数据结构与算法——2-3-4树 原创

mb5fe18fab305a5 2021-01-21 09:58:03

文章标签 java 文章分类 Java 编程语言 阅读数 201

1）引言

在前几篇文章中介绍了 2-3 树的定义以及插入删除操作。本篇文章将在 2-3 树的基础上更进一步，介绍比 2-3 树更为复杂的数据结构 **2-3-4树** 。之所以介绍 2-3-4 树是因为 2-3-4 树与极为重要的红黑树有着等价关系，通过先学习2-3-4 树为后面学习红黑树打下基础，增进对于红黑树的理解。

2） 2-3-4树

2-3 树不再是单纯的二叉树了，因为 2-3 树中除了 2- 节点之外还存在 3- 节点。在 2-3 树的基础上进一步扩展， 2-3-4 树在 2-3 树的基础上添加 4- 节点。4- 节点可以存储 3 个键值，最多可以拥有 4 棵子树。

3）定义

（1）每个节点每个节点有 1、2 或 3 个 key ，分别称为 2- 节点，3- 节点，4- 节点。
（2）所有叶子节点到根节点的长度一致（也就是说叶子节点都在同一层）。
（3）每个节点的 key 从左到右保持了从小到大的顺序，两个 key 之间的子树中所有的 key 一定大于它的父节点的左 key ，小于父节点的右 key 。

返回顶部

例如图 3.1 所示的一棵 2-3-4 树:

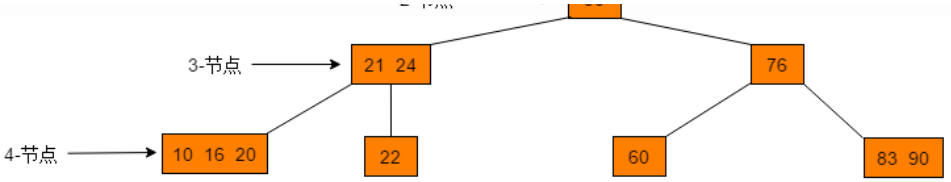
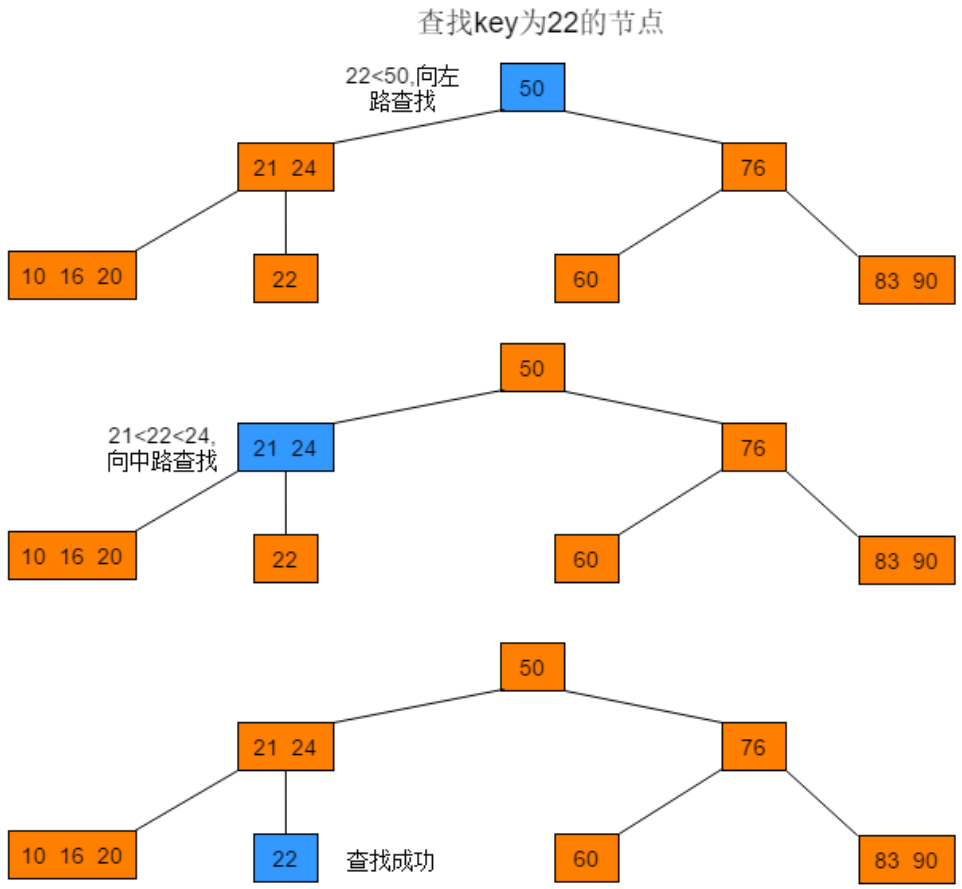


