

CST2018 2-2 Circuit

徐良钦

问题描述

- ❖ 给定 n 个64位的二进制数 $\{a_i\}$ ，对于第 i 个位置，求出 $j \in [i-k-1, i+k+1]$ ， $j \neq i$ 且 a_j 与 a_i 异或值最大。如有多个 j 满足，取最小的 j 。
- ❖ 异或：二进制异或，相同为0，不同为1
- ❖ $2 \leq n \leq 500000$
- ❖ $0 \leq k \leq 500000$
- ❖ $2s, 256M$

问题描述

- ❖ 给定 n 个64位的二进制数 $\{a_i\}$ ，对于第 i 个位置，求出 $j \in [i-k-1, i+k+1]$ ， $j \neq i$ 且 a_j 与 a_i 异或值最大。如有多个 j 满足，取最小的 j 。
- ❖ 比如 $n=5$ ， $k=1$ ， $\{a_i\}=\{001, 010, 010, 100, 101\}$
- ❖ $i=1$ ， $j \in [1, 3] \neq 1$ ，当 $j=2$ 时有最大值 $a_1 \oplus a_2 = 011$
- ❖ $i=2$ ， $j \in [1, 4] \neq 2$ ，当 $j=4$ 时有最大值 $a_2 \oplus a_4 = 110$
- ❖ $i=4$ ， $j \in [2, 5] \neq 4$ ，当 $j=2$ 时有最大值 $a_4 \oplus a_2 = 110$

