红黑树

二叉搜索树

基本性质

1. 若它的左子树不为空，左子树上所有节点的值都小于它的根节
2. 若它的右子树不为空，右子树上所有的节点的值都大于（等于）它的根节点

操作

1. 插入
2. 删除
3. 查找

没有平衡性质，最坏情况下退化成链表

自平衡二叉搜索树（简称平衡二叉树）

AVL树

平衡性质

1. 任何节点的两个子树的高度差不大于1

操作

1. 查找
2. 插入
3. 删除

红黑树

平衡性质

1. 所有节点非红即黑
2. 根结点是黑色
3. 所有叶子节点都是黑色结点（NIL结点）
4. 红色结点的左右子节点都是黑色的

5. 对于任意节点而言，从起开始其到叶子点树NIL指针的每条路径都包含相同数目的黑节点

B树（自平衡多叉搜索树）

平衡性质（维持树高的规则）

1. 每一个节点最多有 m 个子节点
2. 每一个非叶子节点（除根节点）最少有 ⌈$m/2$⌉ 个子节点
3. 如果根节点不是叶子节点，那么它至少有两个子节点
4. 有 k 个子节点的非叶子节点拥有 $k-1$ 个键
5. 所有的叶子节点都在同一层

2-3-4树（4阶B树）

> **2-3-4 Trees**
>
> A 2-3-4 tree is a balanced search tree having three types of nodes- (i) a 2-node has one key and two child nodes (just like binary search tree node), (ii) a 3-node has two keys and three child nodes,
>
> https://algorithmtutor.com/Data-Structures/Tree/2-3-4-Trees/

平衡性质

1. 每一个节点最多有4个子节点
2. 非叶子节点，2-节点有1个元素2个子节点
3. 非叶子节点，3-节点有2个元素3个子节点
4. 非叶子节点，4-节点有3个元素4个子节点
5. 所有的叶子节点都在同一层

通过2-3-4树理解红黑树

> 2-3-4树过程可视化（4阶B树）

> **B-Tree Visualization**
>
> B-Trees Algorithm Visualizations
>
> https://www.cs.usfca.edu/~galles/visualization/BTree.html

