19. 高级树、AVL 树 和 红黑树

复习

*Tree
- Binary Tree
- Binary Search Tree
- Pre - Order : root - left - right
- In - Order : left - root - right
- Post - Order : left - right - root

平衡树

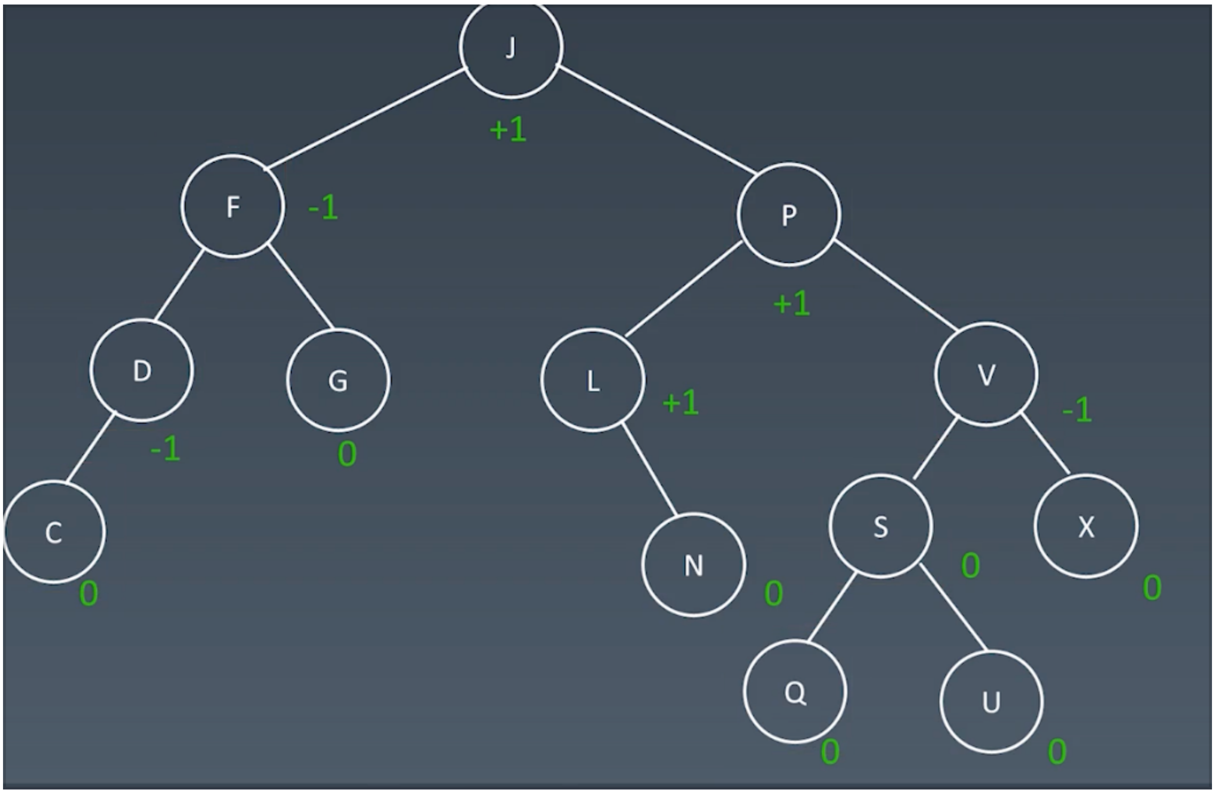
保证性能的关键

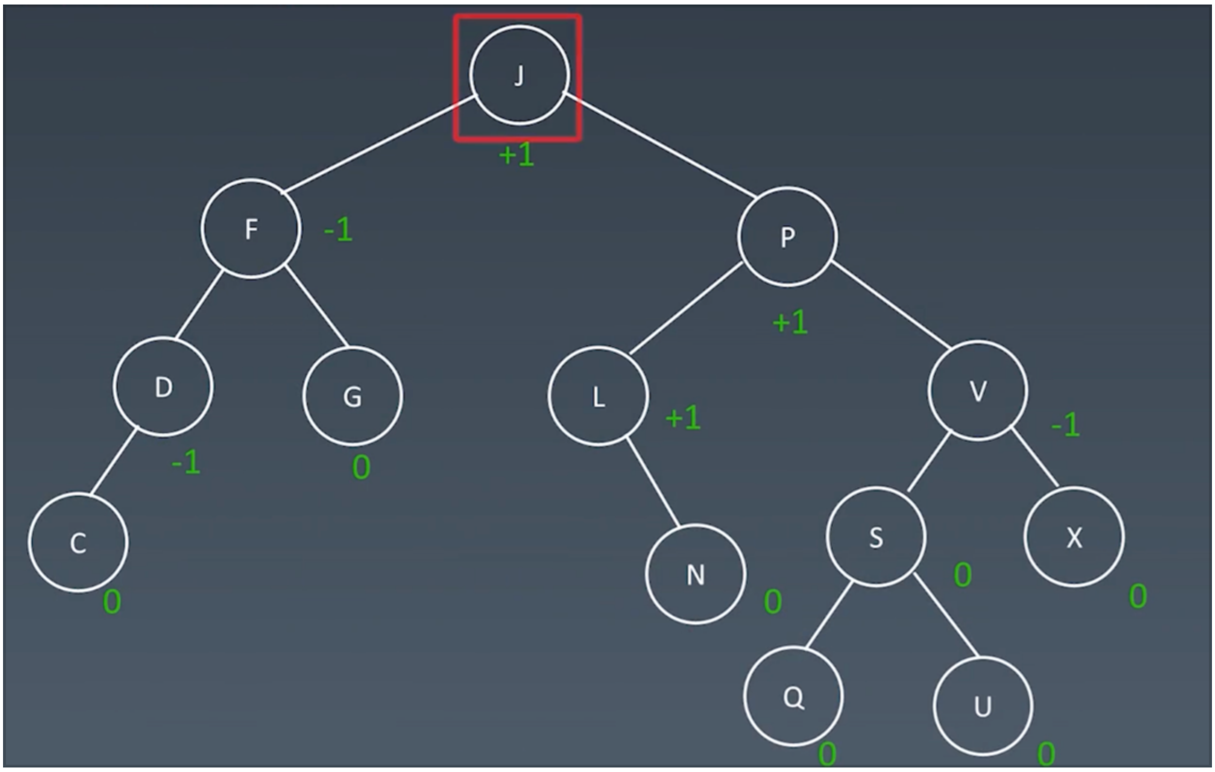
1. 保证二维维度！ ➜ 左右子树节点平衡（ Recursively ）
2. Balanced
3. Self Balancing Binary Search Tree
    - 2 - 3 Tree
    - AA Tree
    - AVL Tree
    - B - Tree
    - Splay Tree
    - Treap

