算法导论　第七周作业　10 月 29 日　周四

PB18151866 龚小航

4.1 红黑树:

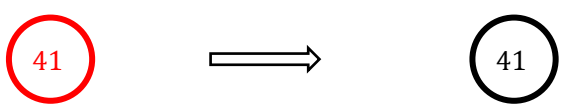
(***a***). 将关键字 41, 38, 31, 12, 19, 8 连续地插入一棵初始为空的红黑树之后，试画出该结果树。

(***b***). 对于上一问得到的红黑树，依次删除 8, 12, 19，试画出每次删除操作后的红黑树。

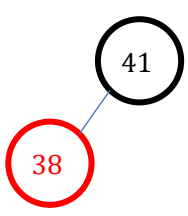
解：对这两个问题分别分析:

(***a***) 作出每一步得到的结果:

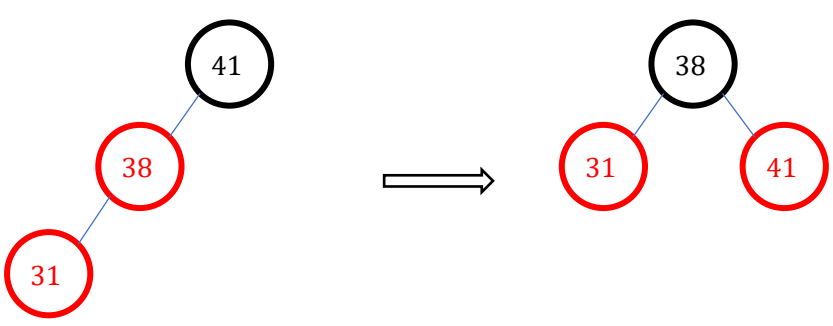
插入 41:



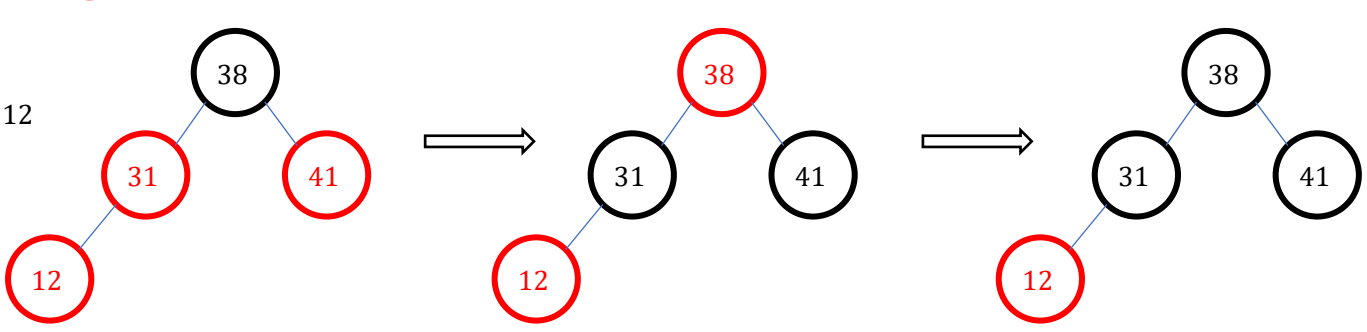
插入 38:



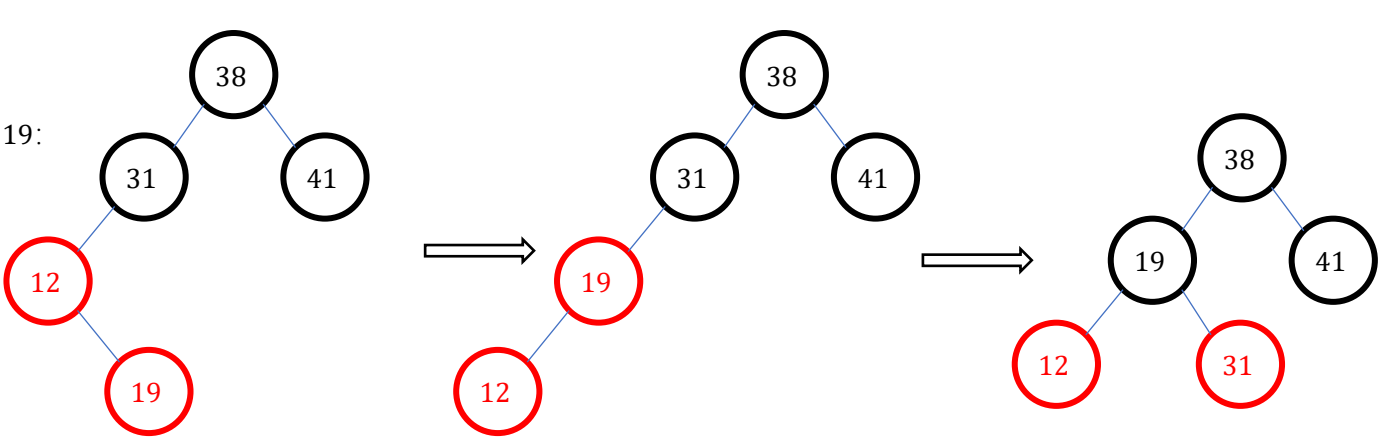
插入 31:



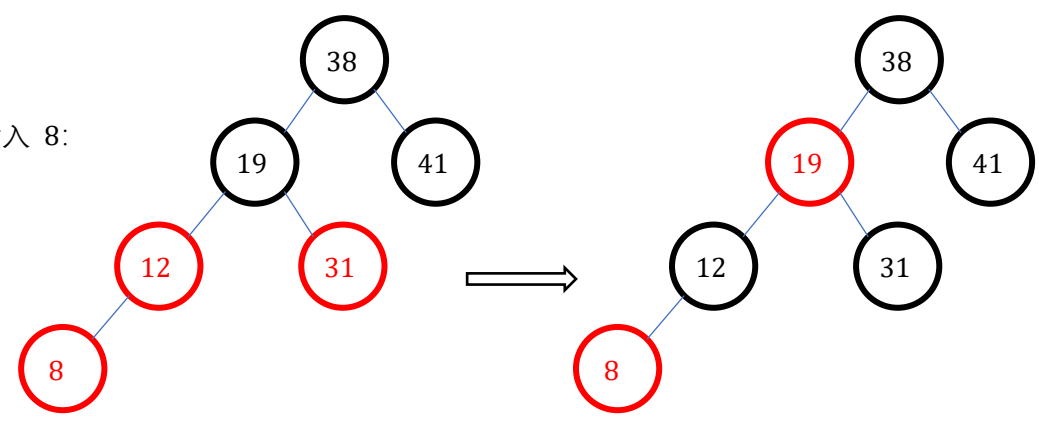
插入 12



插入 19:

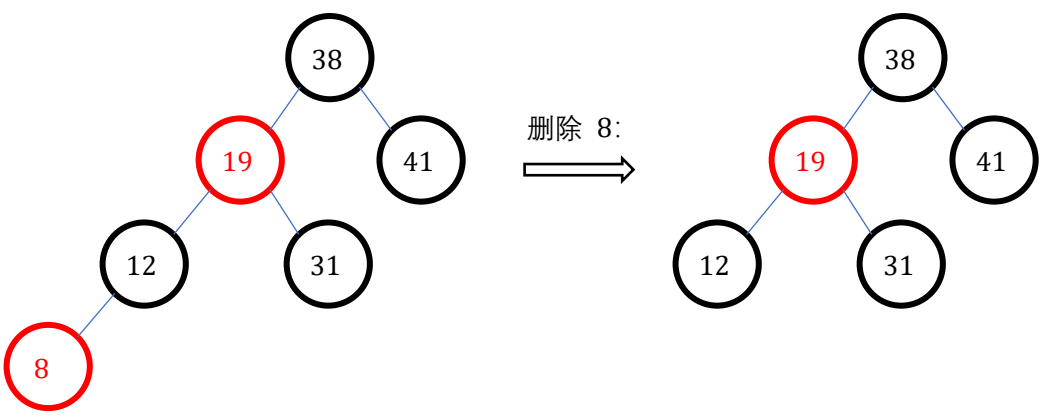


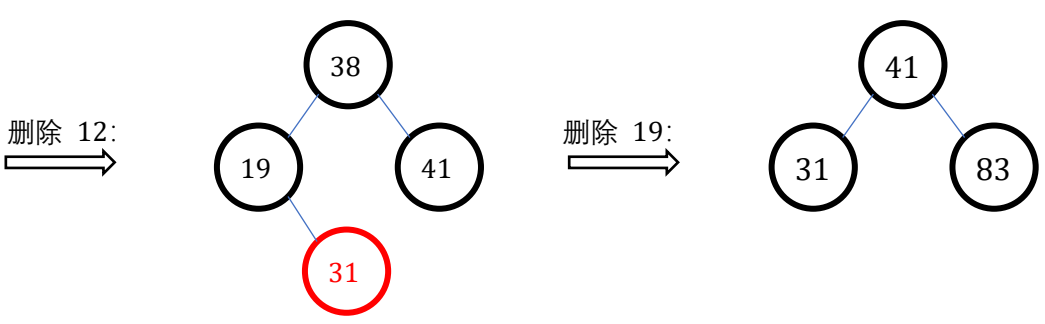
插入 8:



最终红黑树如上图所示。

(***b***) 画出每步删除之后的结果。删除可以分为两步，第一步为删除相应的节点，第二步是通过旋转与重新着色使之重新成为一棵红黑树。当删除的是叶子节点时，直接删去即可；若被删节点有一个孩子，删去它并用孩子代替它；若有两个孩子，则将其后继替换它并删除后继。





