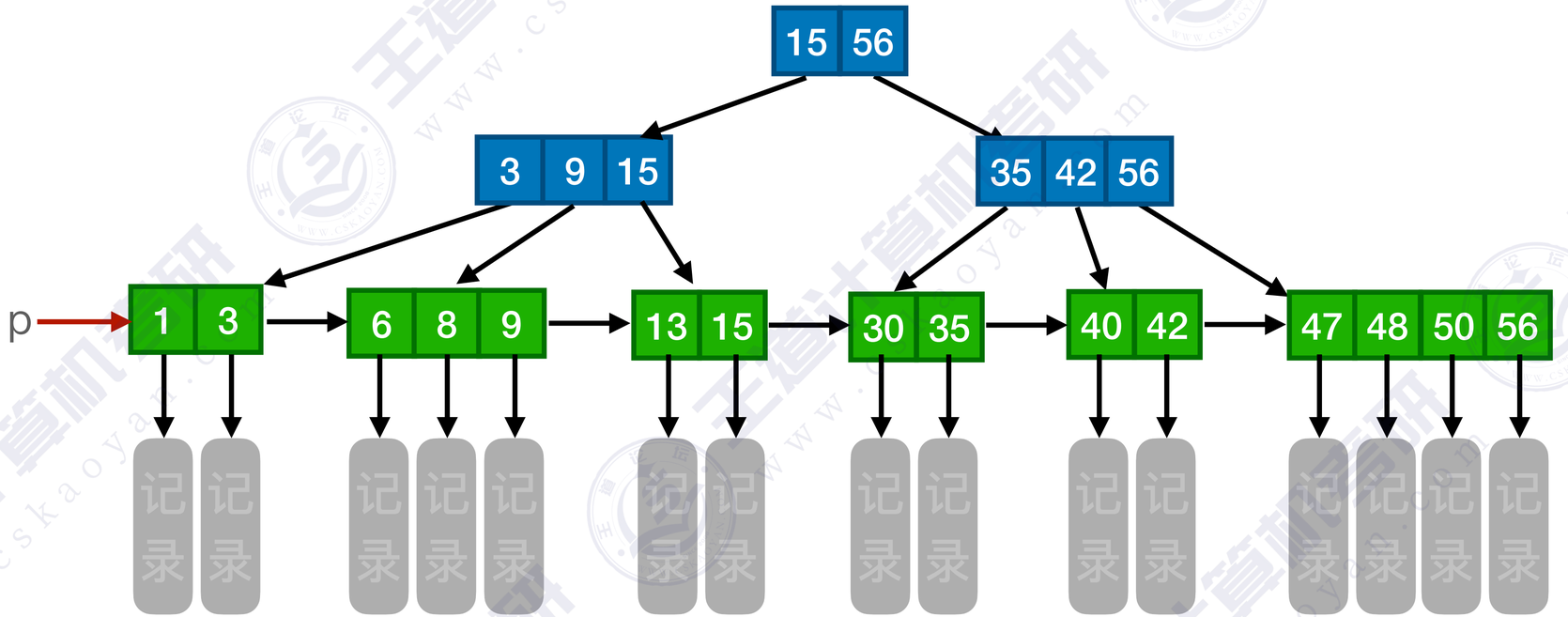
本节内容

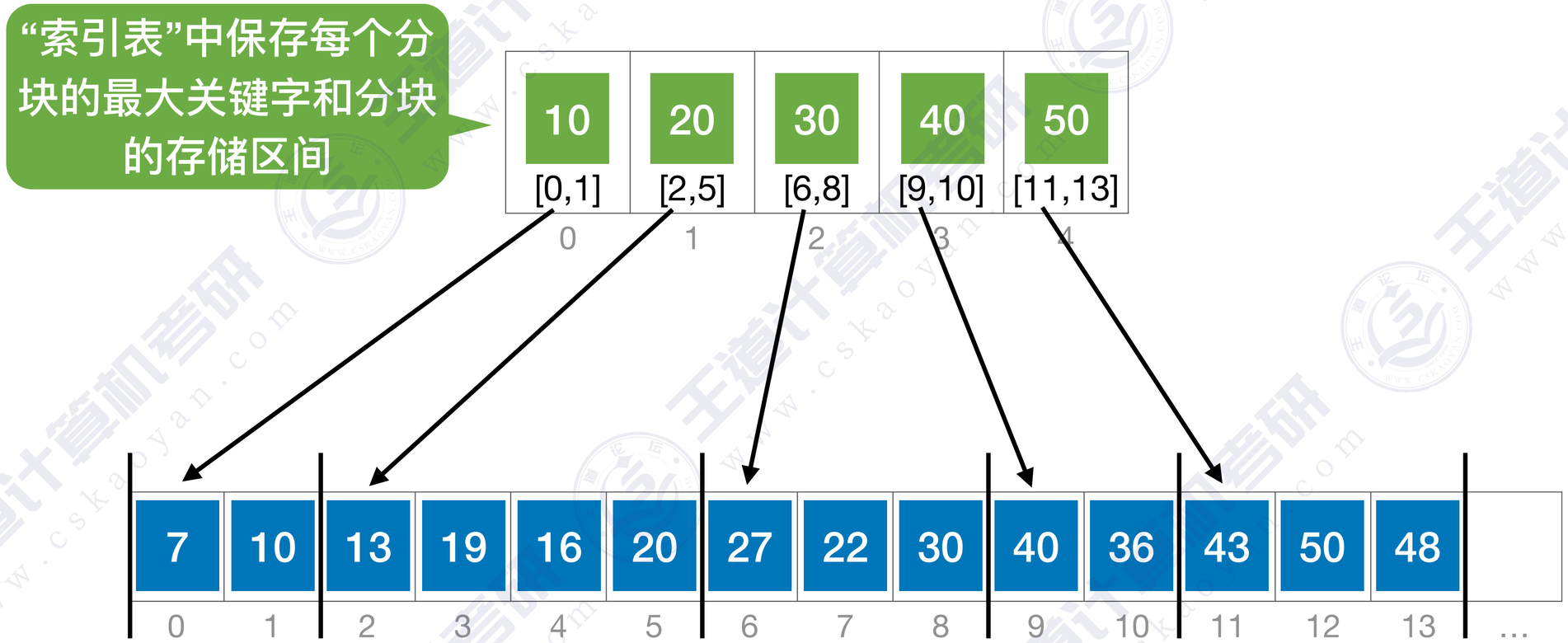
B+树

B+树



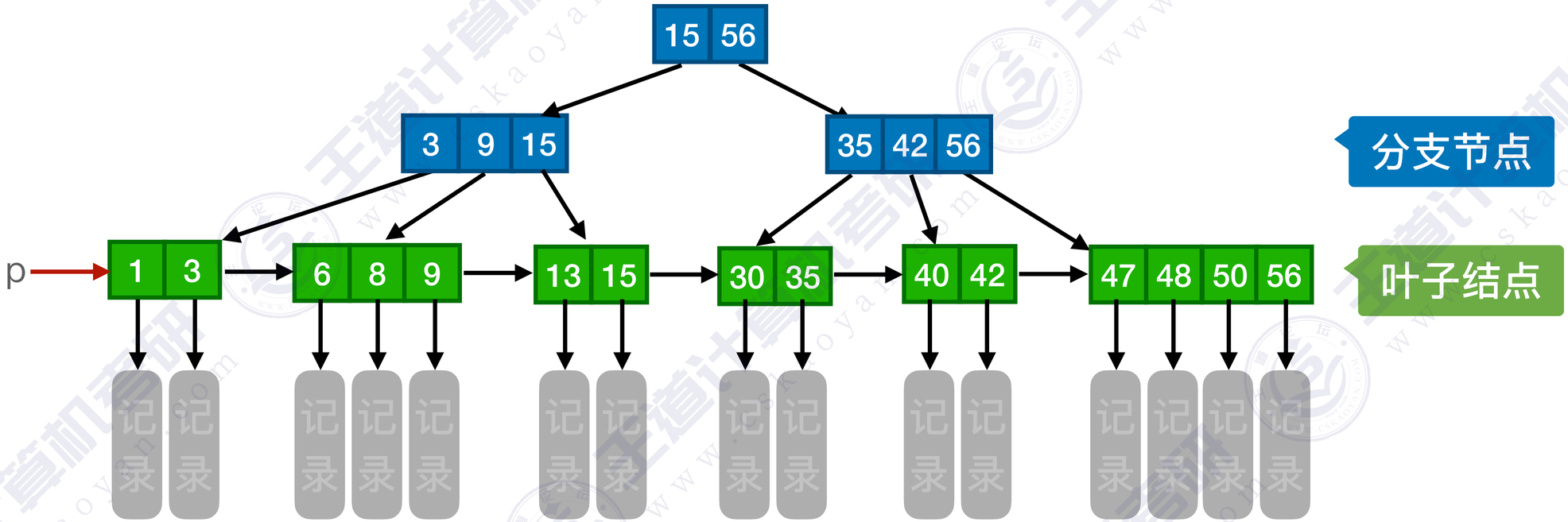
注：以上是一棵4阶B+树

对比：分块查找



特点：块内无序、块间有序

B+树



注：以上是一棵4阶B+树

一棵m阶的B+树需满足下列条件：

1）每个分支结点最多有m棵子树（孩子结点）。