> 红黑树过程可视化

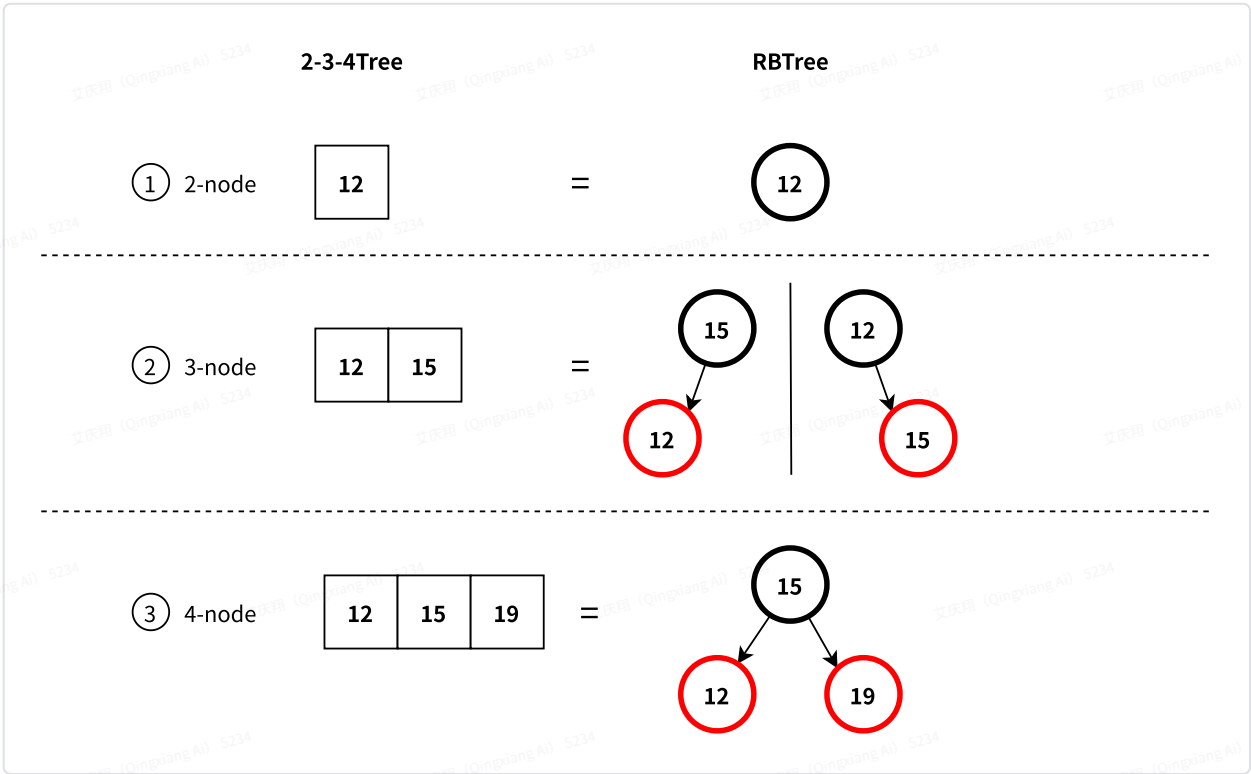
Red/Black Tree Visualization

Red/Black Tree Algorithm Visualizations

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

1. 2-/3-/4-节点对应的红黑树子树

a. 2-3-4树对应的每个红黑树子树高度（路径上黑色节点数量）都为1



b. 因此，一棵2-3-4树对应多棵红黑树

2. 操作

a. 如何维持树高（调整）

i. 说明

1. 调整的目的是为了解决不平衡的情况
2. 调整是递归操作，如果发生了需要调整的情况要一直向上调整直到满足平衡条件
3. 插入会发生插入调整，一直向上调整直到平衡。插入和插入调整是两个过程
4. 删除会发生删除调整，一直向上调整直到平衡。删除和删除调整是两个过程

ii. 分类

1. 2-3-4树
   - 旋转(rotate)
   - 分裂(split)

