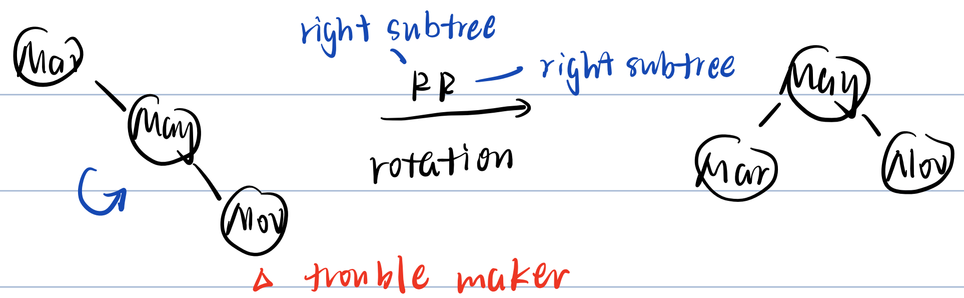
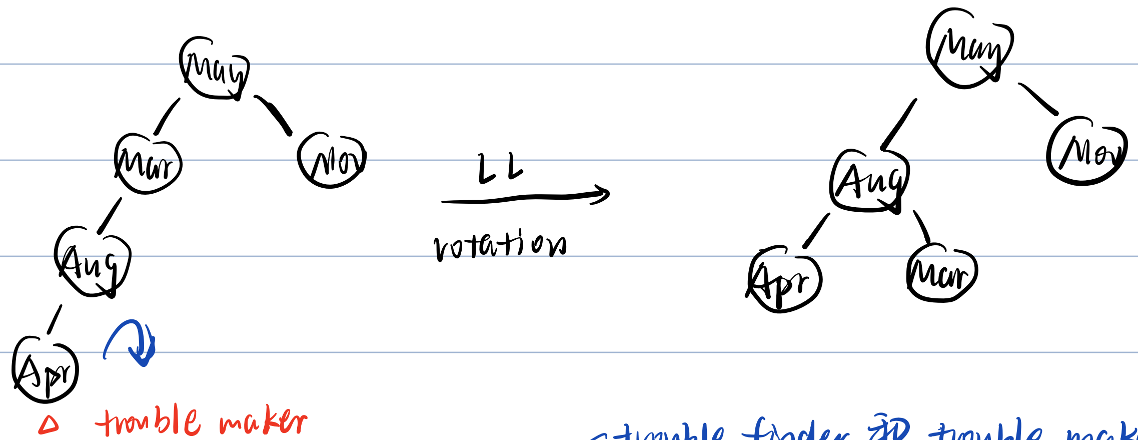
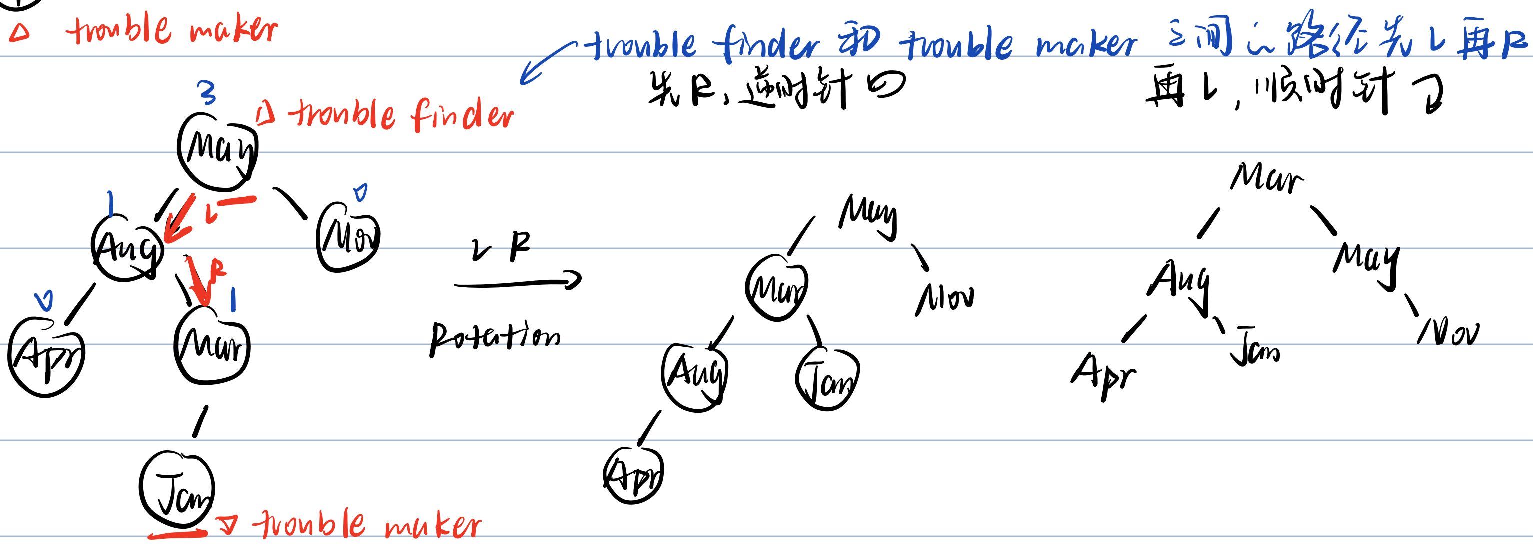
AVL Tree

