

漫画：什么是 B+ 树？

2017-07-13 算法爱好者

(点击上方公众号，可快速关注)

来源：伯乐专栏作者/玻璃猫，微信公众号 - 梦见 (dreamsee321)

如有好文章投稿，请点击 → [这里了解详情](#)

之前已介绍了 B 树的原理和应用，没看过的童鞋，请点击下面的链接：

[《漫画：什么是 B 树？》](#)

这一次我们来介绍 B+ 树。

大黄，上次你说的 B- 树，我
基本上明白了。那 B+ 树又是
个啥？



B+ 树是基于 B- 树的一种变体，
有着比 B- 树更高的查询性能。



哇，居然可以比 B- 树性能更高。

那 B+ 树究竟长什么样子呢？



在细说 B+ 树之前，我们先来回顾一下 B- 树的几大特征。



一个m阶的B树具有以下几个特征：

- 1.根结点至少有两个子女。
- 2.每个中间节点都包含 $k-1$ 个元素和 k 个孩子，其中 $m/2 \leq k \leq m$
- 3.每一个叶子节点都包含 $k-1$ 个元素，其中 $m/2 \leq k \leq m$
- 4.所有的叶子结点都位于同一层。
- 5.每个节点中的元素从小到大排列，节点当中 $k-1$ 个元素正好是 k 个孩子包含的元素的值域分划。

B+ 树和 B- 树有一些共同点，但是 B+ 树也具备一些新的特征。



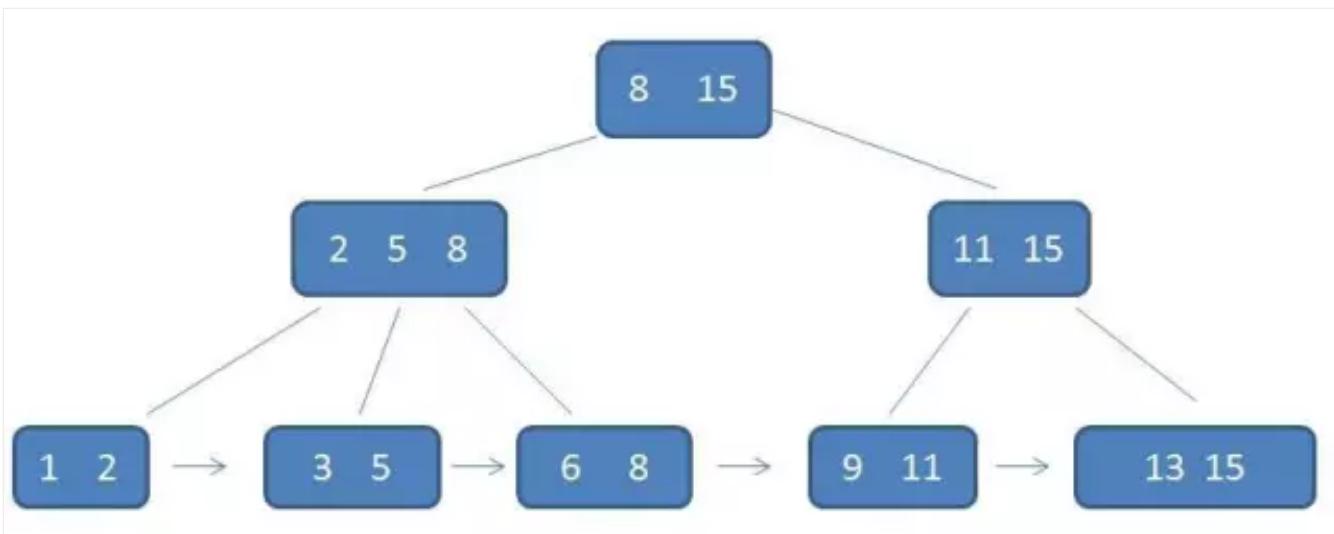
一个m阶的B+树具有如下几个特征：

1. 有k个子树的中间节点包含有k个元素（B树中是k-1个元素），每个元素不保存数据，只用来索引，所有数据都保存在叶子节点。
2. 所有的叶子结点中包含了全部元素的信息，及指向含这些元素记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接。
3. 所有的中间节点元素都同时存在于子节点，在子节点元素中是最大（或最小）元素。

概念好繁琐，看不懂哎.....