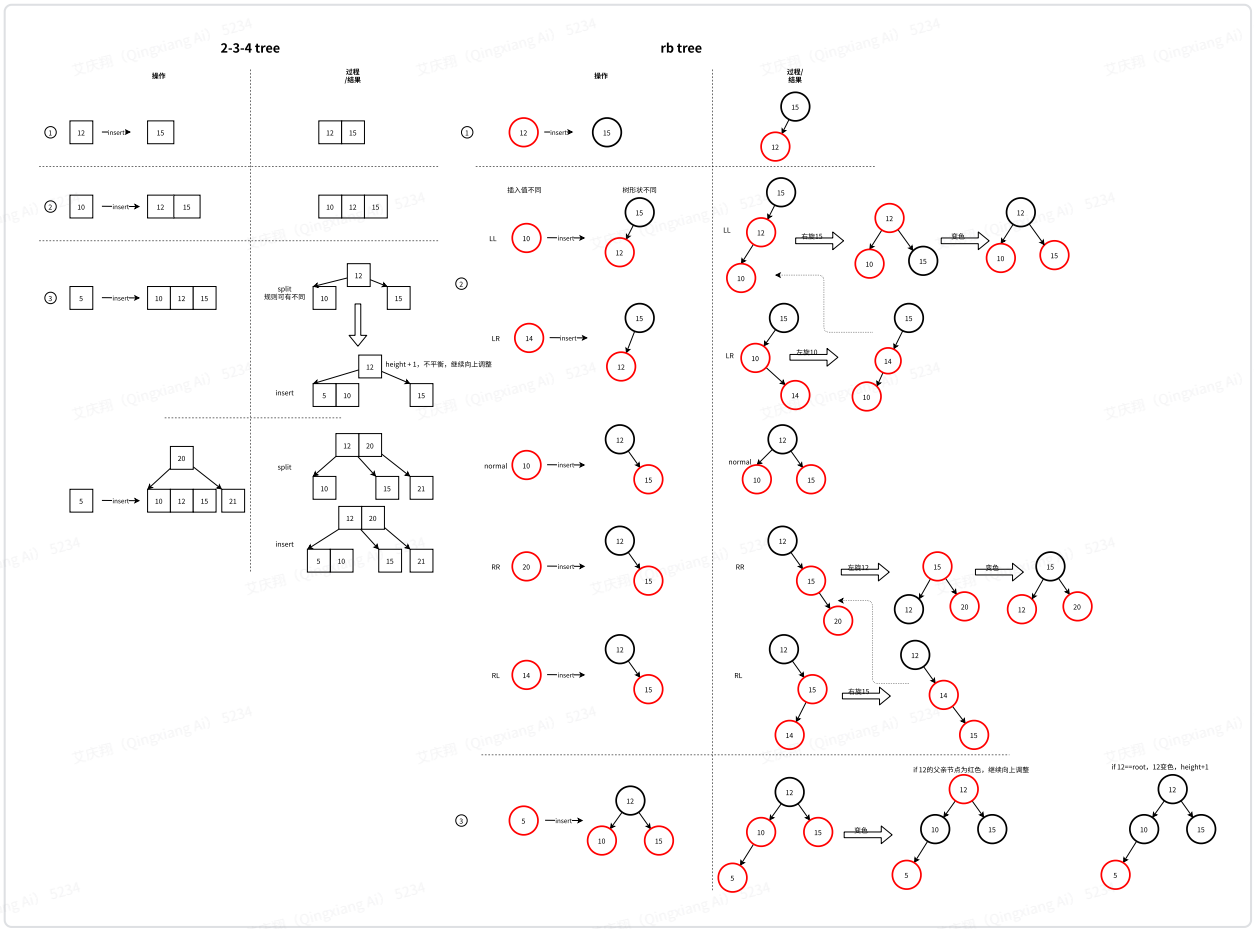
- 合并(merge)

2. 红黑树
   a. 旋转(rotate)
   b. 变色(discolor)

