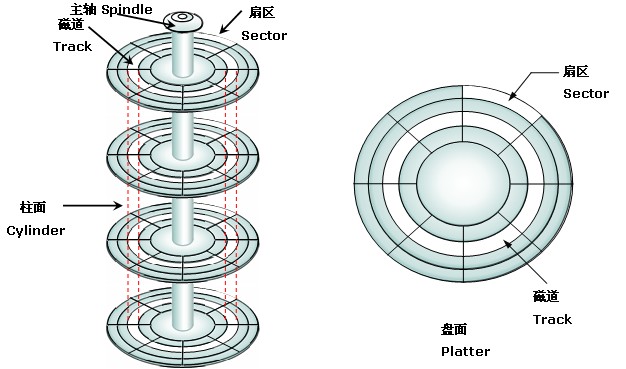
# B-树与B+树

## B-树



在磁盘寻址的过程。磁盘顺序访问的速度比随机访问要快。在随机访问的时候，磁盘需要重新寻址，引起磁盘转动，磁盘转动的时间比磁盘访问的时间要大几个数量级。所以在设计磁盘存储数据结构时候，尽可能的减少磁盘的随机访问次数。

采用多叉树能设计磁盘存储减少磁盘的随机访问。在选择多叉树的时候，需要维护多叉树的平衡。故B-树作为了磁盘寻址的首选。那在磁盘访问的过程，B-树又是如何工作的。

磁盘与内存的数据交换单位是扇区。在设计B-树的时候，将B-树的每一个结点大小设计为扇区大小。这样B-树的子树分支就能达到几百乃至几千。比如一颗B-树的结点有1024颗子结点，这样一颗高度为3的B-树，即可存储1G数量的扇区。

那如何来减少磁盘的随机访问次数。减少次数是一个相对的。相比较二叉树来说，访问磁盘的次数要明显少很多。因此在数据库的设计方面都是采用B-树以及B-树的衍生树。

### **设计案例题**

设计一个图片存储索引组件

1. 图片的数量是巨大的。

2. 可以添加图片，可以删除图片

3. 可以通过图片名称进行查找

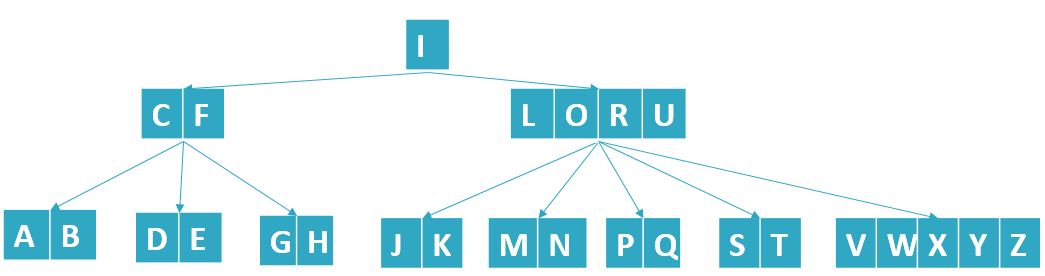
4. 添加，删除，查找都是对单一图片进行操作

### B-树定义

一颗M阶B树T，满足以下条件

1. 每个结点至多拥有M课子树
2. 根结点至少拥有两颗子树
3. 除了根结点以外，其余每个分支结点至少拥有M/2课子树
4. 所有的叶结点都在同一层上
5. 有k课子树的分支结点则存在k-1个关键字，关键字按照递增顺序进行排序
6. 关键字数量满足ceil(M/2)-1 <= n <= M-1

