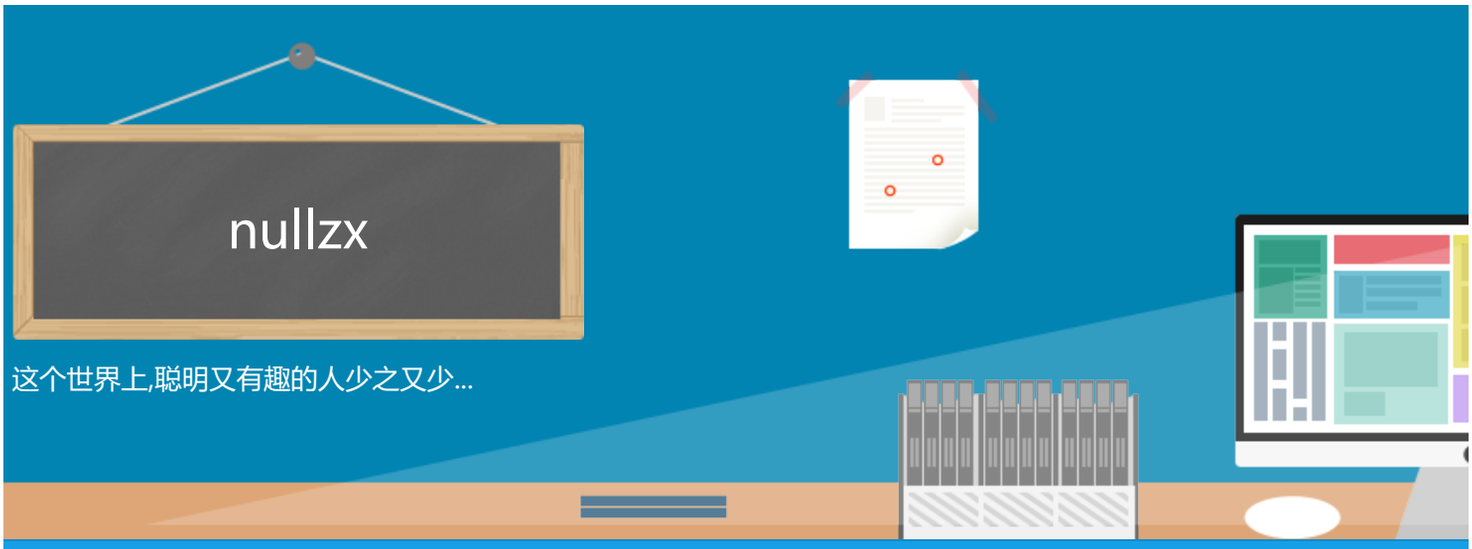


从2-3-4树到红黑树（上）

欢迎探讨，如有错误敬请指正

如需转载，请注明出处 http://www.cnblogs.com/nullzx/

相关博客：

从2-3-4树到红黑树（中）

从2-3-4树到红黑树（下）

1. 2-3-4树的定义

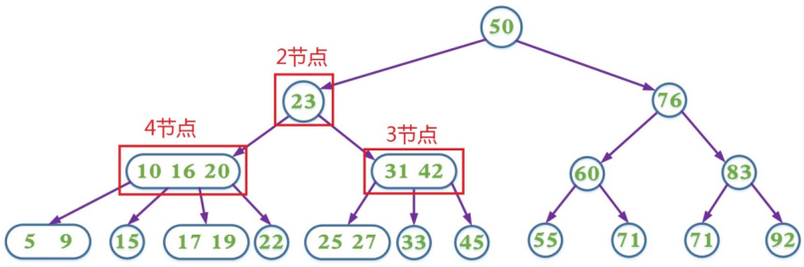
2-3-4树是一种阶为4的B树。它是一种自平衡的数据结构，可以保证在O(lgn)的时间内完成查找、插入和删除操作。它主要满足以下性质：

（1）每个节点每个节点有1、2或3个key，分别称为2（孩子）节点，3（孩子）节点，4（孩子）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。