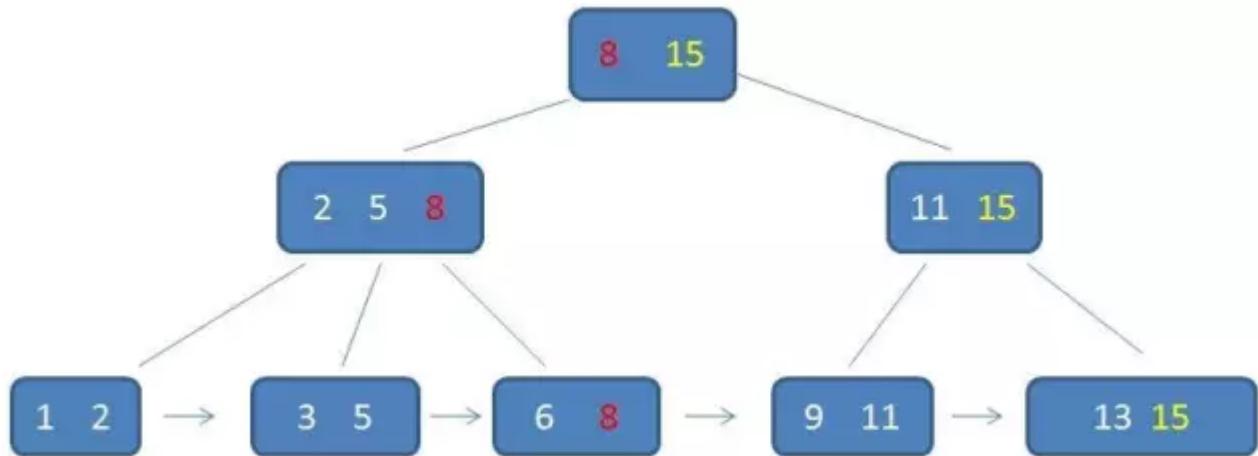
别急，我们用具体例子来看一看
B+ 树的结构。



这是什么怪树，不但节点之间含有
重复元素，而且叶子节点还用指针
连在一起。



你说的这些正是 B+ 树的几个特点。首先，每一个父节点的元素都出现在子节点中，是子节点的最大（或最小）元素。



在上面这棵树中，根节点元素 8 是子节点 2, 5, 8 的最大元素，也是叶子节点 6, 8 的最大元素。



根节点元素 15 是子节点 11, 15
的最大元素，也是叶子节点
13, 15 的最大元素。



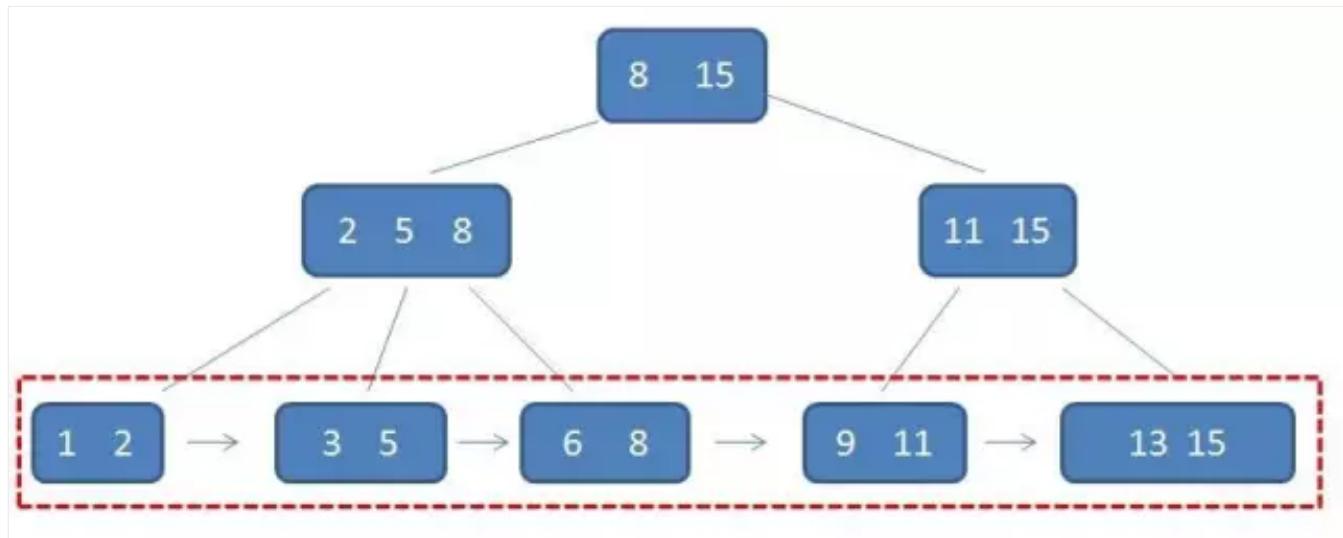
需要注意的是，根节点的最大元素（这里是 15），也就等同于整个 B+ 树的最大元素。以后无论插入删除多少元素，始终要保持最大元素在根节点当中。



至于叶子节点，由于父节点的元素都出现在子节点，因此所有叶子节点包含了全量元素信息。



并且每一个叶子节点都带有指向下一个节点的指针，形成了一个有序链表。



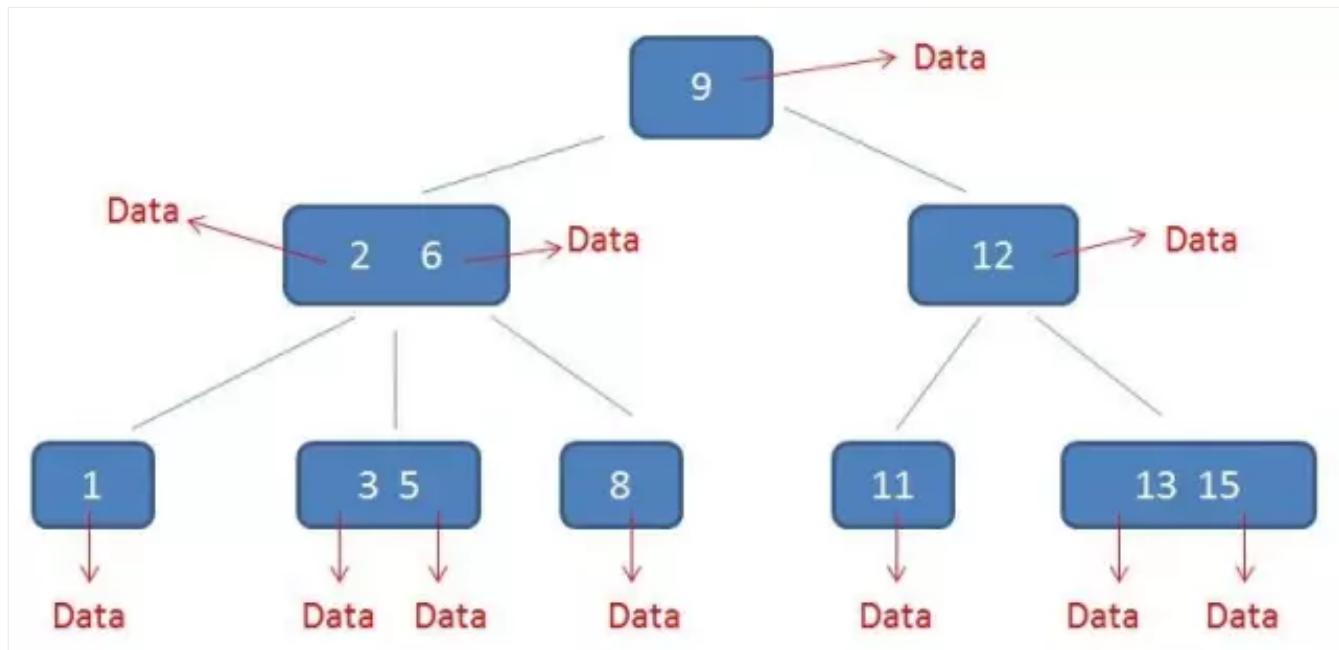
B+ 树还具有一个特点，这个特点是在索引之外，确是至关重要的特点。那就是 [卫星数据] 的位置。