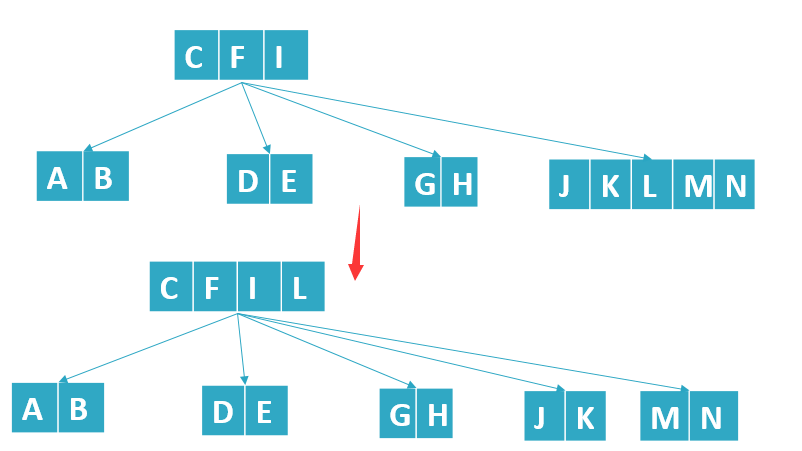
B-树如下图所示，



### B-树添加

假设B-树的最大关键字数量为5， 最大子树数量为6， 则归并数T为3

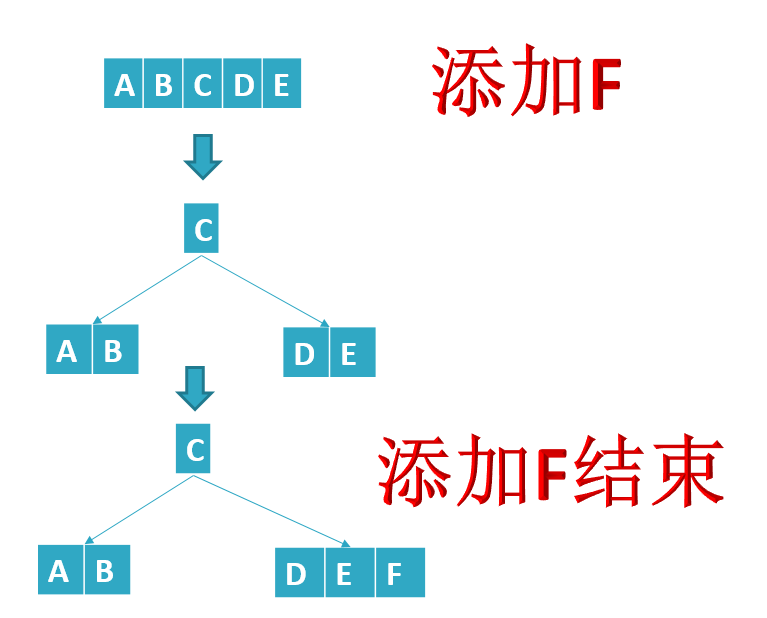
1. 结点分裂



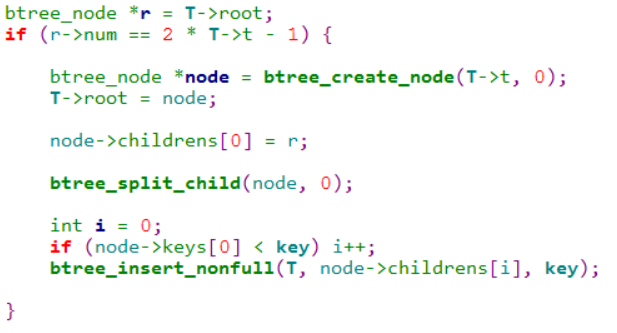
具体实现代码如下：



1. 根结点分裂