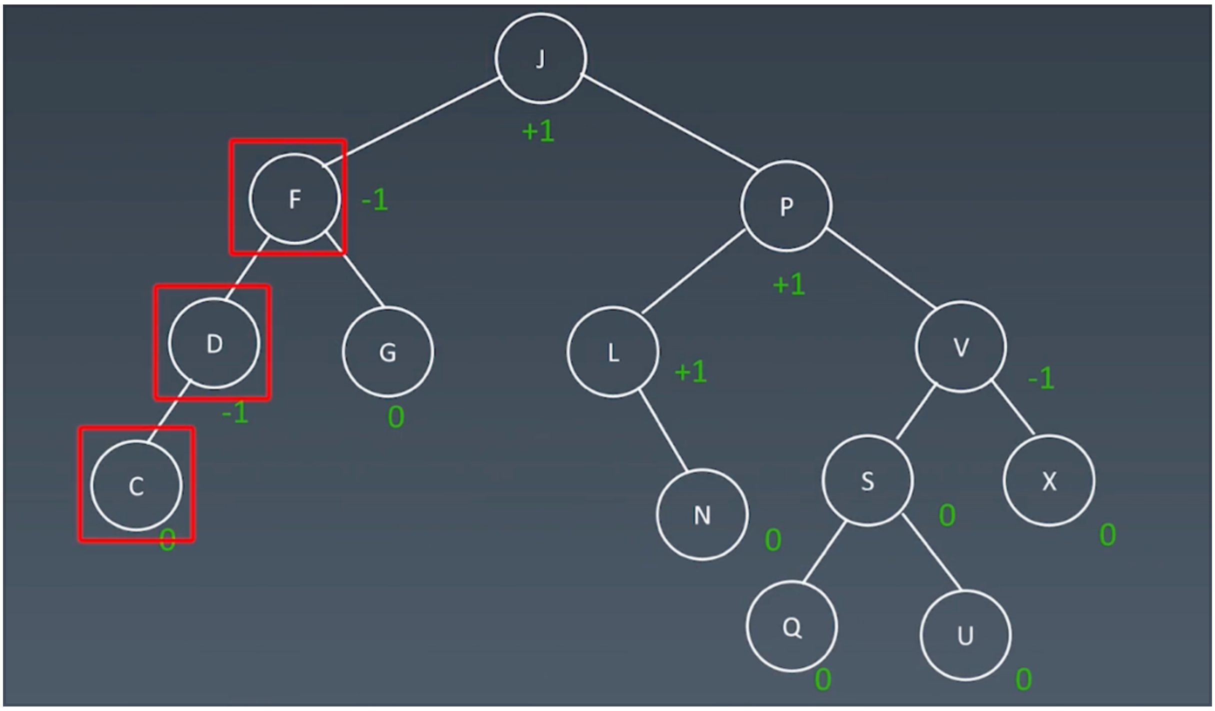
AVL 树

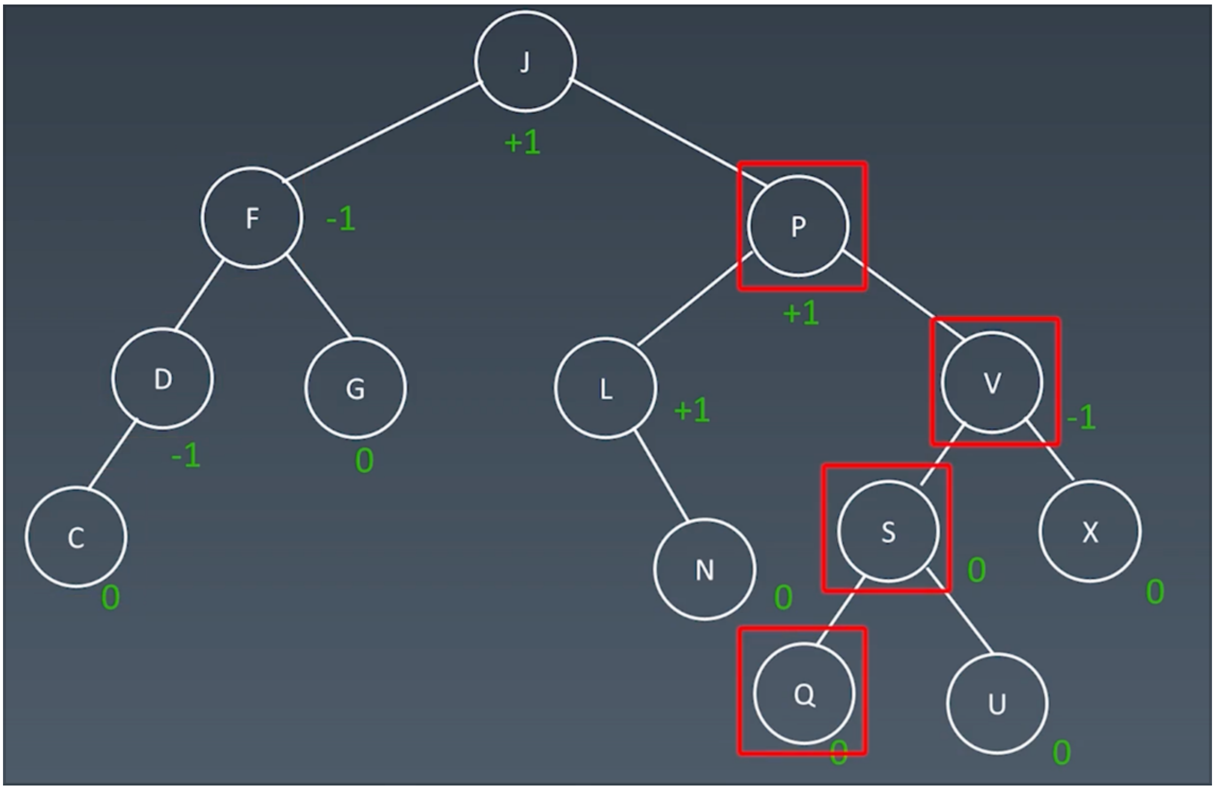
1. 发明者：G.M. Adelson - Velsky 和 Evgenii Landis
2. 平衡因子 Balance Factor
    - 左子树的高度减去它的右子树的高度（有时相反）
    - balance factor = { -1, 0, 1 }
3. 四种旋转操作来进行平衡
    1. 左旋 ： 右右子树
    2. 右旋 ： 左左子树
    3. 左右旋 ：左右子树
    4. 右左旋 ：右左子树
4. 不足：节点需要存储额外信息，且调整次数频繁
5. Self Balancing Binary Search Tree

