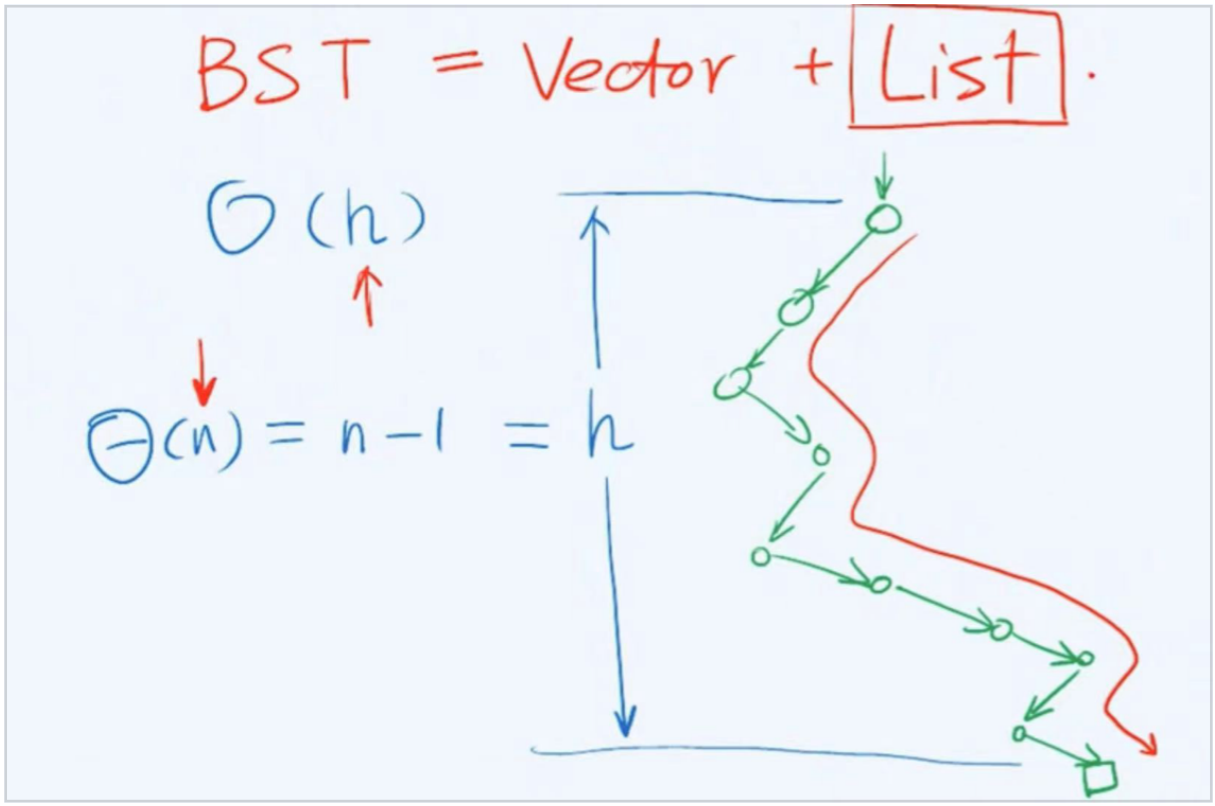
08C 平衡

#数据结构邓神

极端情况

BST = Vector + List
我们目前的BST还有些粗糙
时间复杂度都为 O(h)
所以我们要好好控制高度！！！
也就是平衡，因为平衡就是高度最低的情况