图3.1

4） 查找

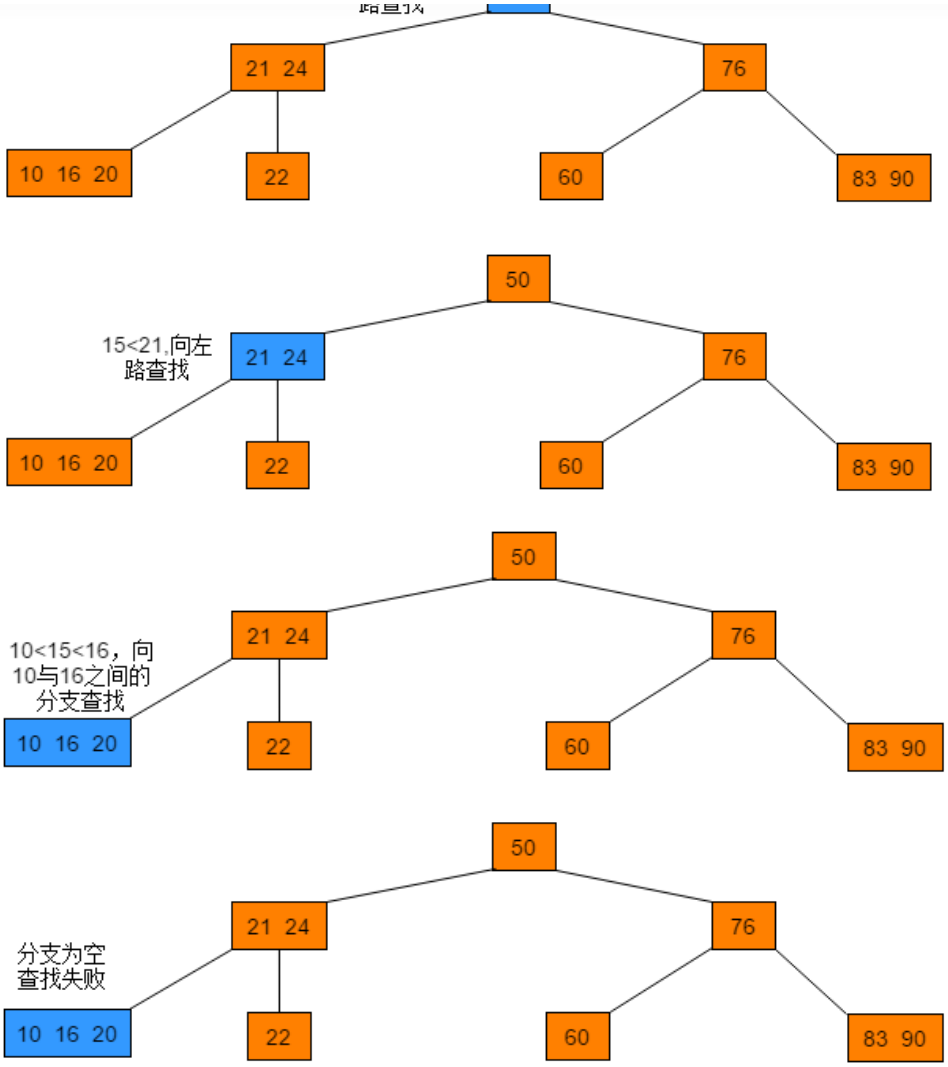
2-3-4 树的查找类似了二叉树的查找过程，通过键值的比较来决定遍历方向。

例如在图3.1所示树中查找22：



例如在图 3.1 所示树中查找 15：

返回顶部

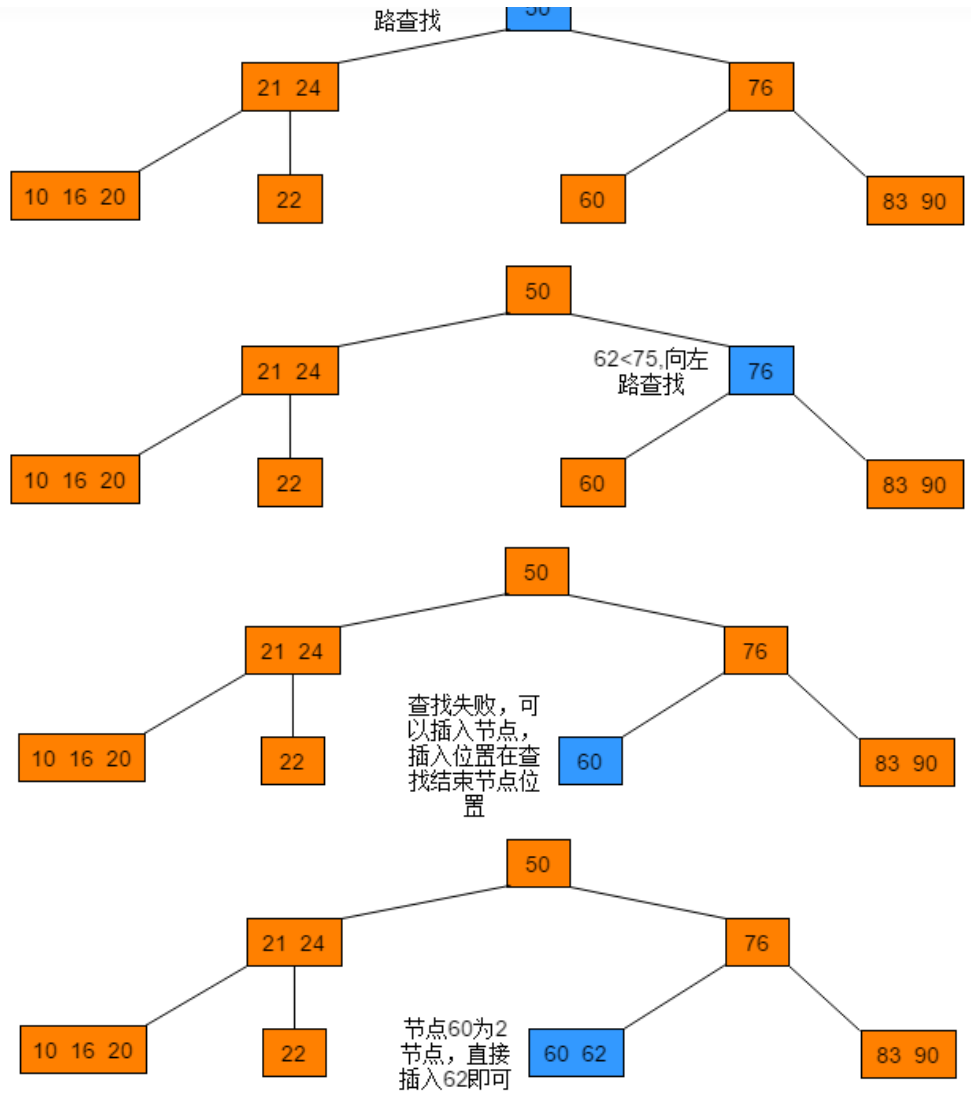


5） 插入

如果 2-3-4 树中已存在当前插入的 key ， 则插入失败，否则最终一定是在叶子节点中进行插入操作，因为查找过程的结束位置在叶子节点。

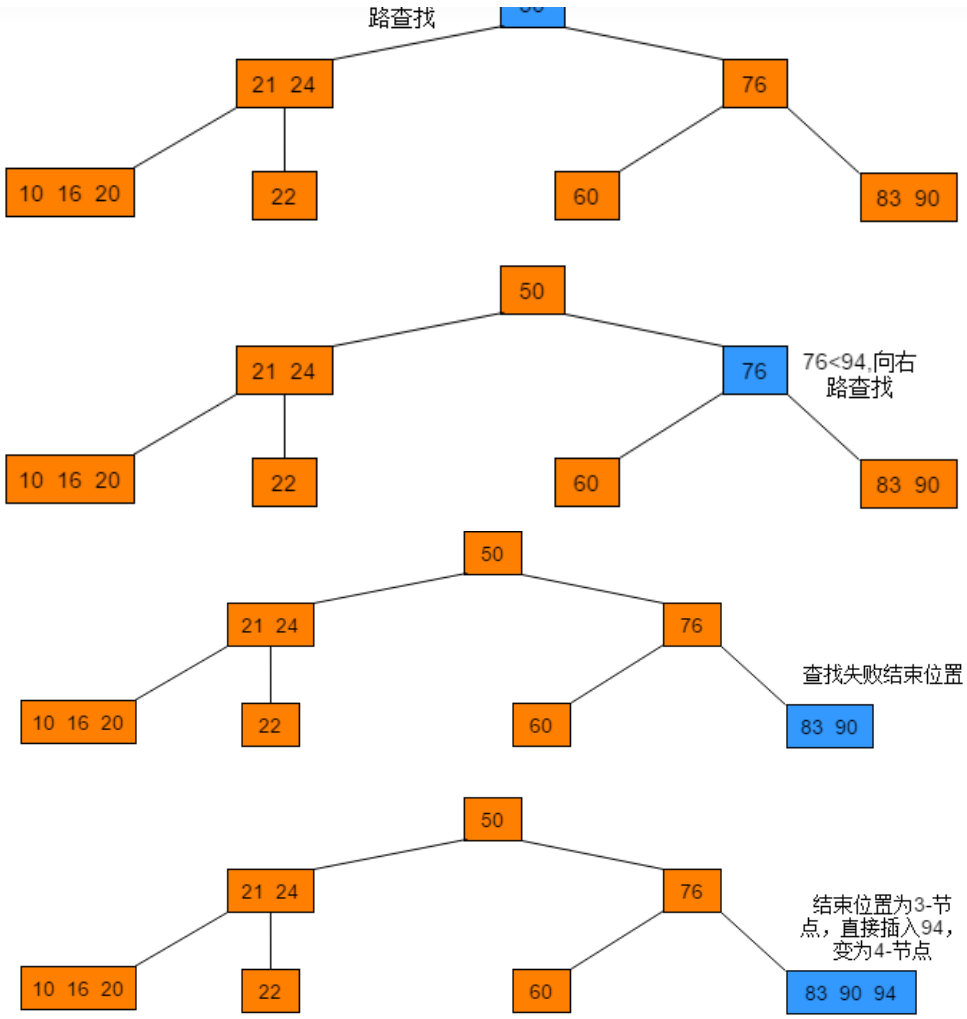