2. 插入操作

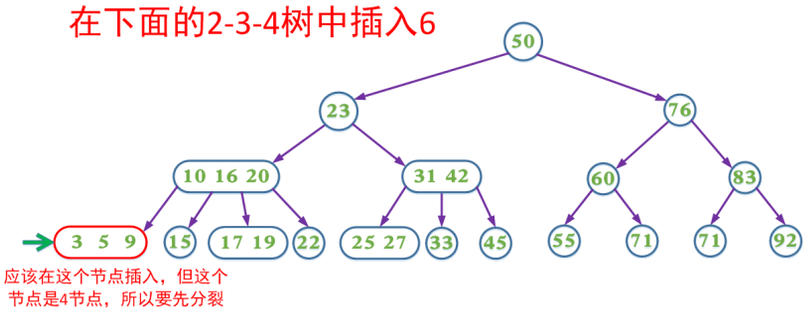
（1）如果2-3-4树中已存在当前插入的key，则插入失败，否则最终一定是在叶子节点中进行插入操作

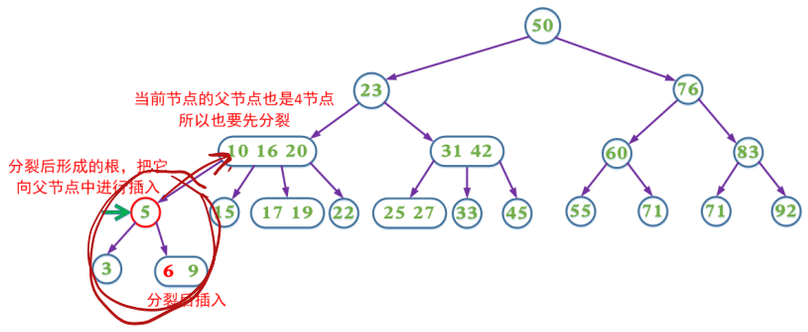
（2）如果待插入的节点不是4节点，那么直接在该节点插入

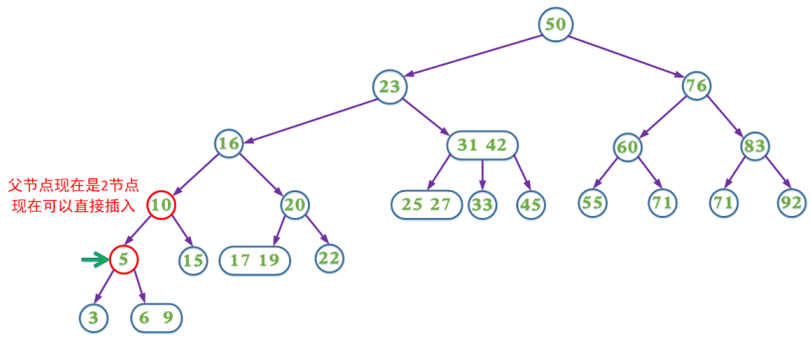
（3）如果待插入的节点是个4节点，那么应该先分裂该节点然后再插入。一个4节点可以分裂成一个根节点和两个子节点（这三个节点各含一个key）然后在子节点中插入，我们把分裂形成的根节点中的key看成向上层插入的key，然后重复第2步和第3步。

