

Lecture 27: B 树

27.1 B 树基础概念

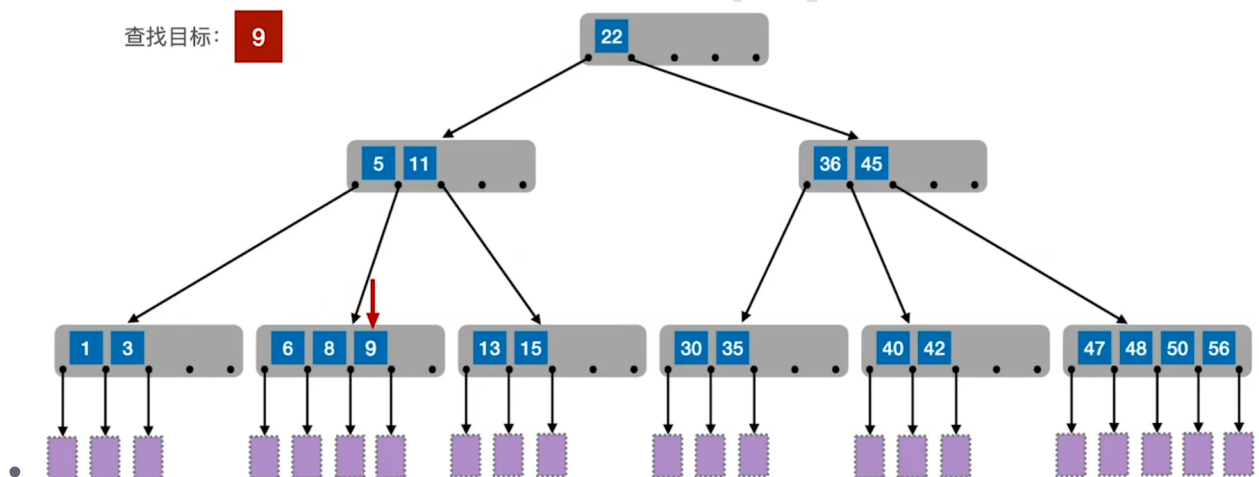
27.1.1 引入: m 叉排序树

引入: m 叉排序树

- 二叉排序树是把每一个问题递归分成了两个子问题;
- 而 B 树是把每一个问题递归分成了 n 个子问题;
- 比如五叉排序树: 最少 1 个关键字, 2 个分叉最多 4 个关键字, 5 个分叉
- 结点内关键字有序;

查找示例

- 示例:
 - 在五叉排序树中查找关键字 9;
- 图示:



基础概念: m 叉排序树 $\rightarrow$ B 树

- 效率:
 - 若每个结点内关键字太多导致树变高, 要查更多层结点, 效率低
- 方法: 保证 m 叉排序树的效率
 - 策略一: m 叉查找树中, 规定除了根节点外, 任何结点至少有 $\lceil m/2 \rceil$ 个分叉, 即至少含有 $\lceil m/2 \rceil - 1$ 个关键字;
 - 策略二: m 叉查找树中, 规定对于任何一个结点, 其所有子树的高度都要相同;
- 什么是 B 树:
 - 满足了策略一和策略二的 m 叉排序树, 即为 B 树;

27.1.2 B 树的定义

基础概念

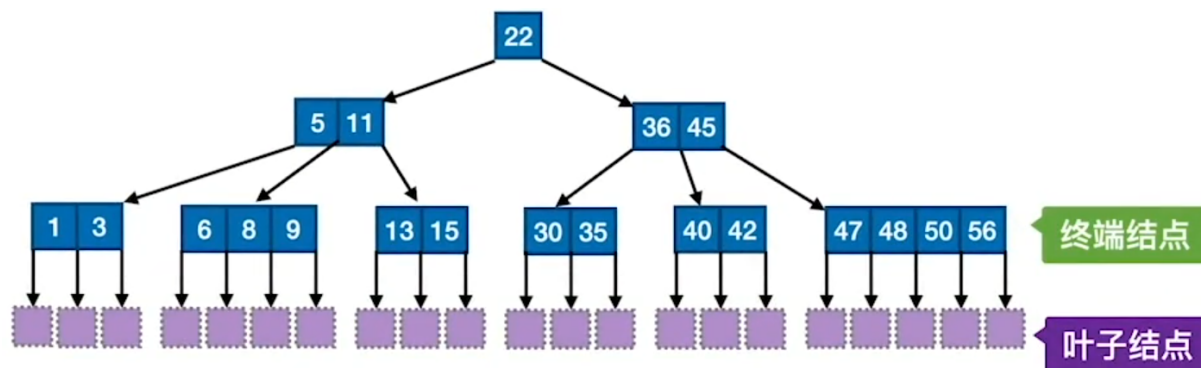
- 概念:

- B 树，又称**多路平衡查找树**，B 树中所被允许的孩子个数的最大值称为 B 树的阶，通常用 m 表示；

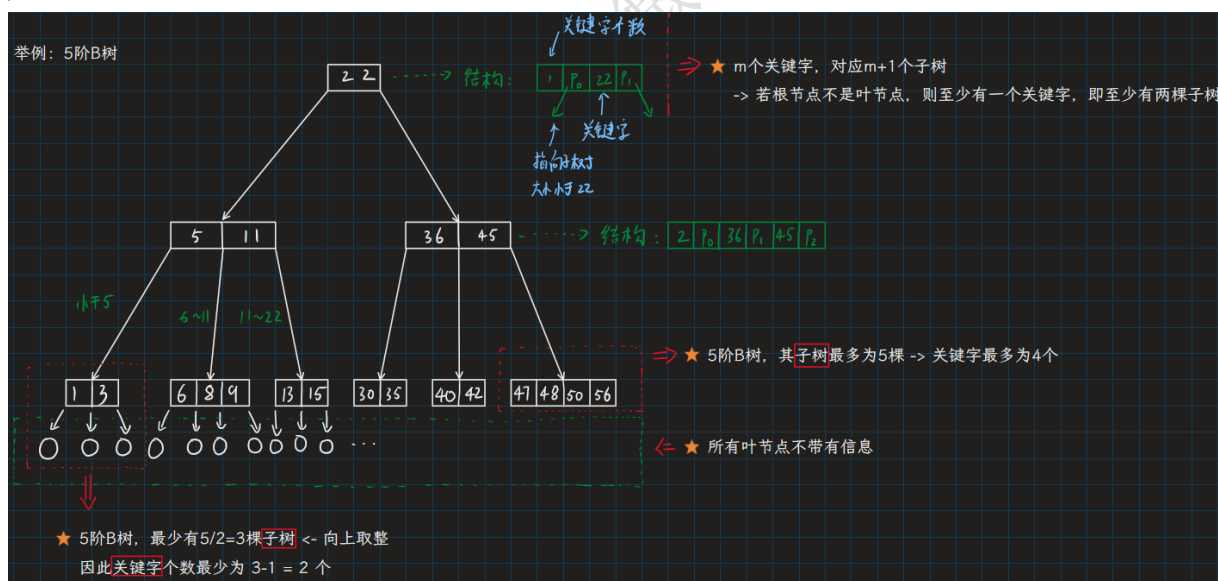
性质：

- 一棵 m 阶 B 树或为空树，或为满足如下特性的 m 叉树：
 1. 树中每个结点至多有 m 棵子树，即至多含有 $m - 1$ 个关键字；
 2. 若根结点不是终端结点，则至少有两棵子树；
 3. 除根结点外的所有非叶结点至少有 $\lceil m/2 \rceil$ 棵子树，即至少含有 $\lceil m/2 \rceil - 1$ 个关键字；
 4. 所有的叶结点都出现在同一层次上，并且**不带信息**（可以视为外部结点或类似于折半查找判定树的查找失败结点，**实际上这些结点不存在**，指向这些结点的指针为空）

图示：



解析：

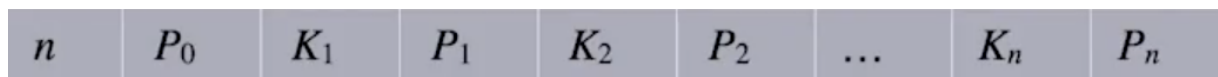


非叶节点

概念：

- 其中， $K_i (i = 1, 2, \dots, n)$ 为结点的关键字，且满足 $K_1 < K_2 < \dots < K_n$; $P_i (i = 0, 1, \dots, n)$ 为指向子树根结点的指针，且指针 P_{i-1} 所指子树中所有结点的关键字均小于 K_i , P_i 所指子树中所有结点的关键字均大于 K_i ;
- 并且 $n: (\lceil m/2 \rceil - 1 \leq n \leq m - 1)$