2）非叶根结点至少有两棵子树，其他每个分支结点至少有 $\lceil m/2 \rceil$ 棵子树。

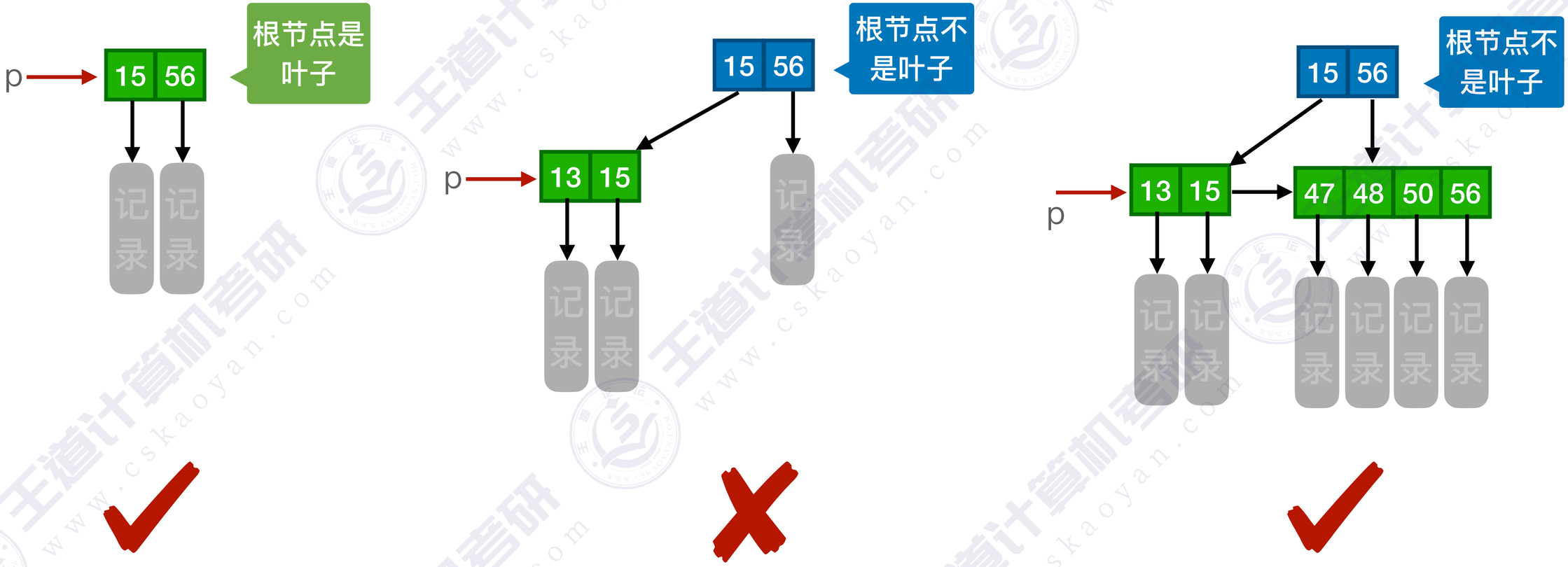
3）结点的子树个数与关键字个数相等。

4）所有叶结点包含全部关键字及指向相应记录的指针，叶结点中将关键字按大小顺序排列，并且相邻叶结点按大小顺序相互链接起来。

支持顺序查找

5）所有分支结点中仅包含它的各个子结点中关键字的最大值及指向其子结点的指针。

B+树