我们看一下如果不平衡的极端情况

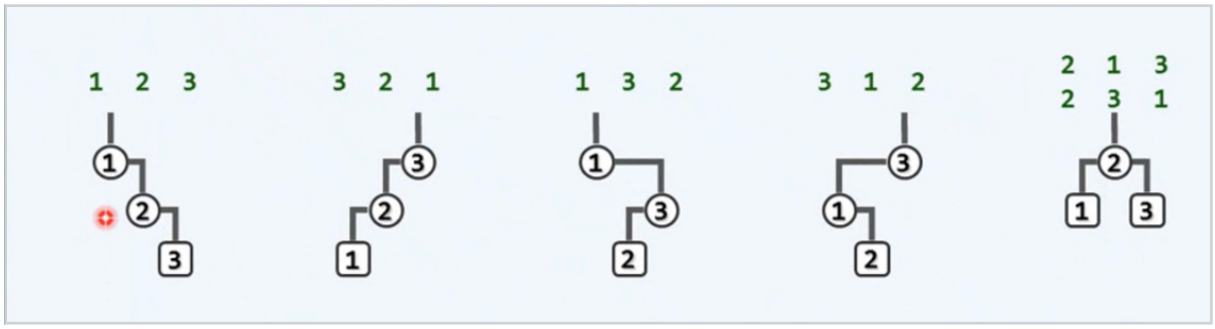


整颗树已经完全退化为List，完全失去了BST的优势
这是我们所不能接受的！

平均高度

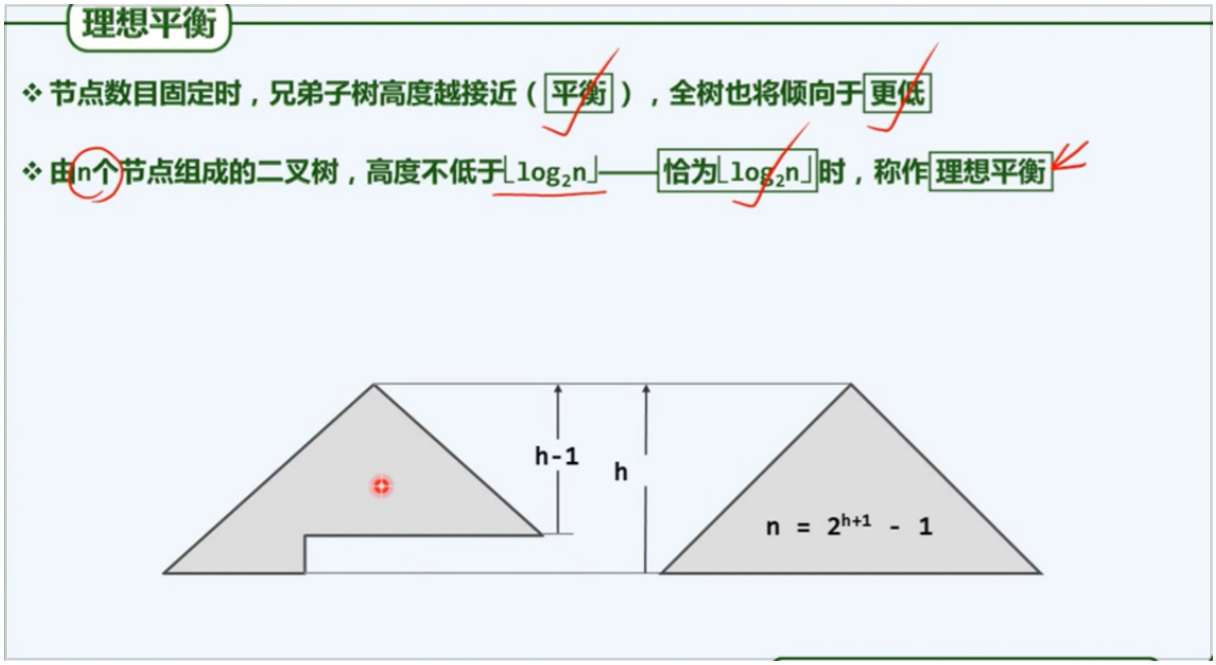
随机生成：log(n) [过于乐观]~ 随机组成：sqrt(n) ???
哪一个更为可信
后者更为可信，前者是有重复的
因为：不同的关键码序列是可能生成同一个BST