5.1 非 4- 节点插入

如果待插入的节点不是 4- 节点，那么直接在该节点插入。
例如在 2- 节点插入：



例如在 3- 节点插入：

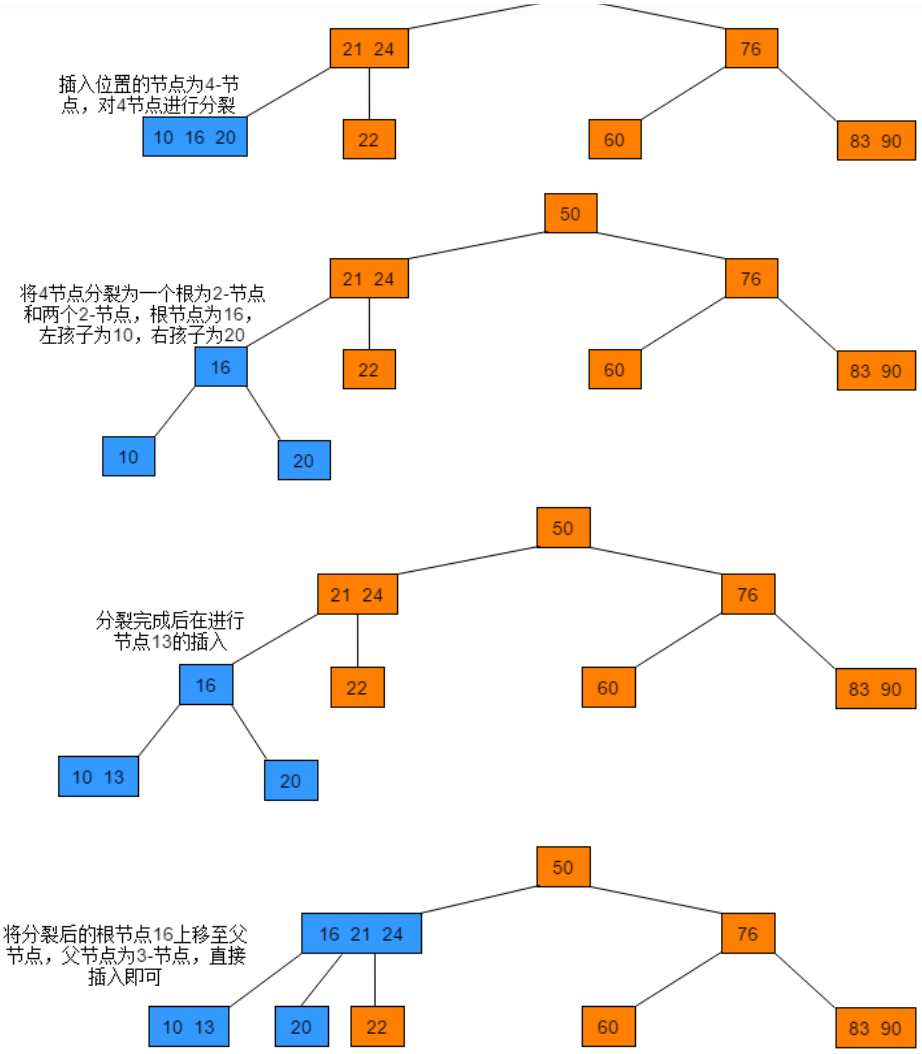
返回顶部



5.2 4- 节点插入

如果待插入的节点是个 4- 节点，那么应该先分裂该节点然后再插入。一个 4- 节点可以分裂成一个根节点和两个子节点（这三个节点各含一个 key ）然后在子节点中插入，我们把分裂形成的根节点中的 key 看成向上层插入的 key ，然后重复 5.1 和 5.2 。