- 例子：J 节点 平衡因子 = 右子树高度 4 - 左子树高度 3 = 1



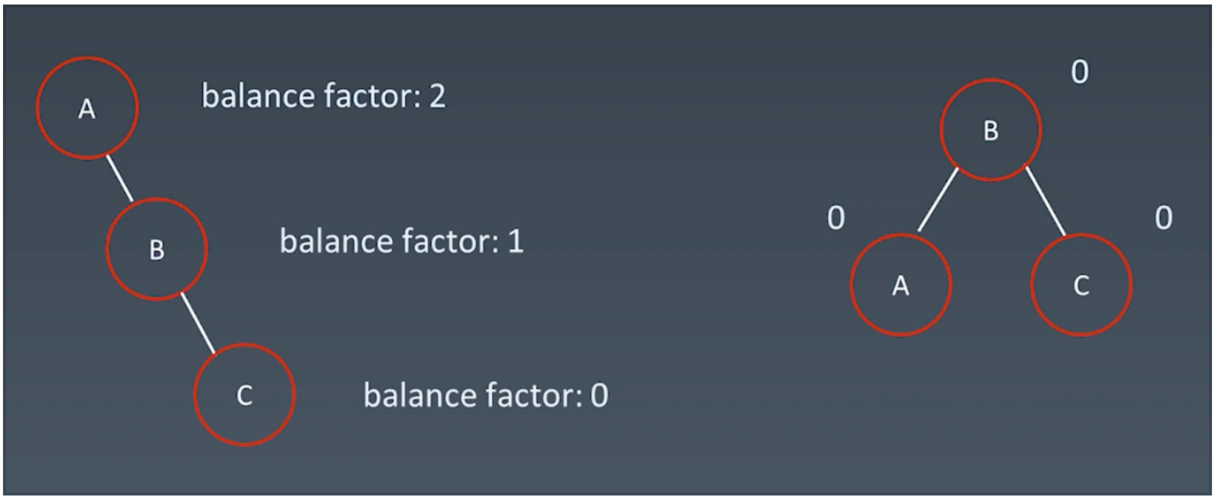




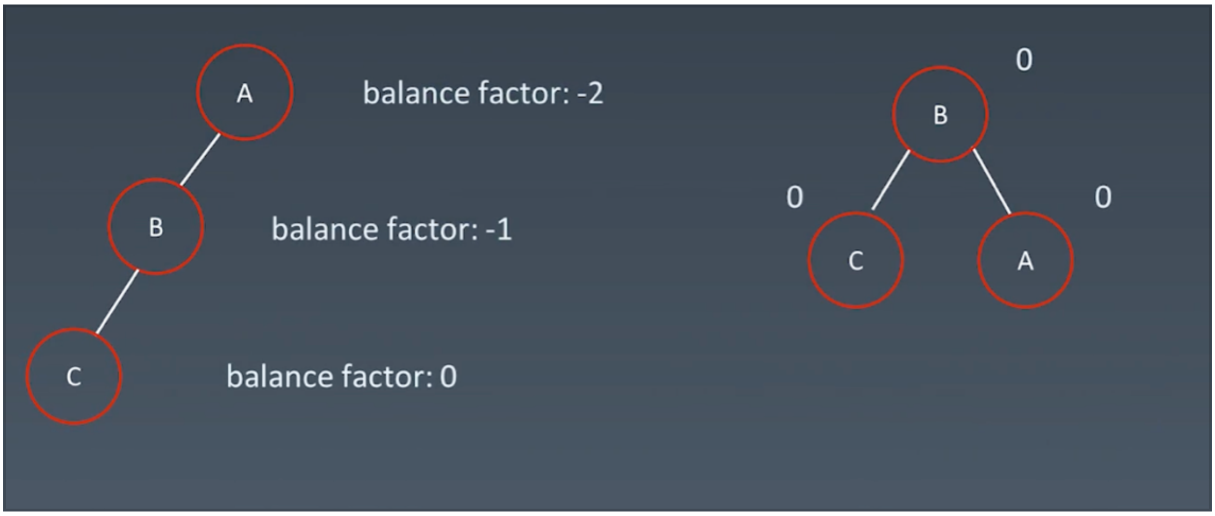


四种旋转操作

1. 左旋：右右子树

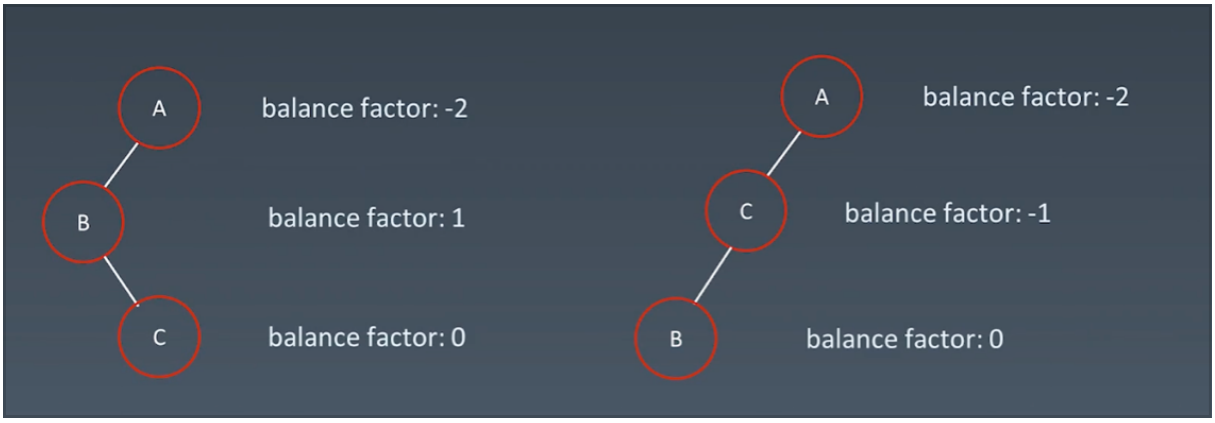


2. 右旋：左左子树

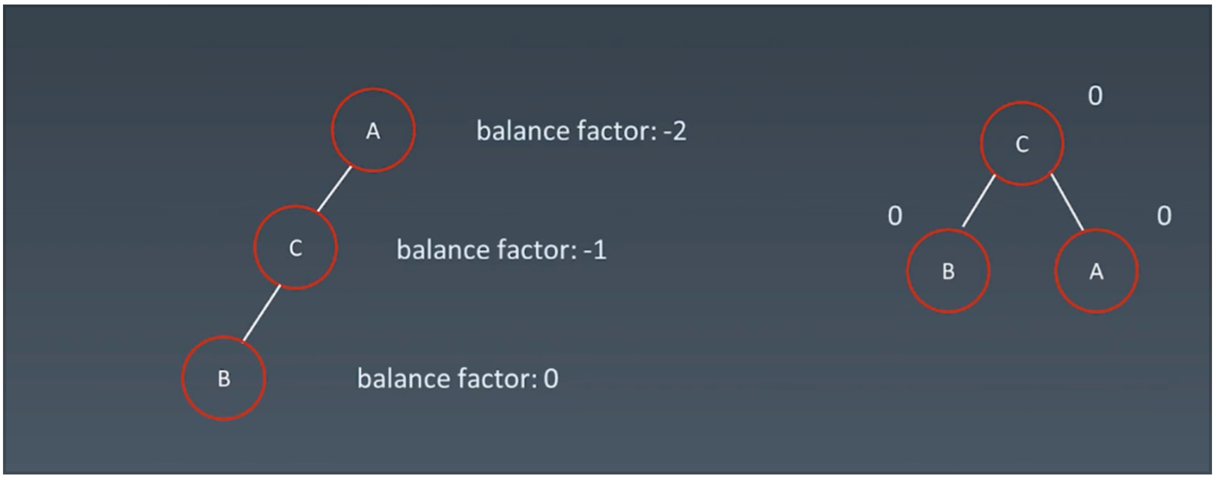


3. 左右旋：左右子树

先对B节点子树左旋

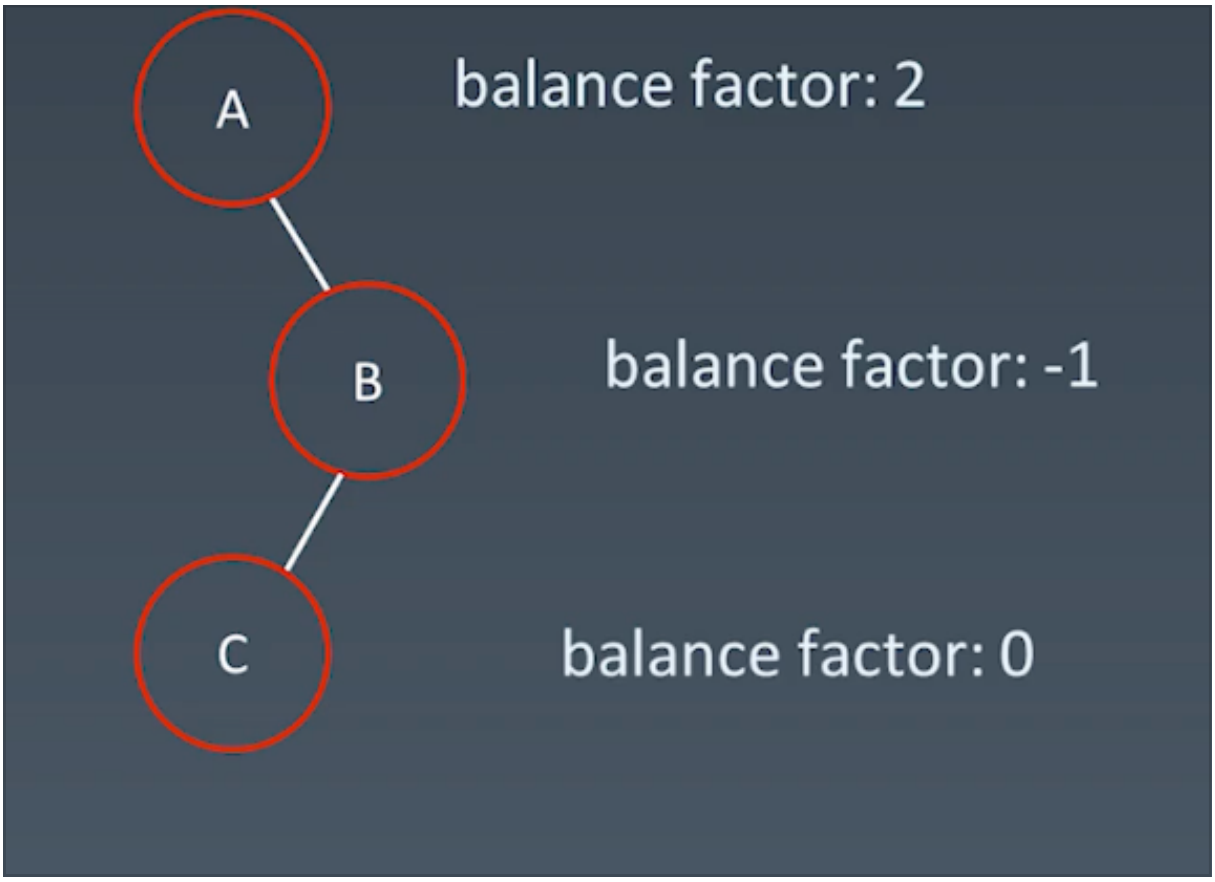