b. 添加

i. 添加操作仅发生在叶子结点

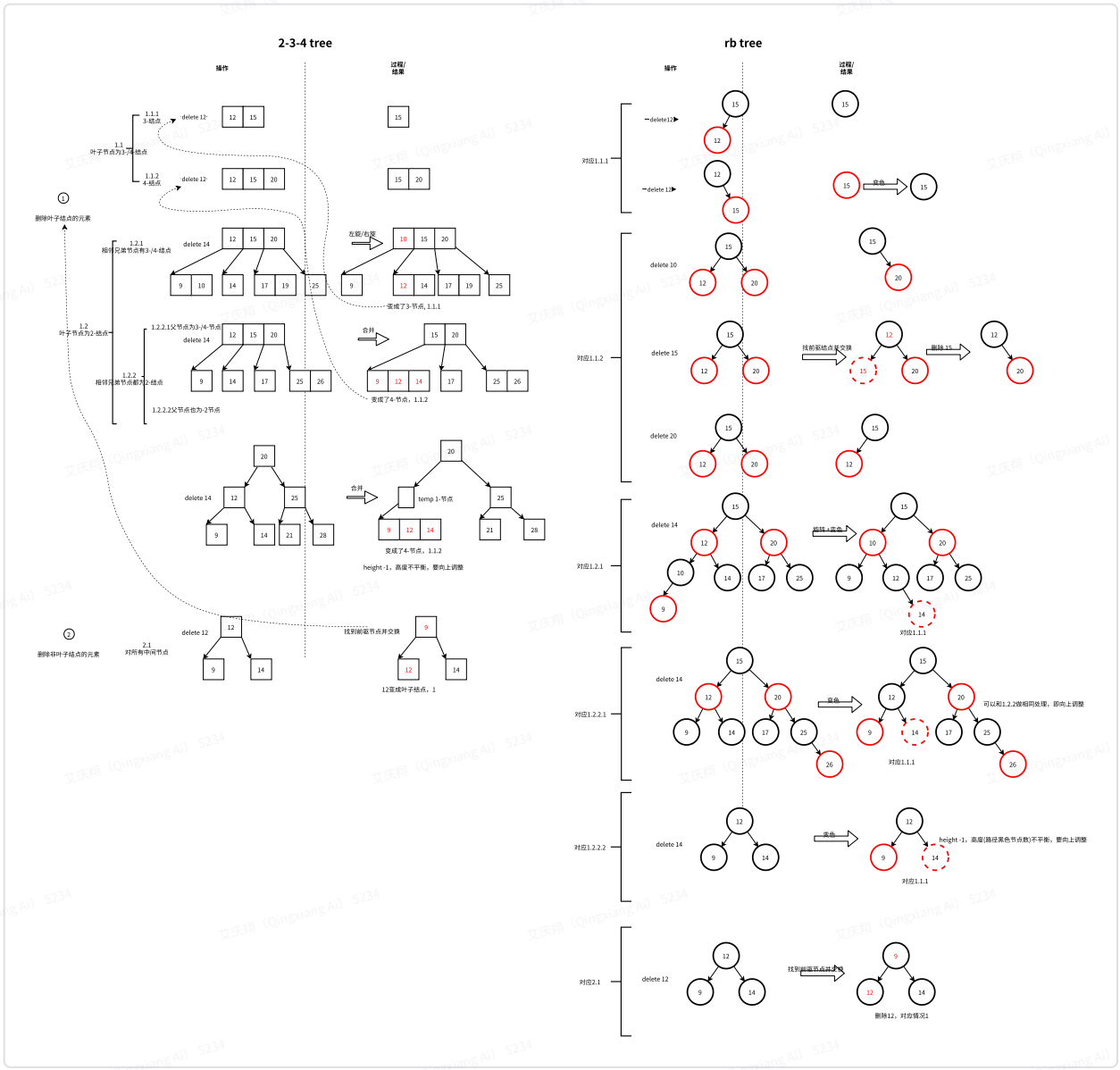
ii. 红黑树新增节点是红色的



c. 删除

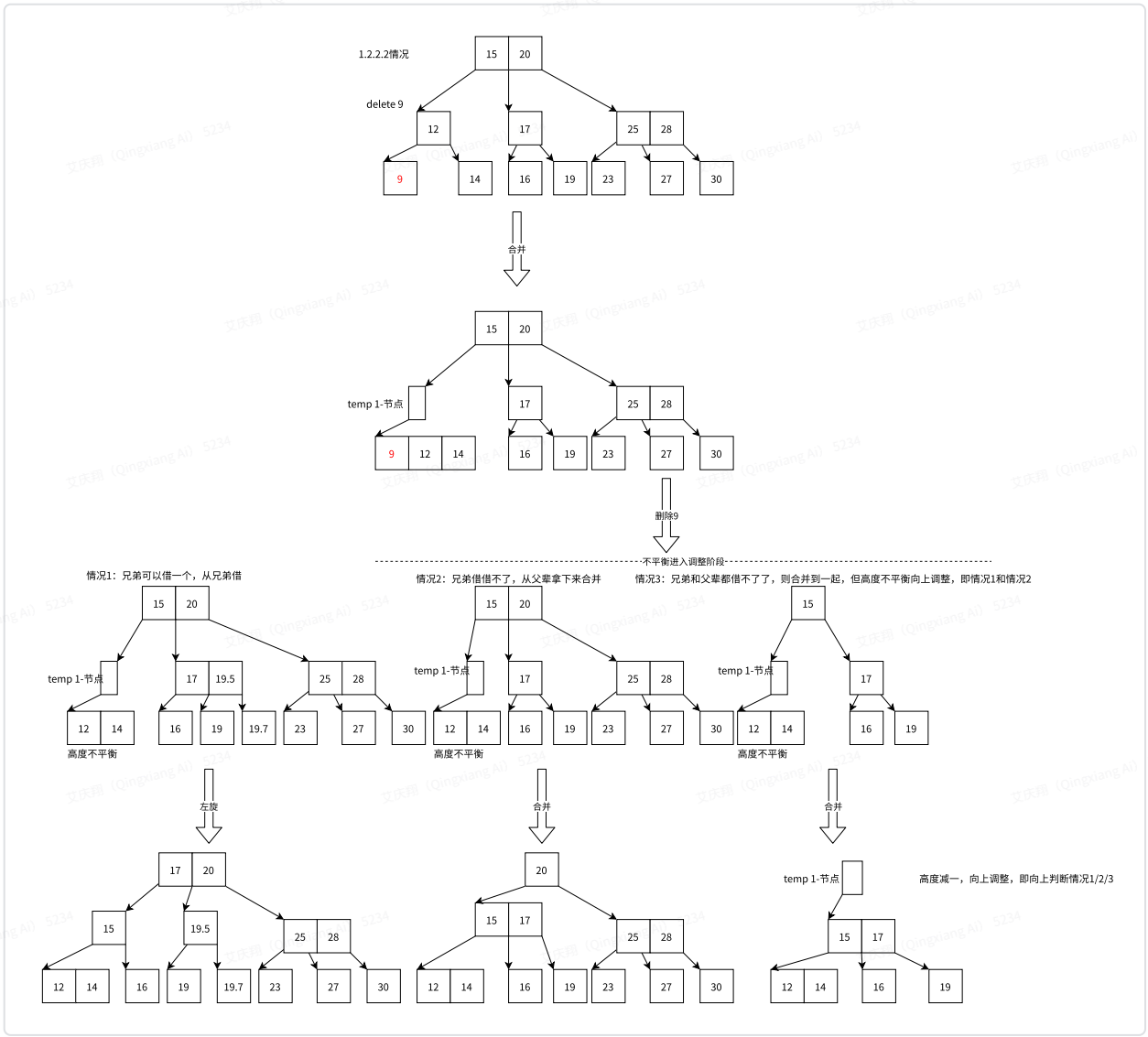
i. 如果删除操作没有发生在叶子结点，则转化成删除叶子结点

ii. 如果删除操作没有发生在3-/4-节点上，则转换成3-/4-节点

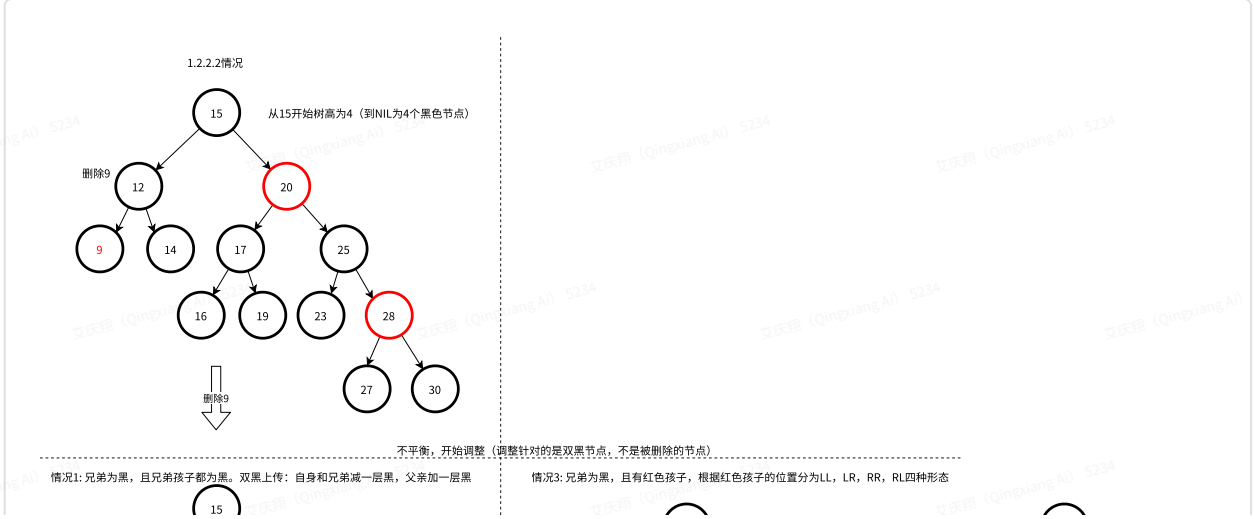


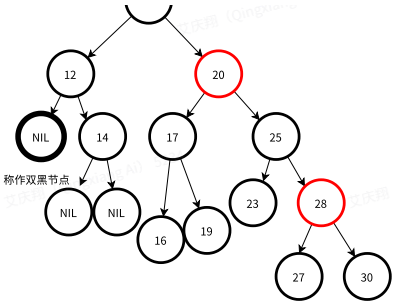
d. 删除调整过程

- 2-3-4树