问题分析

❖要是我们有一个这样的数据结构就好了！

- 插入一个数字
- 删除一个数字
- 查询与给定的数字异或值最大的数

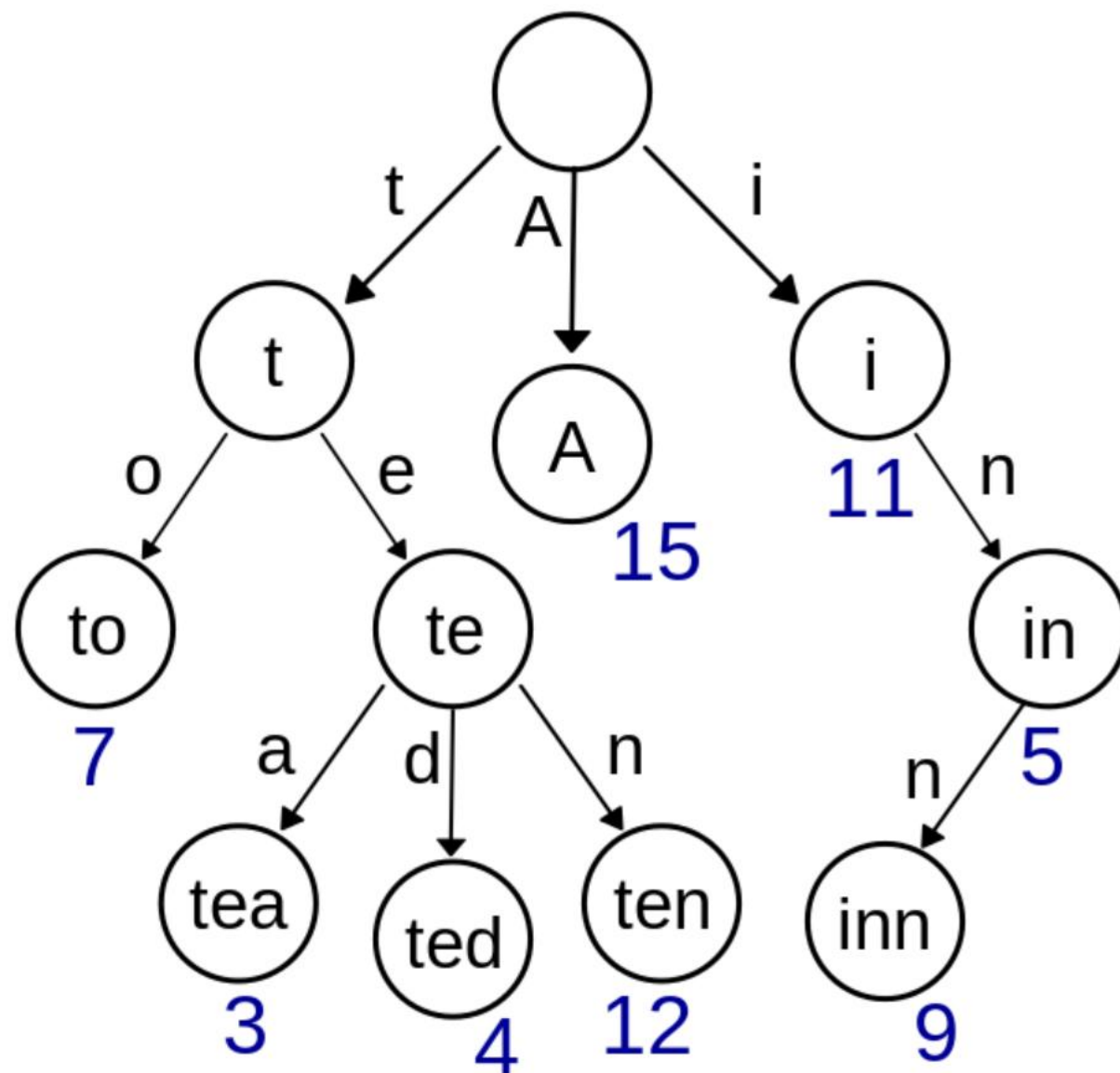
❖Trie树！

Trie树

- ❖ 一棵多叉树
- ❖ 每条边是一个字符
- ❖ 从根走到某个节点路径上的字符连起来就是一个串
- ❖ 标记某些节点是终止节点

Trie树

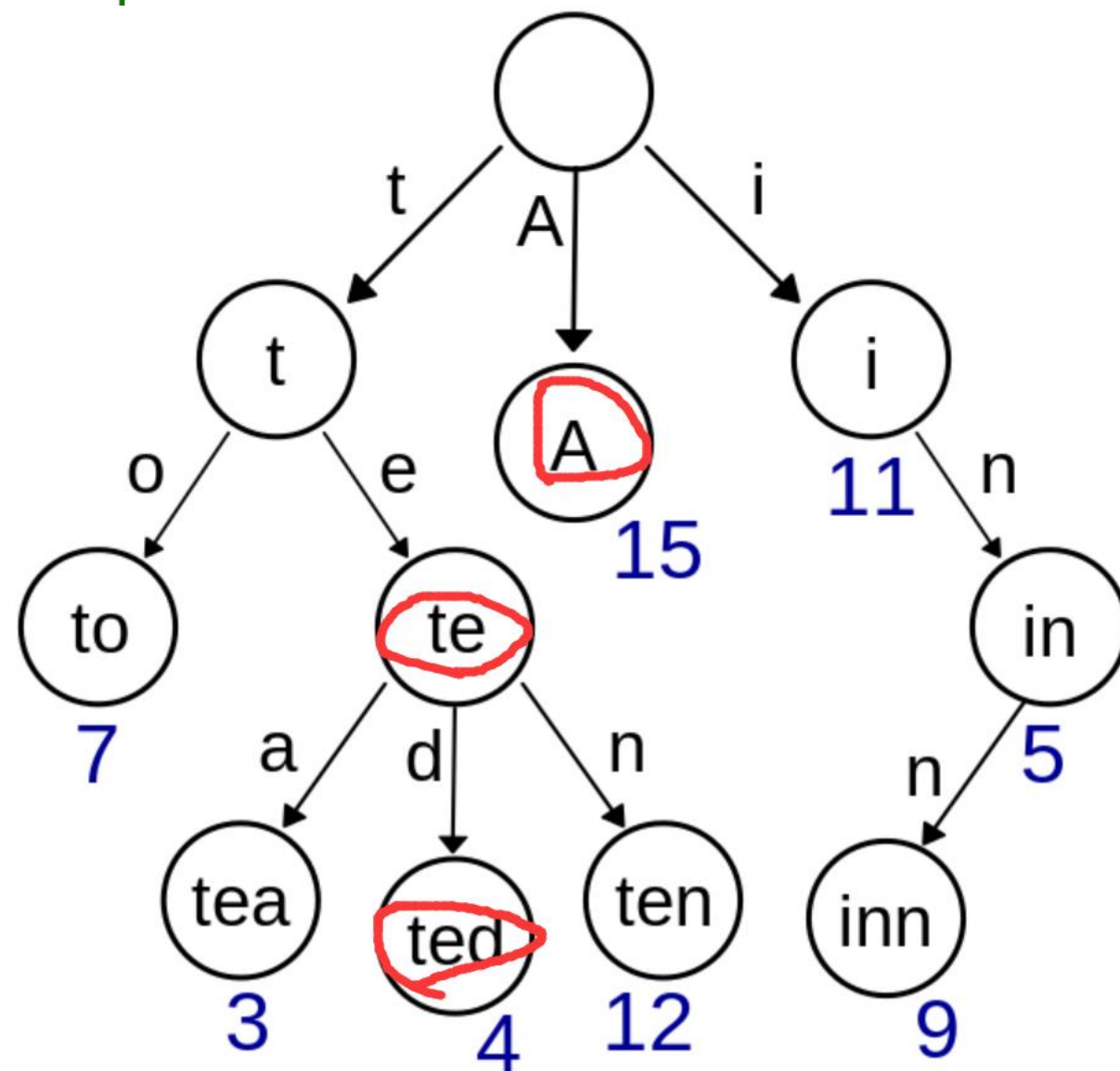
- ❖ 一棵多叉树
- ❖ 每条边是一个字符
- ❖ 从根走到某个节点路径上的字符
- ❖ 标记某些节点是终止节点



