

# B树和B+树的插入、删除图文详解

简介：本文主要介绍了B树和B+树的插入、删除操作。写这篇博客的目的是发现没有相关博客以举例的方式详细介绍B+树的相关操作，由于自身对某些细节也感到很迷惑，通过查阅相关资料，对B+树的操作有所顿悟，写下这篇博客以做记录。由于是自身对B+树的理解，肯定有考虑不周的情况，或者理解错误的地方，请留言指出。

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/>

## 1. B树

### 1. B树的定义

B树也称B-树，它是一颗多路平衡查找树。我们描述一颗B树时需要指定它的阶数，阶数表示了一个结点最多有多少个孩子结点，一般用字母m表示阶数。当m取2时，就是我们常见的二叉搜索树。

一颗m阶的B树定义如下：

- 1) 每个结点最多有m-1个关键字。
- 2) 根结点最少可以只有1个关键字。

### 公告

本人学识渊博、经验丰富，代码风骚、效率恐怖，c/c++、java、php无不精通，熟练掌握各种框架，深山苦练20余年，一天只睡4小时，千里之外定位问题，瞬息之间修复上线。身体强壮、健步如飞，可连续编程100小时不休息，讨论技术方案5小时不喝水，上至带项目、出方案，下至盗账号、威胁pm，啥都能干。泡面矿泉水已备好，学校不支持编程已辍学，家人不支持编程已断绝关系，老婆不支持编程已离婚，小孩不支持编程已送养。

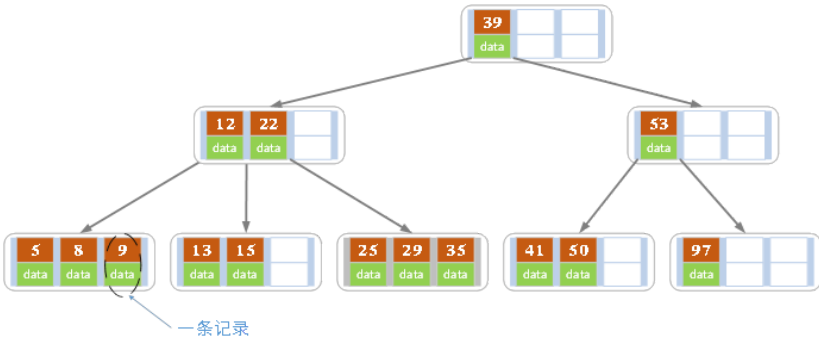
昵称： nullzx  
园龄： 6年  
粉丝： 275  
关注： 4  
[+加关注](#)

< 2021年11月 >						
日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

### 最新随笔

1.笨办法理解动态规划算法

- 3) 非根结点至少有 $\text{Math.ceil}(m/2)-1$ 个关键字。
- 4) 每个结点中的关键字都按照从小到大的顺序排列，每个关键字的左子树中的所有关键字都小于它，而右子树中的所有关键字都大于它。
- 5) 所有叶子结点都位于同一层，或者说根结点到每个叶子结点的长度都相同。



$2 \leq m < 4$

上图是一颗阶数为4的B树。在实际应用中的B树的阶数m都非常大（通常大于100），所以即使存储大量的数据，B树的高度仍然比较小。每个结点中存储了关键字（key）和关键字对应的数据（data），以及孩子结点的指针。我们将一个key和其对应的data称为一个记录。但为了方便描述，除非特别说明，后续文中就用key来代替（key，value）键值对这个整体。在数据库中我们将B树（和B+树）作为索引结构，可以加快查询速速，此时B树中的key就表示键，而data表示了这个键对应的条目在硬盘上的逻辑地址。

## 1.2 B树的插入操作

插入操作是指插入一条记录，即（key，value）的键值对。如果B树中已存在需要插入的键值对，则用需要插入的value替换旧的value。若B树不存在这个key，则一定是在叶子结点中进行插入操作。

- 1) 根据要插入的key的值，找到叶子结点并插入。
- 2) 判断当前结点key的个数是否小于等于 $m-1$ ，若满足则结束，否则进行第3步
- 3) 以结点中间的key为中心分裂成左右两部分，然后将这个中间的key插入到父结点中，这个key的左子树指向分裂后的左半部分，这个key的右子支指向分裂后的右半部分，然后将当前结点指向父结点，继续进行第3步。