图示：



m 阶 B 树性质总结

- 1. 根节点的子树数 $\in [2, m]$, 关键字数 $\in [1, m-1]$ 其他结点的子树数 $\in [\lceil m/2 \rceil, m]$ 。关键字数 $\in [\lceil m/2 \rceil - 1, m - 1]$;
- 2. 对任一结点, 其所有子树高度都相同;
- 3. 关键字的值:

子树0 < 关键字1 < 子树1 < 关键字2 < 子树2 <

- 4. 核心性质: 高度的最小值与最大值
 - h 的最小值 $\rightarrow$ 关键字个数最多时 $\rightarrow$ 每一个节点有最多的关键字数: $m - 1$, 每一层有最多节点数: $1 + m + m^2 + \dots + m^{h-1} \rightarrow$ 最大关键字数:
 $(m - 1)(1 + m + m^2 + \dots + m^{h-1}) = (m^h - 1) \geq n$

$$\log_m(n + 1) \leq h \leq \log_{\lceil m/2 \rceil} \frac{n + 1}{2} + 1$$

27.2 B 树的插入与删除

27.2.1 插入

概念

- 核心要求:
 - 1. 对 m 阶 B 树—除根节点外, 结点关键字个数 $\lceil m/2 \rceil - 1 \leq n \leq m - 1$;
 - 2. 子树 0 < 关键字 1 < 子树 1 < 关键字 2 < 子树 2 < ...
- 过程:
 - 新元素一定是插入到最底层“终端节点”, 用“查找”来确定插入位置在插入 key 后, 若导致原结点关键字数超过上限, 则从中间位置 ($\lceil m/2 \rceil$) 将其中的关键字分为两部分, **左部分包含的关键字放在原结点中, 右部分包含的关键字放到新结点中, 中间位置 ($\lceil m/2 \rceil$) 的结点插入原结点的父结点;**
 - 若此时导致其父结点的关键字个数也超过了上限, 则继续进行这种分裂操作, 直至这个过程传到根结点为止, 进而导致 B 树高度增1;

示例

- 背景:
 - 5 阶 B 树—结点关键字个数 $\lceil m/2 \rceil - 1 \leq n \leq m - 1$, 即: $2 \leq n \leq 4$
- 过程:
 - 新元素一定是**插入到最底层“终端节点”**, 用“**查找**”来确定**插入位置**;
 - 在插入 key 后, 若导致原结点关键字数超过上限, 则从中间位置 ($\lceil m/2 \rceil$) 将其中的关键字分为两部分, 左部分包含的关键字放在原结点中, 右部分包含的关键字放到新结点中, 中间位置 ($\lceil m/2 \rceil$) 的结点插入原结点的父结点;

27.2.2 删除

概念: 删除

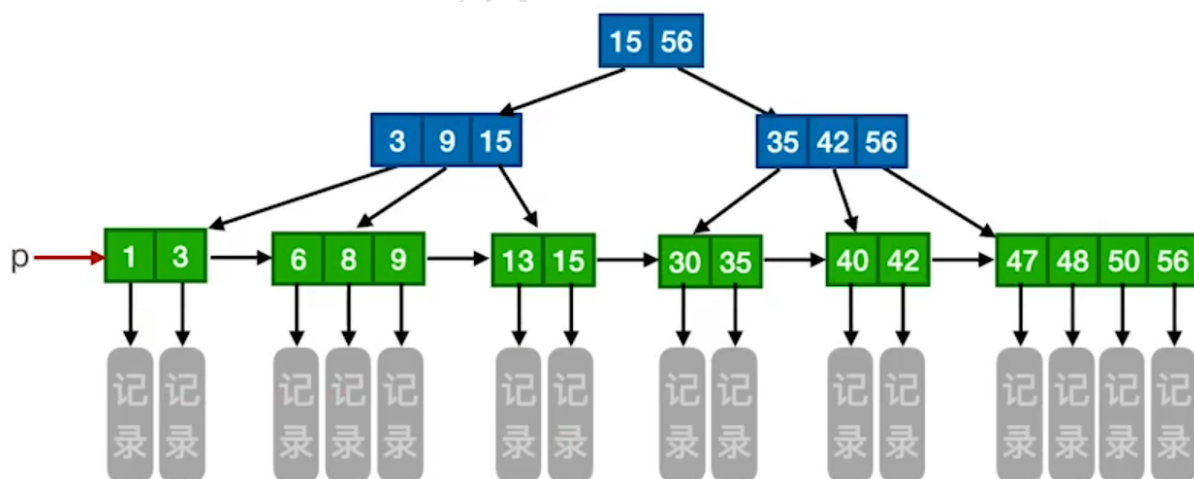
- 若被删除关键字在终端节点，则直接删除该关键字（要注意节点关键字个数是否低于下限 $\lceil m/2 \rceil - 1$ ）
- 若被删除关键字在非终端节点，则用直接前驱或直接后继来替代被删除的关键字
 - 直接前驱：当前关键字左侧指针所指子树中“最右下”的元素；
 - 直接后继：当前关键字右侧指针所指子树中“最左下”的元素；
- 对非终端结点关键字的删除，必然可以转化为对终端结点的删除操作；