Trie树

❖给定的字典是{A, te, ted}

❖则这仨是终止节点



Trie树

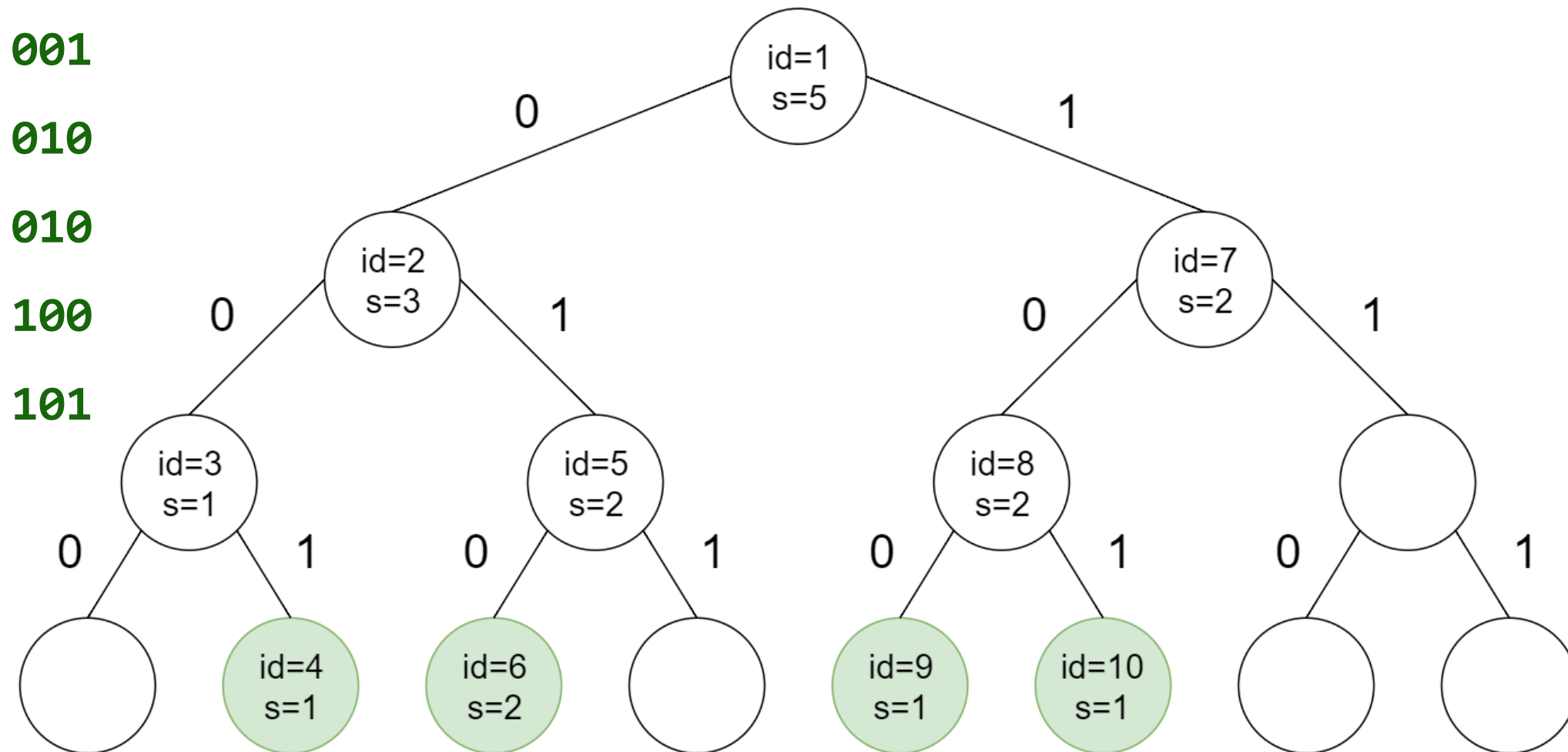
- ❖ 二进制数字的Trie树呢？
- ❖ 将二进制这个字符串直接插入到Trie树里就行了！
- ❖ 但要注意，所有数字的长度必须相等（用前导0补齐）
- ❖ 比如64位二进制数的5就是