$\Rightarrow$  等效为同时增加一个父节点，一个子节点

- 2.EclipseEE的Web开发环境配置（使用Tomcat作为Web服务器）
- 3.二分类神经网络公式推导过程
- 4.B+树在磁盘存储中的应用
- 5.JAVA NIO工作原理及代码示例
- 6.B树和B+树的插入、删除图文详解
- 7.Java8 中 ConcurrentHashMap工作原理的要点分析
- 8.二分搜索以及其扩展形式
- 9.Tarjan算法：求解图的割点与桥（割边）
- 10.生产者消费者模型的正确姿势

### 随笔分类

- C语言(1)
- Java 并发编程(15)
- Java基础(7)
- 机器学习(1)
- 算法(17)
- 正则表达式(1)

### 阅读排行榜

1. B树和B+树的插入、删除图文详解(159455)
2. Kosaraju算法解析: 求解图的强连通分量(37798)

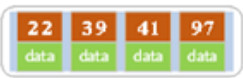
下面以5阶B树为例，介绍B树的插入操作，在5阶B树中，结点最多有4个key, 最少有2个key

a) 在空树中插入39



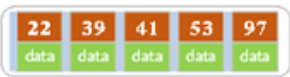
此时根结点就一个key，此时根结点也是叶子结点

b) 继续插入22，97和41



根结点此时有4个key

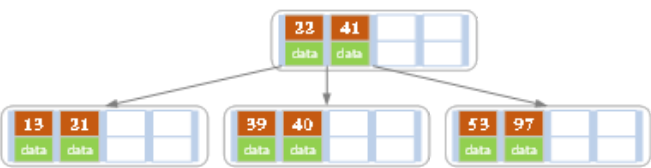
c) 继续插入53



插入后超过了最大允许的关键字个数4，所以以key值为41为中心进行分裂，结果如下图所示，分裂后当前结点指针指向父结点，满足B树条件，插入操作结束。当阶数m为偶数时，需要分裂时就不存在排序恰好在中间的key，那么我们选择中间位置的前一个key或中间位置的后一个key为中心进行分裂即可。



d) 依次插入13，21，40，同样会造成分裂，结果如下图所示。



e) 依次插入30，27，33；36，35，34；24，29，结果如下图所示。



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(15541)

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的工作原理及使用示例(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)

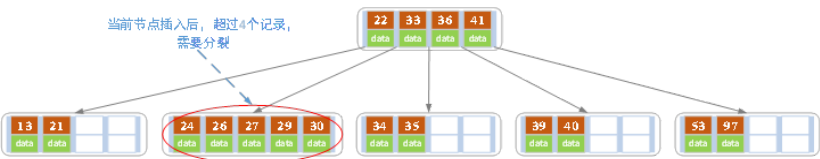
3. Tarjan算法：求解图的割点与桥（割边）(10)

4. Kosaraju算法解析: 求解图的强连通分量(10)

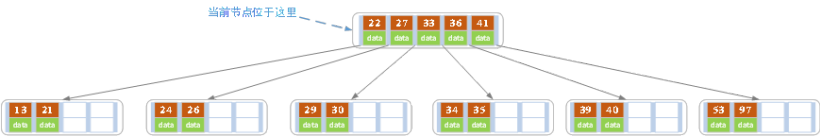
5. 1000行代码徒手写正则表达式引擎【1】--JAVA中正则表达式的使用(6)

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)