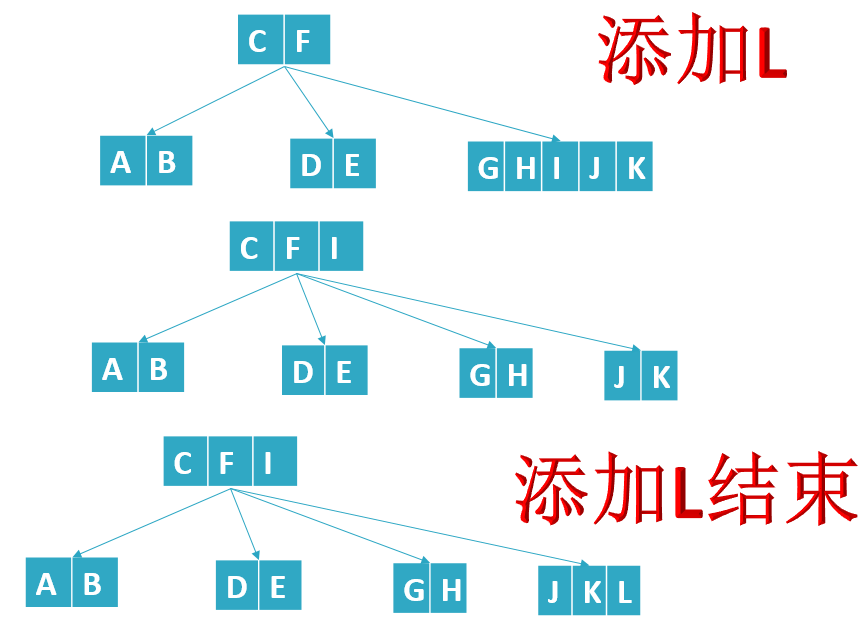


具体实现代码如下：

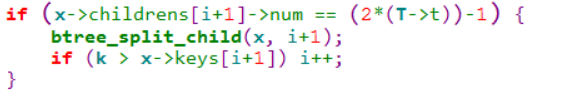


结点分裂代码：

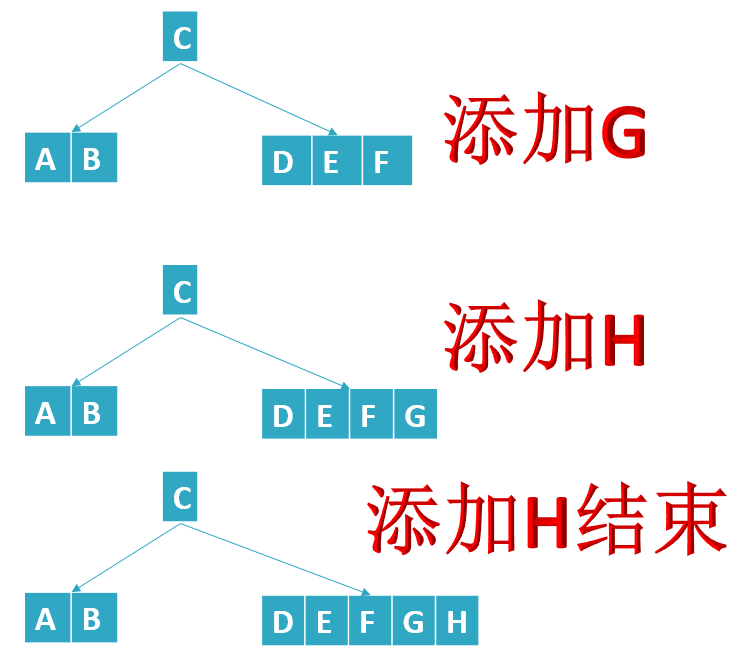
1. 叶子结点分裂



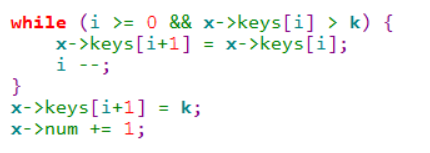
具体实现代码：



1. 如果是叶子结点，且不大于最大关键字数量，即可立即插入