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。

在下面的2-3-4树中插入6

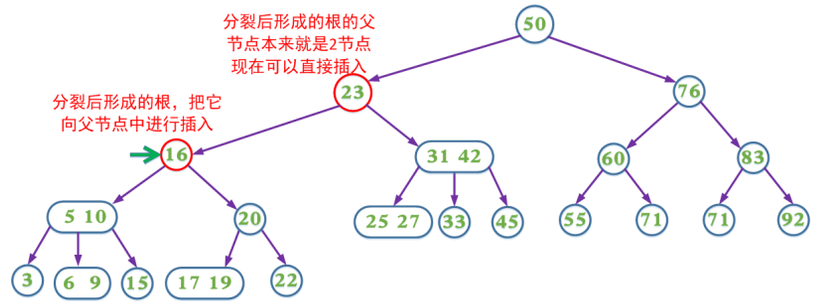


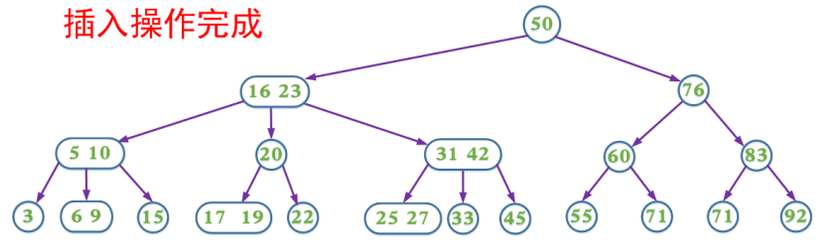




随笔分类

阅读排行榜





3. 删除操作

（1）如果2-3-4树中不存在当前需要删除的key，则删除失败。

（2）如果当前需要删除的key不位于叶子节点上，则用后继key覆盖，然后在它后继

key所在的子支中删除该后继key。

（3）如果当前需要删除的key位于叶子节点上：

（3.1）该节点不是2节点，删除key，结束

（3.2）该节点是2节点，删除该节点：

（3.2.1）如果兄弟节点不是2节点，则父节点中的key下移到该节点，兄弟节点中的一个key上移

（3.2.2）如果兄弟节点是2节点，父节点是个3节点或4节点，父节点中的key与兄弟节点合并

（3.2.3）如果兄弟节点是2节点，父节点是个2节点，父节点中的key与兄弟节点中的key合并，形成一个3节点，把此节点看成当前节点（此节点实际上是下一层的节点），重复步骤3.2.1到3.2.3

如果是在2节点（叶子节点）中进行删除，每次删除会减少一个分支，如果删除操作导致根节点参与合并，则2-3-4树会降低一层。

3. Tarjan算法：求解图的割点与桥（割边）(23890)

4. 多模字符串匹配算法之AC自动机—原理与实现(20232)

5. 快速排序算法原理及实现（单轴快速排序、三向切分快速排序、双轴快速排序）(19216)

6. 线程池的工作原理及使用示例(15540)

7. C语言的标准输入输出(13392)

8. Java并发包中Semaphore的工作原理、源码分析及使用示例(12614)

9. 线程池ThreadPoolExecutor、Executors参数详解与源代码分析(12231)

10. ScheduleThreadPoolExecutor的工作原理与使用示例(11682)

评论排行榜

1. B树和B+树的插入、删除图文详解(39)

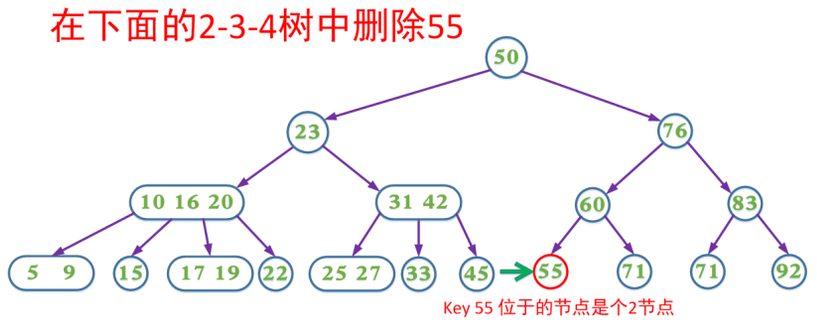
2. 多模字符串匹配算法之AC自动机—原理与实现(11)

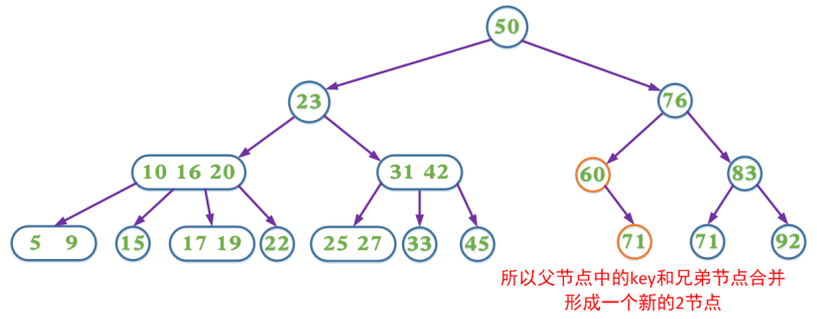
3. Tarjan算法：求解图的割点与桥（割边）(10)