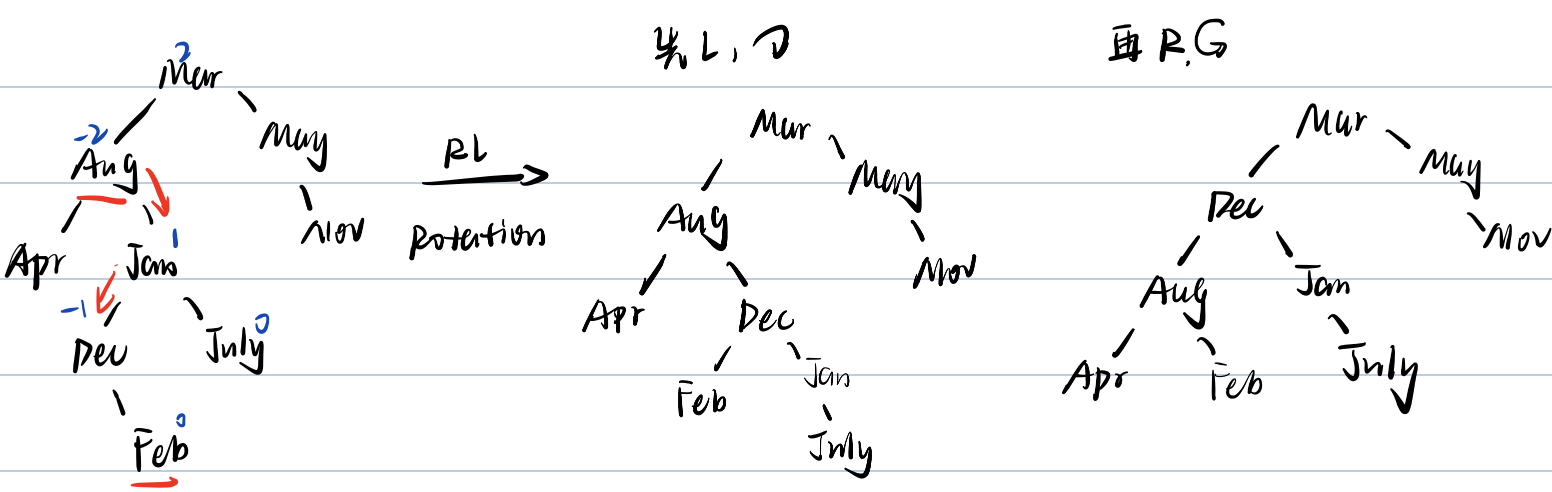
$O(\log N)$ AVL - [illegible] splay tree

$BF(node) = |h_L - h_R| \le 1$ 每一次加入都做检查来调整









To start: 记 n_h 为高度 h 的树的节点数, $n_h = n_{h-1} + n_{h-2} + 1$,

由斐波那契, $n_h = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^{k+3} - 1$ ⇒ h (高度) $= O(\ln n)$