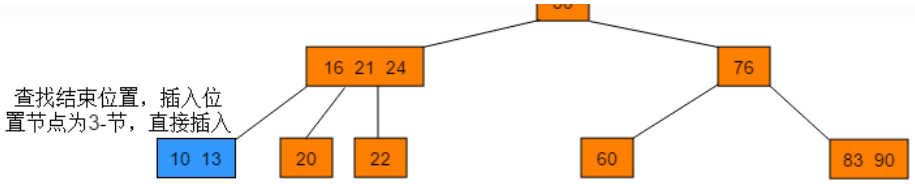
图解:

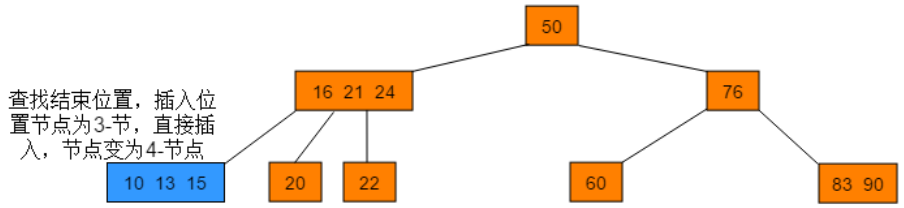


5.3 根节点分裂

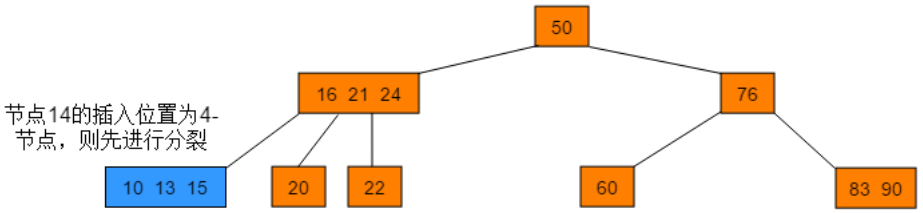
如果是在 4 节点中进行插入，每次插入会多出一个分支，如果插入操作导致根节点分裂，则 2-3-4 树会生长一层。