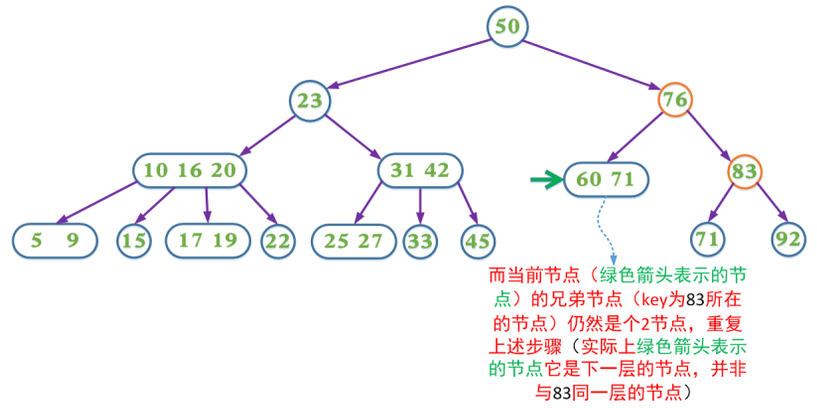
4. Kosaraju算法解析: 求解图的强连通分量(10)

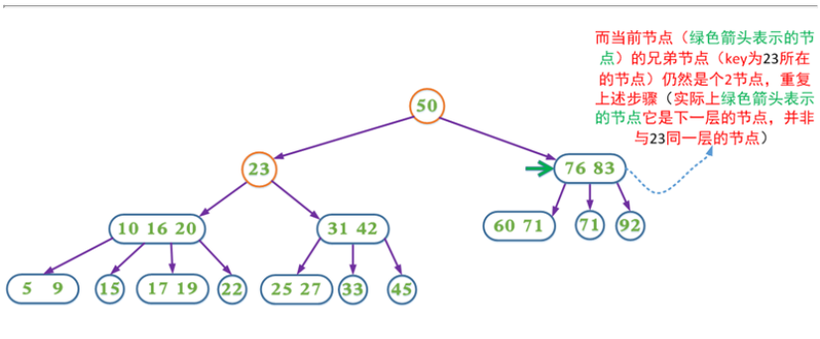
5. 1000行代码徒手写正则表达式引擎【1】--JAVA中正则表达式的使用(6)

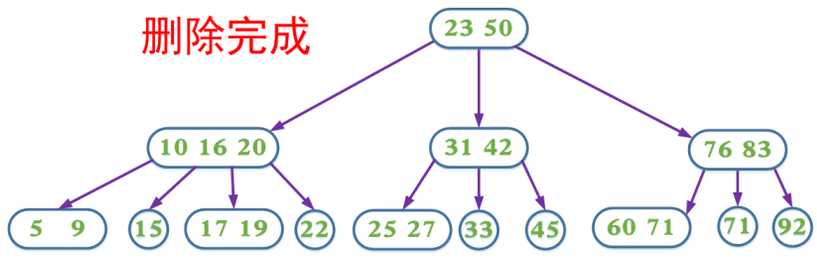
6. 快速排序算法原理及实现（单轴快速排序、三向切分快速排序、双轴快速排序）(6)











4. 带有预分裂的插入操作

上面的插入以及删除操作在某些情况需要不断回溯来调整树的结构以达到平衡。为了消除回溯过程，在插入操作过程中我们可以采取预分裂的操作，即我们在插入的搜索路径中，遇到4节点就分裂（分裂后形成的根节点的key要上移，与父

7. Java8 中 ConcurrentHashMap工作原理的要点分析(5)

8. 笨办法理解动态规划算法(4)

9. B+树在磁盘存储中的应用(4)

10. 生产者消费者模型的正确姿势(4)