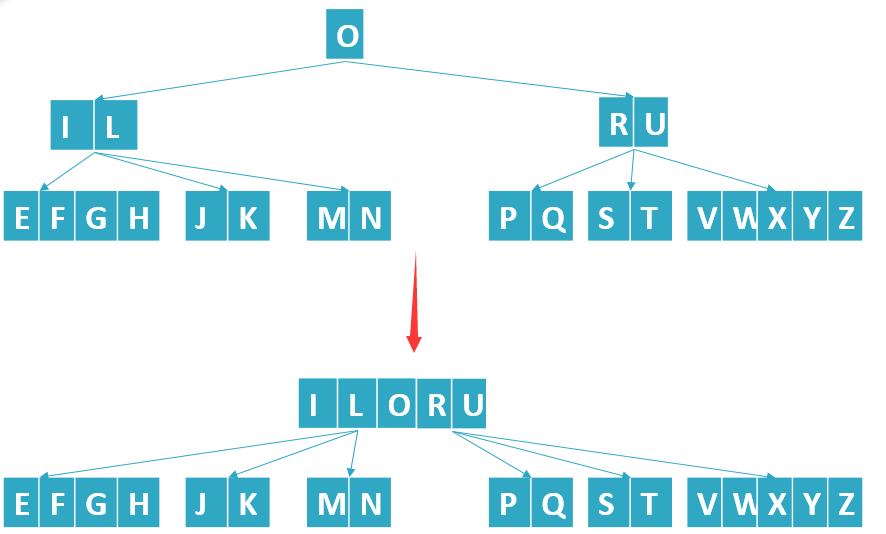
实现代码：



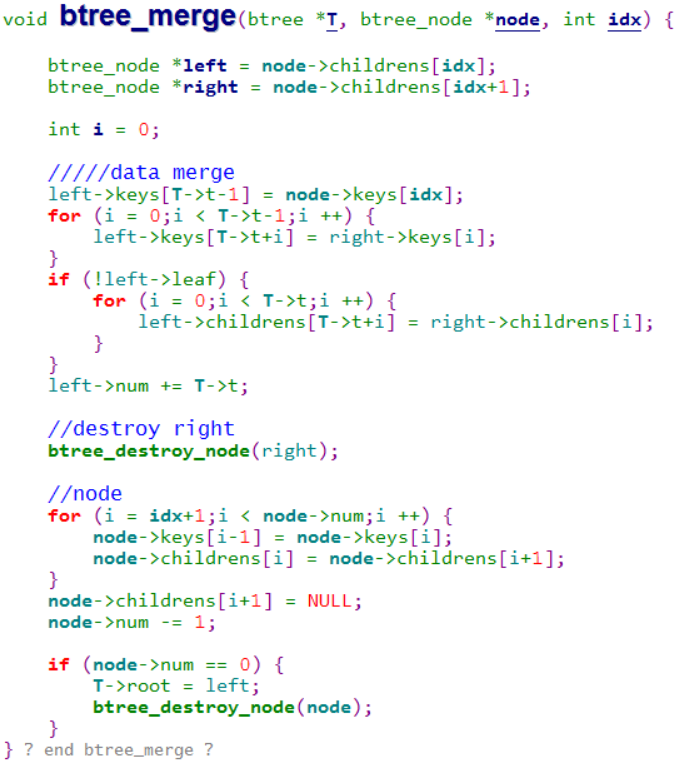
### B-树删除

1. 结点归并

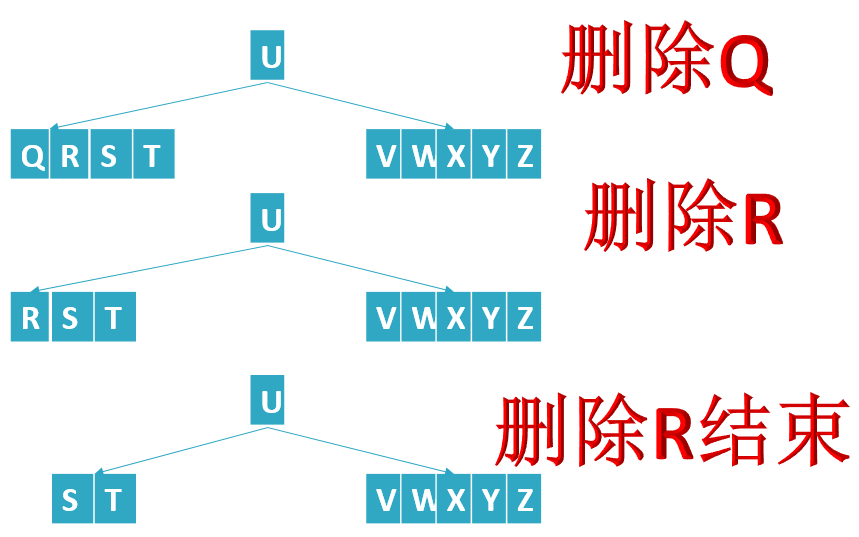
如下图所示：



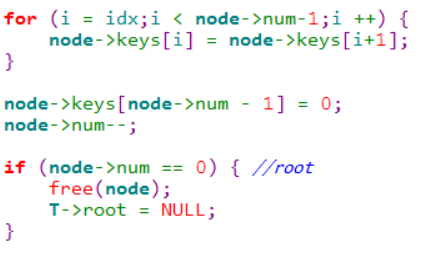
实现代码如下：



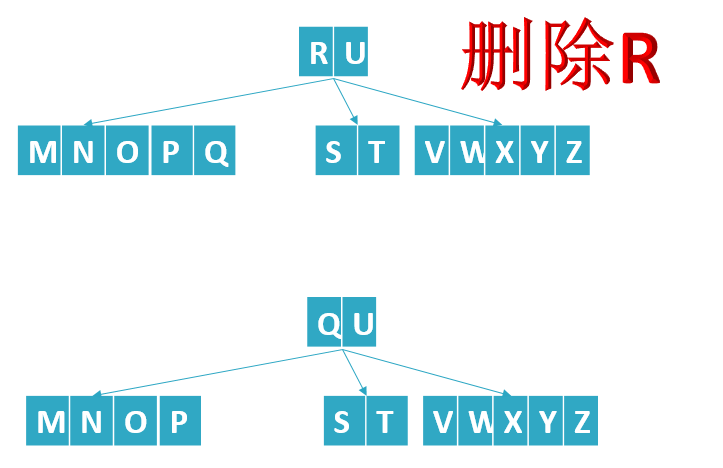