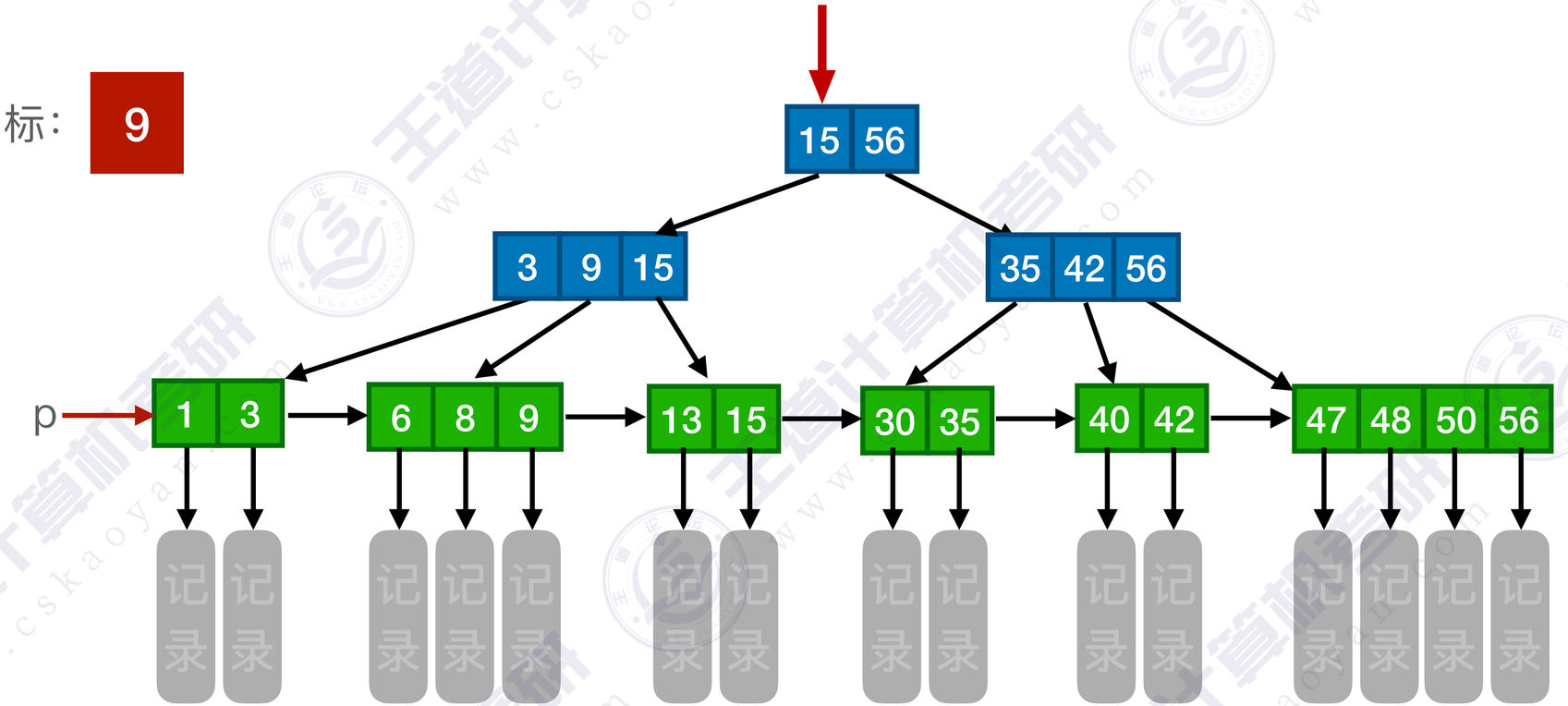


可以理解为：要追求“绝对平衡”，即所有子树高度要相同

2）非叶根结点至少有两棵子树，其他每个分支结点至少有 $\lceil m/2 \rceil$ 棵子树。

B+树的查找

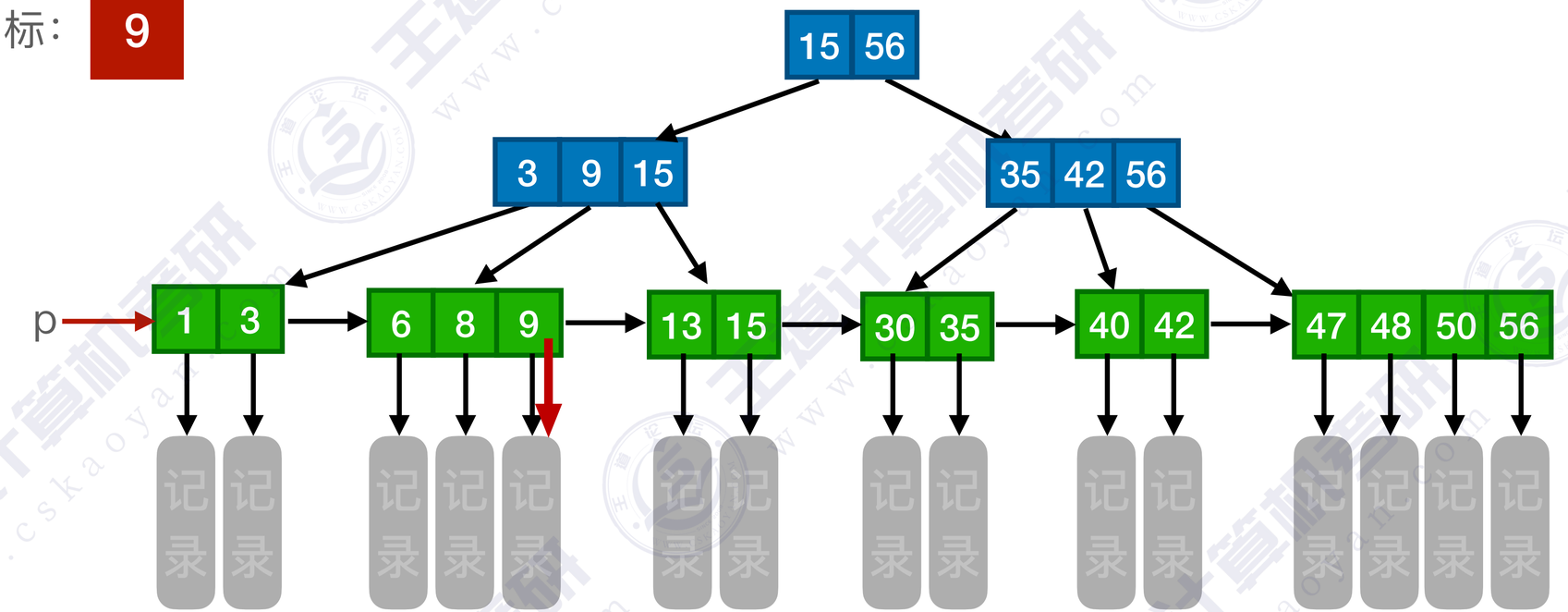
查找目标：9



多路查找

B+树的查找

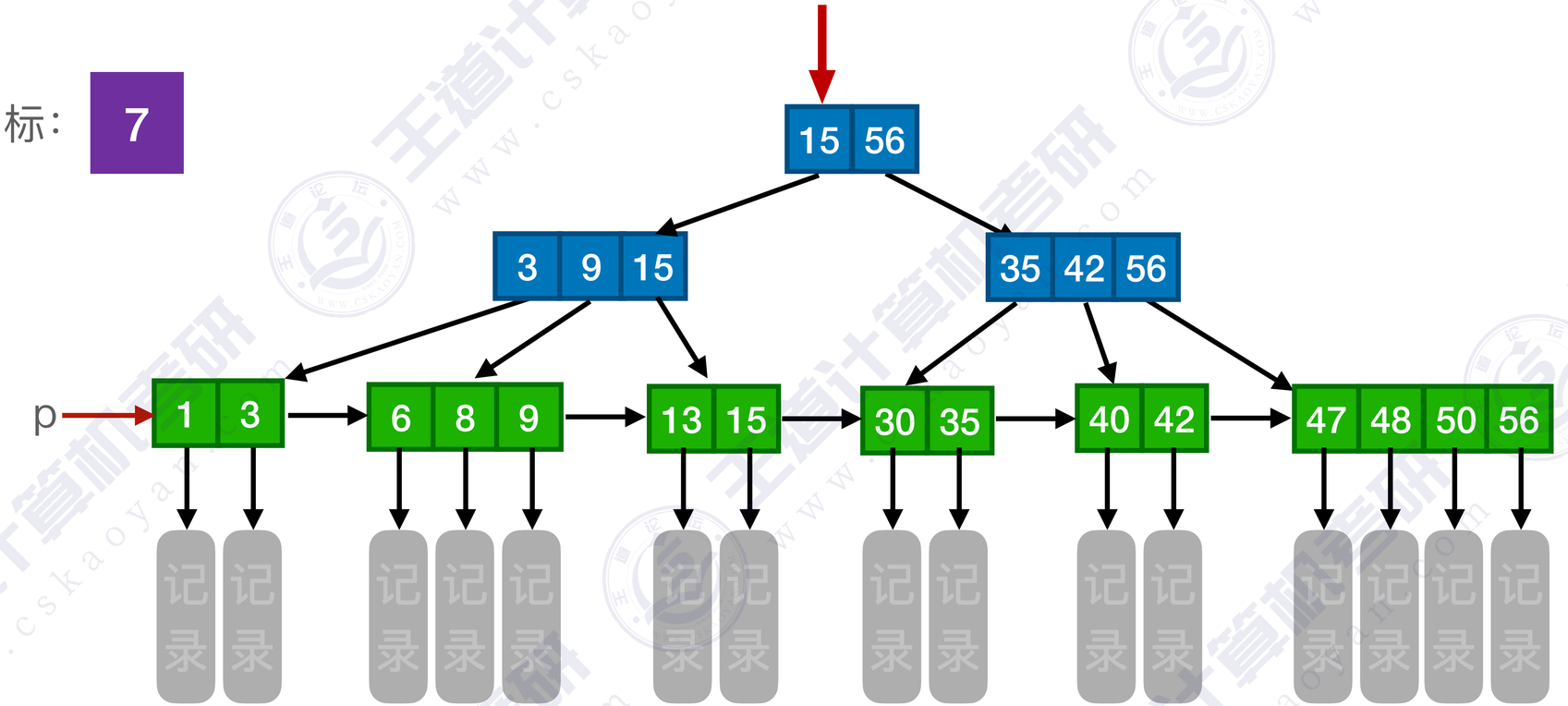