推荐排行榜

1. B树和B+树的插入、删除图文详解(95)

2. Tarjan算法：求解图的割点与桥（割边）(19)

3. Kosaraju算法解析: 求解图的强连通分量(12)

4. 从2-3-4树到红黑树（上）(9)

5. 索引优先队列的工作原理与简易实现(8)

最新评论

1. Re:索引优先队列的工作原理与简易实现

牛逼

--_LittleBee

2. Re:1000行代码徒手写正则表达式引擎【1】--JAVA中正则表达式的使用

博主，可以分享下源码吗

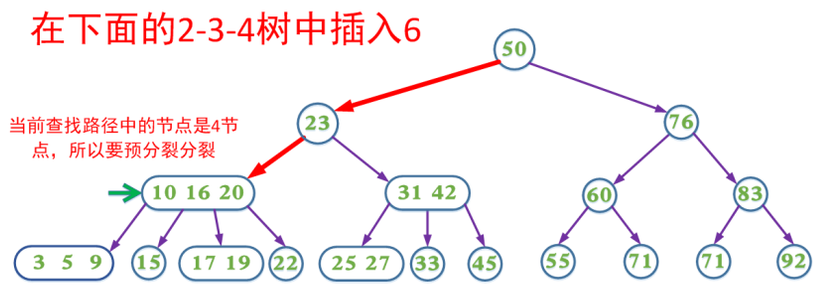
--a5365958

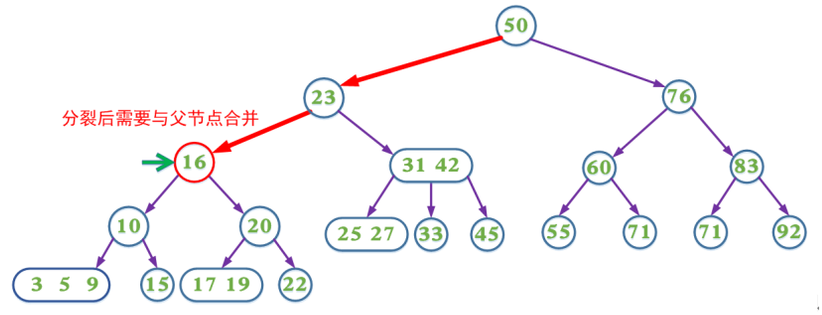
3. Re:自顶向下归并排序和自底向上的归并排序

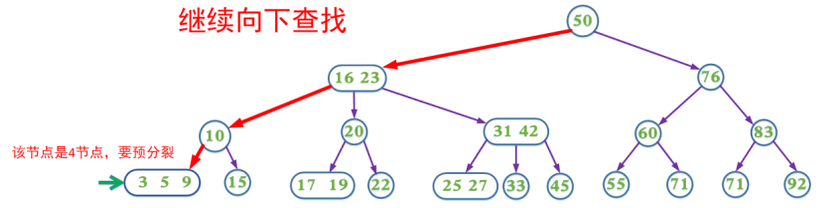
感谢

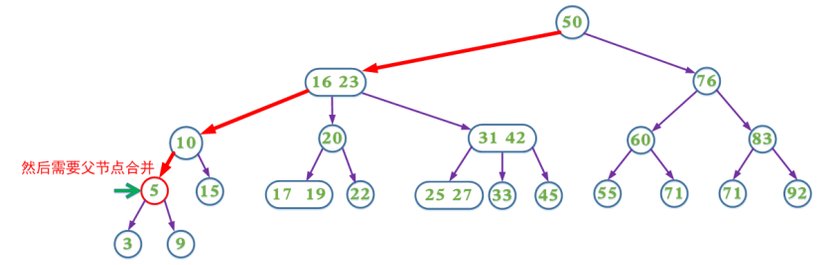
--苍迹

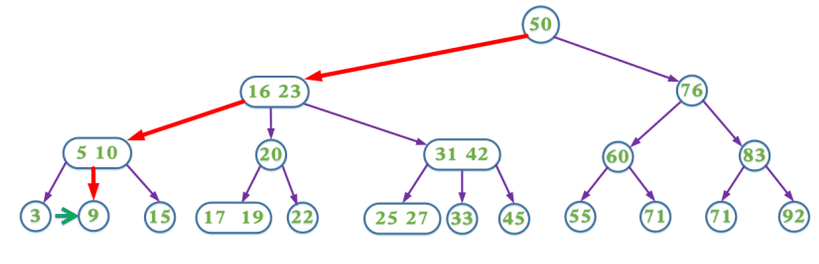
节点中的key合并）这样可以保证找到需要插入节点时可以直接插入（即该节点一定不是4节点）