Splay tree

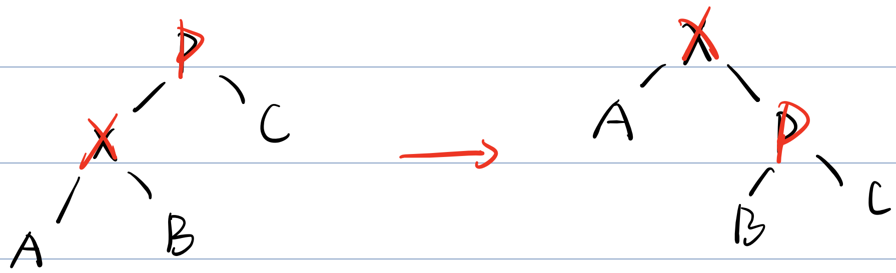
⇒ 对一个空树连续M次操作, 时间复杂度 $O(m\log n)$.

每次访问、插入的时候, 都将该点移动到root.

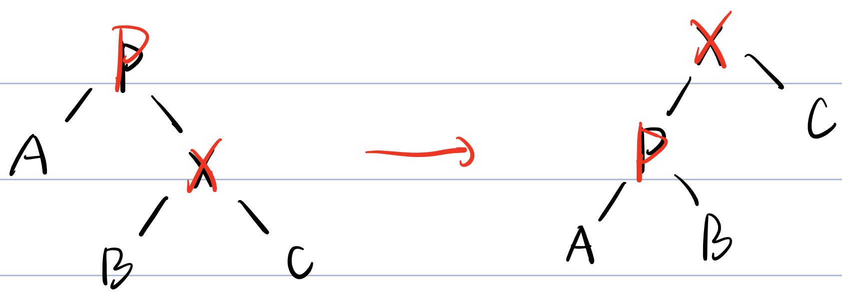
insertion 插入:

node X, parent P, grandparent G.

1. if P is the root → rotate X and P

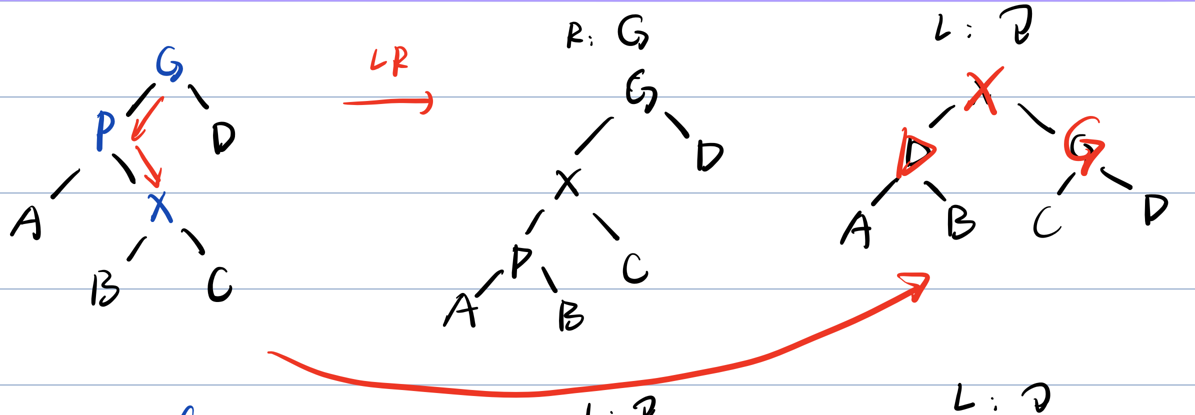


or.