查找目标：9



多路查找

B+树的查找

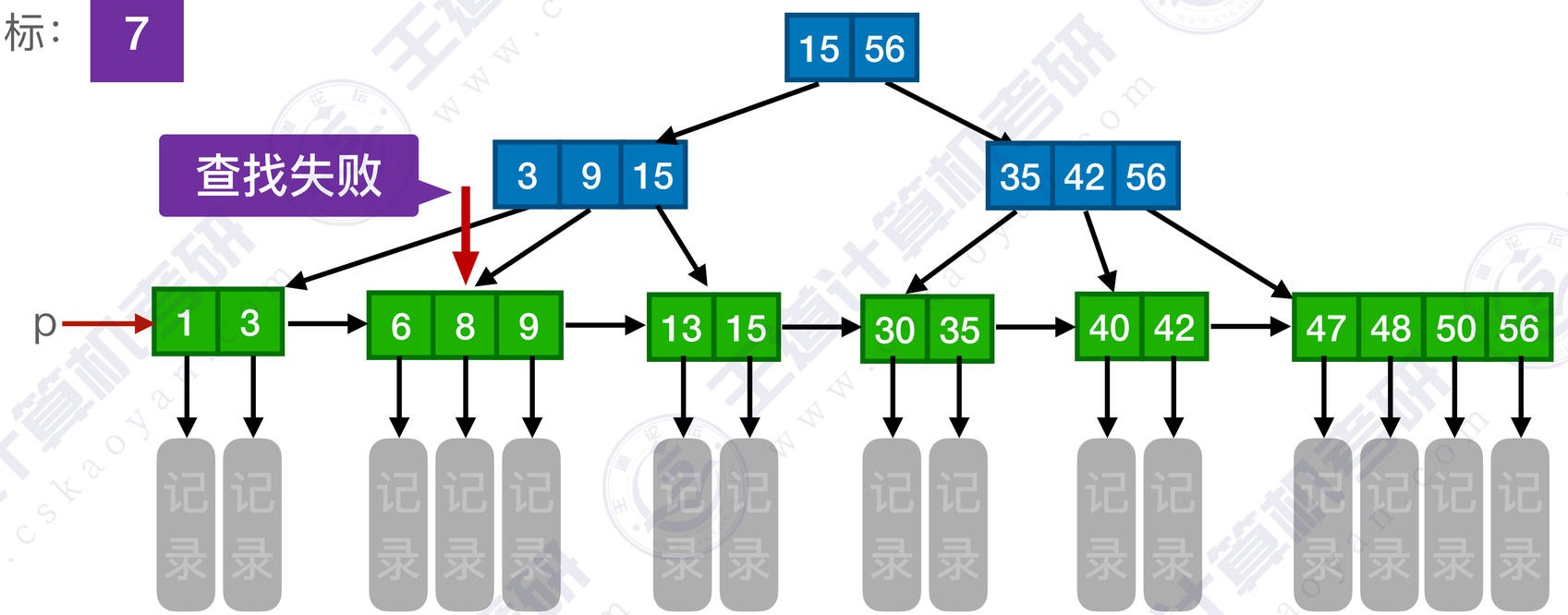
查找目标：7



多路查找

B+树的查找

查找目标：7

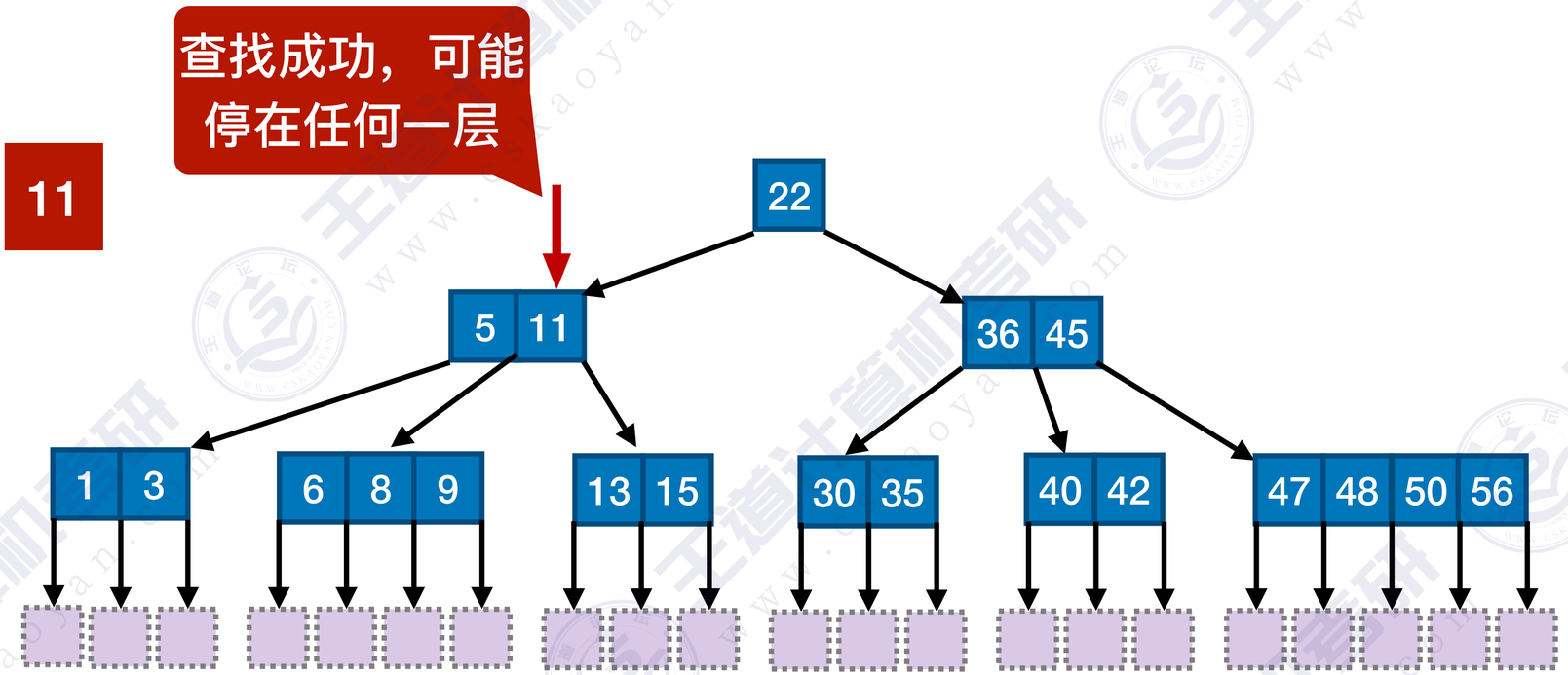


多路查找

B+树中，无论查找成功与否，最终一定都要走到最下面一层结点

对比：B树的查找

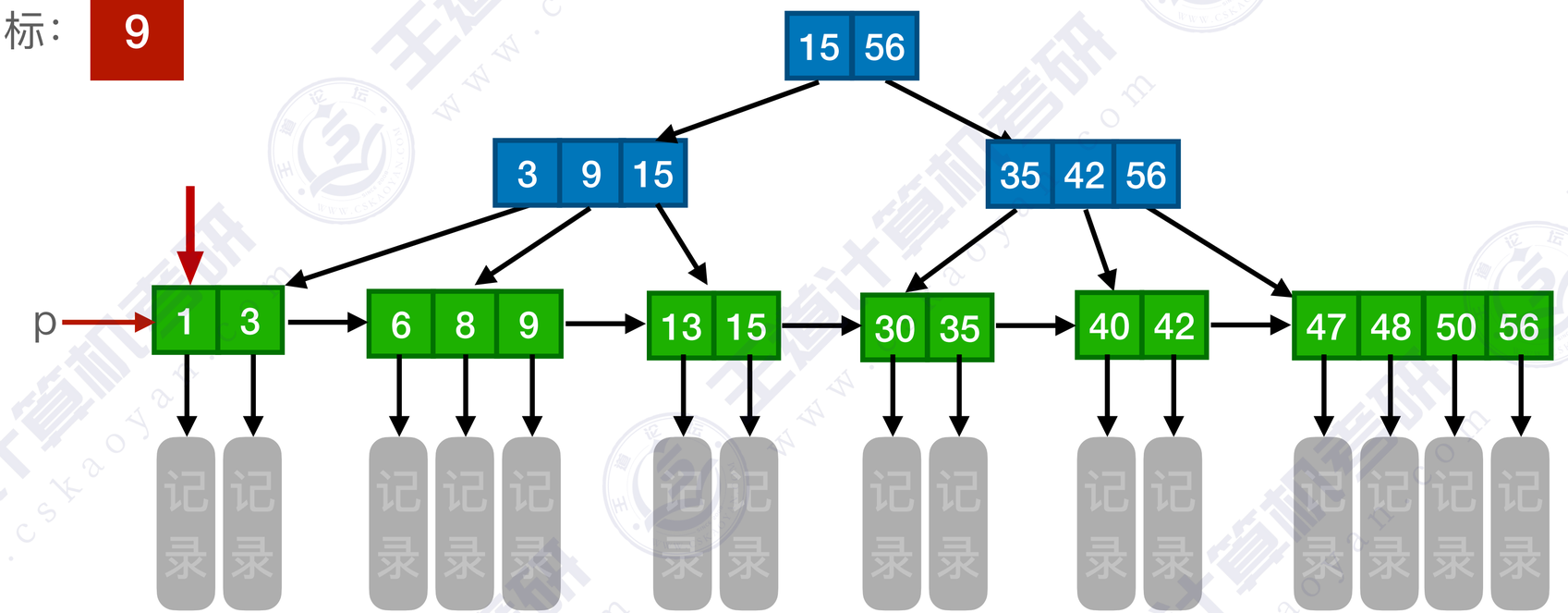
查找目标：11