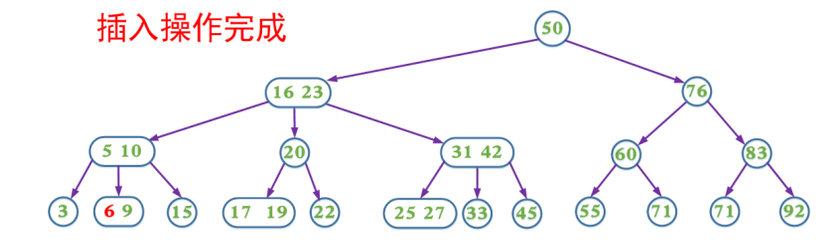












5. 带有预合并的删除操作

4. Re:B树和B+树的插入、删除图文详解

楼主，你写的很好，但是似乎在B+树上存在一些问题，严蔚敏的数据结构一书中的B+树有些出入。

--骆闻舟的小娇妻

5. Re:Kosaraju算法解析: 求解图的强连通分量

@ztx666nb 有些题目用不了tarjan...

--liqa

6. Re:B树和B+树的插入、删除图文详解

错了吧？B-树中的结点存的是data的地址，而不是data

--Binvail

7. Re:B+树在磁盘存储中的应用

“我们这时不妨再假设一个记录的大小是1KB，那么一个叶子结点可以存4个记录。而对于索引结点（大小也是4KB），由于只需要存key值和相应的指针，所以一个索引结点可能可以存储100~150个分支，我们不...

--三维感知小白

8. Re:B+树在磁盘存储中的应用

@发挥哥 我理解的是4万...