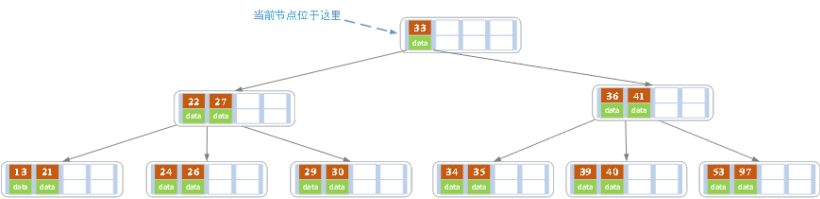
f) 插入key值为26的记录，插入后的结果如下图所示。



当前结点需要以27为中心分裂，并向父结点进位27，然后当前结点指向父结点，结果如下图所示。

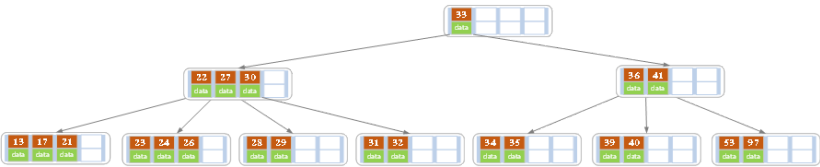


进位后导致当前结点（即根结点）也需要分裂，分裂的结果如下图所示。



分裂后当前结点指向新的根，此时无需调整。

g) 最后再依次插入key为17,28,29,31,32的记录，结果如下图所示。



在实现B树的代码中，为了使代码编写更加容易，我们可以将结点中存储记录的数组长度定义为m而非m-1，这样方便底层的结点由于分裂向上层插入一个记录时，上层有多余的位置存储这个记录。同时，每个结点还可以存储它的父结点的引用，这样就不必编写递归程序。

一般来说，对于确定的m和确定类型的记录，结点大小是固定的，无论它实际存储了多少个记录。但是分配固定结点大小的方法会存在浪费的情况，比如key为28,29所在的结点，还有2个key的位置没有使用，但是已经不可能继续在插入任何值了，因为这个结点的前序key是27,后继key是30,所有整数值都用完了。所以如果记录先按key的大小排好序，再插入到B树中，结点的使用率就会很低，最差情况下使用率仅为50%。

最优 => 分层？

- 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

### 1.3 B树的删除操作

删除操作是指，根据key删除记录，如果B树中的记录中不存  
对应key的记录，则删除失败。

1) 如果当前需要删除的key位于非叶子结点上，则用后继key  
(这里的后继key均指后继记录的意思) 覆盖要删除的key，  
然后在后继key所在的子支中删除该后继key。此时后继key一  
定位于叶子结点上，这个过程和二叉搜索树删除结点的方式  
类似。删除这个记录后执行第2步

2) 该结点key个数大于等于 $\text{Math.ceil}(m/2)-1$ ，结束删除操  
作，否则执行第3步。

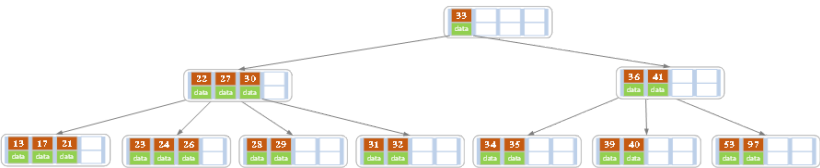
3) 如果兄弟结点key个数大于 $\text{Math.ceil}(m/2)-1$ ，则父结点  
中的key下移到该结点，兄弟结点中的一个key上移，删除操  
作结束。

否则，将父结点中的key下移与当前结点及它的兄弟结点中的  
key合并，形成一个新的结点。原父结点中的key的两个孩子  
指针就变成了一个孩子指针，指向这个新结点。然后当前结  
点的指针指向父结点，重复上第2步。

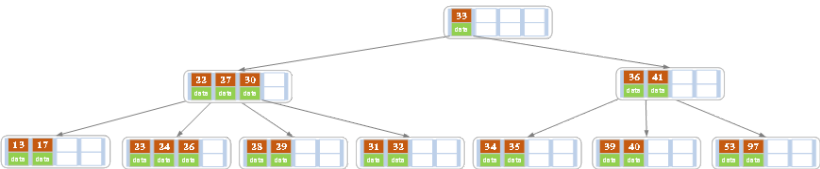
有些结点它可能即有左兄弟，又有右兄弟，那么我们任意选  
择一个兄弟结点进行操作即可。

下面以5阶B树为例，介绍B树的删除操作，5阶B树中，结点最  
多有4个key，最少有2个key

a) 原始状态



b) 在上面的B树中删除21，删除后结点中的关键字个数仍然  
大于等2，所以删除结束。



c) 在上述情况下接着删除27。从上图可知27位于非叶子结  
点上，所以用27的后继替换它。从图中可以看出，27的后继为

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。

--路闻舟的小娇妻

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1  
KB，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