27.3 B+ 树

27.3.1 B+ 树基础概念

概念

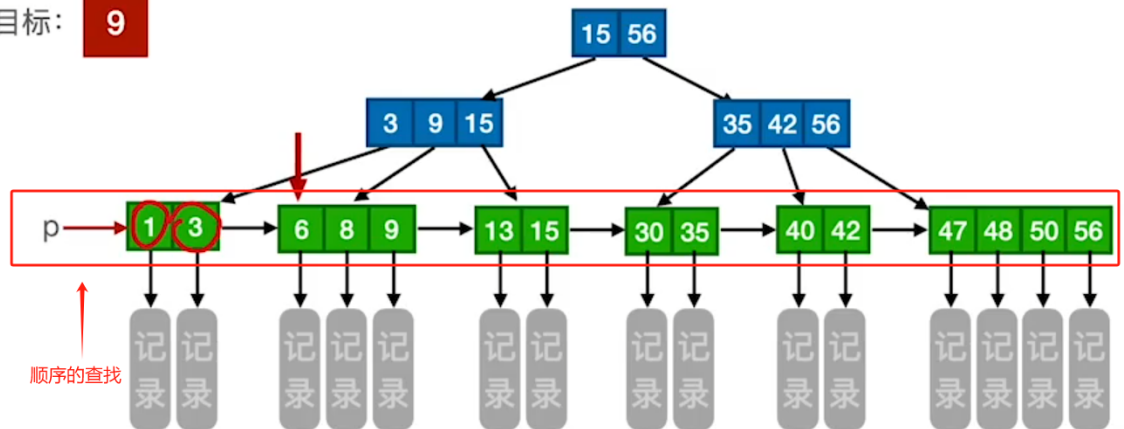
- 性质：
 1. 每个分支结点最多有 m 棵子树（孩子结点）；
 2. 非叶根结点至少有两棵子树，其他每个分支结点至少有 $\lceil m/2 \rceil$ 棵子树；
 3. 结点的子树个数与关键字个数相等；
 - 这一点和 B 树不同；
 4. 所有叶结点包含全部关键字及指向相应记录的指针，叶结点中将关键字按大顺序排列，并且相邻叶结点按大小顺序相互链接起来；
 - 所以 B+ 中的结点支持顺序查找；
 5. 所有分支结点中仅包含它的各个子结点中关键字的最大值及指向其子结点的指针；
- 图示：4 阶 B+ 树



- 图示：B+ 树的顺序查找

查找目标:

9



27.3.2 B+ 树对比 B 树

性质对比

- 对比 1:
 - B+树:
 - 结点中的 n 个关键字对应 n 棵子树;
 - B 树:
 - 结点中的 n 个关键字对应 $n+1$ 棵子树;
- 对比 2:
 - B+ 树:
 - 根节点的关键字数 n 属于 $[1, m]$;
 - 其他结点的关键字数 n 属于 $[\lceil m/2 \rceil, m]$;
 - B 树:
 - 根节点的关键字数 n 属于 $[1, m-1]$;
 - 其他结点的关键字数 n 属于 $[\lceil m/2 \rceil - 1, m-1]$;
- 对比 3:
 - B+ 树:
 - 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:
 - 在 B 树中, 各结点中包含的关键字是不重复的;
- 对比 4:
 - B+ 树:
 - 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:
 - B 树的结点中都包含了关键字对应的记录的存储地址;
- 对比 5:
 - 在内存中的访问: 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：

总结

	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”）	

获取笔记持续更新可加微信：Wan