--三维感知小白

9. Re:从2-3-4树到红黑树（中）

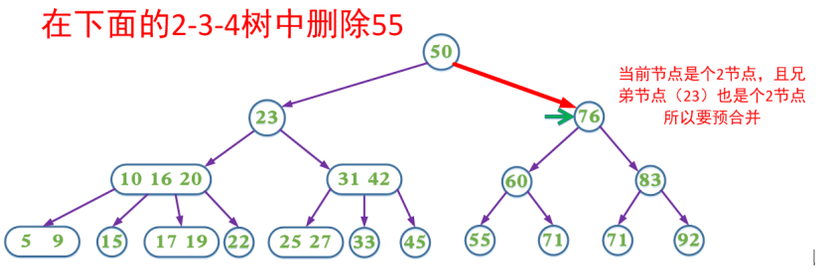
不错,感谢,收藏了

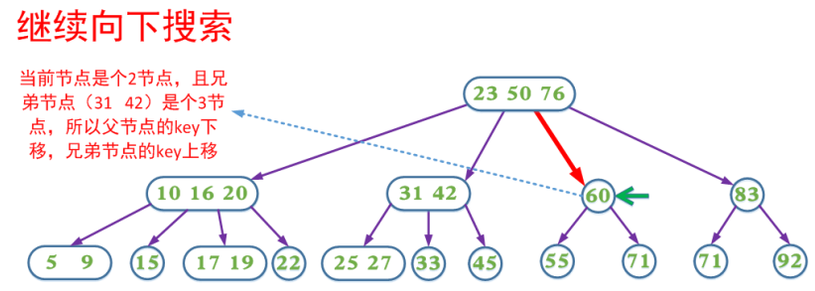
--繁星春水

10. Re:Tarjan算法：求解图的割点与桥（割边）

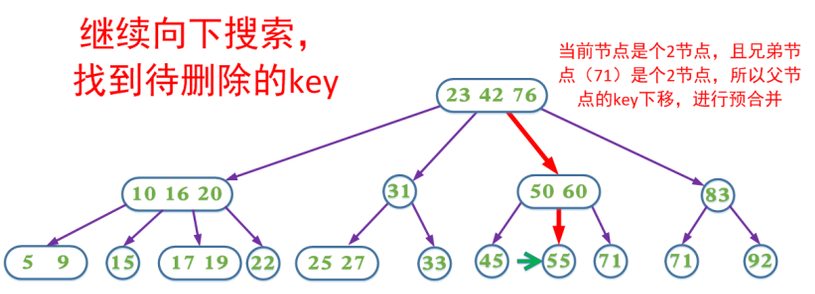
写得特别好，特别是图片实例讲解，终于学会了。谢谢！

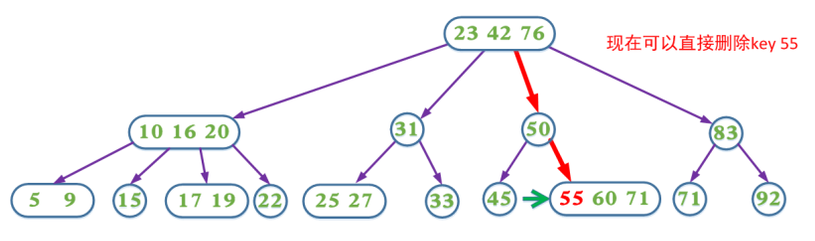
在删除过程中，我们同样可以采取预合并的操作，即我们在删除的搜索路径中（除根节点，因为根节点没有兄弟节点和父节点），遇到当前节点是2节点，如果兄弟节点也是2节点就合并（该节点的父节点中的key下移，与自身和兄弟节点合并）；如果兄弟节点不是2节点，则父节点的key下移，兄弟节点中的key上移。这样可以保证，找到需要删除的key所在的节点时可以直接删除（即要删除的key所在的节点一定不是2节点）。

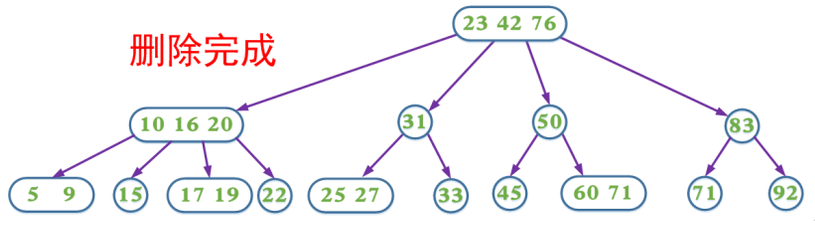




这里包含key为60的节点也可以选择让父节点中的key 76下移和兄弟节点中的83合并，两种方式都能达到B树的平衡，这也是在2-3-4树对应的红黑树中使用的方式。







--大凉龙雀

11. Re:B树和B+树的插入、删除图文详解

b+树叶子结点数和父节点中的key数量是不是搞错了，，，b+也不是这样的的呀

--persistanceBoy

12. Re:B树和B+树的插入、删除图文详解

@求教一下一个问题。 在B树的删除操作中，加入将第 d 步删除32改为删除40，会出现个没有列举出来的情况，请教一下这种情况改如何处理啊，谢谢！ “加入”改为“假如”...

--大海-f

13. Re:B树和B+树的插入、删除图文详解

求教一下一个问题。
在B树的删除操作中，加入将第 d 步删除32改为删除40，会出现个没有列举出来的情况，请教一下这种情况改如何处理啊，谢谢！

--大海-f

14. Re:从2-3-4树到红黑树（上）

@Route66 第一点，我可能看懂你的意思了，删除之后树的高度得一致，得平衡...

--season-qd

15. Re:多模字符串匹配算法之AC自动机—原理与实现

需要说明的是，当指针位于结点b（图中曲线经过了两次b,这里指第二次的b，即目标字符串“ijabdf”中的b）,这时读取文本串字符下标为9的字符（即‘d’）时，由于b的所有孩子结点（这里恰好只有一个孩子...

--vinceall

16. Re:B树和B+树的插入、删除图文详解