000...000101

一共64位

Trie树

❖ 每个节点记录一个值 s ，表示该节点以及子树终止节点的个数

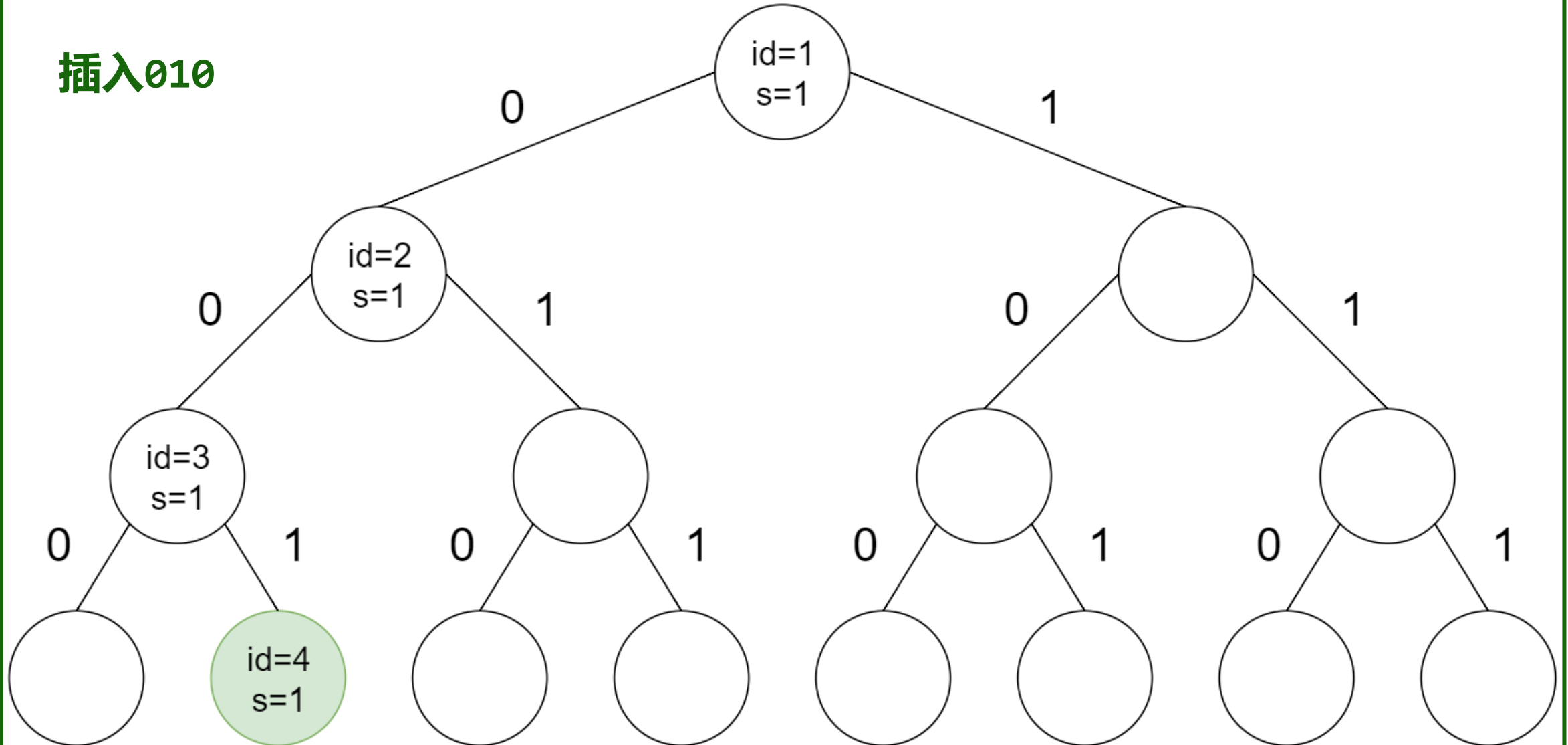


Trie树

❖ 插入操作：从根走到新建的终止节点，并且路径上所有点s加1