再对A节点子树右旋

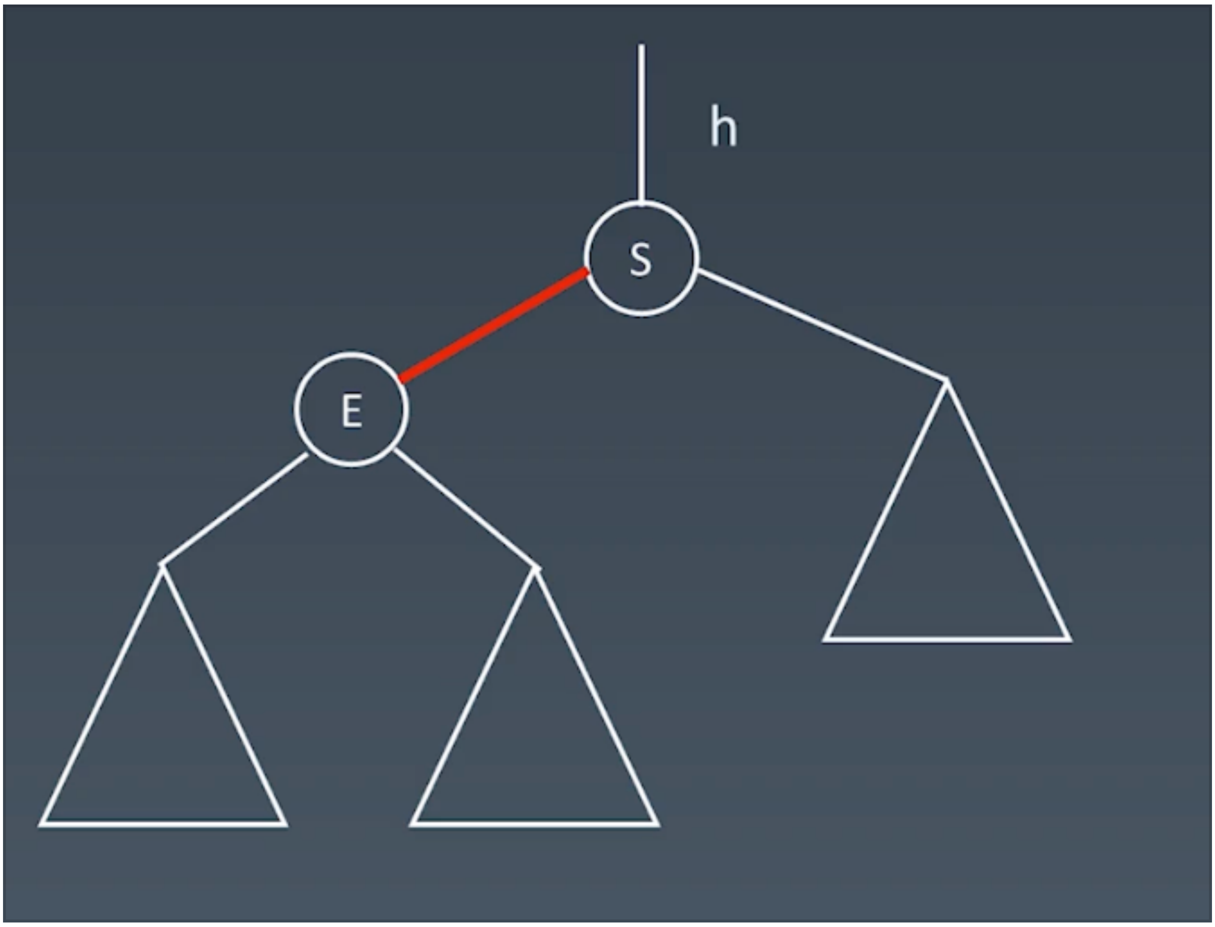


3. 右左旋 ： 右左子树

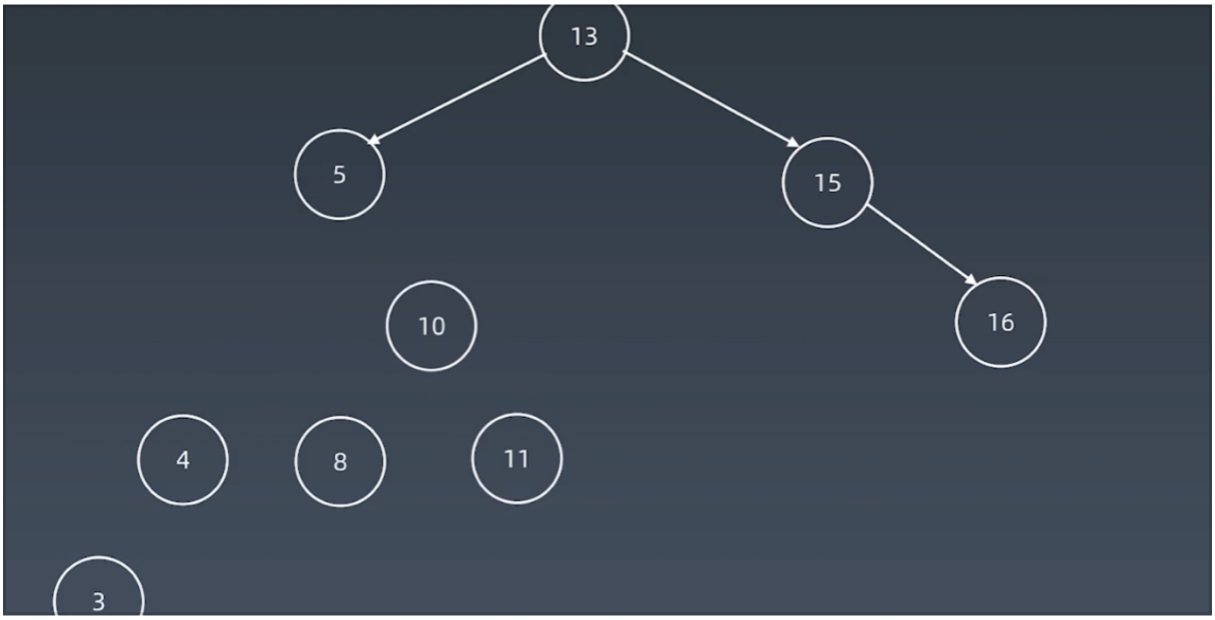
先对B节点子树右旋，再对A节点子树左旋



带有子树的旋转演示

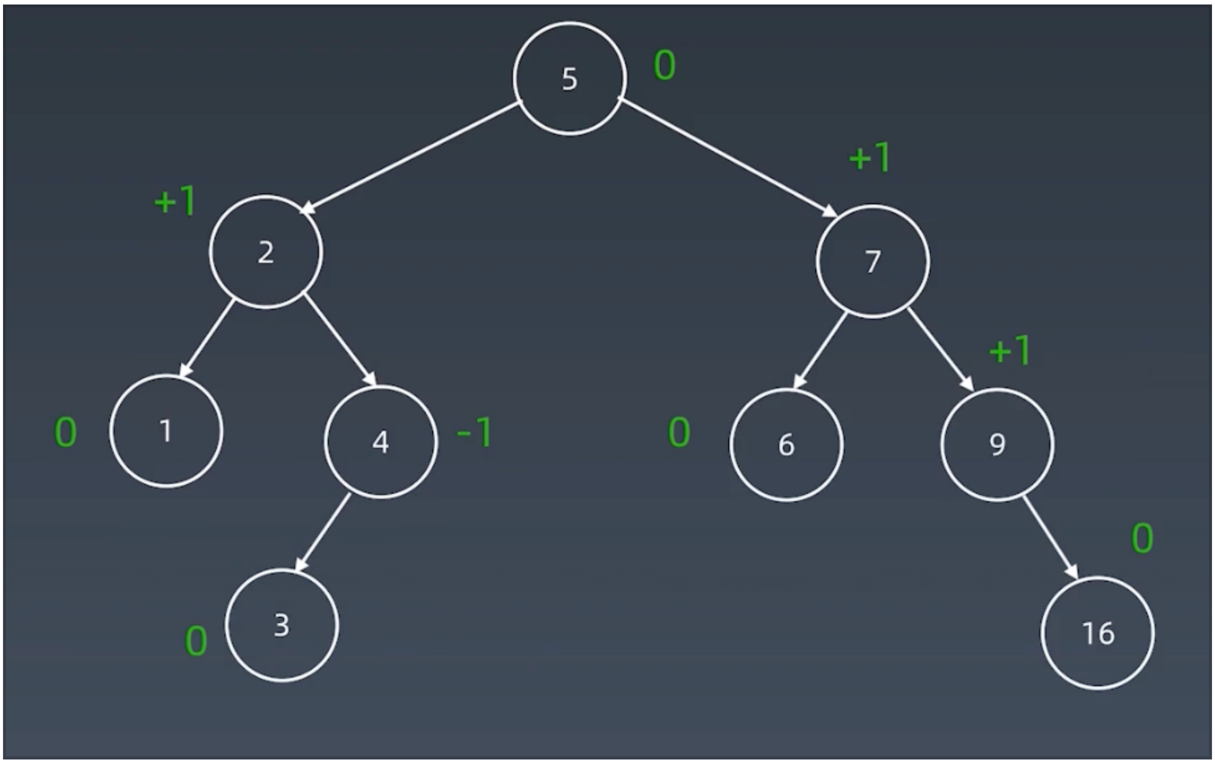


- 参考动画
- 例子1：带有子树的右旋

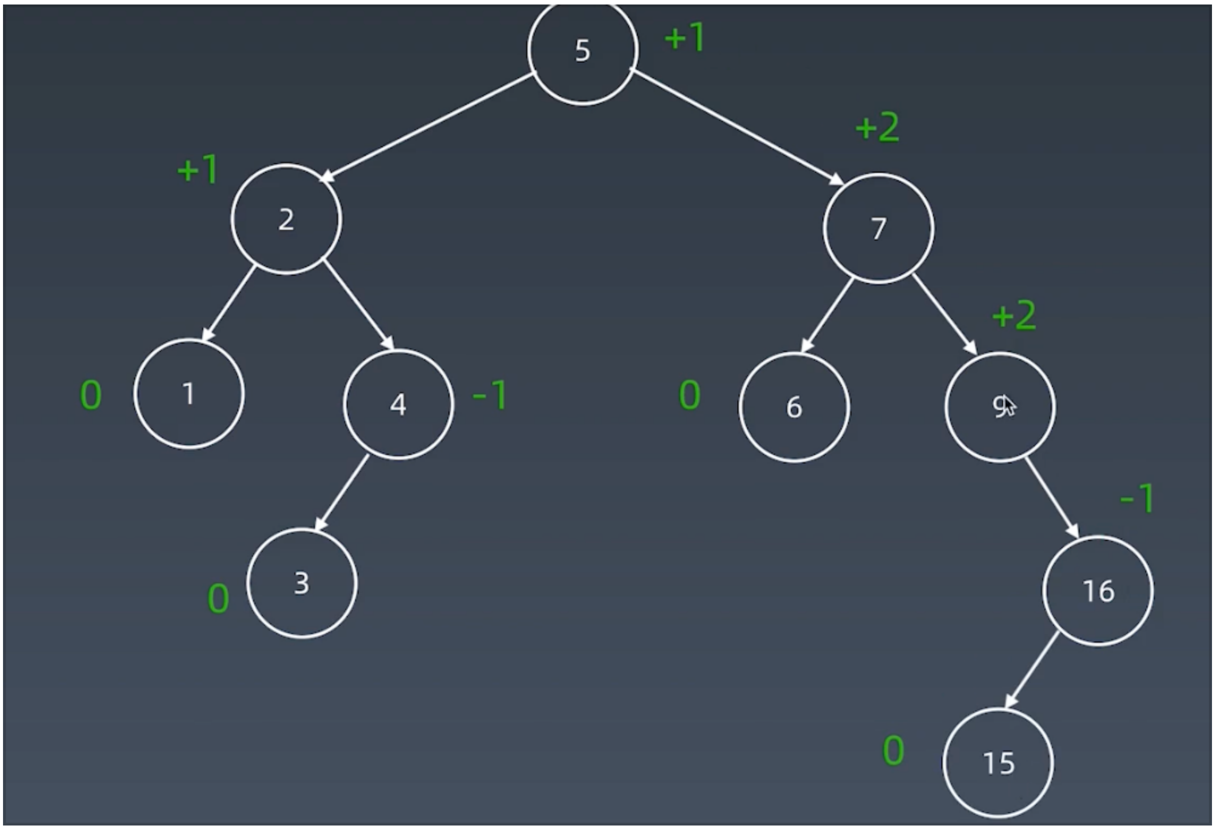


- 例子2 ： 增加 15