所谓卫星数据，指的是索引元素所指向的数据记录，比如数据库中的某一行。在 B-树中，无论中间节点还是叶子节点都带有卫星数据。



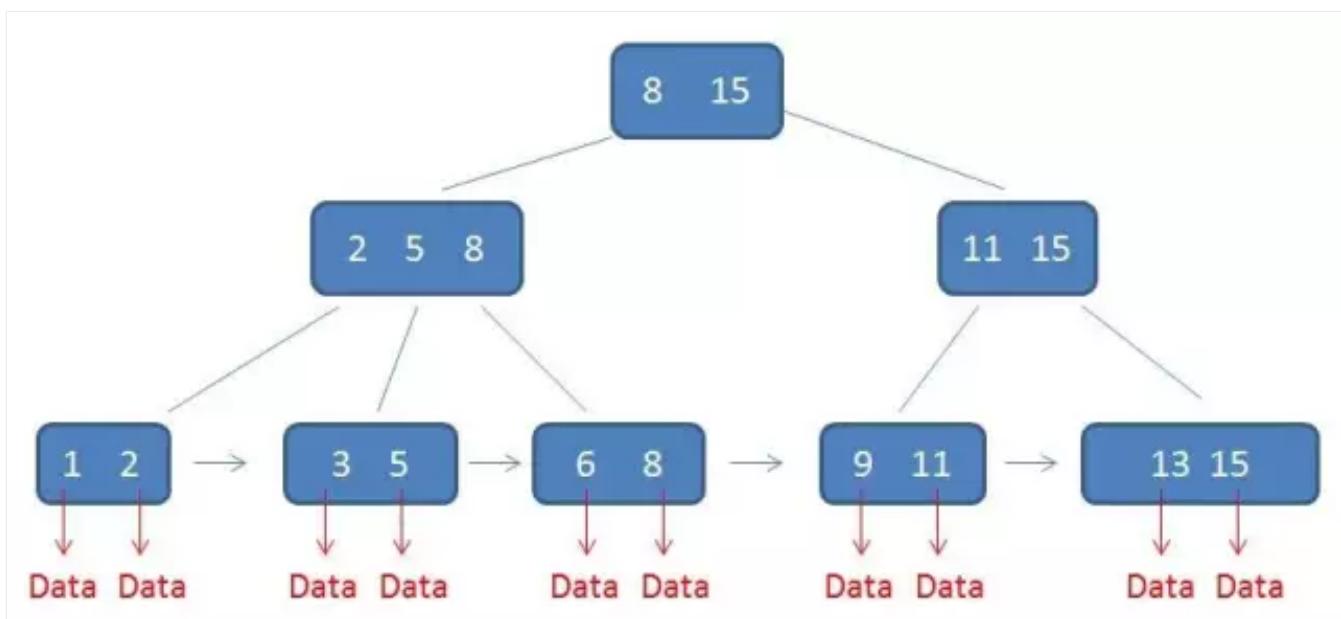
B-树中的卫星数据 (Satellite Information)：



而在 B+ 树当中，只有叶子节点带有卫星数据，其余中间节点仅仅是索引，没有任何数据关联。



B+树中的卫星数据（Satellite Information）：



需要补充的是，在数据库的聚集索引（Clustered Index）中，叶子节点直接包含卫星数据。在非聚集索引（NonClustered Index）中，叶子节点带有指向卫星数据的指针。

大体明白了。B+ 树设计成这样子，
究竟有什么好处呢？



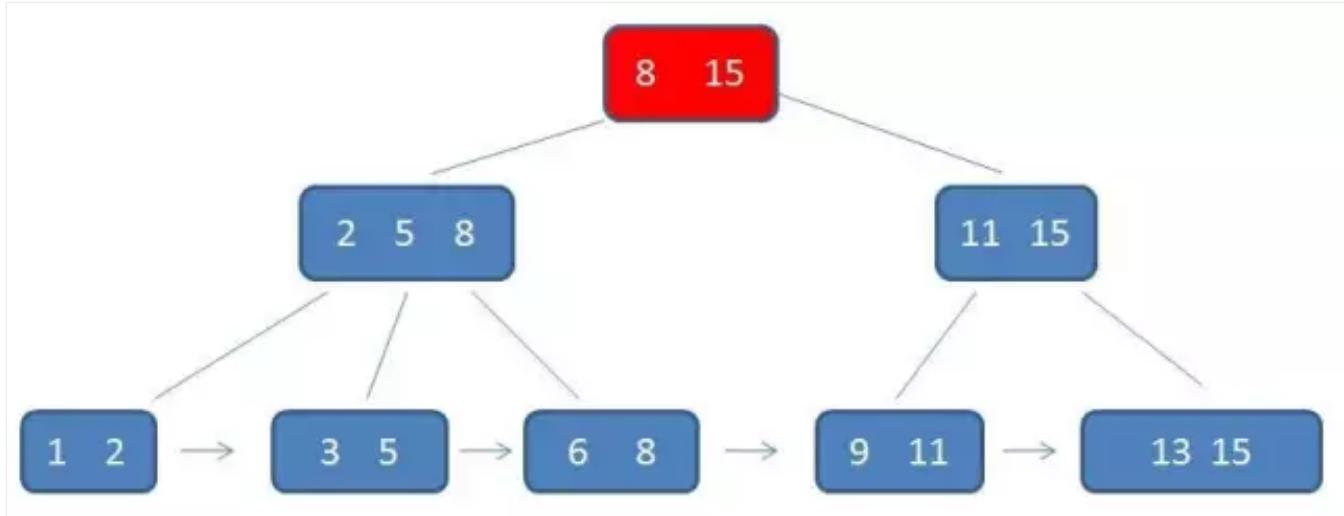
B+ 树的好处主要体现在查询性能上。
下面我们分别通过单行查询和范围
查询来做分析。