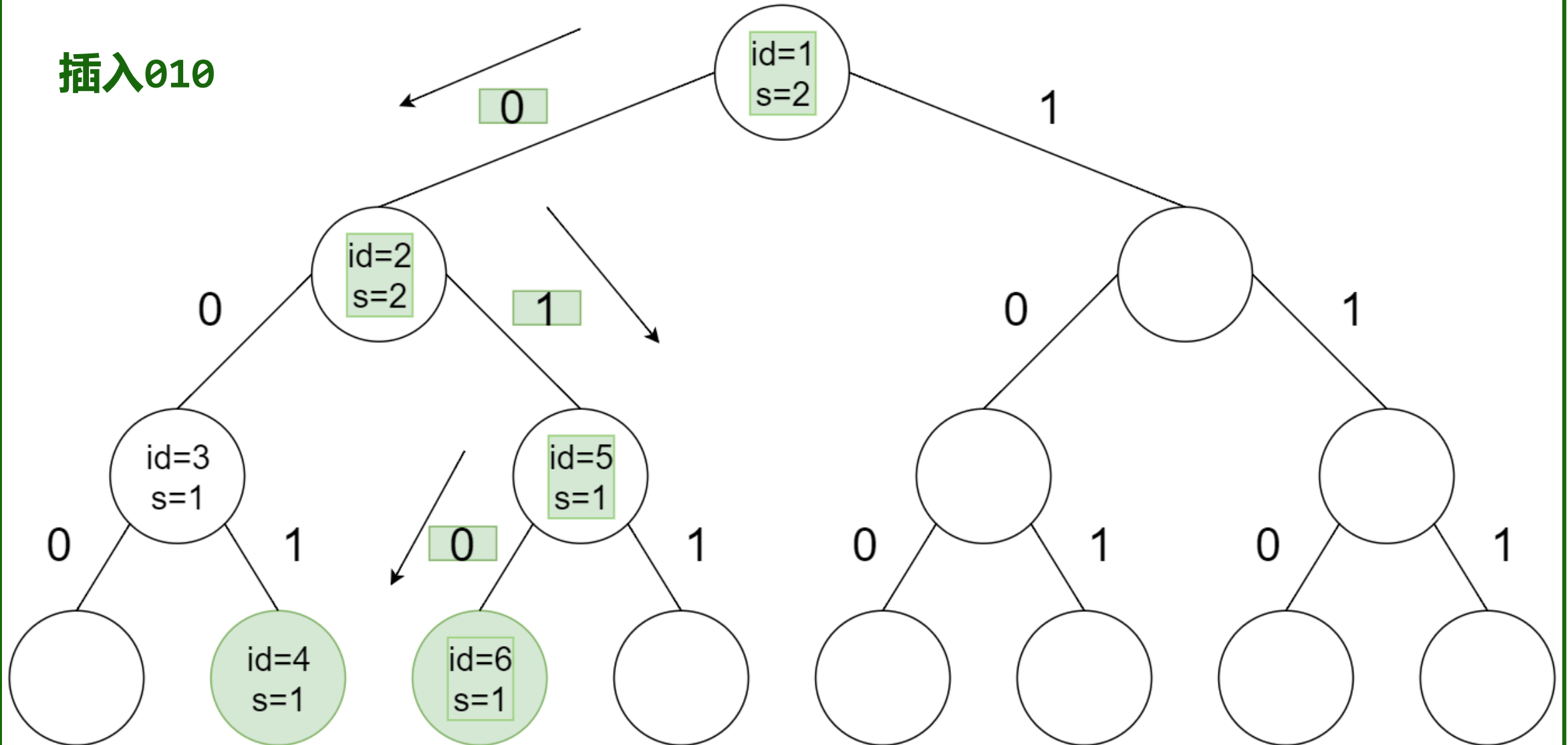
Trie树

插入010



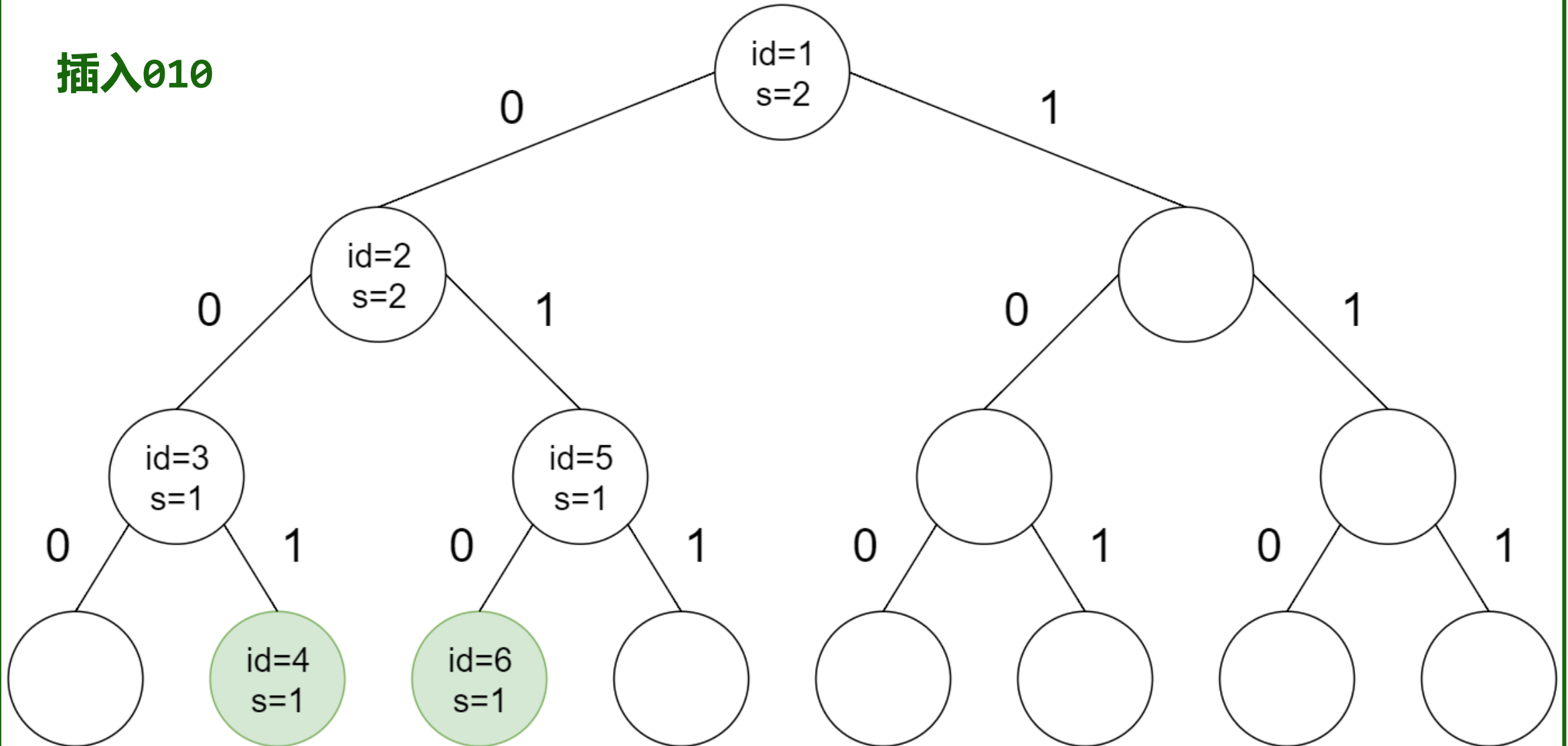
Trie树

插入010



Trie树