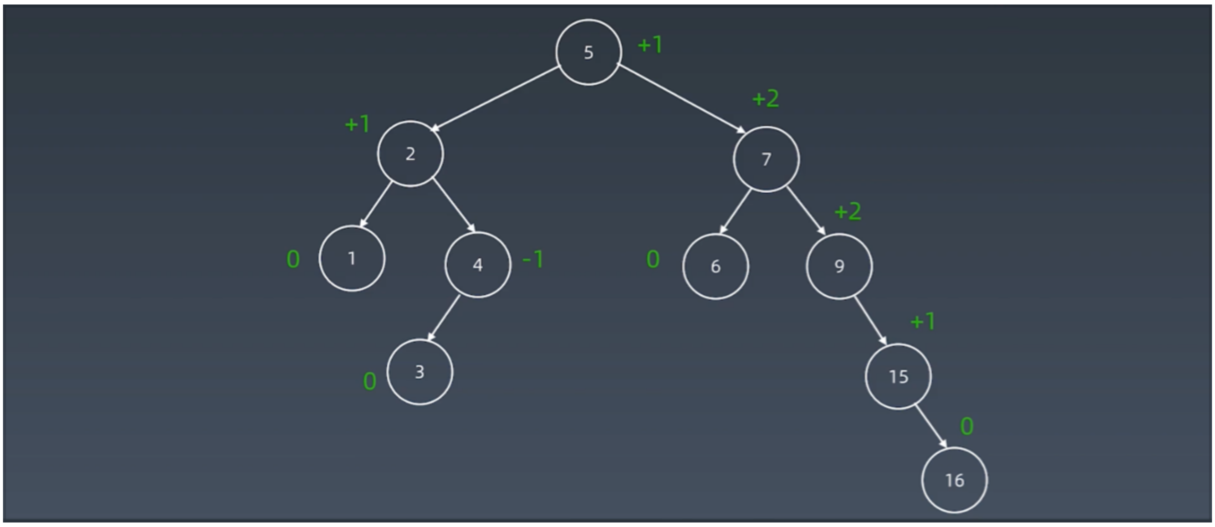
  原二叉搜索树



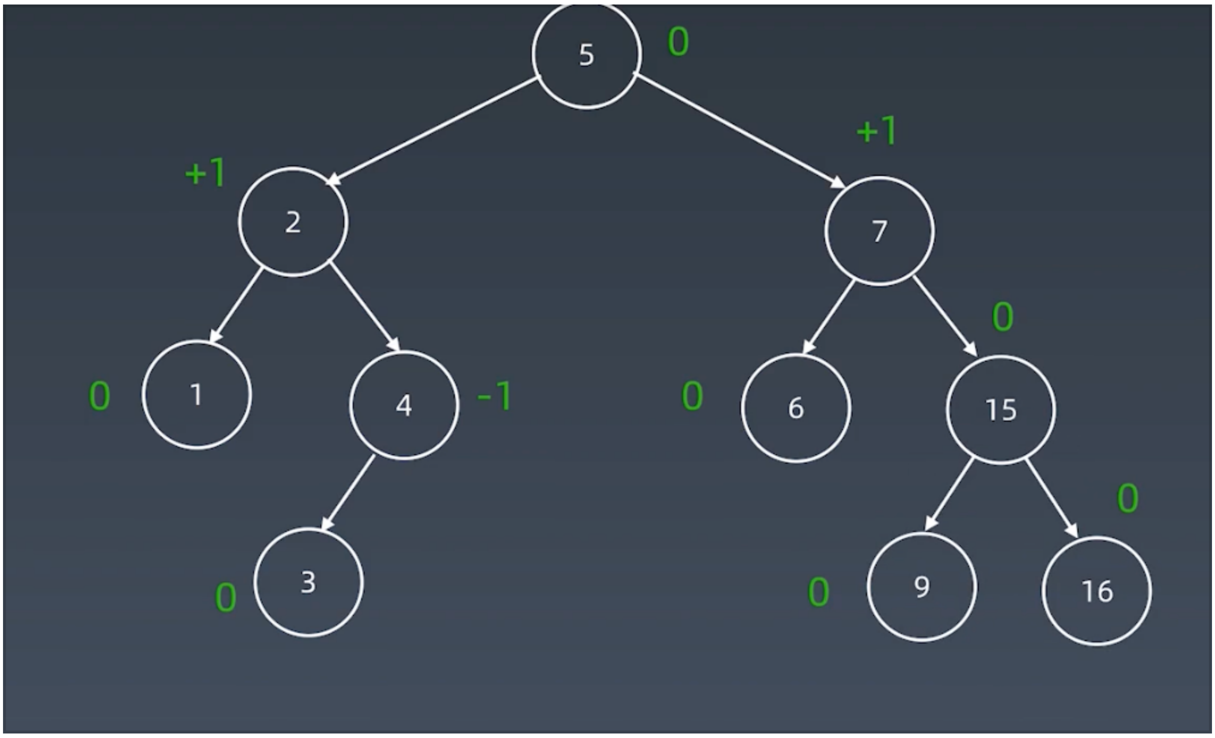
增加 15 节点，原二叉搜索树不平衡



右左子树，先对16节点进行右旋



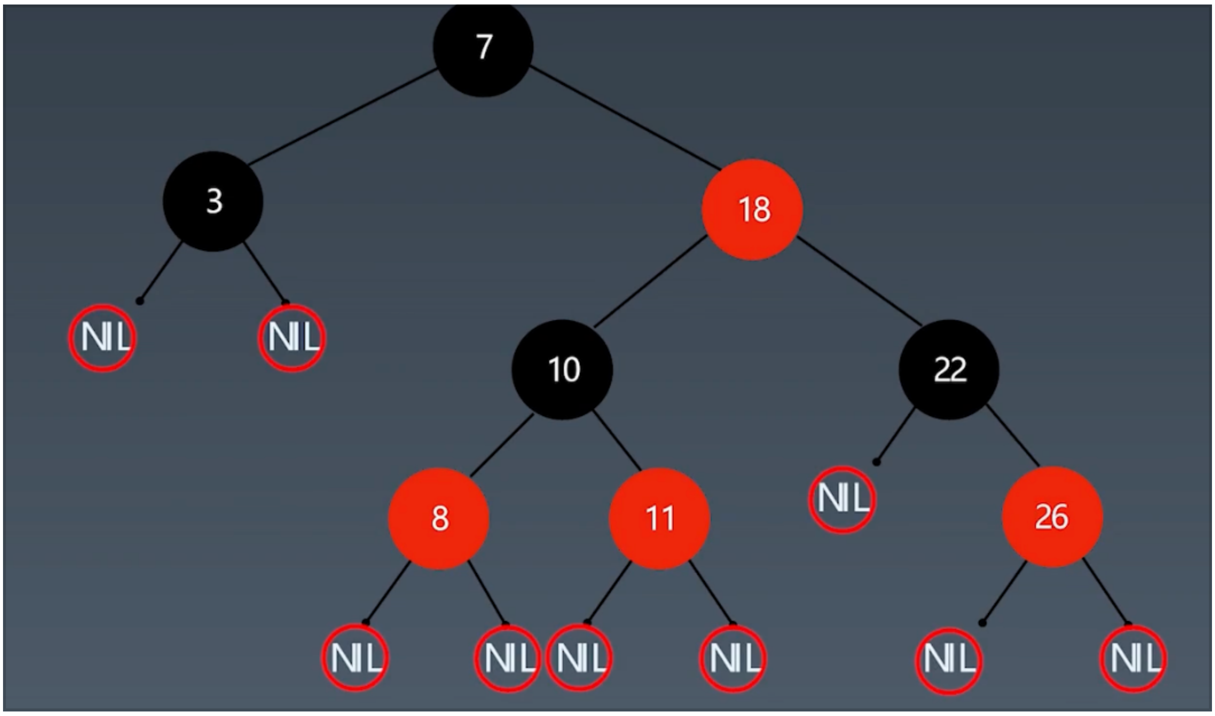
再对15节点进行左旋

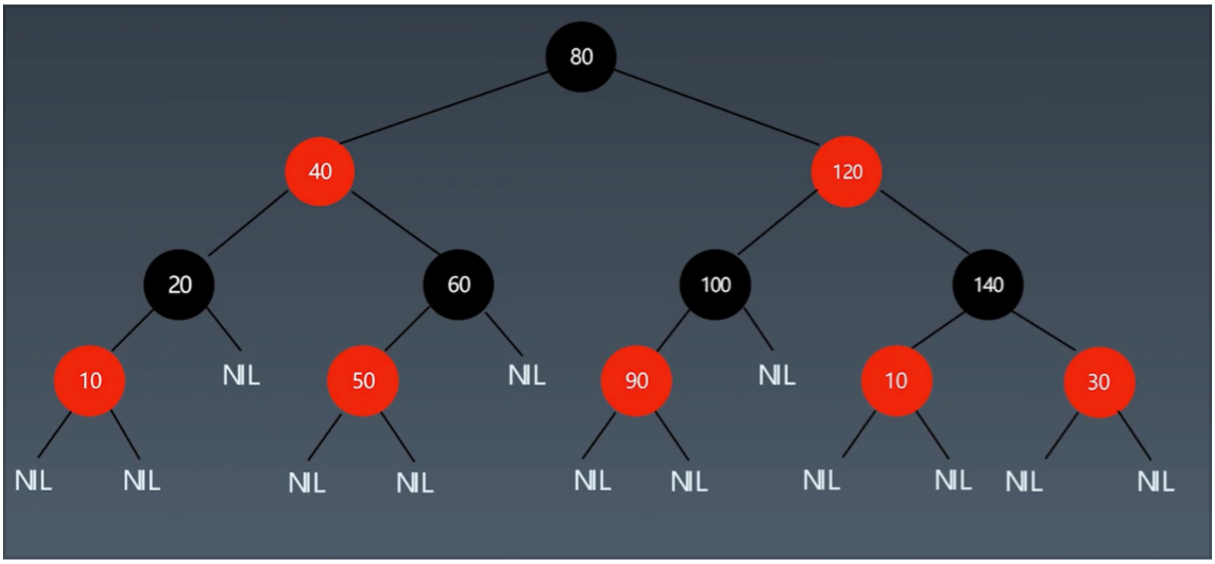


红黑树 Red - Black Tree

定义

红黑树是一种 近似平衡 的二叉搜索树（Binary Search Tree）, 它能够确保任何一个节点的左右子树的 高度差小于两倍.





红黑树是满足如下条件的二叉搜索树 ：

- 每个节点要么是红色，要么是黑色；
- 根节点是黑色；
- 每个叶节点 (NIL 节点、空节点）是黑色的；
- 不能有相邻接的两个红色节点；
- 从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点 ➜ 确保任何一个节点的左右子树的高度差小于两倍.

关键性质

从根到叶子的最长的可能路径不多余最短的可能路径的两倍长 ➜ 任何一个节点的左右子树的高度差小于两倍.

对比

- AVL trees provide faster lookups than Red Black Trees because they are more strictly balanced.
- Red Black Trees provide faster insertion and removal operations than AVL trees as fewer rotations are done due to relatively relaxed balancing.
- AVL trees store balance factors or heights with each node, thus requires storage for an integer per node whereas Red Black Tree requires only 1 bit of information per node.
- Red Black Trees are used in most of the language libraries like map, multimap, multisetin C++ whereas AVL trees are used in databases where faster retrievals are required.

#Algorithm/Part II : Theory/Data Structure#