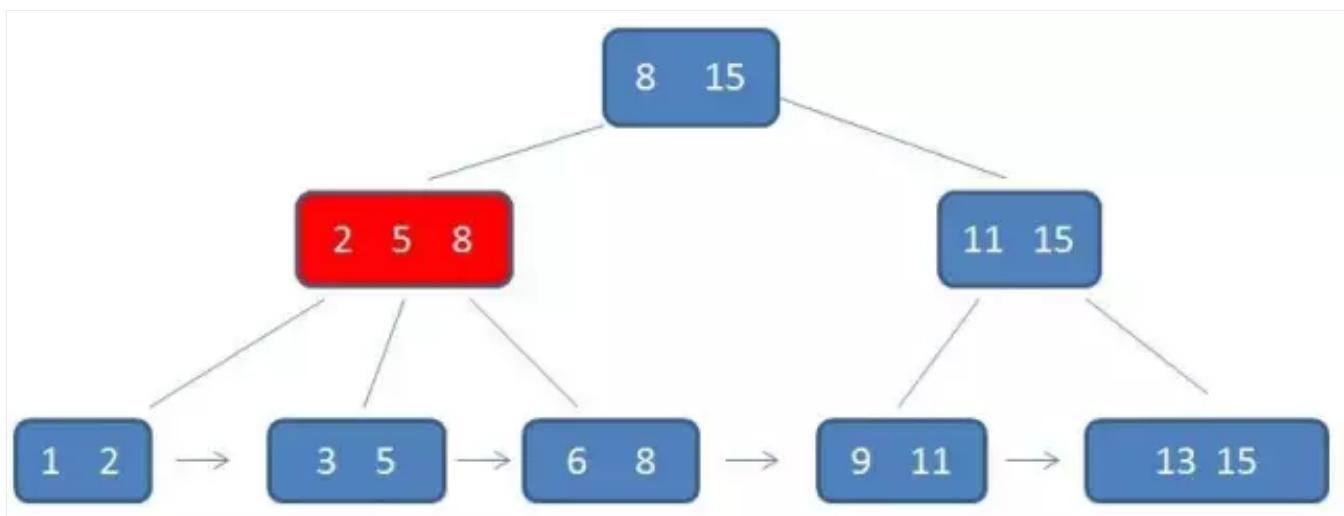
在单元素查询的时候，B+ 树会自顶
向下逐层查找节点，最终找到匹配
的叶子节点。比如我们要查找的是
元素 3



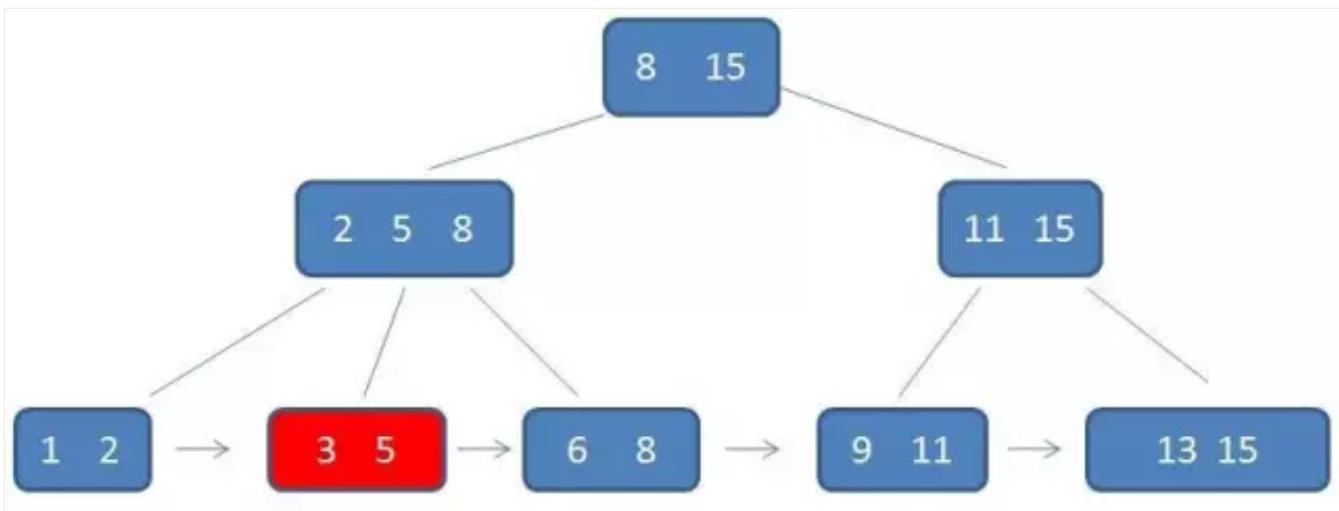
第一次磁盘IO:



第二次磁盘IO:



第三次磁盘IO:



查询流程看起来跟 B- 树差不多嘛。



不，有两点不同。首先，B+ 树的中间节点没有卫星数据，所以同样大小的磁盘页可以容纳更多的节点元素。



这意味着，数据量相同的情况下，
B+ 树的结构比 B- 树更加“矮胖”，
因此查询时 IO 次数也更少。



其次，B+ 树的查询必须最终查找到叶
子节点，而 B- 树只要找到匹配元素
即可，无论匹配元素处于中间节点还
是叶子节点。



因此，B- 树的查找性能并不稳定（最
好情况是只查根节点，最坏情况是查
到叶子节点）。而 B+ 树的每一次查找
都是稳定的。



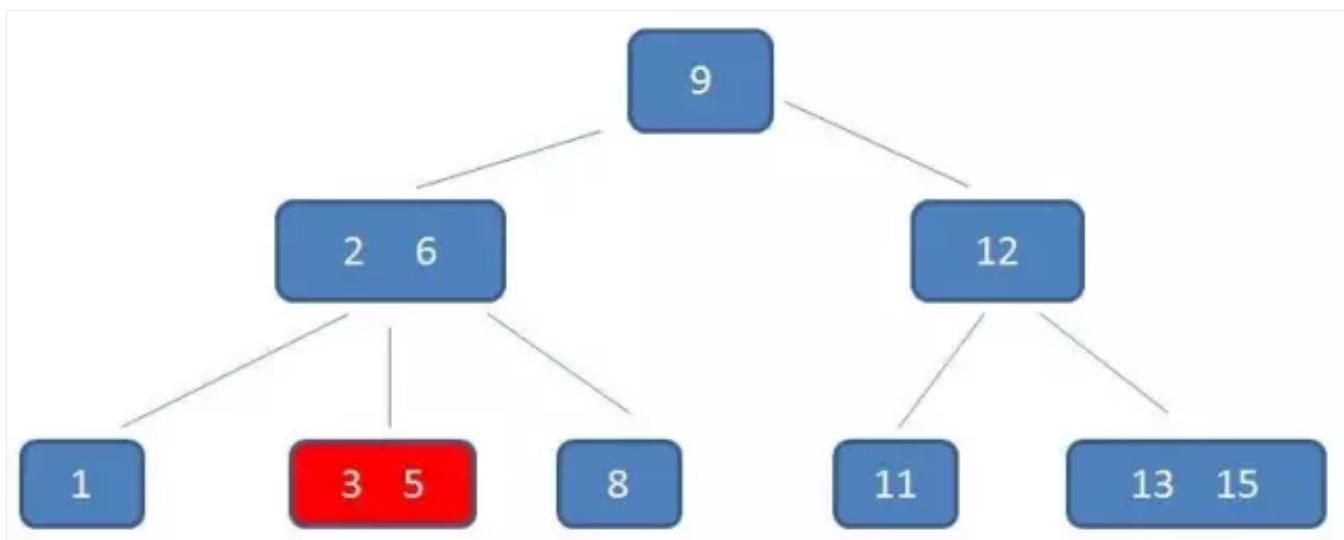


下面我们再来看看范围查询。B- 树如何做范围查询呢？只能依靠繁琐的中序遍历。比如我们要查询范围为 3 到 11 的元素：

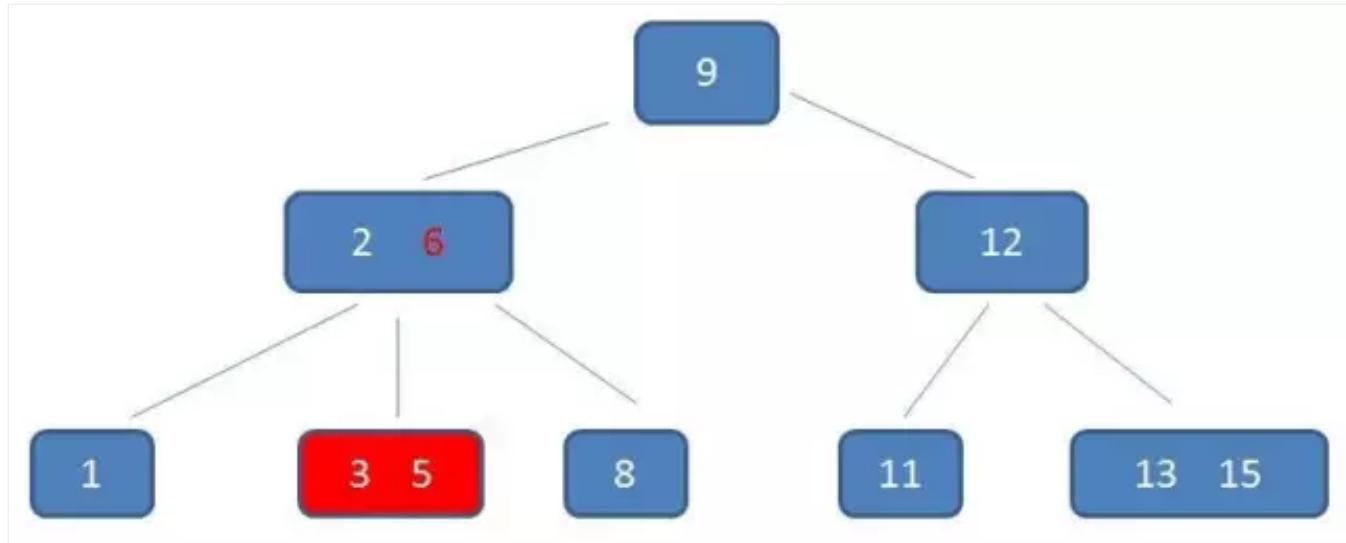


B-树的范围查找过程