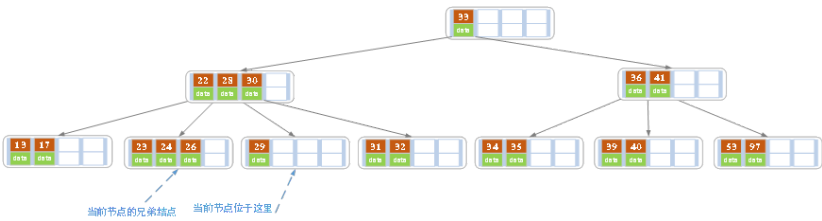
--繁星春水

10. Re:Tarjan算法：求解图的割点与桥（割  
边）

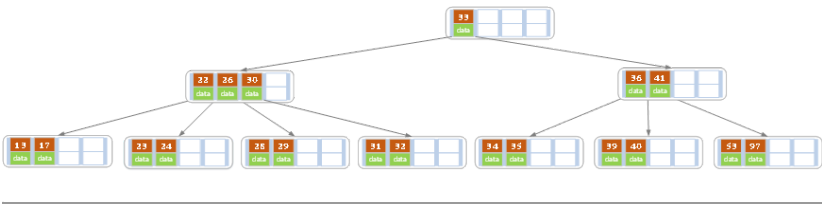
写得特别好，特别是图片实例讲解，终于学  
会了。谢谢！



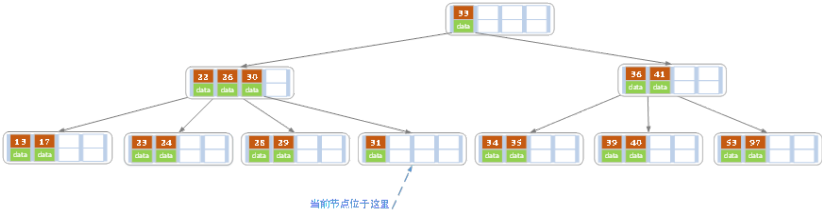
28，我们用28替换27，然后在28（原27）的右孩子结点中删除28。删除后的结果如下图所示。



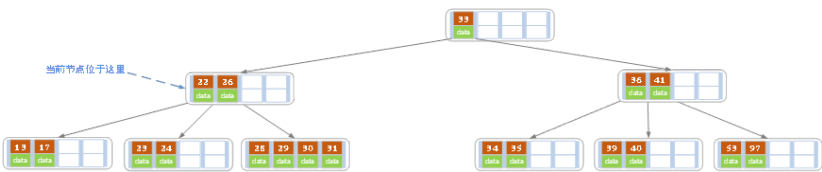
删除后发现，当前叶子结点的记录的个数小于2，而它的兄弟结点中有3个记录（当前结点还有一个右兄弟，选择右兄弟就会出现合并结点的情况，不论选哪一个都行，只是最后B树的形态会不一样而已），我们可以从兄弟结点中借取一个key。所以父结点中的28下移，兄弟结点中的26上移，删除结束。结果如下图所示。



d) 在上述情况下接着32，结果如下图。

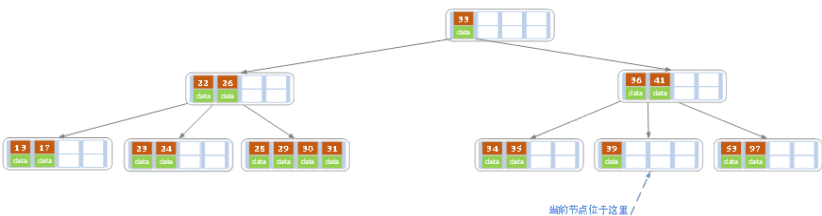


当删除后，当前结点中只key，而兄弟结点中也仅有2个key。所以只能让父结点中的30下移和这个两个孩子结点中的key合并，成为一个新的结点，当前结点的指针指向父结点。结果如下图所示。



当前结点key的个数满足条件，故删除结束。

e) 上述情况下，我们接着删除key为40的记录，删除后结果如下图所示。



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+也不是这样的呀

--persistenceBoy

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

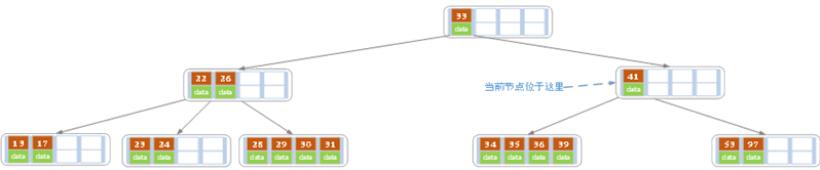
15. Re:多模字符串匹配算法之AC自动机—原理与实现

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...