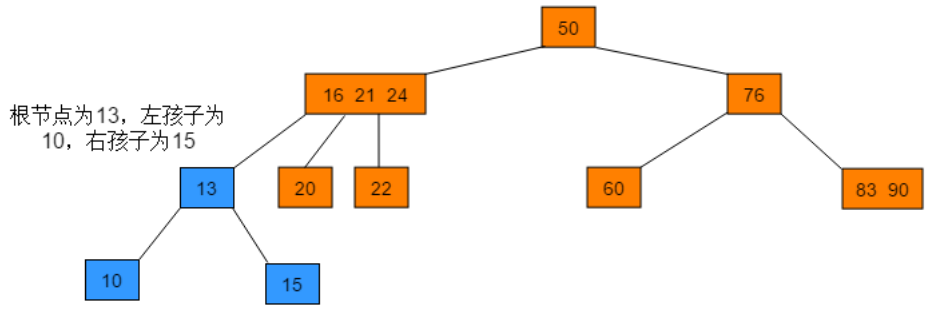
图解：

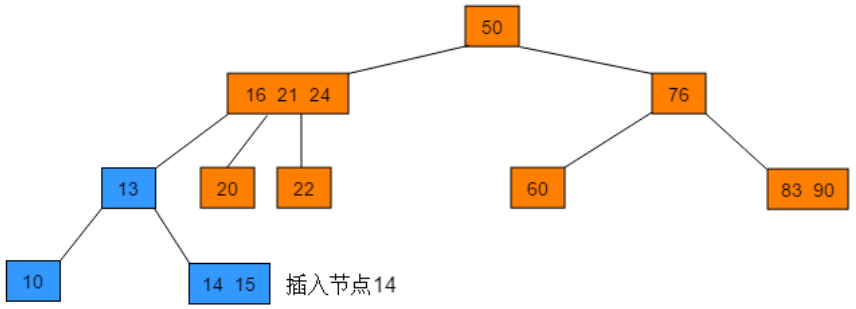


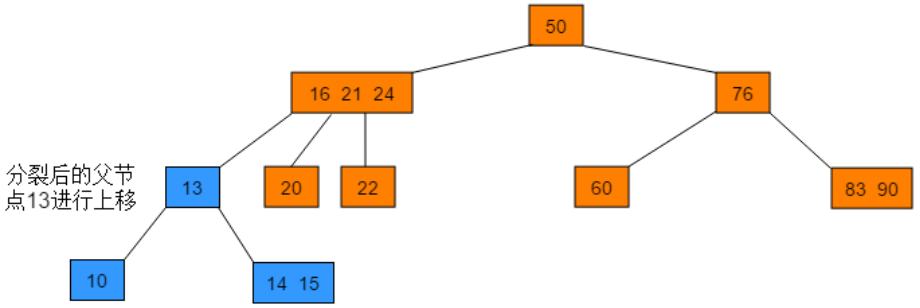


继续插入key为14的节点

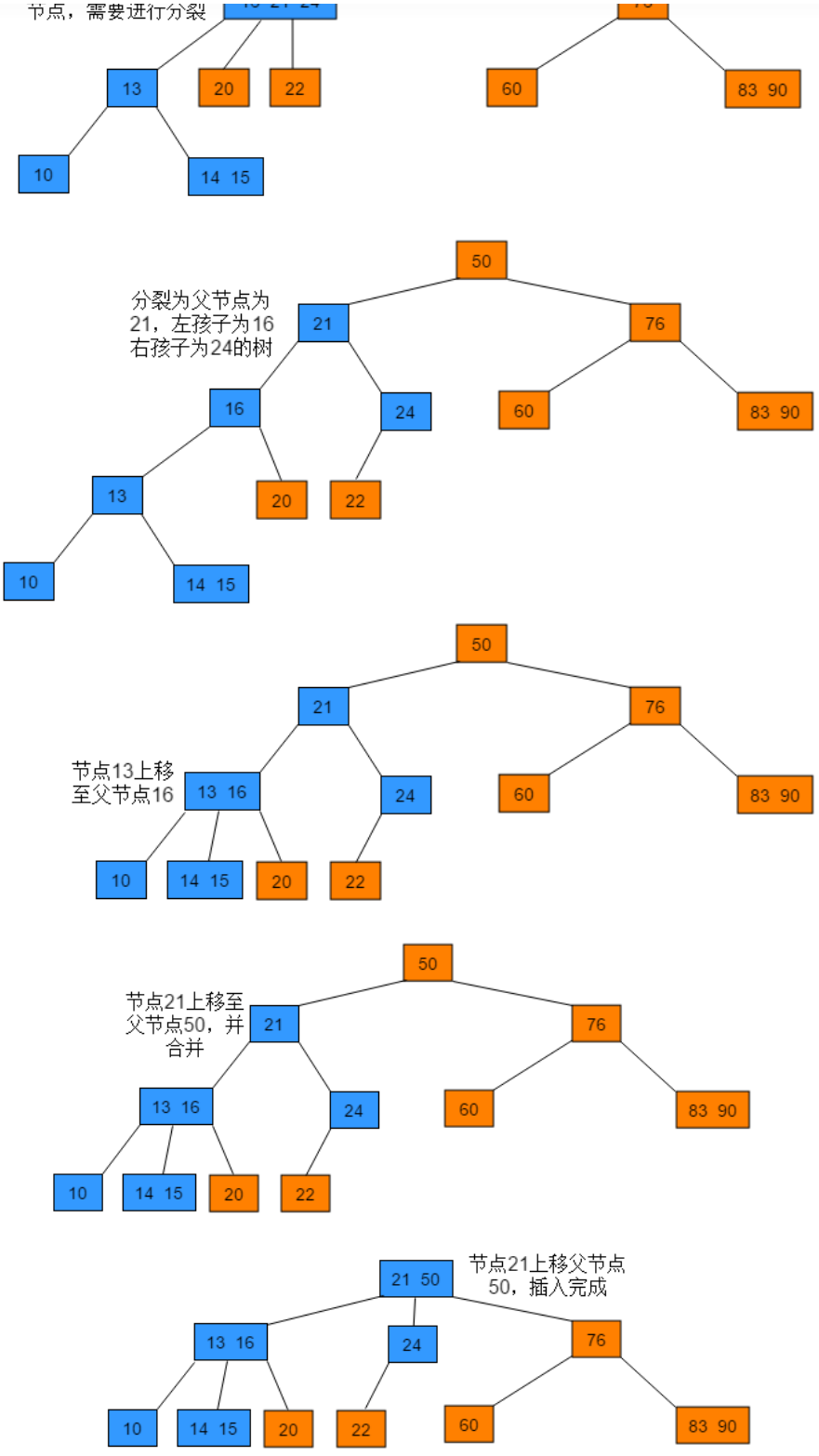








返回顶部

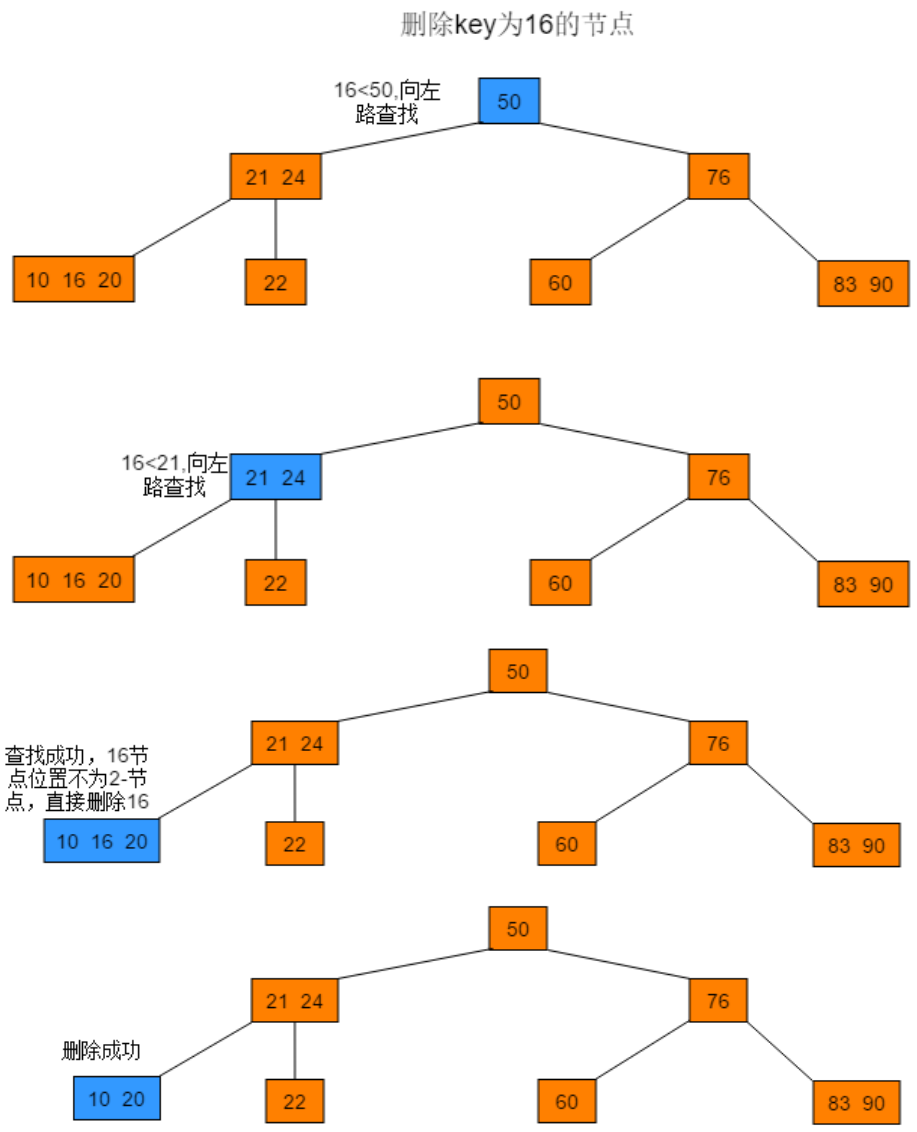


返回顶部

6） 删除

执行删除之前需要对删除 key 进行查找，若查找失败则无法删除。查找成功，才可删除 key 。删除节点情况有以下几种：

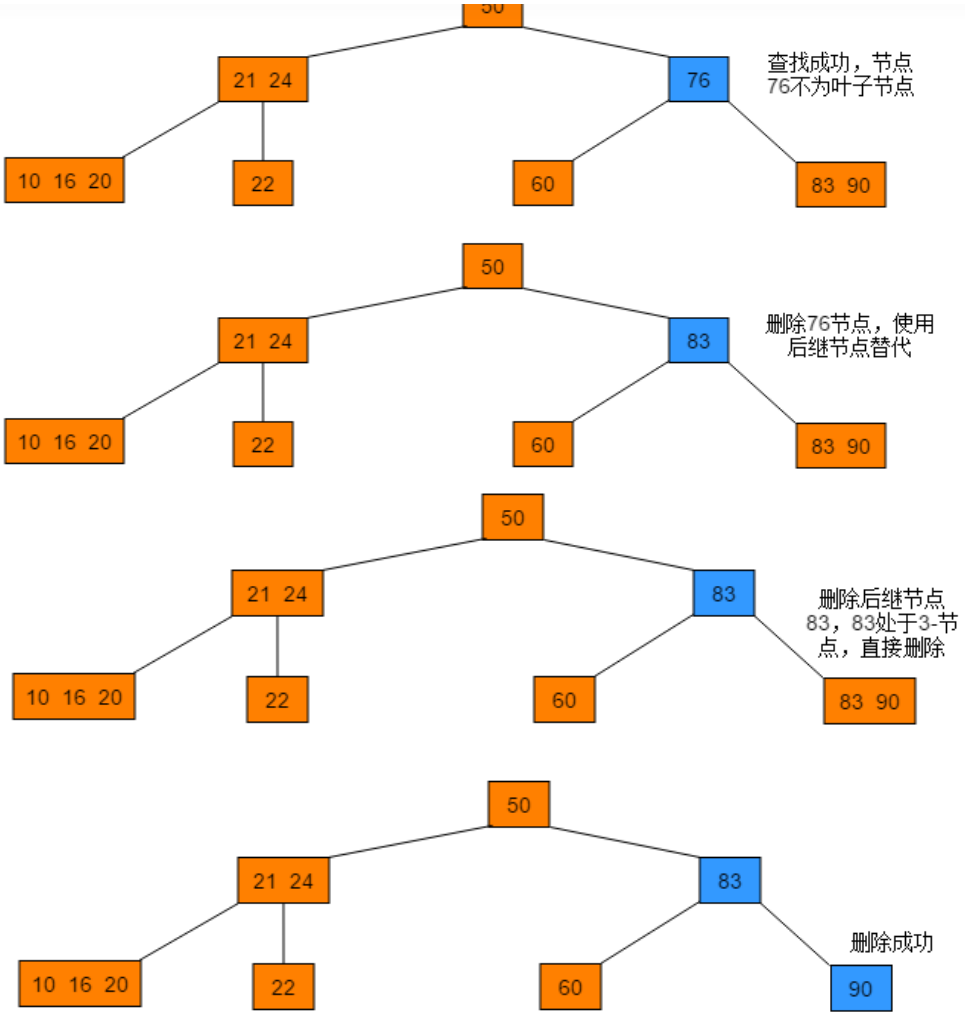
6.1 删除的节点不为 2- 节点



6.2 删除非叶子节点