插入010

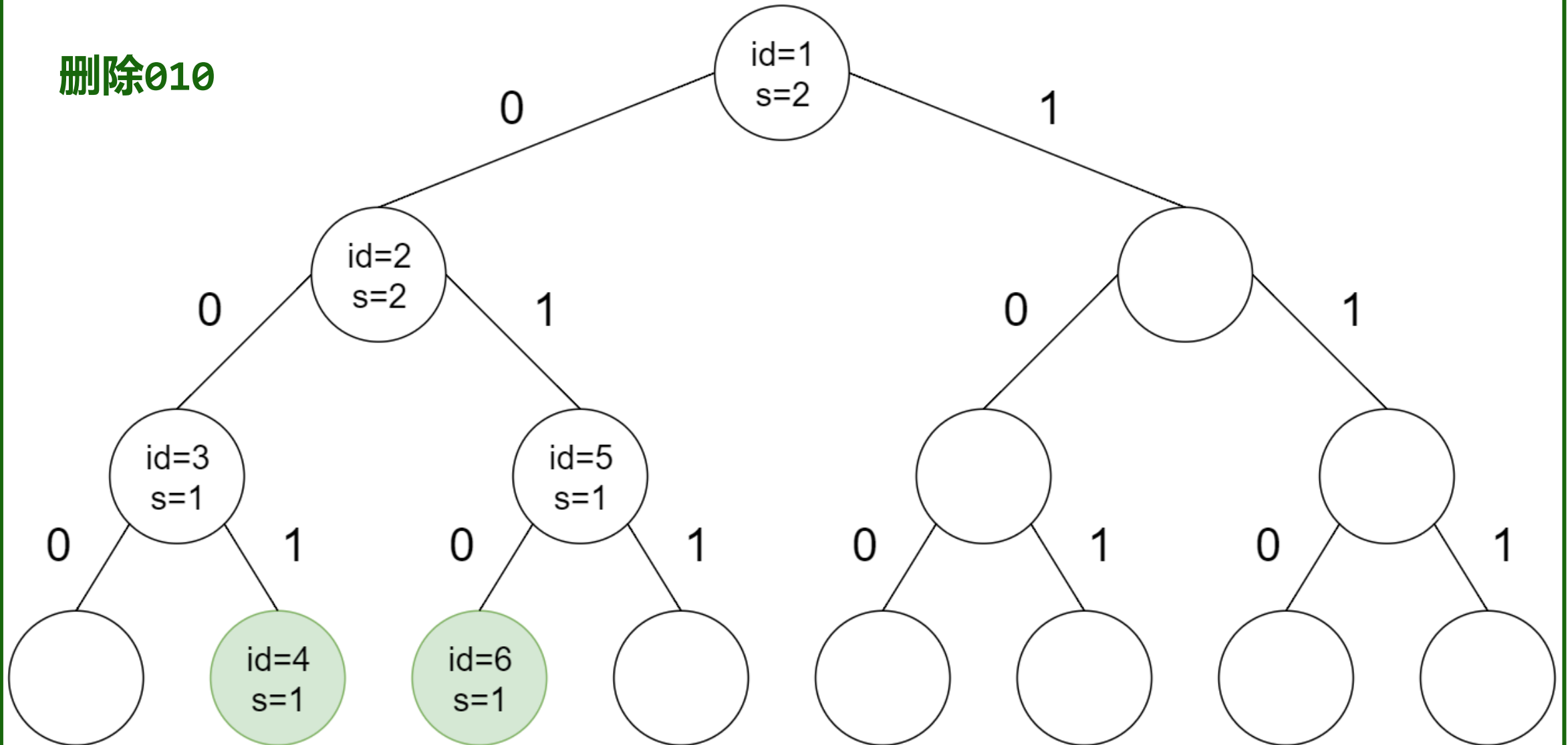


Trie树

- ❖ 删除操作：从根走到新建的终止节点，并且路径上所有点s减1
- ❖ 和插入操作基本一样

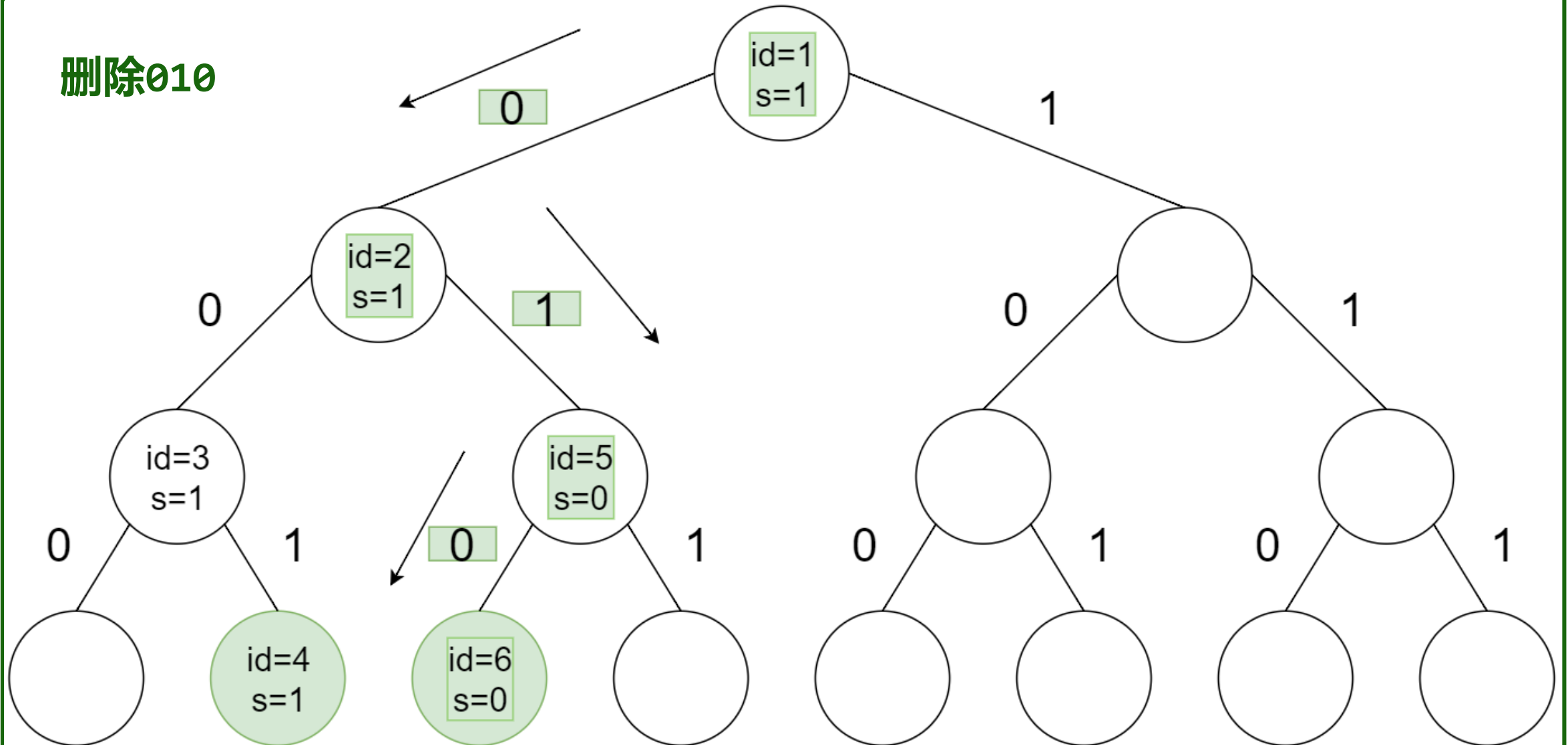