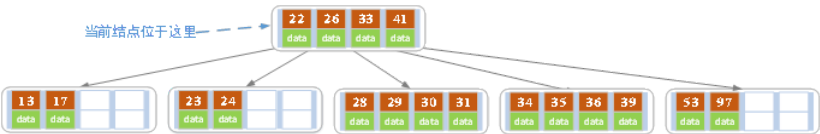
--vinceall

16. Re:B树和B+树的插入、删除图文详解

同理，当前结点的记录数小于2，兄弟结点中没有多余key，所以父结点中的key下移，和兄弟（这里我们选择左兄弟，选择右兄弟也可以）结点合并，合并后的指向当前结点的指针就指向了父结点。



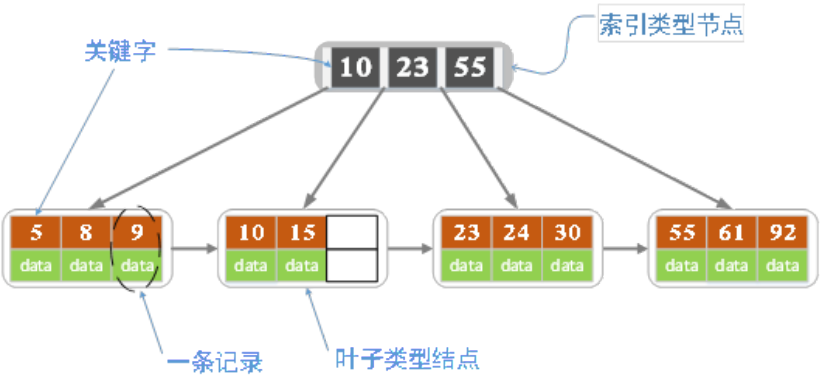
同理，对于当前结点而言只能继续合并了，最后结果如下所示。



合并后结点当前结点满足条件，删除结束。

## 2. B+树

### 2.1 B+树的定义



各种资料上B+树的定义各有不同，一种定义方式是关键字个数和孩子结点个数相同。这里我们采取维基百科上所定义的方式，即关键字个数比孩子结点个数小1，这种方式是和B树基本等价的。上图就是一颗阶数为4的B+树。

除此之外B+树还有以下要求。

- 1) B+树包含2种类型的结点：内部结点（也称索引结点）和叶子结点。根结点本身即可以是内部结点，也可以是叶子结点。根结点的关键字个数最少可以只有1个。
- 2) B+树与B树最大的不同是内部结点不保存数据，只用于索引，所有数据（或者说记录）都保存在叶子结点中。

图文并茂

--李哲操的博客

#### 17. Re:B树和B+树的插入、删除图文详解

B树插入操作里面g步骤写错了吧？  
最后再依次插入key为17,28,29,31,32的记录  
我看图上插入的是：17,23,28,31,32

--Lawliet147

#### 18. Re:多模字符串匹配算法之AC自动机—原理与实现

目前看过的写的最清晰的一篇文章，谢谢大佬~

--ba哥

#### 19. Re:多模字符串匹配算法之AC自动机—原理与实现

楼主公告也太逗了。。

--ba哥

#### 20. Re:B树和B+树的插入、删除图文详解

写的好呀，非常棒

--路远且行

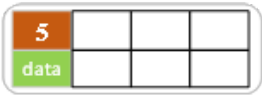
- 3) m阶B+树表示了内部结点最多有m-1个关键字（或者说内部结点最多有m个子树），阶数m同时限制了叶子结点最多存储m-1个记录。
- 4) 内部结点中的key都按照从小到大的顺序排列，对于内部结点中的一个key，左树中的所有key都小于它，右子树中的key都大于等于它。叶子结点中的记录也按照key的大小排列。
- 5) 每个叶子结点都存有相邻叶子结点的指针，叶子结点本身依关键字的大小自小而大顺序链接。