当删除的节点是非叶子节点，无论待删除节点的 key 是多少个，先使用中序遍历找到待删除节点的后继节点，然后将后继节点与待删除节点位置互换，此时就将问题转化为删除节点为叶子节点（平衡树的非叶子节点中序遍历后继节点肯定叶子节点），如果该叶子是非 2- 节点，则与 2.4.1 一样，如果该节点是 2- 节点，则跟后面的 2.4.3 情形一样。

图解：



6.3 删除的叶子节点为 2- 节点

当删除的叶子节点是 2- 节点，则将节点删除后，需要对树进行调整，调整规则如下：

1）当前节点的父节点是 2- 节点，兄弟节点不为 2- 节点，则将兄弟节点的一个 key 上移成父节点，而父节点下移成子节点，此时树满足 2-3-4 树，完成调整。

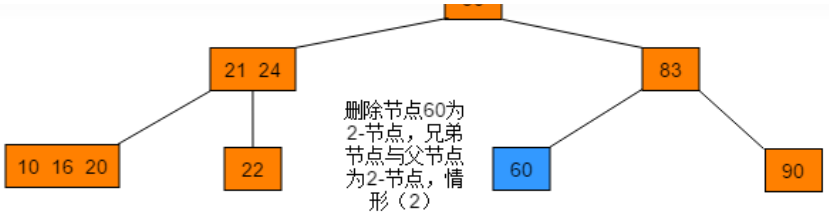
（2）当前节点的父节点是 2-节点，兄弟节点也为 2- 节点，则此时将父节点与兄弟节点合并，将合并后的节点看成当前节点，然后重复的判断，即判断合并后的当前节点的兄弟节点与父节点的情况，然后走对应的（1）（2）（3）处理，直到满足 2-3-4 树，完成调整。

