m阶B-树 balance tree

根据数据规模确定B-树的阶数m，计算树中key数量范围

B-树的插入、构建从最底层非叶子结点开始的，B-树是由下往上的。

m阶B-树，每个节点最多有m子树，最多有m-1个key；根节点至少有一个key，非根节点至少有 ceil(m/2)个key

B-树成长过程

B-树由下往上长，始终在最低层非叶子节点上插入新节点，当关键字个数超过m-1时，要分裂成2个节点，中间的key成为根节点。

寻找插入点，是从根节点往下查找

删除节点，删除父子换，借，左右合并

B-树，适合单个查找，不适合范围查找。

叶子结点不包含关键字，所以可以把叶子结点看成在树里实际上并不存在外部结点，指向这些外部结点的指针为空，叶子结点的数目正好等于树中所包含的关键字总个数加1。

m阶B+树

节点中key的数量和子节点的个数相等

叶子节点有指针连接起来

所有非叶子节点key值，在叶子节点中存在（也就是说，父节点key值，在子节点中存在）

根节点最大值，就是整个树的最大值（父节点key值是子节点中key值最大的）

只有叶子节点带有数据

B+树作为索引的优点

1、IO次数更少：节点没有数据，可以加载更多的索引到内存中，查找过程减少IO

2、相比于B-树，节点中关键字可以多一个，所以B+树更矮胖

查询每次都要到叶子节点才能取出数据。

范围查询更方便（叶子节点是一个有序链表）

通常在b+树上有两个头指针，一个指向根结点，一个指向关键字最小的叶子结点。

B+树相对于B-树的优势

b+树的中间节点不保存数据，所以磁盘页能容纳更多节点元素，更“矮胖”；

b+树查询必须查找到叶子节点，b树只要匹配到即可不用管元素位置，因此b+树查找更稳定（并不慢）；

对于范围查找来说，b+树只需遍历叶子节点链表即可，b树却需要重复地中序遍历，

https://www.cnblogs.com/xueqiuqiu/articles/8779029.html

2-3查找树

两种节点类型：

2——1个key两个链接（子节点）

3——2个key三个链接（子节点）

一颗完美的平衡的2-3树所有的空链（没有节点的链）到根节点的距离是相同的

查找过程

1、判断根节点命中

2、如果未命中，确定在哪一条链，然后在子树中重复第一步

结果：在子树中找到，或者到最下面的空链，未找到。

红黑树

红黑树由2-3树拆分得到，红链为左链

红链：将两个2节点连接起来构成三一节点

黑链：2-3树中的普通链接

二叉树的查询效率很高，但是数据库却选用B+树，是因为索引的数据量会很大的，索引只能部分加载到内存，因为存储的原因，树的高度越高，IO的次数就越多，对查询效率影响很大。

所以选择M阶B+树，相对矮胖的树。