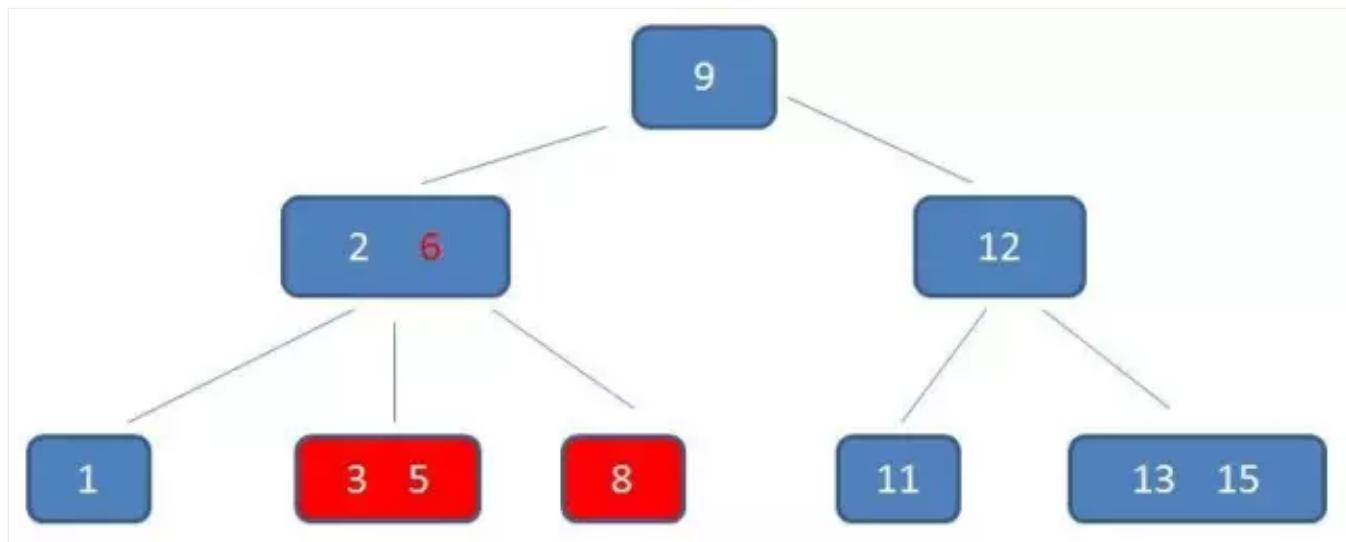
自顶向下，查找到范围的下限（3）：



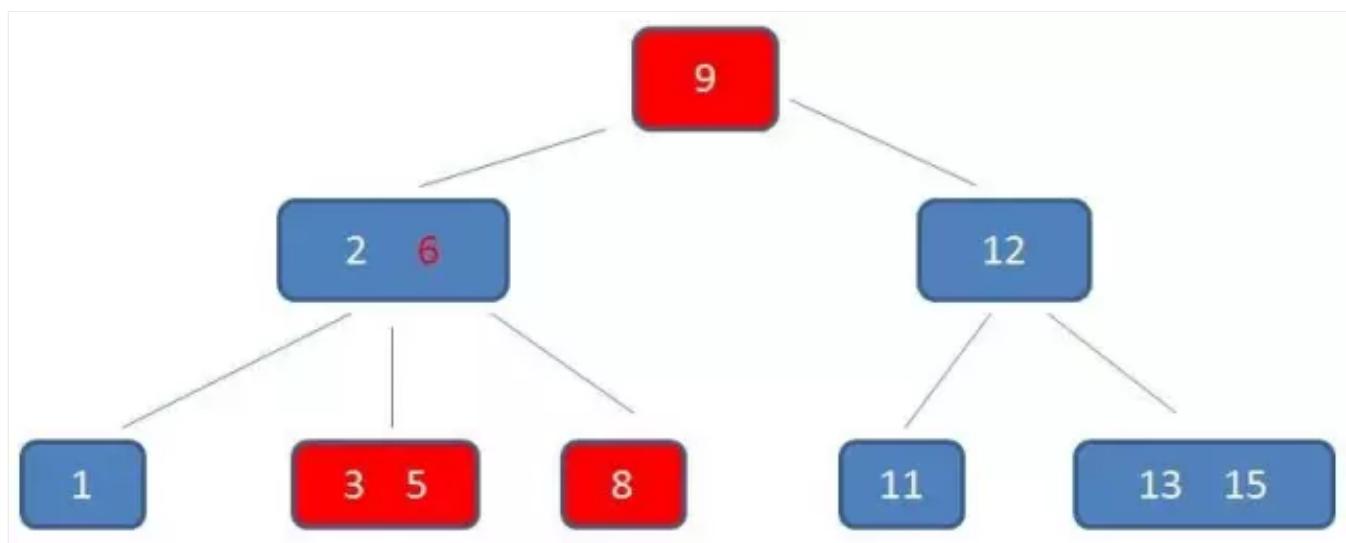
中序遍历到元素6：



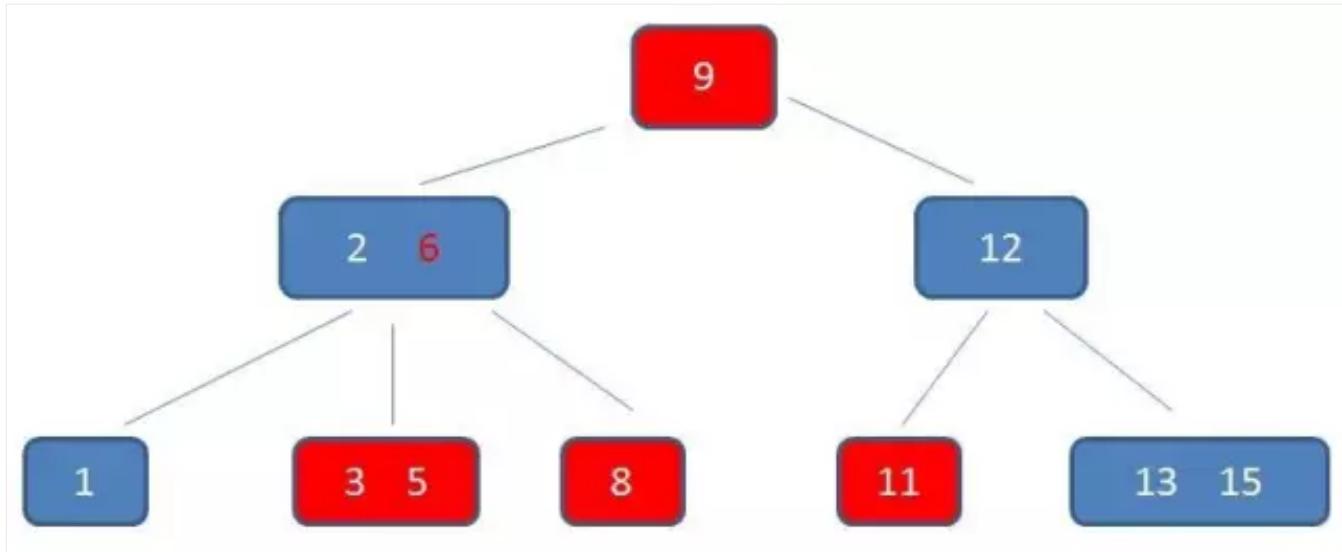
中序遍历到元素8：



中序遍历到元素9：



中序遍历到元素11，遍历结束：



B- 树的范围查询确实很繁琐呀。

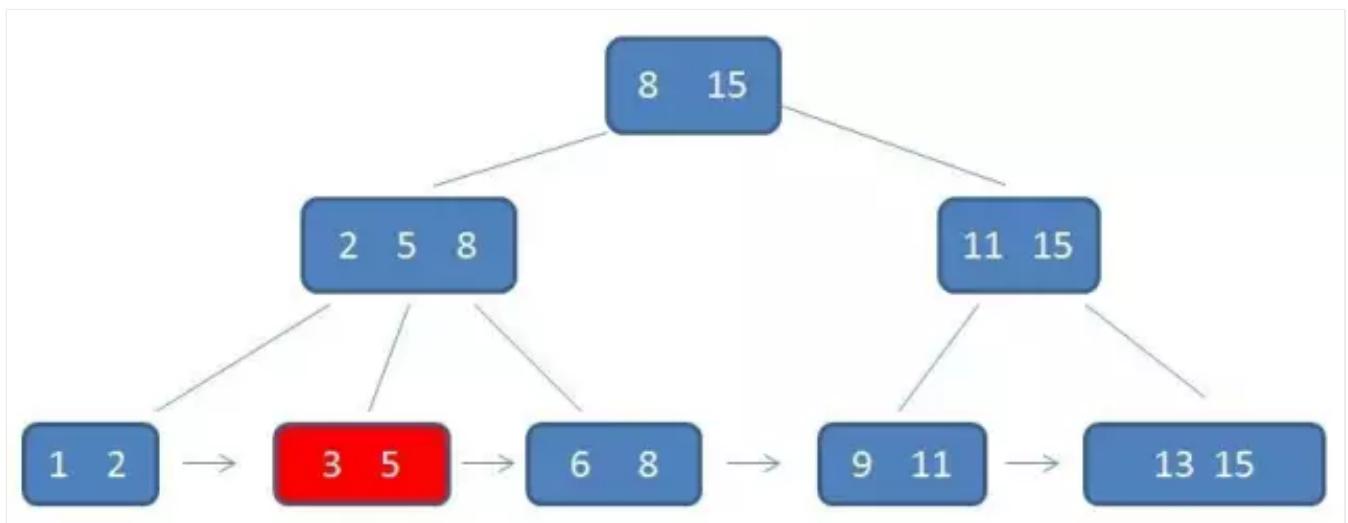


反观 B+ 树的范围查询，则要简单得多，只需要在链表上做遍历即可：

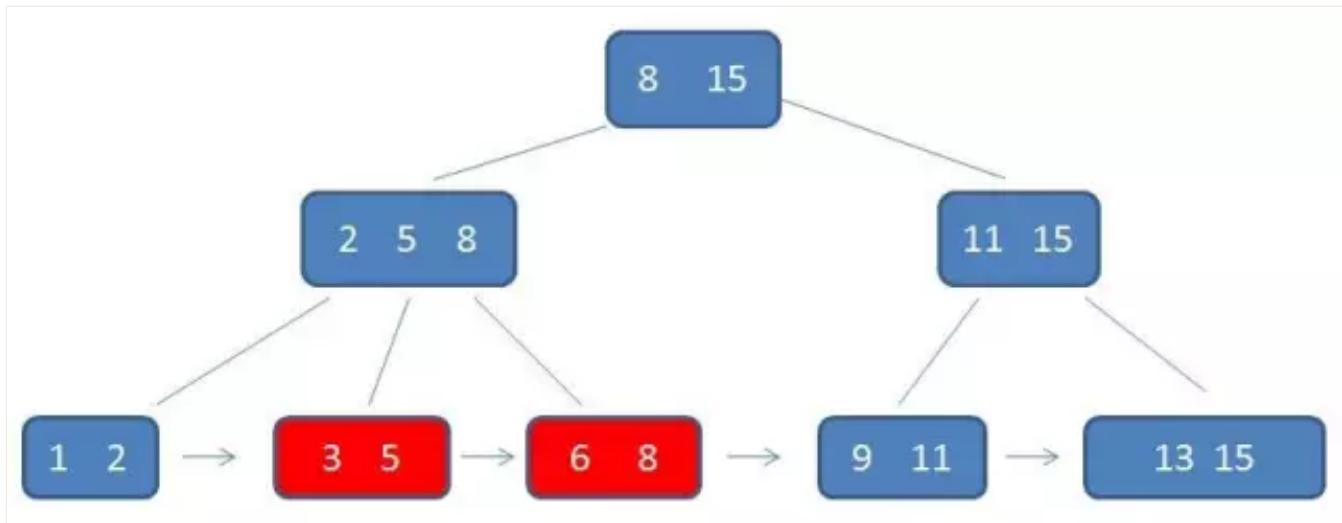


B+树的范围查找过程

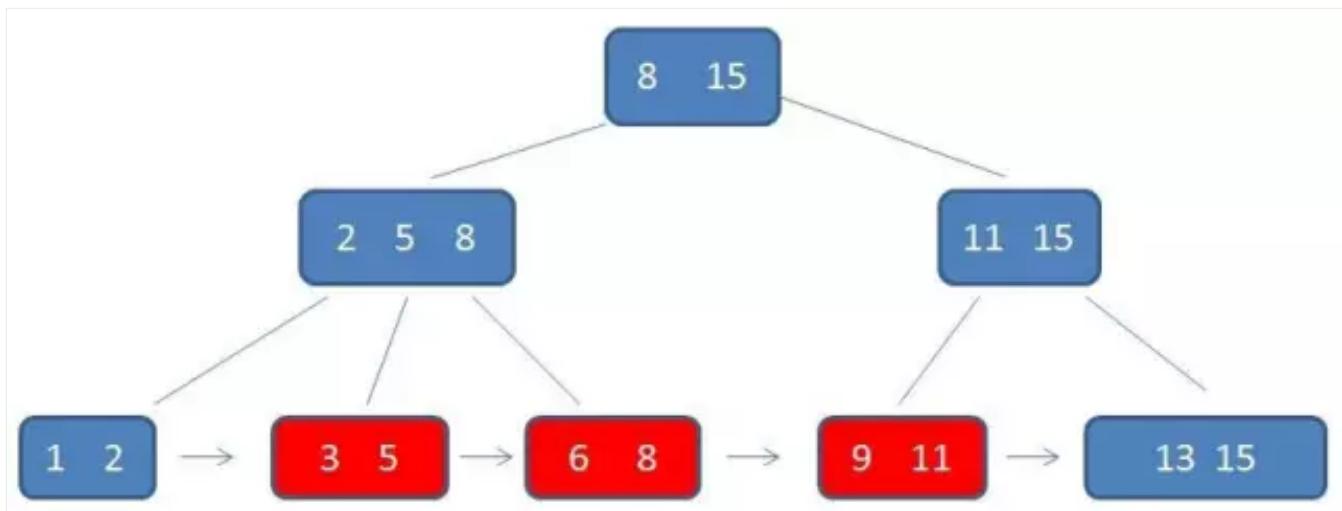
自顶向下，查找到范围的下限（3）：