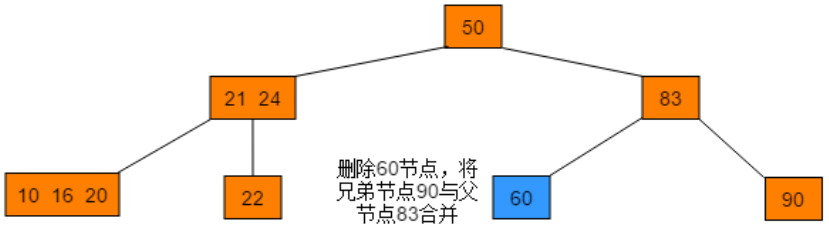
（3）当前节点的父节点不为 2- 节点，即此时有两个或三个兄弟节点，此时需要根据相邻兄弟节点情形进[illegible]，规则如下：

返回顶部

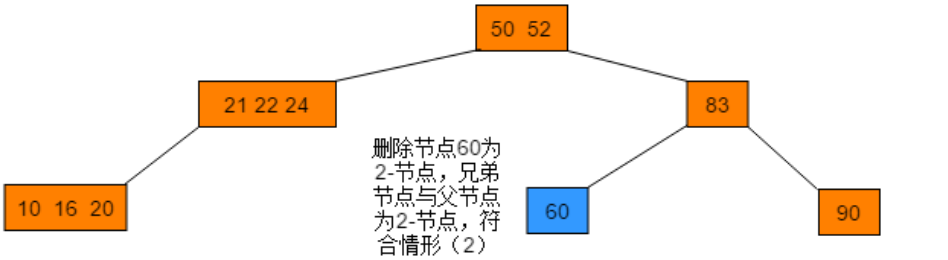
> （3）-a：若当前节点的相邻兄弟节点为非 3 个 key ，则父节点的一个 key 下移，与相邻兄弟节点合并，此时树满足2-3树，完成调整。
> （3）-b：若当前节点的相邻兄弟节点为 3 个 key ，则父节点的一个 key 下移成 1 个 key 的节点，相邻兄弟节点的一个 key 上移与父节点合并，此时树满足 2-3-4 树，完成调整。