1. 关键字在叶子结点中，直接删除



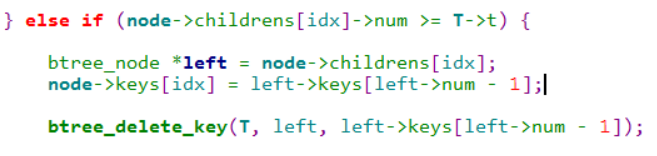
实现代码如下：



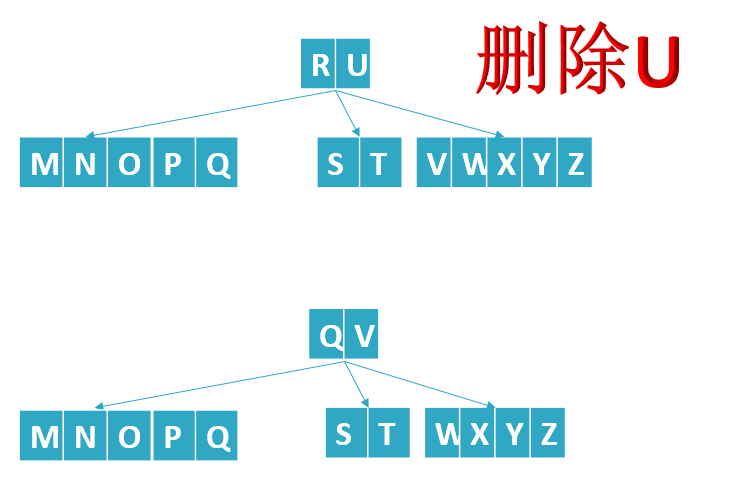
1. 当前结点为内结点，左孩子至少包含T个关键字



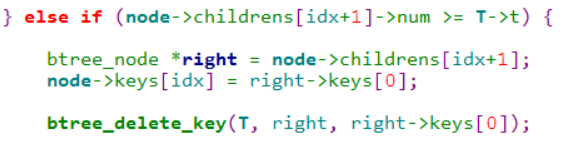
实现代码如下：



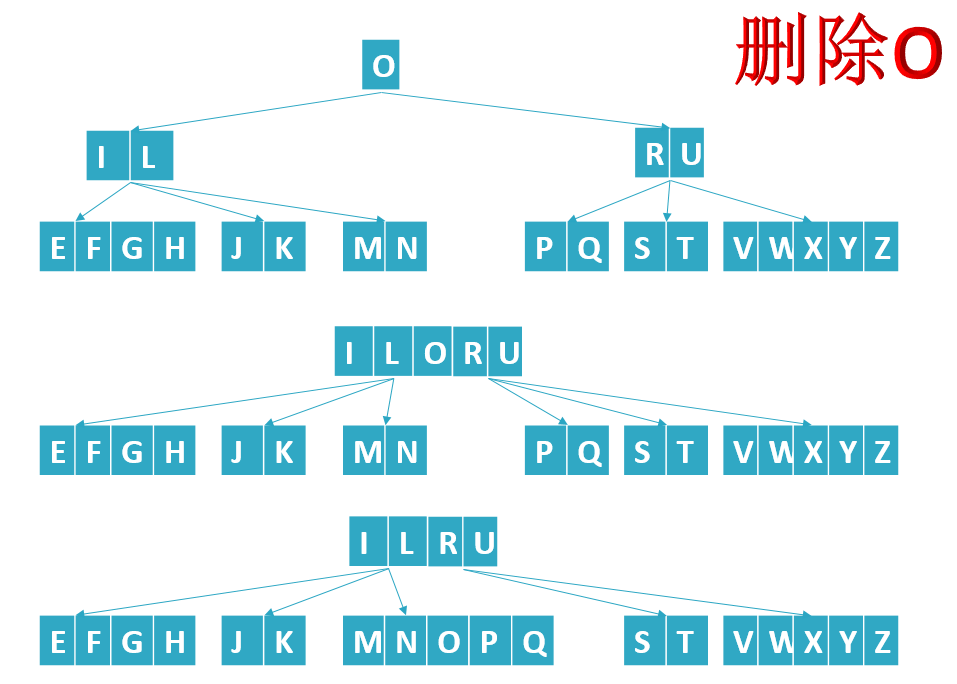