## 2.2 B+树的插入操作

- 1) 若为空树，创建一个叶子结点，然后将记录插入其中，此时这个叶子结点也是根结点，插入操作结束。
- 2) 针对叶子类型结点：根据key值找到叶子结点，向这个叶子结点插入记录。插入后，若当前结点key的个数小于等于m-1，则插入结束。否则将这个叶子结点分裂成左右两个叶子结点，左叶子结点包含前m/2个记录，右结点包含剩下的记录，将第m/2+1个记录的key进位到父结点中（父结点一定是索引类型结点），进位到父结点的key左孩子指针向左结点,右孩子指针向右结点。将当前结点的指针指向父结点，然后执行第3步。
- 3) 针对索引类型结点：若当前结点key的个数小于等于m-1，则插入结束。否则，将这个索引类型结点分裂成两个索引结点，左索引结点包含前(m-1)/2个key，右结点包含m-(m-1)/2个key，将第m/2个key进位到父结点中，进位到父结点的key左孩子指向左结点，进位到父结点的key右孩子指向右结点。将当前结点的指针指向父结点，然后重复第3步。

下面是一颗5阶B树的插入过程，5阶B数的结点最少2个key，最多4个key。

a) 空树中插入5



b) 依次插入8, 10, 15

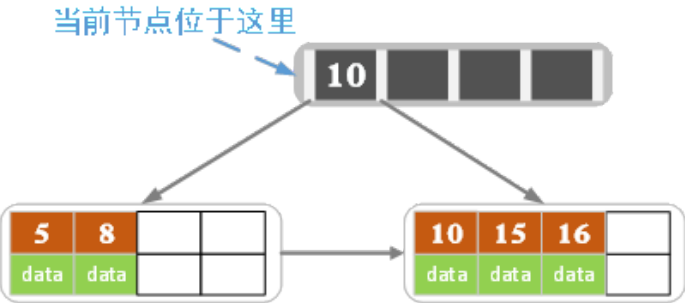


5	8	10	15
data	data	data	data

c) 插入16

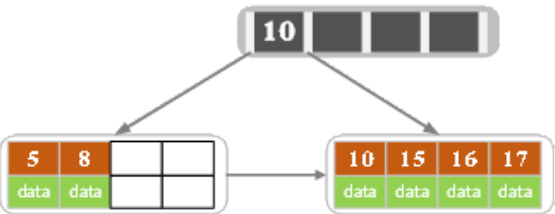
5	8	10	15	16
data	data	data	data	data

插入16后超过了关键字的个数限制，所以要进行分裂。在叶子结点分裂时，分裂出来的左结点2个记录，右边3个记录，中间key成为索引结点中的key，分裂后当前结点指向了父结点（根结点）。结果如下图所示。