注：以上是一棵5阶B树

B+树的查找

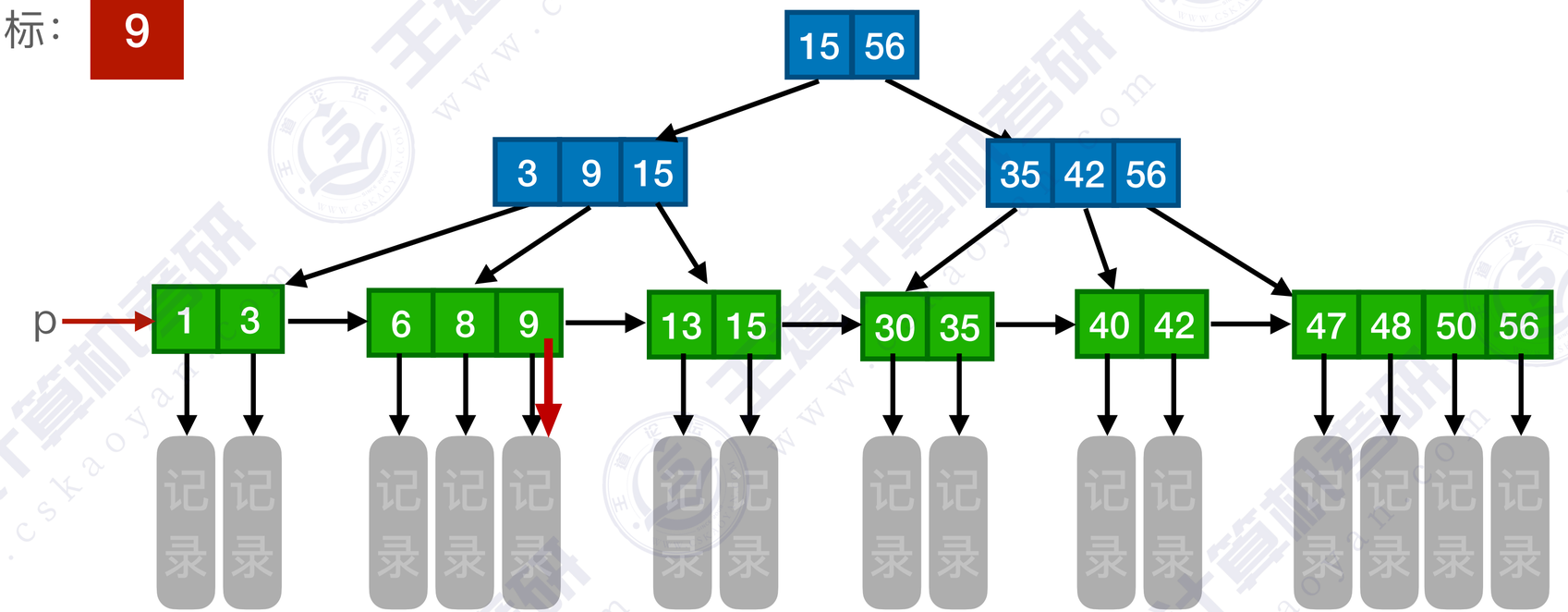
查找目标：9



顺序查找

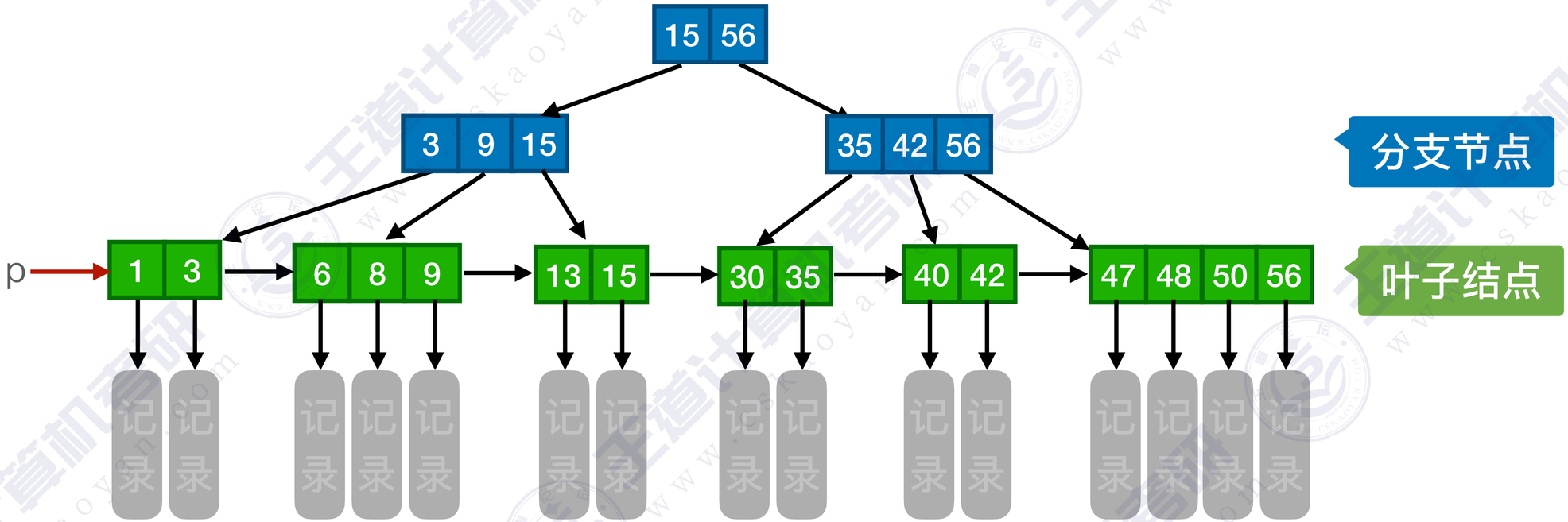
B+树的查找

查找目标：9



顺序查找

B+树 VS B树

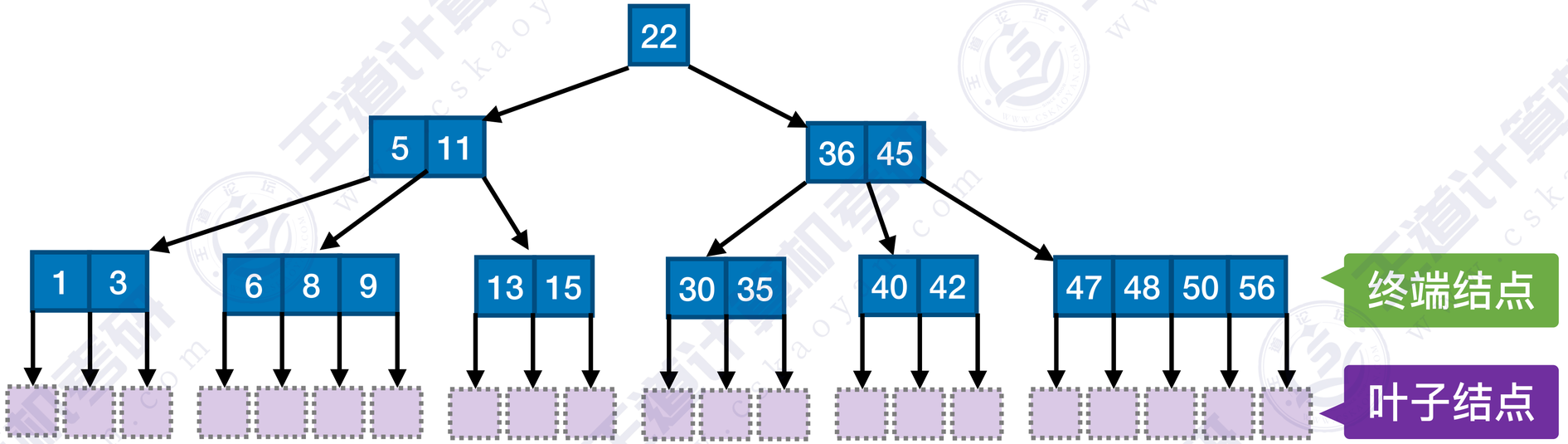


注：以上是一棵4阶B+树

m阶B+树：

1） 结点中的n个关键字对应n棵子树