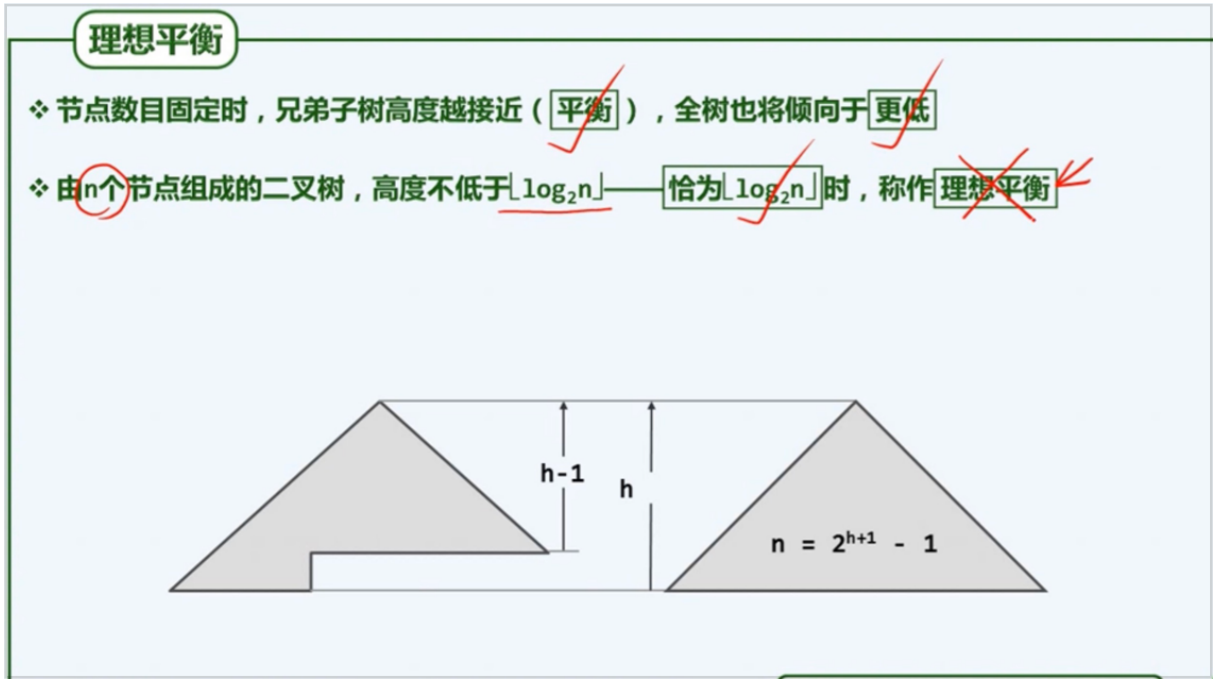
但是这不是一个好消息，意味这在随机意义下，我们无法对树高满意



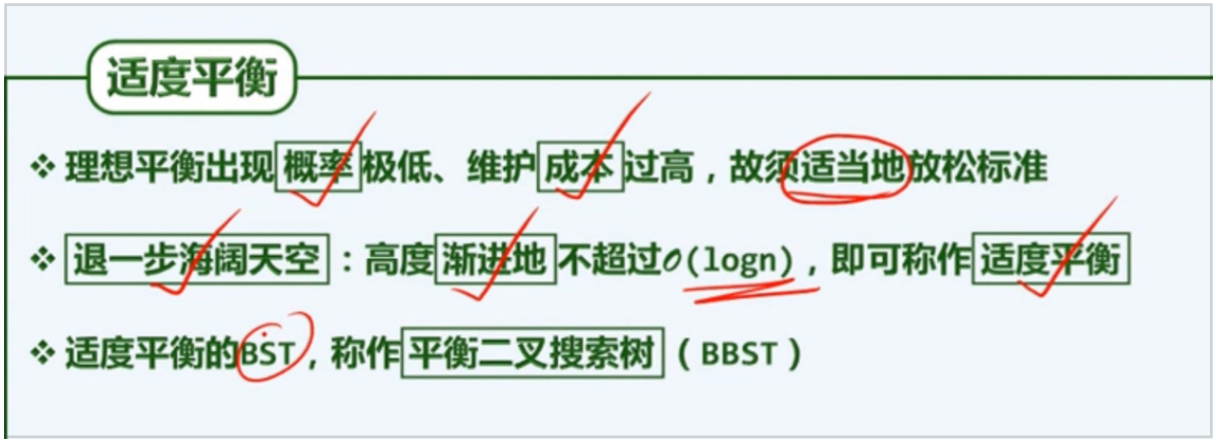
理想 + 适度

理想平衡



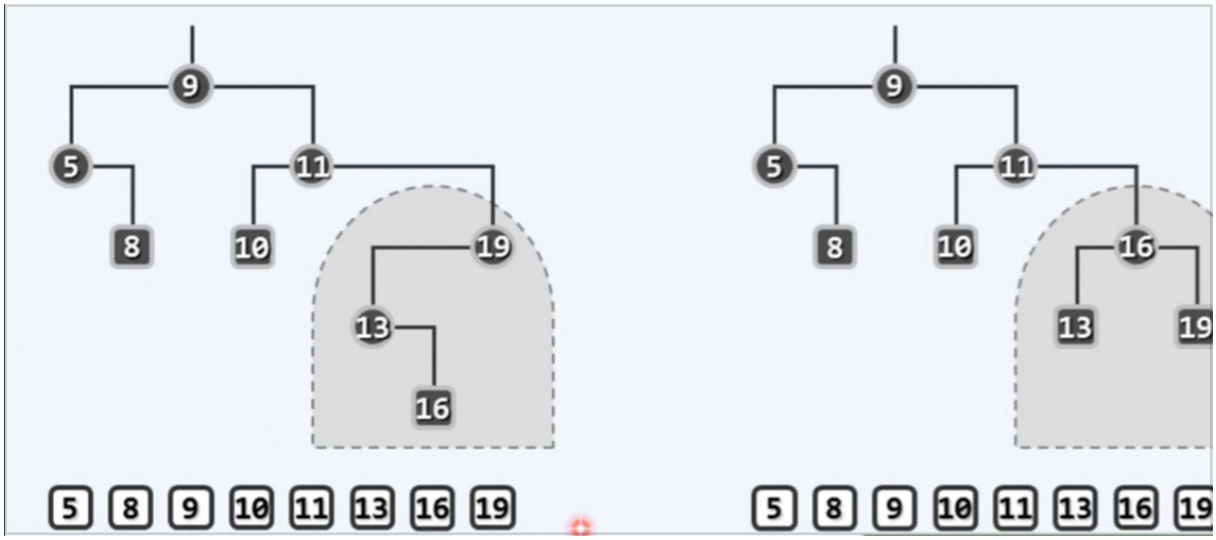


适度平衡



BBST : 平衡二叉搜索树 Balanced Binary Search Tree