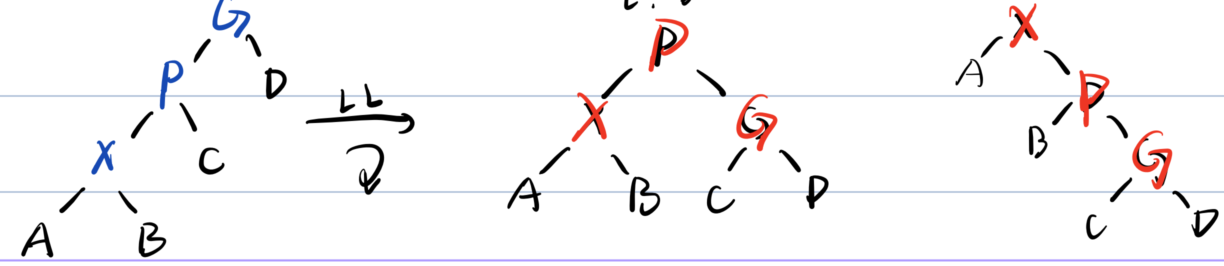


2. if P is not the root.

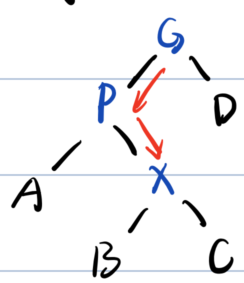
zig-zag



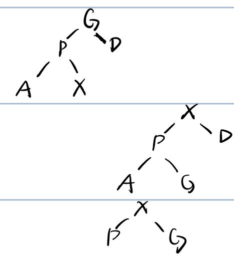
zig-zig



zig-zag (LR 或 RL)　zig左旋, zag右旋.



1. 取下子节点

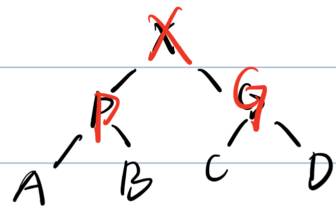


2. 将X换到最高位置

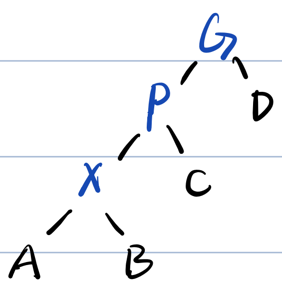
3. 将P、G放在X两侧

4. 接上子节点

5.



zig-zig (LL 或 RR)



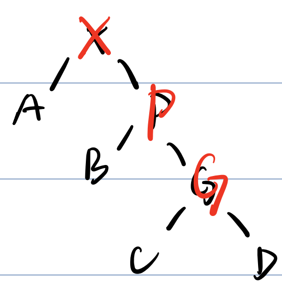
1. 取下子节点

2. 将X换到最高位置

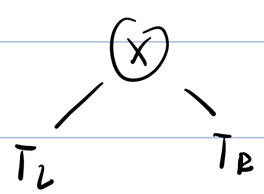
3. 将P、G放在X两侧 !! 按照LL/RR排列

4. 连上子节点

5.

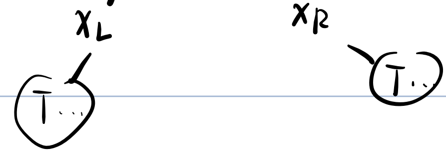


deletion 操作.



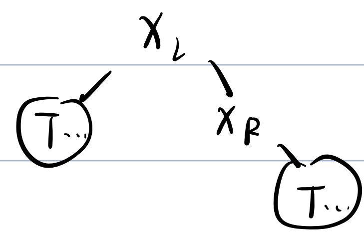
1. find X.　(X一定root)

2. 删 X.



3. find max T_L : X_L

4. 取 X_L 为新 root，接上 T_R:



Amortized analysis 均摊分析

M个连续节点插入 at most $O(M \log N)$

→ amortized time bound.

worst-case bound ≥ amortized bound ≥ average-case bound

△ 复杂度上届

Aggregate analysis 考虑最坏的情况

worst-case time $T(n)$, average / amortized cost $= T(n)/n$

Accounting method

对一个 amortized cost (设定)，如果当前步骤所做的复杂度小于 amortized cost，就会产生 credit (存款) 留给后面的步骤.

要求设计的 amortized cost 要大于实际的付出.

Potential method

构造一个势能函数，势能函数代表了当前累计的 credit.

(比如之前的操作复杂度比均摊复杂度大，势能增加)

最后的势能减去最初的势能为正值.