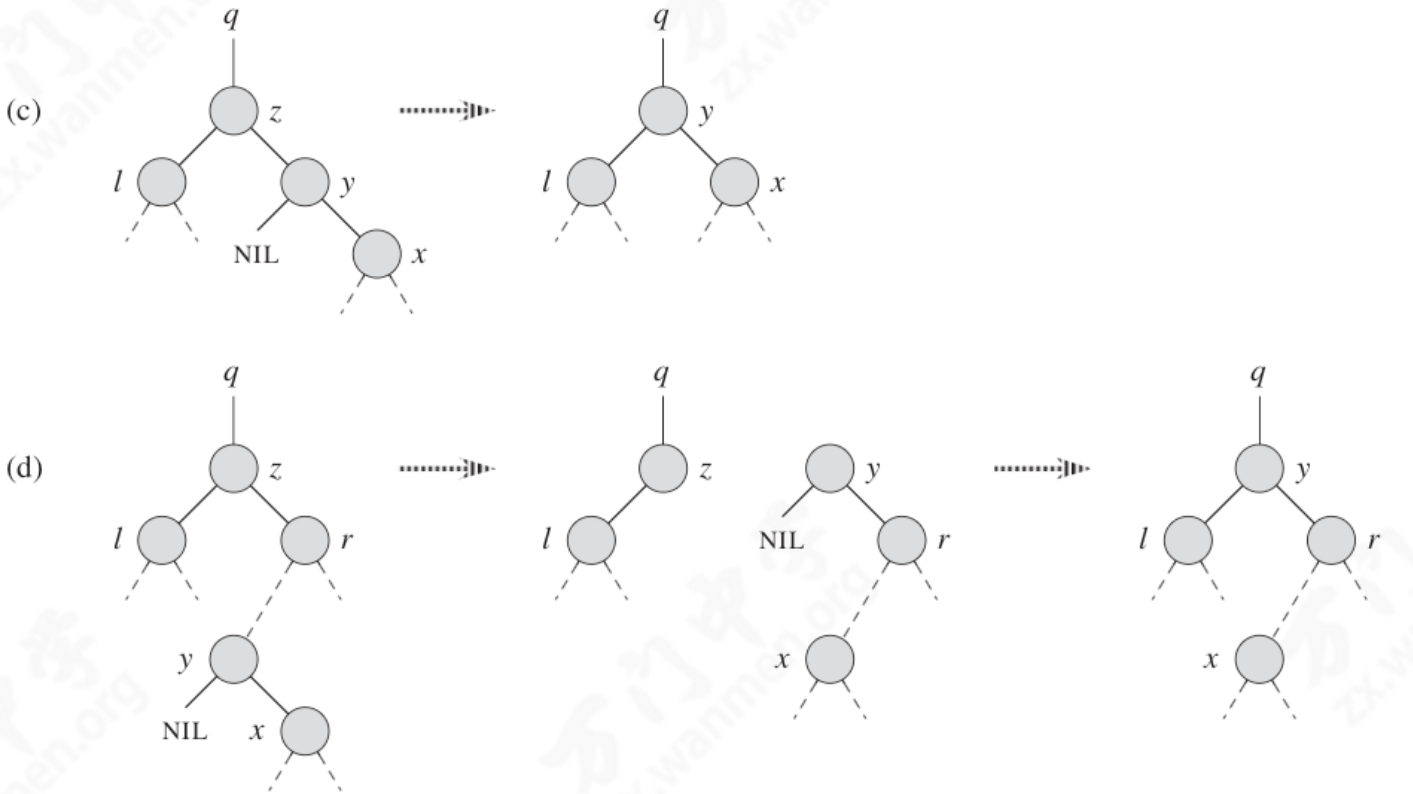
二叉查找树的删除

从一颗二叉查找树中删除一个节点，需要分四种情况讨论，伪代码作为练习留给同学们。



二叉查找树的删除

从一颗二叉查找树中删除一个节点，需要分四种情况讨论，伪代码作为练习留给同学们。



随机构造的二叉查找树

- 我们已经知道，二叉查找树上的各基本操作的运行时间都是 O(h)，h 为树的高度。
- 但是随着元素的插入或删除，树的高度会发生变化。
- 例如，如果各元素按严格增长的顺序插入，那么构造出的树就是一个高度为 n - 1 的链。
- 如果各元素按照随机的顺序插入，则构造出的二叉查找树的期望高度为 O(log n)。
- 这里省略证明，但这是一个重要的结论。
- 当看见题目中出现“数据是随机构造的”时，要能够记起这个结论哦！

下节课再见