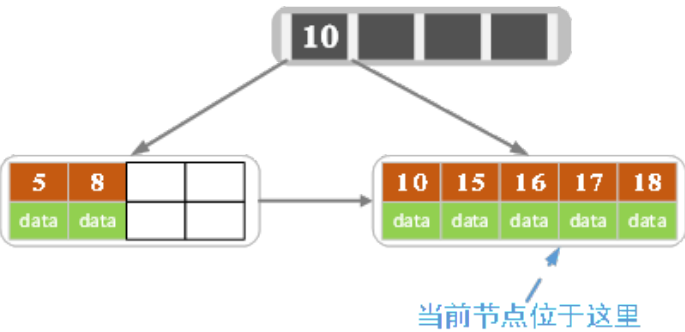


当然我们还有另一种分裂方式，给左结点3个记录，右结点2个记录，此时索引结点中的key就变为15。

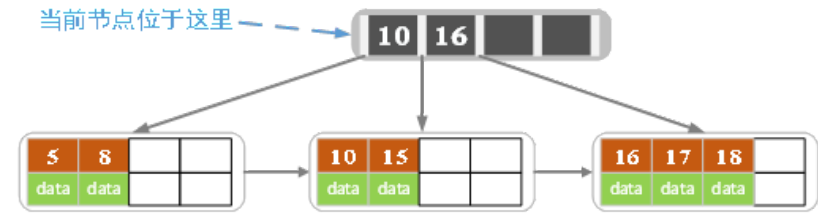
d) 插入17



e) 插入18，插入后如下图所示

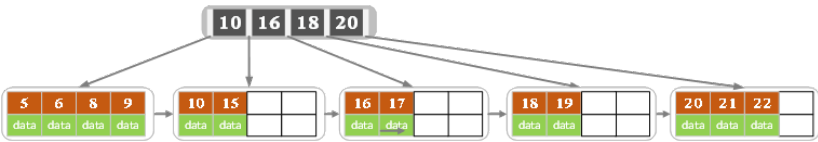


当前结点的关键字个数大于5，进行分裂。分裂成两个结点，左结点2个记录，右结点3个记录，关键字16进位到父结点（索引类型）中，将当前结点的指针指向父结点。

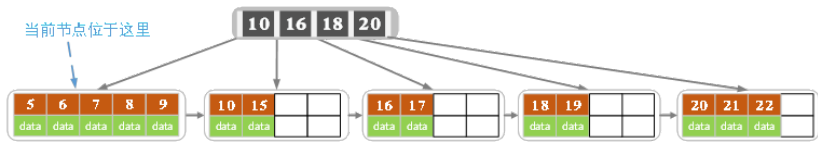


当前节点的关键字个数满足条件，插入结束。

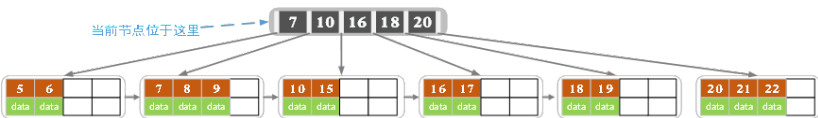
f) 插入若干数据后



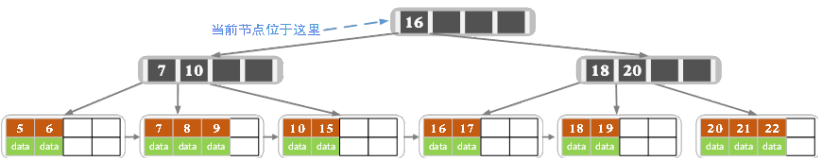
g) 在上图中插入7，结果如下图所示



当前节点的关键字个数超过4，需要分裂。左结点2个记录，右结点3个记录。分裂后关键字7进入到父结点中，将当前结点的指针指向父结点，结果如下图所示。



当前节点的关键字个数超过4，需要继续分裂。左结点2个关键字，右结点2个关键字，关键字16进入到父结点中，将当前结点指向父结点，结果如下图所示。



当前节点的关键字个数满足条件，插入结束。

### 2.3 B+树的删除操作

如果叶子结点中没有相应的key，则删除失败。否则执行下面的步骤

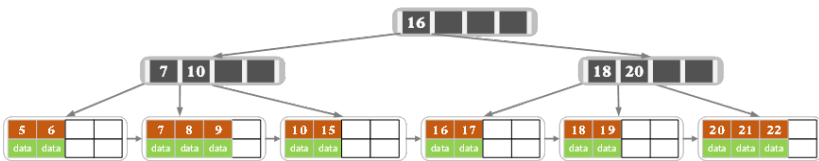
1) 删除叶子结点中对应的key。删除后若结点的key的个数大于等于 $\lceil (m-1)/2 \rceil - 1$ ，删除操作结束，否则执行第2步。

- 2) 若兄弟结点key有富余 (大于 $\text{Math.ceil}(m-1)/2 - 1$ ) , 向兄弟结点借一个记录, 同时用借到的key替换父结 (指当前结点和兄弟结点共同的父结点) 点中的key, 删除结束。否则执行第3步。
- 3) 若兄弟结点中没有富余的key, 则当前结点和兄弟结点合并成一个新的叶子结点, 并删除父结点中的key (父结点中的这个key两边的孩子指针就变成了一个指针, 正好指向这个新的叶子结点), 将当前结点指向父结点 (必为索引结点), 执行第4步 (第4步以后的操作和B树就完全一样了, 主要是为了更新索引结点)。
- 4) 若索引结点的key的个数大于等于 $\text{Math.ceil}(m-1)/2 - 1$ , 则删除操作结束。否则执行第5步
- 5) 若兄弟结点有富余, 父结点key下移, 兄弟结点key上移, 删除结束。否则执行第6步
- 6) 当前结点和兄弟结点及父结点下移key合并成一个新的结点。将当前结点指向父结点, 重复第4步。