B+树 VS B树

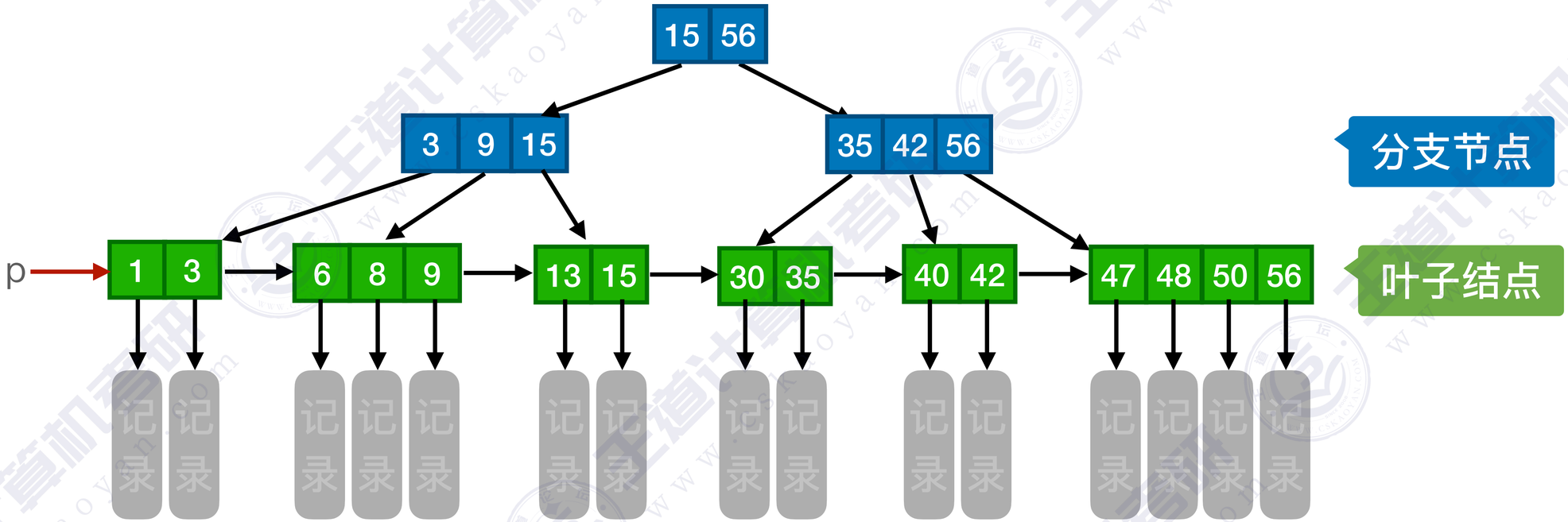


注：以上是一棵5阶B树

m阶B树：

1） 结点中的n个关键字对应n+1棵子树

B+树 VS B树



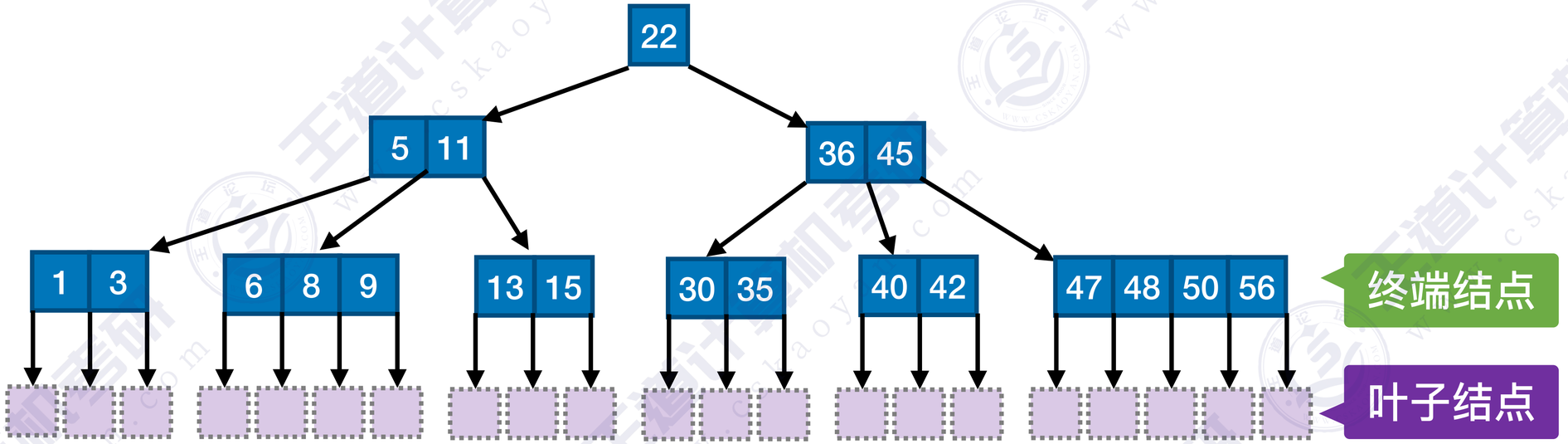
注：以上是一棵4阶B+树

m阶B+树：

2） 根节点的关键字数n∈[1, m]

其他结点的关键字数n∈[$\lceil m/2 \rceil$, m]

