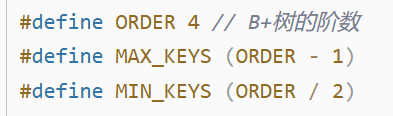
key值最大、最小



# B+树的插入

B+树的插入首先在叶子节点中进行，若插入后叶子节点中的关键字个数大于m，则需要进行分裂操作。分裂需要两个步骤：

节点分裂：将原节点中的关键字和子节点平均分配到两个新的节点中，原节点中的第m/2个（下取整，下同）关键字上升到其父节点中（父节点中关键字的个数加1），若没有父节点（原节点为根节点），则创建一个新的根节点。

调整索引：当分裂操作使得节点中关键字的个数超过m-1个时，需要对父节点进行分裂操作（即使父节点的关键字个数没有超过m-1，但只要其有子节点的关键字个数超过m-1，也需要进行分裂，以保证B+树的性质）。这个分裂过程可能会递归向上进行，甚至可能导致根节点的分裂，从而增加树的高度。

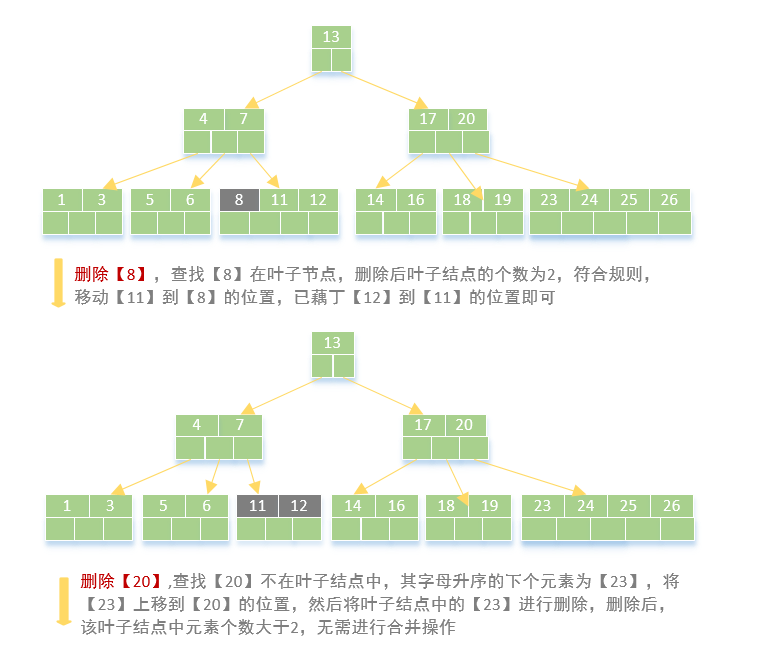
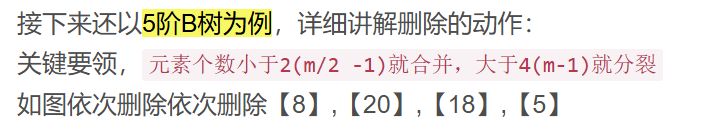
插入操作伪代码:

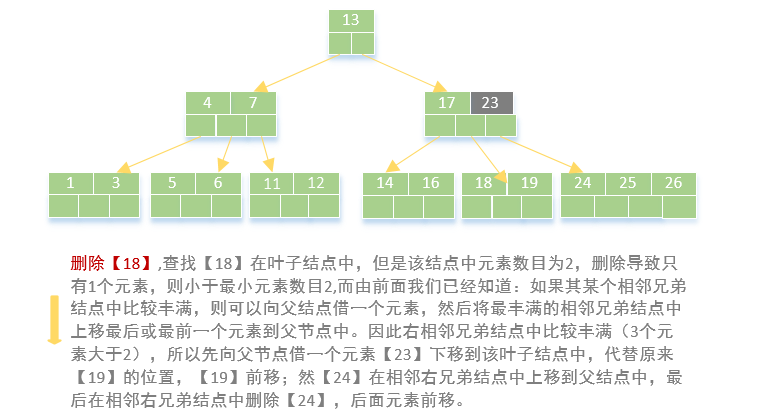


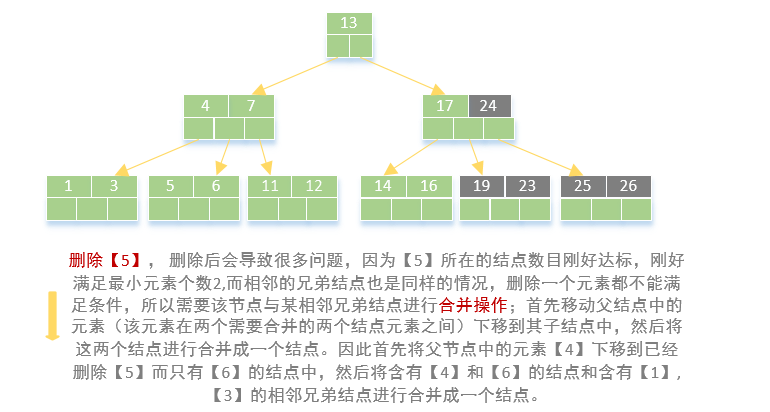
# B+树的删除

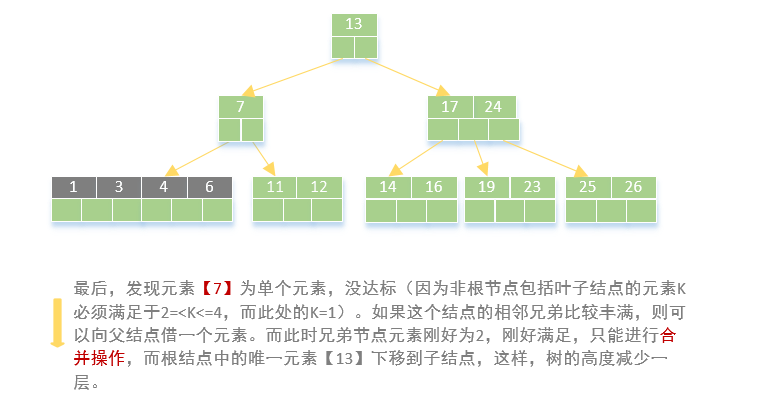
B+树的删除操作稍微复杂一些，需要考虑多种情况。如果被删除的关键字位于非叶子节点，则需要用其后继节点（或前驱节点）中的最小（或最大）关键字替换，并转化为在叶子节点中的删除。如果被删除的关键字位于叶子节点，且该叶子节点中的关键字个数大于等于ceil(m/2)，则可以直接删除。否则，需要考虑合并叶子节点或者从相邻节点借调关键字。以下是删除操作的伪代码实现：

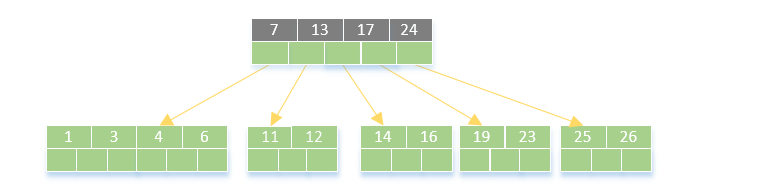












# B+树的检索、插入和删除

检索

在B+树中检索关键码key的方法与B树的检索方式相似，但若在内部节点中找到检索的关键码时，检索并不会结束，要继续找到B+树的叶子结点为止。

插入

与B树的插入操作相似，总是插到叶子结点上。当叶节点中原关键码的个数等于m时，该节点分裂成两个节点，分别使关键码的个数为 (m+1)/2 （向上取整）和 (m+1)/2 （向下取整）。

删除

仅在叶节点删除关键码。若因为删除操作使得节点中关键码数少于 m/2（向下取整）时，则需要调整或者和兄弟节点合并。合并的过程和B树类似，区别是父节点中作为分界的关键码不放入合并后的节点中。