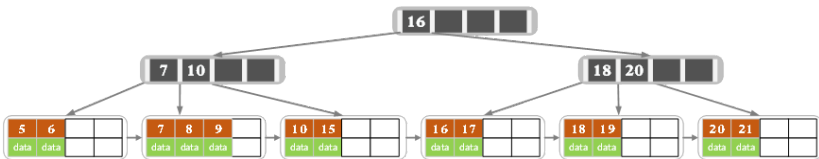
注意, 通过B+树的删除操作后, 索引结点中存在的key, 不一定在叶子结点中存在对应的记录。

下面是一颗5阶B树的删除过程, 5阶B数的结点最少2个key, 最多4个key。

a) 初始状态

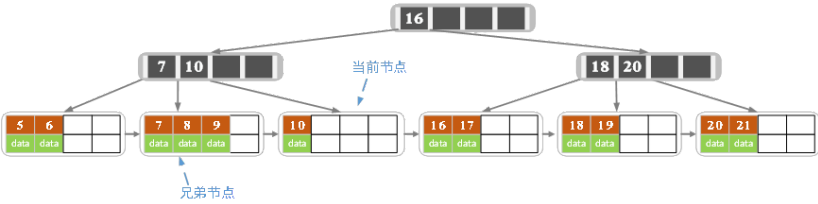


b) 删除22, 删除后结果如下图

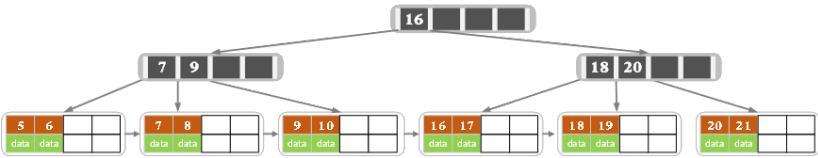


删除后叶子结点中key的个数大于等于2, 删除结束

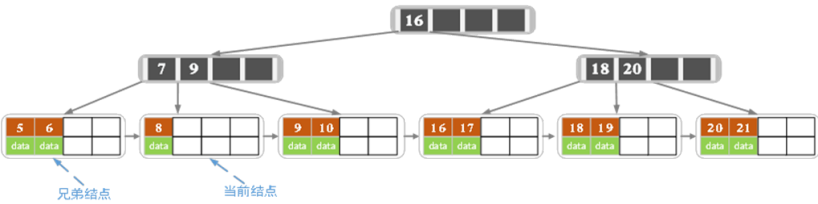
c) 删除15, 删除后的结果如下图所示



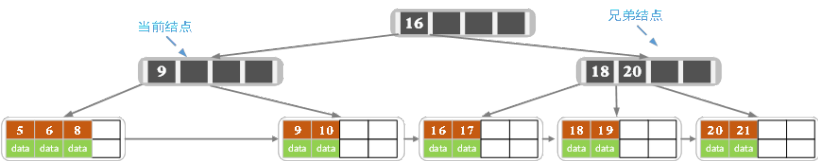
删除后当前结点只有一个key, 不满足条件, 而兄弟结点有三个key, 可以从兄弟结点借一个关键字为9的记录, 同时更新将父结点中的关键字由10也变为9, 删除结束。



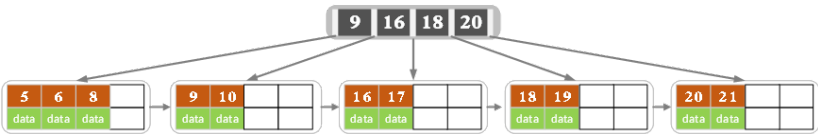
d) 删除7, 删除后的结果如下图所示



当前结点关键字个数小于2, (左) 兄弟结点中的也没有富余的关键字 (当前结点还有个右兄弟, 不过选择任意一个进行分析就可以了, 这里我们选择了左边的), 所以当前结点和兄弟结点合并, 并删除父结点中的key, 当前结点指向父结点。



此时当前结点的关键字个数小于2, 兄弟结点的关键字也没有富余, 所以父结点中的关键字下移, 和两个孩子结点合并, 结果如下图所示。



3. 参考内容

- [1] [B+树介绍](#)
- [2] [从MySQL Bug#67718浅谈B+树索引的分裂优化](#)
- [3] [B+树的几点总结](#)