Ledl扩充的数据结构、动态有序统计和区间树

augmenting data structures

normally, rather than designing data structures from scratch, you tend to take existing data structures and build your functionality into them.

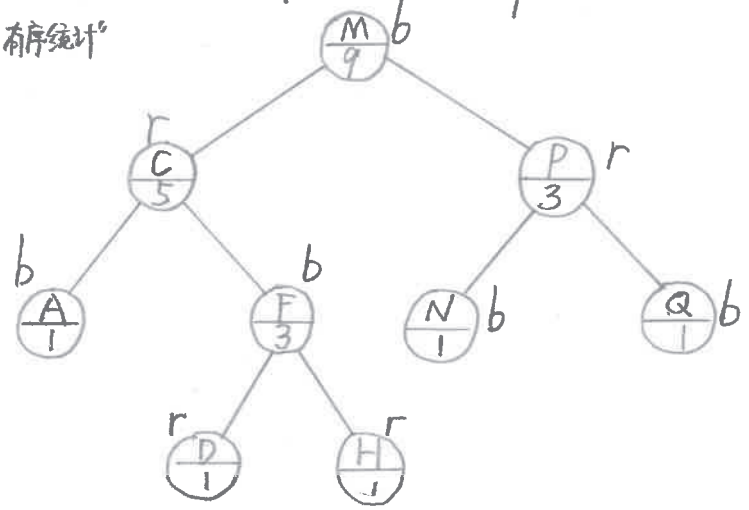
dynamic order statistics

OS-Select (i): return the i-th smallest item in dynamic set

OS-Rank (x): return the rank of x in the sorted order of dynamic set

the basic idea is to keep the sizes of subtrees in the nodes of a red-black tree

"order statistics 有序统计"



(record the subtree sizes in the red-black tree)

x.size = x.left.size + x.right.size + 1 (= the rank of x)

Trick: sentinel (标记法), 即 dummy record (伪记录) for nil (nil.size = 0)

根据这个，开始写 OS-Select (i) 的代码。

```
OS-Select (x, i) // the i-th smallest in the subtree rooted at x
    k ← x.left.size + 1   // k = rank(x)
    if i = k then return x
    if i < k then return OS-Select (x.left, i)
    else return OS-Select (x.right, i-k)
```

Question: why not just let nodes keep its ranks themselves?

Answer: 难以维护。比如插入一个最小的元素，所有的记录都要被修改。

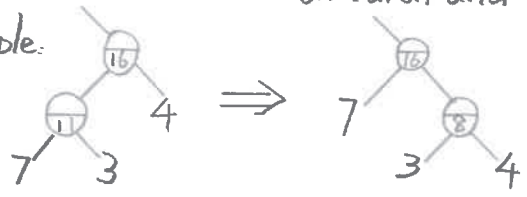
modifying ops: insert, delete

strategy: update subtree sizes when inserting and deleting, (O(lgn) time)

but must handle rebalancing

- r-b color changes: no effect on the size of subtrees
- rotations: look at children and fix up in O(1) time

Example:



data-structure augmentation

methodology: (Ex: OS-trees)

1. choose an underlying data structure (RB-tree)
2. determine what additional information we wish to maintain in the data-structure (subtree sizes)
3. verify that the information can be maintained for the modifying operations
4. develop new operations that use the information you stored (OS-Select, OS-Rank)

usually, must play with interactions between steps

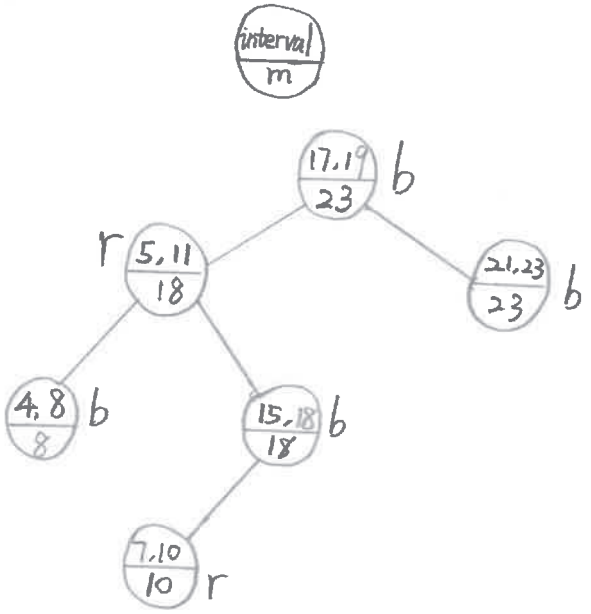
Example: interval trees

maintain a set of intervals, e.g. time intervals

say $i = [7, 10]$, $i.low = 7$, $i.high = 10$

目标 查询(Query): 给定一个区间, 查询集合里所有与给定区间发生重合的区间有哪些。

1. 选择红黑树, 选择 interval 的 lower endpoint 作为关键字
2. 在一个结点里存储这个结点的子树的最大值 m



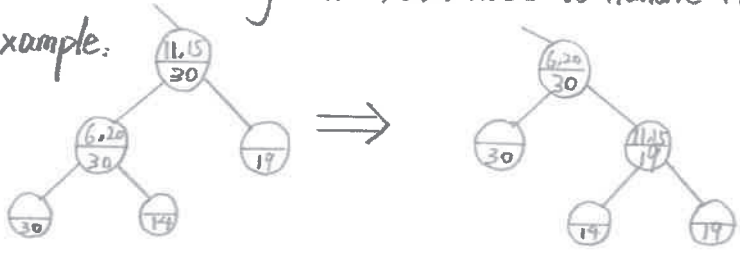
$m = \max\{x.high, x.left.m, x.right.m\}$

3. modifying operations

- insert (O(lgn) time)

fix m's on the way down, but need to handle rotations

Example:



- delete
  留作习题答案略

4. new operations

```
Interval-Search(i) // find an interval that overlaps i
    x ← root
    while x ≠ nil and (i.low > x.interval.high or x.interval.low > i.high)
        do //
            if x.left ≠ nil and i.low ≤ x.left.m
                then x ← x.left
            else
                then x ← x.right
    return x
```

Time = $O(\lg n)$

to list all overlaps,

"拿到一个就删一个，直到拿完，最后再放回去" $O(k \lg n)$

"输出敏感：时间与输出数量有关"

Correctness Analysis

Theorem: let $L=\{i' \in x.left\}$, $R=\{i' \in x.right\}$,

if search goes right, then $\{i' \in L, i' \text{ overlaps } i\} = \phi$

if search goes left, then $\{i' \in L, i' \text{ overlaps } i\} = \phi$

$\Rightarrow \{i' \in R, i' \text{ overlaps } i\} = \phi$

proof: suppose search goes right,

if $x.left = nil$, done since $L = \phi$

otherwise, $i.low > x.left.m$

$= j.high$ for some $j \in L$

no other intervals in L has a larger high endpoint than $j.high$

therefore, $\{i' \in L, i' \text{ overlaps } i\} = \phi$

suppose search goes left, and $\{i' \in L, i' \text{ overlaps } i\} = \phi$

then $i.low \leq x.left.m = j.high$ for some $j \in L$

since $j \in L$ and j doesn't overlap i

$\Rightarrow i.high < j.low$

$\because \forall i' \in R, j.low \leq i'.low$

$\therefore \{i' \in R, i' \text{ overlaps } i\} = \phi$

Q.E.D.