通过链表指针，遍历到元素6, 8：



通过链表指针，遍历到元素9, 11，遍历结束：



哇，果然比 B- 树的中序遍历要
简单得多。



综合起来，B+ 树相比 B- 树的优势有三个：1. IO 次数更少； 2. 查询性能稳定； 3. 范围查询简便。



至于 B+ 树的插入和删除，过程与 B- 树大同小异，我再这里就不详细描述了。



最后我们来总结一下，B+ 树的特征和优势：



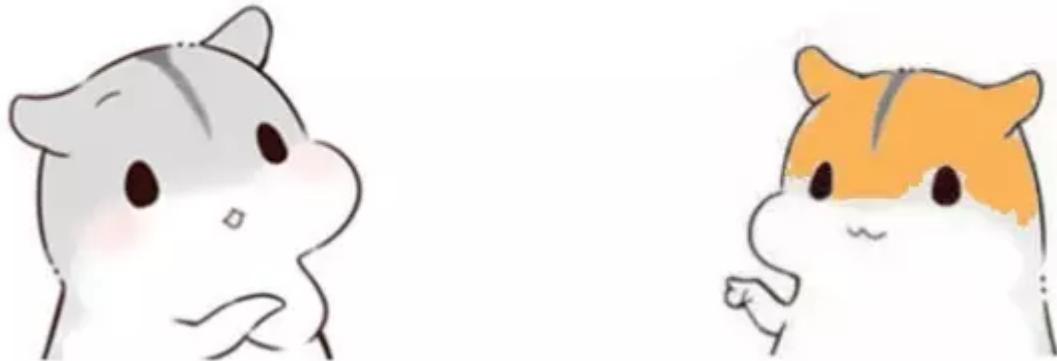
B+树的特征：

1. 有 k 个子树的中间节点包含有 k 个元素（B 树中是 $k-1$ 个元素），每个元素不保存数据，只用来索引，所有数据都保存在叶子节点。
2. 所有的叶子结点中包含了全部元素的信息，及指向含这些元素记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接。
3. 所有的中间节点元素都同时存在于子节点，在子节点元素中是最大（或最小）元素。

B+树的优势：

1. 单一节点存储更多的元素，使得查询的IO次数更少。
2. 所有查询都要查找到叶子节点，查询性能稳定。
3. 所有叶子节点形成有序链表，便于范围查询。

好了，关于 B+ 树我们就介绍到这里，感谢大家的支持！



漫画算法系列

- 漫画算法：最小栈的实现
- 漫画算法：判断 2 的乘方
- 漫画算法：找出缺失的整数
- 漫画算法：辗转相除法是什么鬼？
- 漫画算法：什么是动态规划？（整合版）

- 漫画算法：什么是跳跃表？
- 漫画算法：什么是 B 树？

觉得本文有帮助？请分享给更多人

关注「算法爱好者」，修炼编程内功

算法爱好者

专注算法相关内容



微信号：AlgorithmFans



长按识别二维码关注

伯乐在线旗下微信公众号

商务合作QQ：2302462408

内容转载自公众号

梦见

了解更多 >