  - 用临时状态1-节点代表不平衡的情况（1-节点即没有元素，只有一个孩子的节点，用于保持树高平衡）（如何判断1-节点：有且仅有一个孩子）
  - 删除调整的过程就是不断尝试将1-节点填充（从父/兄弟节点借若干个节点填充过来）直到变成普通节点的过程（矛盾转移）
  - 调整到根结点变成1-节点，1-节点的孩子变成新的根结点，树高减1

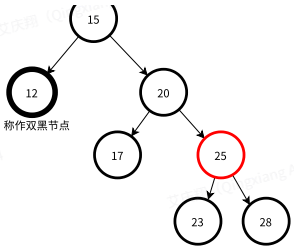


- 红黑树
  - 用临时状态双黑节点代表不平衡的情况（双黑节点即删除了黑色节点导致的路径黑色节点数量变化，赋上临时的双黑颜色保持树高平衡）（如何判断双黑节点：节点颜色用0/1/2表示，2即两层黑色，但stl源码没有这么实现，需要继续看）
  - 删除调整过程就是不断尝试将双黑节点上传（自身和兄弟减一层黑，父亲加一层黑）直到双黑节点消失的过程（矛盾转移）
  - 调整到根结点仍是双黑节点，直接变成普通黑节点，树高减一

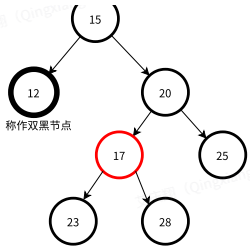




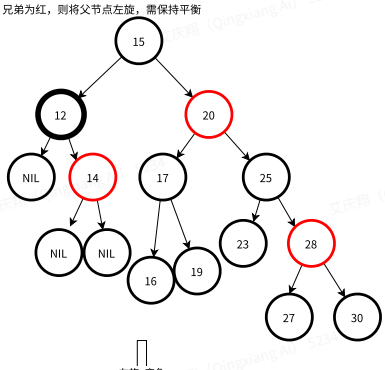
RR形



RL形

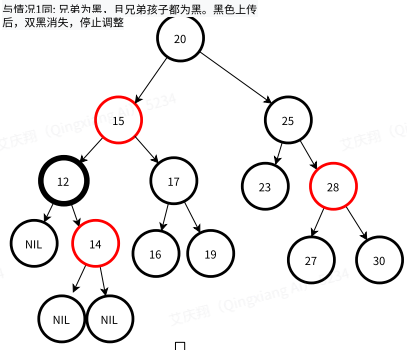


情况2: 兄弟为红，则将父节点左旋，需保持平衡



左旋+变色

与情况1同: 兄弟为黑，且兄弟孩子都为黑。黑色上传后，双黑消失，停止调整



变色