1. 当前结点为内结点，右孩子至少包含T个关键字



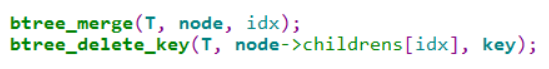
实现代码如下：



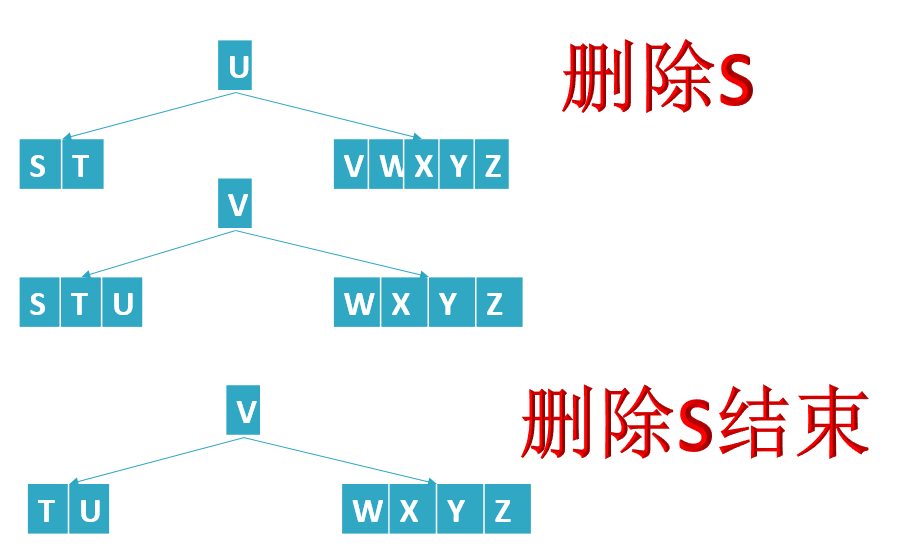
1. 左右孩子结点都是T-1个关键字，将当前key与左右孩子合并，然后在删除。



实现代码



1. 相邻左右结点包含至少有T个结点，且当前孩子的结点是T-1个结点。



实现代码如下：