歧义 = 等价 | 等价BST



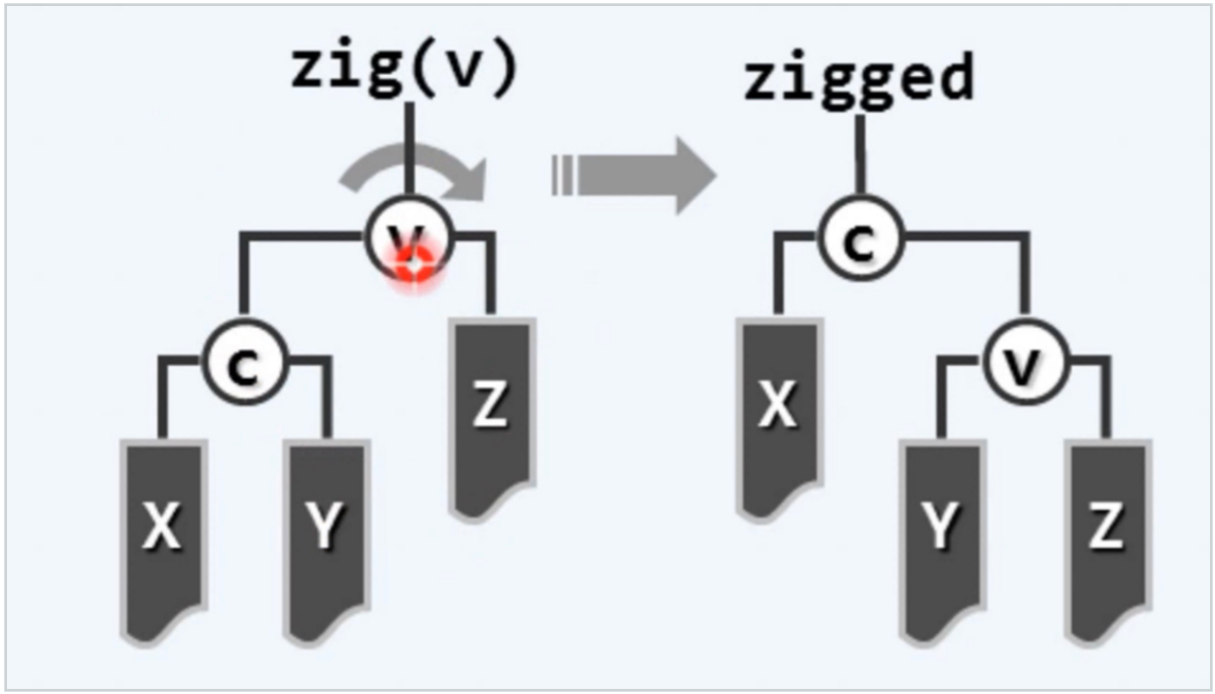
这两颗BST尽管结构不同但是中序遍历序列是完全相同的

而针对BBST这样的问题，歧义性却非常重要，我们可以在不破坏BST情况下优化高度

我们将拓扑结构不相同但是中序遍历相同的BST成为等价的BST

等价变换 + 旋转调整

1. Zig

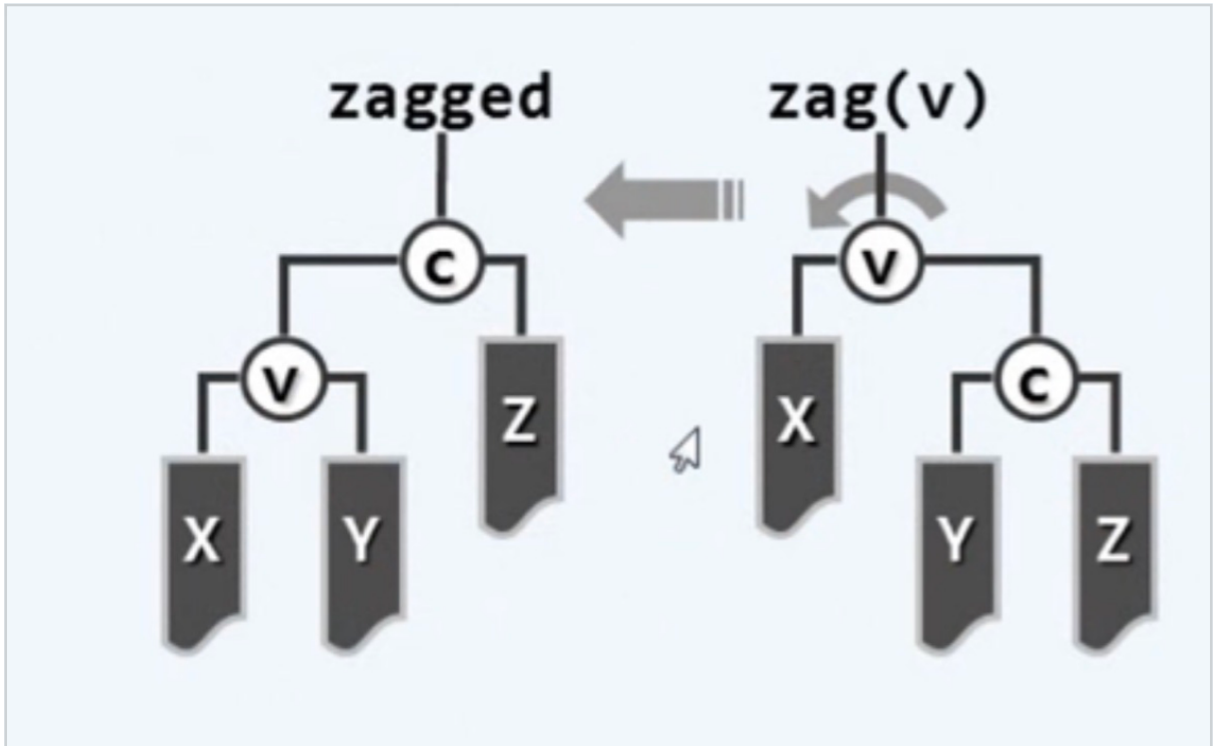


可以这样调整 中序遍历结果不变

可以视为一种旋转

当然我们也可以反过来旋转

2. Zag



各种BBST都定义了平衡的准则：能够保证任何一颗BBST在经过插入或者删除的操作后会暂时的游离到BBST之外，但是又立刻会被其算法本身拉回

1. 我们执行的每一次等价变化都应该限制在常数规模的局部 保证时间复杂度为O(1)
2. 将刚刚失衡的BBST恢复的操作的时间规模不能超过 LOGN 否则会得不偿失