B+树 VS B树



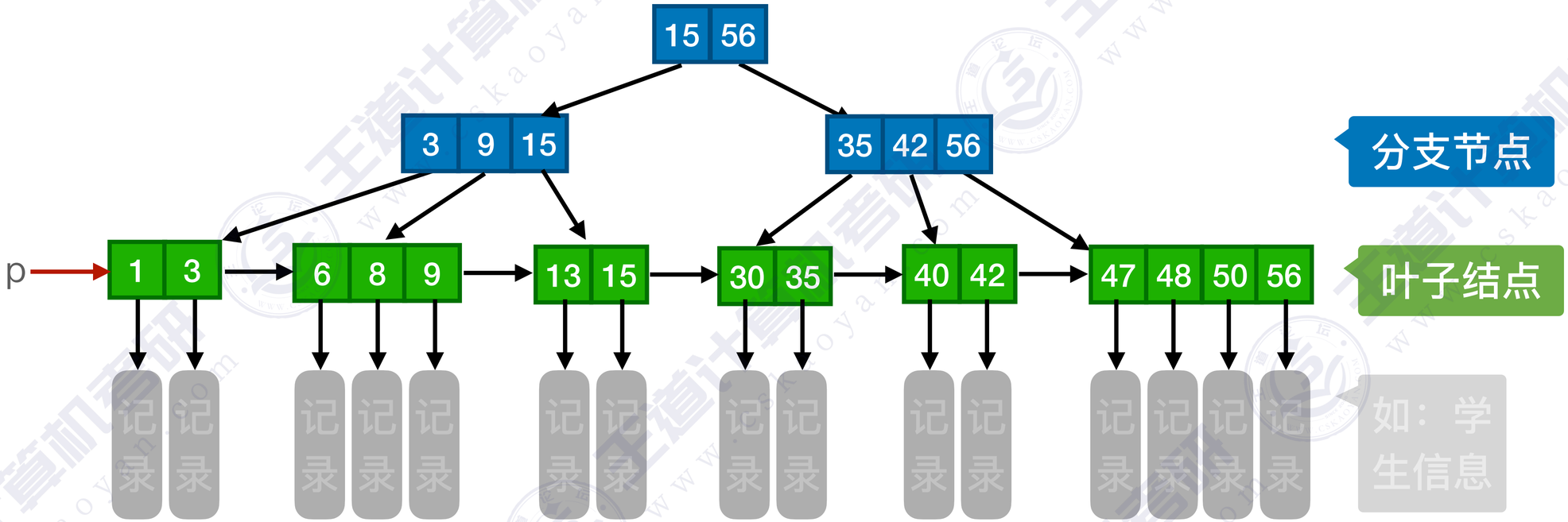
注：以上是一棵5阶B树

m阶B树：

2） 根节点的关键字数n∈[1, m–1]。

其他结点的关键字数n∈[$\lceil m/2 \rceil$–1, m–1]