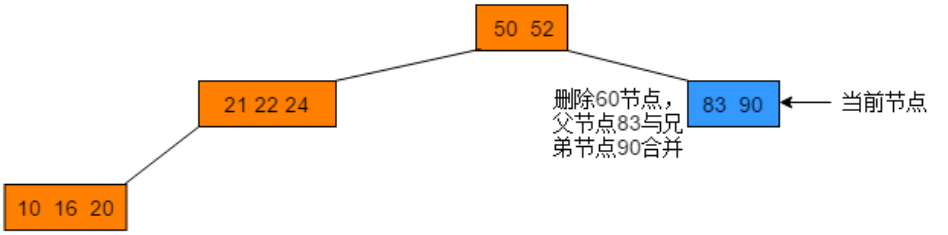
图解：

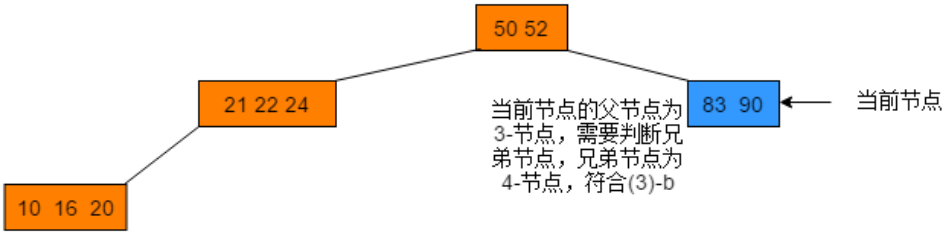




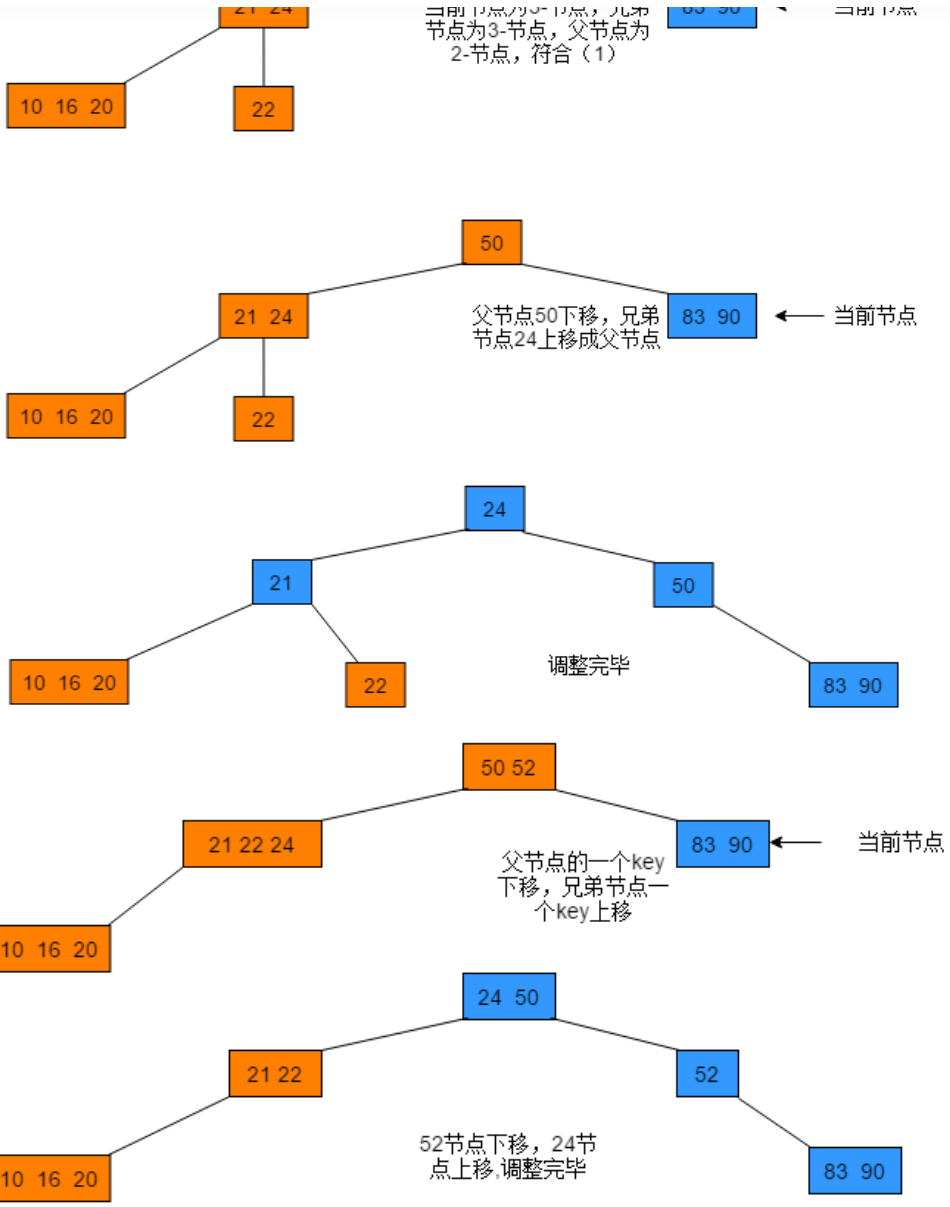
删除key为60的节点



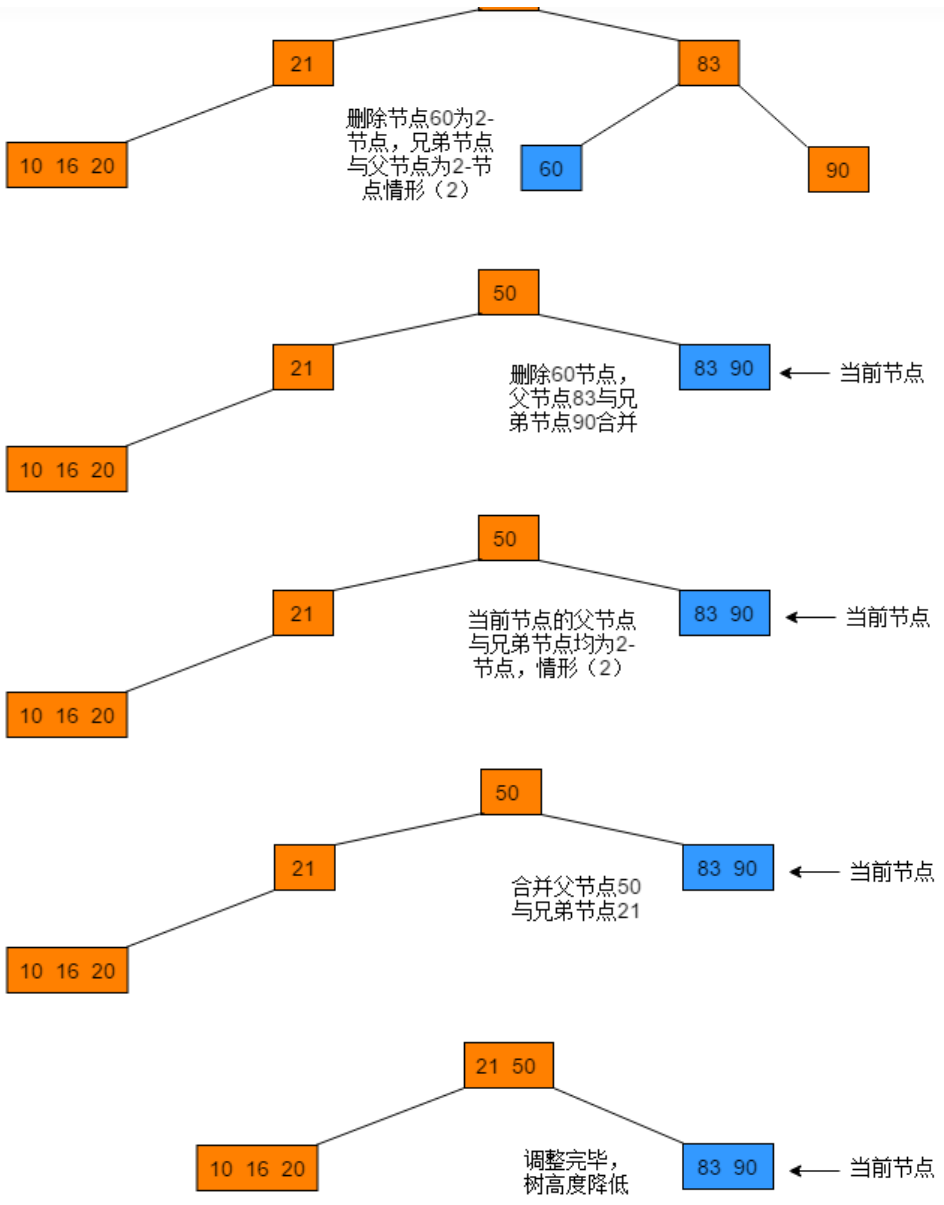




图解：


节点为3-节点，父节点为
2-节点，符合（1）
10 16 20
22
50
21 24
父节点50下移，兄弟
节点24上移成父节点
83 90
当前节点
10 16 20
22
24
21
50
10 16 20
22
调整完毕
83 90
50 52
21 22 24
父节点的一个key
下移，兄弟节点一
个key上移
83 90
当前节点
10 16 20
24 50
21 22
52
52节点下移，24节
点上移,调整完毕
10 16 20
83 90

图解：



7） 结语

本篇文章主要介绍了 2-3-4 树的性质，以及插入删除等操作。介绍 2-3-4 树的目的主要是为了为后续学习红黑树和 B- 树打下一个基础。

返回顶部

上一篇：算法科普：有趣的霍夫曼编码 下一篇：算法科普：有趣的游程编码