第 13 讲: 搜索树

姓名：林凡琪　　**学号：**211240042

评分：_____　　**评阅：**_____

2022 年 5 月 18 日

请独立完成作业，不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

1　作业 (必做部分)

题目 1 (TC 12.1-5)

Argue that since sorting n elements takes $\Omega(n \lg n)$ time in the worst case in the comparison model, any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \lg n)$ time in the worst case.

解答：

反证法: 假设我们可以使用少于 $Omega\ nlgn$ 时间通过基于比较的算法构造二叉搜索树，因为有序树遍历是 $Theta\ n$ ，那么我们可以在小于 $Omega\ nlgn$ 的时间内获得排序的元素，这与 n 元素进行排序在最坏情况下需要 $Omega\ nlgn$ 时间的事实相矛盾。

题目 2 (TC 12.2-9)

Let T be a binary search tree whose keys are distinct, let x be a leaf node, and let y be its parent. Show that $y.key$ is either the smallest key in T larger than $x.key$ or the largest key in T smaller than $x.key$.

解答：

如果 $x = y.left$，则在 x 上调用后续函数将导致 while 循环没有迭代，因此将返回 y。如果 $x = y.right$，则用于调用前置任务（参见练习 3）的 while 循环将不运行，因此将返回 y。

题目 3 (TC 12.3-5)

Suppose that instead of each node x keeping the attribute $x.p$, pointing to x's parent, it keeps $x.succ$, pointing to x's successor. Give pseudocode for SEARCH, INSERT, and DELETE on a binary search tree T using this representation. These procedures should operate in time $O(h)$, where h is the height of the tree T. (*Hint:* You may wish to implement a subroutine that returns the parent of a node.)

解答:
我们不需要更改 $textSEARCH$, 但是必须实现 $textPARENT$.

```
function PARENT(T, x)
    if x == T.root then
        return NIL
    end if
    y = TREE − MAXIMUM(x).succ
    if y == NIL then
        y = T.root
    else
        if y.left == x then
            return y
        end if
        y = y.left
    end if
    while y.right! = x do
        y = y.right
    end while
    return y
end function
procedure INSERT(T,z)
    y = NIL
    x = T.root
    pred = NIL
    while x ≠ NIL do
        y = x
        if z.key < x.key then
            x = x.left
        else
            pred = x
            x = x.right
        end if
    end while
    if y == NIL then
        T.root = z
        z.succ = NIL
    else if z.key < y.key then
        y.left = z
        z.succ = y
        if pred ≠ NIL then
            pred.succc = z
        end if
    else
        y.right = z
        z.succ = y.succ
        y.succ = z
    end if
end procedure
```

We modify TRANSPLANT a bit since we no longer have to keep the pointer of p. 此外，我们必须实现 TREE-PREDECESSOR，这有助于我们在 DELETE 的第 2

```
procedure TRANSPLANT(T, u, v)
    p = PARENT(T, u)
    if p == NIL then
        T.root = v
    else if then u == p.left
        p.left = v
    else
        p.right = v
    end if
end procedure
```

行轻松找到 predecessor。

2 Open Topics

Open Topics 1 (Interval tree)
介绍 Interval tree
参考资料:

- Interval tree

Open Topics 2 (Splay tree)
介绍 Splay tree (平摊分析部分可仅介绍基本思想)。
参考资料:

- Splay tree
- Self-Adjusting Binary Search Trees (Rober Tarjan, JACM85).pdf

3 反馈

```
function TREE-PREDECESSOR(T, x)
    if x.left! = NIL then
        return TREE − MAXIMUM(x.left)
    end if
    y = T.root
    pred = NIL
    while y ≠ NIL do
        if y.key == x.key then
            break
        end if
        if y.key < x.key then
            pred = y
            y.right
        else
            y = y.left
        end if
    end while
    return pred
end function
procedure DELETE(T, z)
    pred = TREE − PREDECESSOR(T, z)
    pred.succ = z.succ
    if z.left == NIL then
        TRANSPLANT(T, z, z.right)
    else if z.right == NIL then
        TRANSPLANT(T, z, z.left)
    else
        y = TREE − MIMIMUM(z.right)
        if PARENT(T, y)! = z then
            TRANSPLANT(T, y, y.right)
            y.right = z.right
        end if
        TRANSPLANT(T, z, y)
        y.left = z.left
    end if
end procedure
```