4.2 假设我们希望记录一个区间集合的最大重叠点，即被最多数目区间所覆盖的那个点。

(***a***) 证明：在最大重叠点中，一定存在一个点是其中一个区间的端点。

(***b***) 设计一个数据结构，使得它能够有效地支持 INTERVAL − INSERT、INTERVAL − DELETE，以及返回最大重叠点的 FIND − POM 操作。

解：分析说明：

(***a***)：选中一个最大重叠点，如果它不是某个区间端点，那么让这个点逐渐左移或右移，直到遇到一个区间端点为止。此时，新得到的端点所在区间数目要么增加(若出现这种情况说明原来取的点不是最大重叠点)要么不变，所以最大重叠点一定是一个区间的端点。

(***b***)：【课本 199 页】将所有端点组织成红黑树，如课本 199 页所示，再附加一些额外信息以及额外操作即可。该树节点的值是每个端点的值。左端点关联+1 值，而右端点关联-1 值。使用分治算法，最大重叠点要么在左子树，要么在右子树，要么是自己。

增加的信息：对于节点 x 而言：

$f(x)$：区间的左端点为 $+1$，右端点为 -1

$sum(x)$：以 x 为根的所有节点的 f 值之和

立即可以给出 $sum(x)$ 的计算方法：其中 $left$，$right$ 表示当前节点的左右孩子，。

$$sum(x) = sum\big(\text{left}(x)\big) + f(x) + sum\big(\text{right}(x)\big)$$

实现 INTERVAL − INSERT 操作：

与红黑树一样，使用相同的插入方式。在插入时需要修改若干个节点的 sum 值。具体操作为：将经过的每个节点的 sun 值加上插入节点的 f 值。

实现 INTERVAL − DELETE 操作：

和红黑树一样，使用相同的删除方式。在删除时需要修改若干个节点的 sum 值。具体操作为：将经过的每个节点的 sun 值减去插入节点的 f 值。

实现 FIND − POM 操作：

令 $m(x)$ 表示以 x 为根节点的树的最大重叠点的重叠数。显然 $\sum_{k=i}^{j} f(k)$ 表示重叠的区间数。只需计算出 $m(x)$ 即可。

一棵红黑树的最大重叠点可以用分治算法求解：重叠点分为三种情况（在左子树中、根节点、在右子树中），要分别算出左右子树的 $m(x)$，最后进行比较。而左右子树的$m(x)$ 又可递归调用该算法求出。每个 $m(x)$ 还应该附带位置信息，以指示最大重叠点的位置。

$$m(x) = \max\begin{cases} m\big(\text{left}(x)\big) & \text{最大重叠点在 } x \text{ 左子树中} \\ sum\big(\text{left}(x)\big) + f(x) & x \text{ 是最大重叠点} \\ sum\big(\text{left}(x)\big) + f(x) + m\big(\text{right}(x)\big) & \text{最大重叠点在 } x \text{ 右子树中} \end{cases}$$

4.3 （斐波那契堆删除操作的另一种实现）Pisano 教授提出了下面的 FIB − HEAP − DELETE 过程的一个变种，声称如果删除的结点不是由 H.min 指向的结点，那么该程序运行地更快。

```
PISANO-DELETE(H,x)
1: if x == H.min then
2:     FIB-HEAP-EXTRACT-MIN(H)
3: else
4:     y = x.p
5:     if y ≠ NIL then
6:         CUT(H,x,y)
7:         CASCADING-CUT(H,y)
8:     add x's child list to the root list of H
9:     remove x from the root list of H
```

(***a***) 该教授的声称是基于第 8 行可以在 O(1) 实际时间完成的这一假设，他的程序可以运行的更快。该假设有什么问题吗?

(***b***) 当 x 不是由 H.min 指向时，给出 PISANO − DELETE 实际时间的一个好（紧凑）上界。你给出的上界应该以 x.degree 和调用 CASCADING − CUT 的次数 c 这两个参数来表示。

解： 分别分析：

(a) 第八行不能在 O(1) 时间内完成。将一个节点 x 的子节点连入根节点双向链表中，需要执行的操作为：将根节点双向链表中本来连接 x 的两支连接最左和最右的子节点，总共需要调节四个指针，这一部分所需时间为 $O(1)$；另外还需要做的操作为将 x 的所有孩子节点的父节点指针置空，这一部分所需要的时间为 $O(\text{x.degree})$。因此总的时间复杂度不是常数级别的，和 x 的度数有关。

(b) 当 x 不是斐波那契堆中拥有最小值的节点时，根据算法，将 x 的所有孩子连入根节点双向链表中需要 $O(\text{x.degree})$ 的时间；而做 CASCADING − CUT 一次就将一个节点移到根节点双向链表中，并将父节点标记。因此一次操作所需的时间是常数，c 次调用 CASCADING − CUT 所需要的时间为 $O(c)$。因此 PISANO − DELETE 算法的实际时间上界可以为 O(x.degree + c)