B+树 VS B树

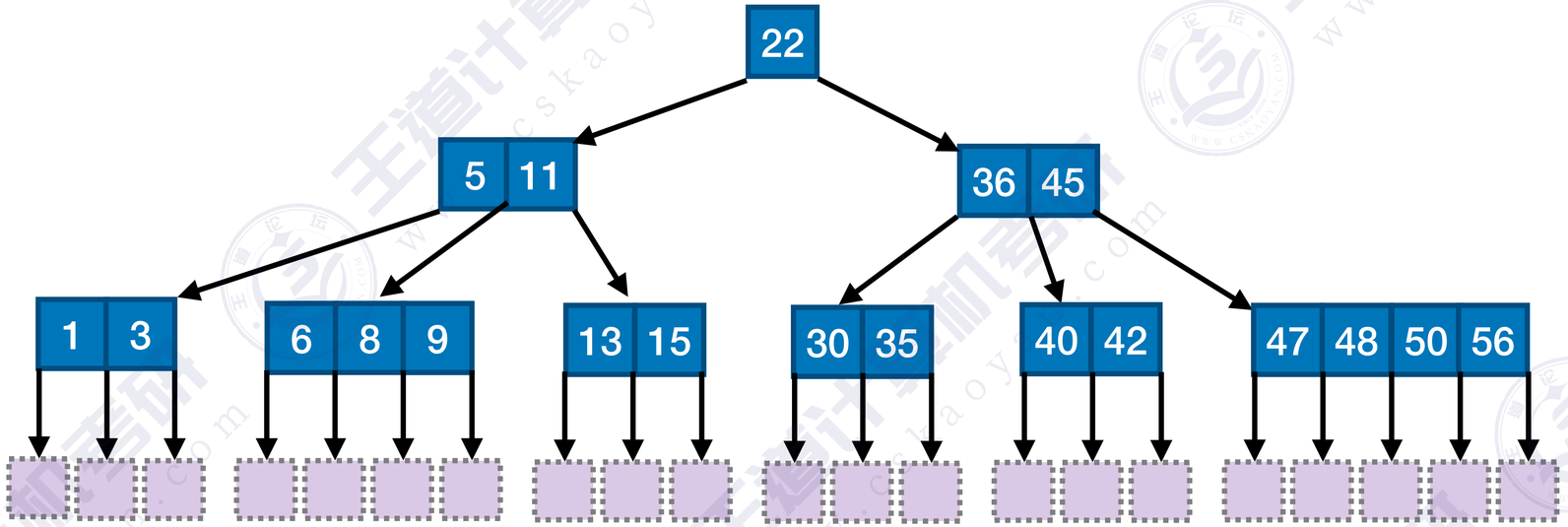


注：以上是一棵4阶B+树

m阶B+树：

3）在B+树中，叶结点包含全部关键字，非叶结点中出现过的关键字也会出现在叶结点中

B+树 VS B树

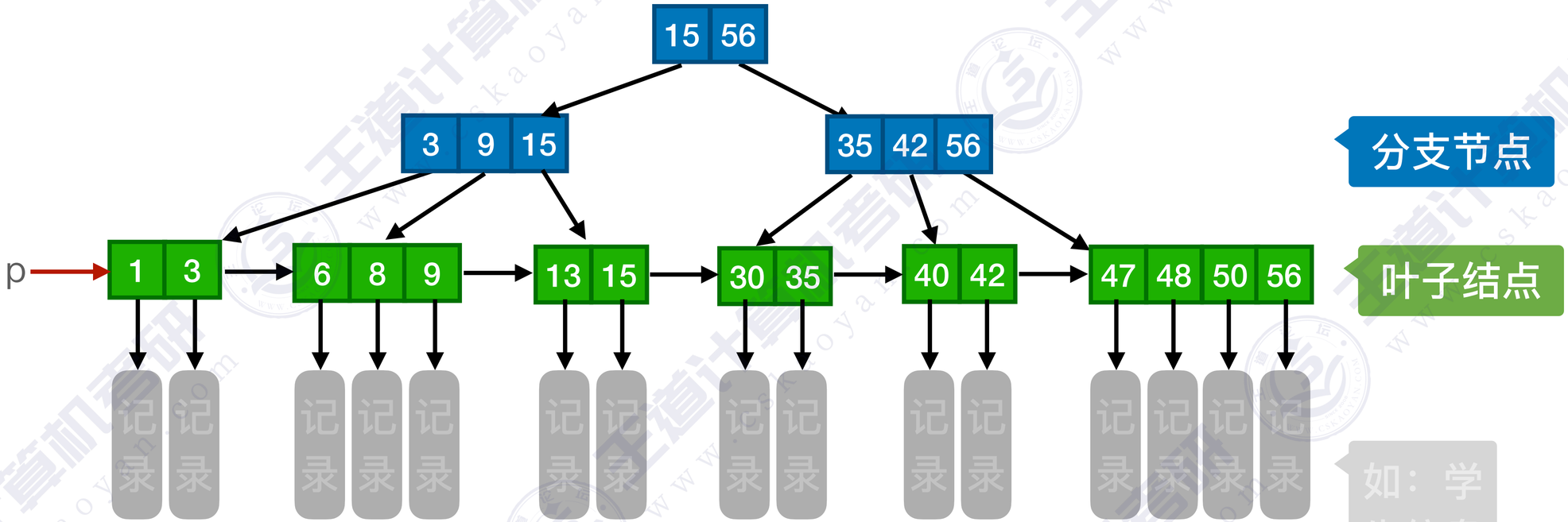


注：以上是一棵5阶B树

m阶B树：

3） 在B树中，各结点中包含的关键字是不重复的

B+树 VS B树

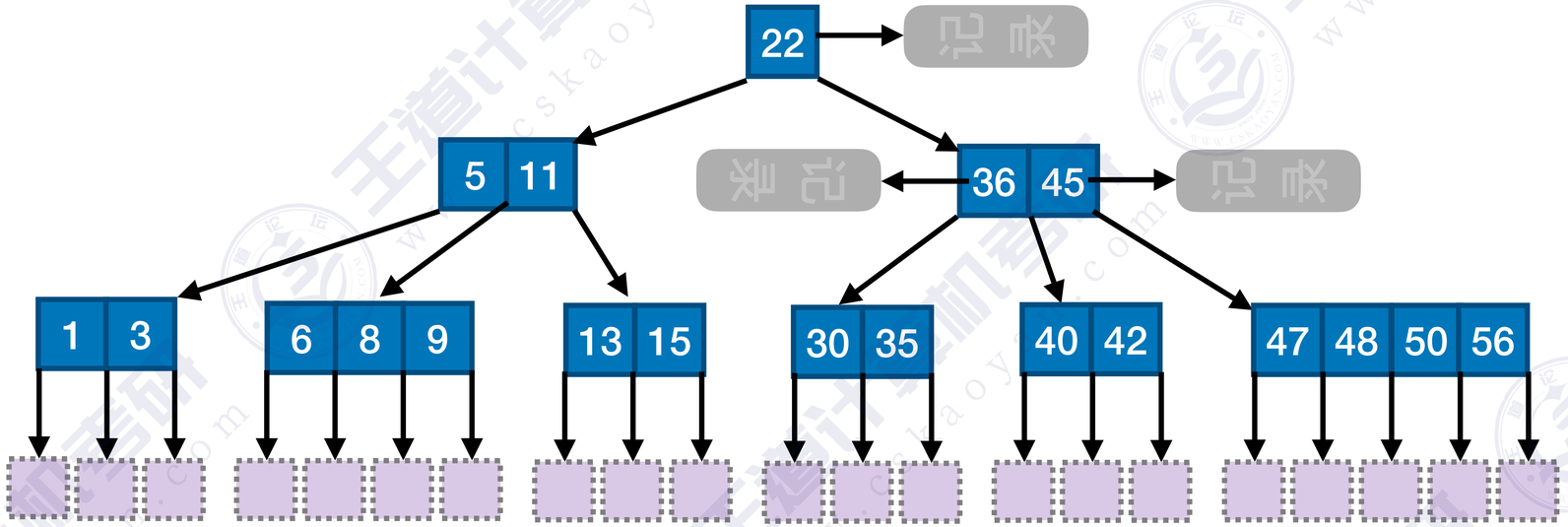


注：以上是一棵4阶B+树

m阶B+树：

4）在B+树中，叶结点包含信息，所有非叶结点仅起索引作用，非叶结点中的每个索引项只含有对应子树的最大关键字和指向该子树的指针，不含有该关键字对应记录的存储地址。

B+树 VS B树

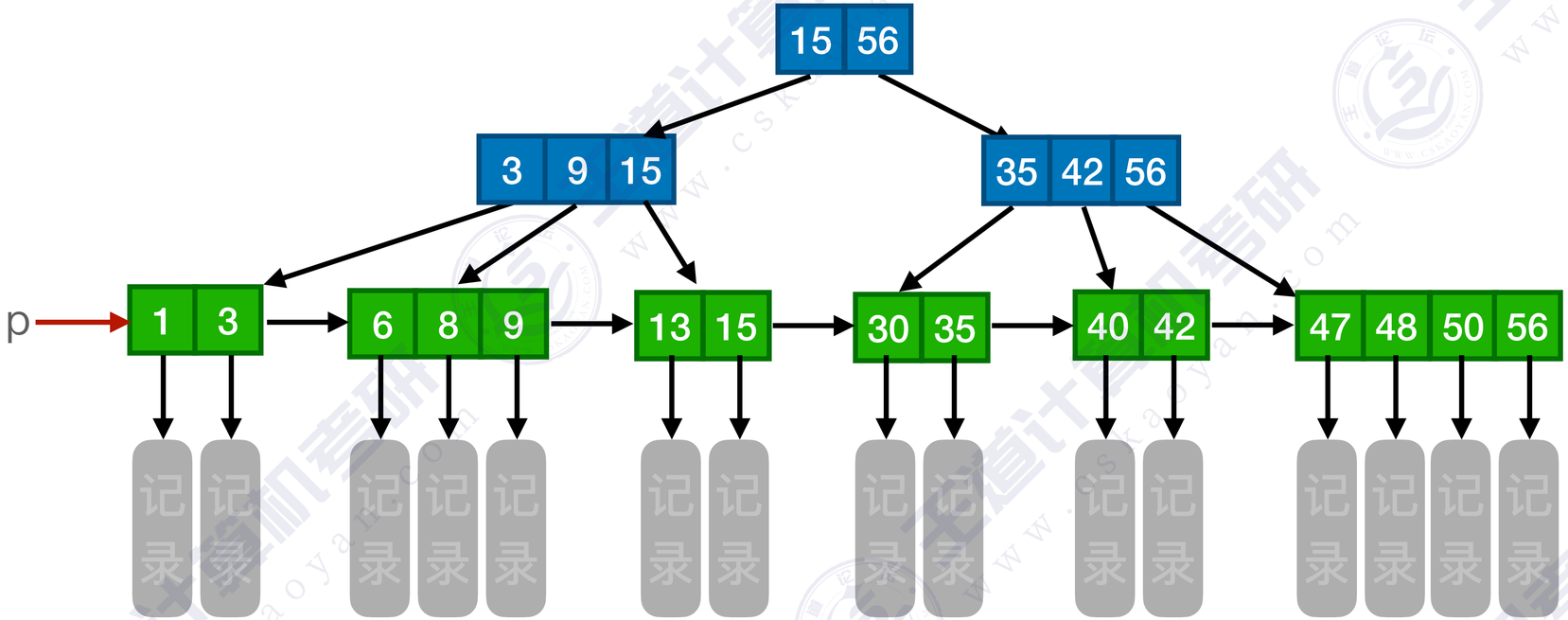


注：以上是一棵5阶B树

m阶B树：

4） B树的结点中都包含了关键字对应的记录的存储地址

B+树 VS B树



典型应用：关系型数据库的“索引”（如MySQL）

在$B+$树中，非叶结点不含有该关键字对应记录的存储地址。
可以使一个磁盘块可以包含更多个关键字，使得B+树的阶更大，树高更矮，读磁盘次数更少，查找更快

知识回顾与重要考点

	m阶B树	m阶B+树
类比	二叉查找树的进化——>m叉查找树	分块查找的进化——>多级分块查找
关键字与分叉	n个关键字对应n+1个分叉（子树）	n个关键字对应n个分叉
结点包含的信息	所有结点中都包含记录的信息	只有最下层叶子结点才包含记录的信息 （可使树更矮）
查找方式	不支持顺序查找。查找成功时，可能停在任何一层结点，查找速度“不稳定”	支持顺序查找。查找成功或失败都会到达最下一层结点，查找速度“稳定”
相同点	除根节点外，最少 $\lceil m/2 \rceil$ 个分叉（确保结点不要太“空”） 任何一个结点的子树都要一样高（确保“绝对平衡”）	