Trie树

删除010



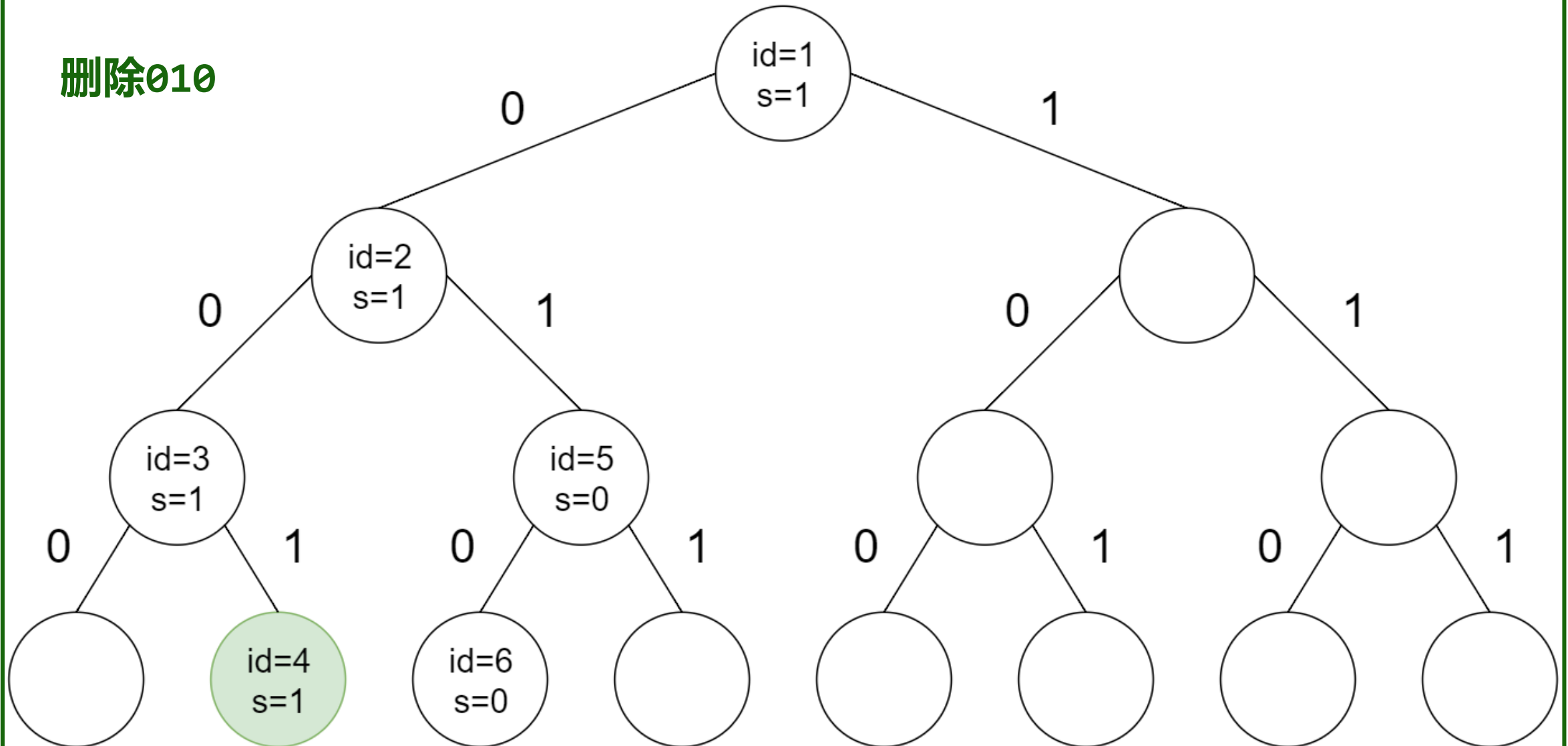
Trie树

删除010



Trie树

删除010

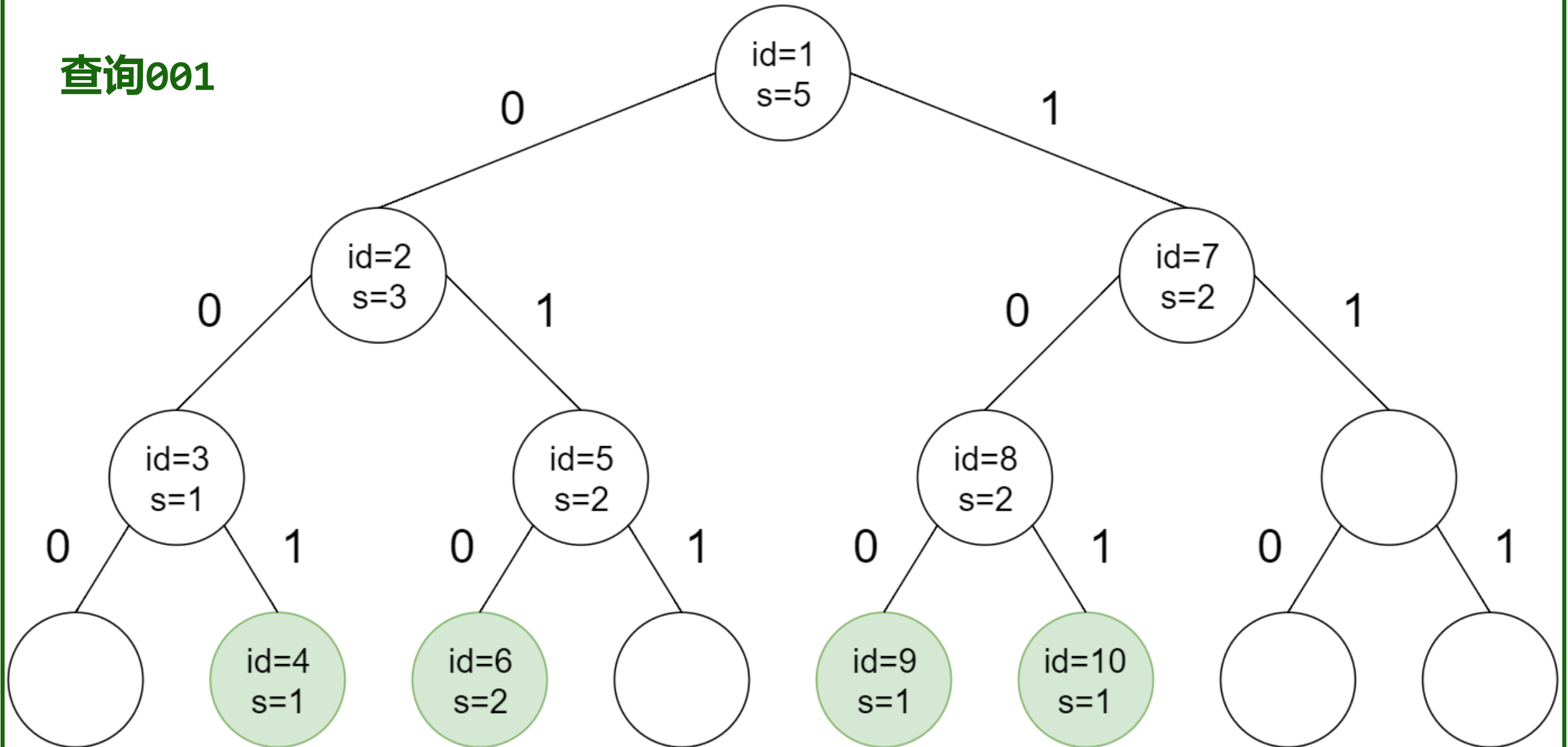


Trie树

- ❖ 查询操作：在每个节点判断是往左走还是往右走，当然是往异或最大的那个方向走
- ❖ 比如在某一层，当前查询的这个数字在这一位是1，那么如果所在节点左边（0）存在终止节点，自然得往左走（异或起来是1）。否则往右走。

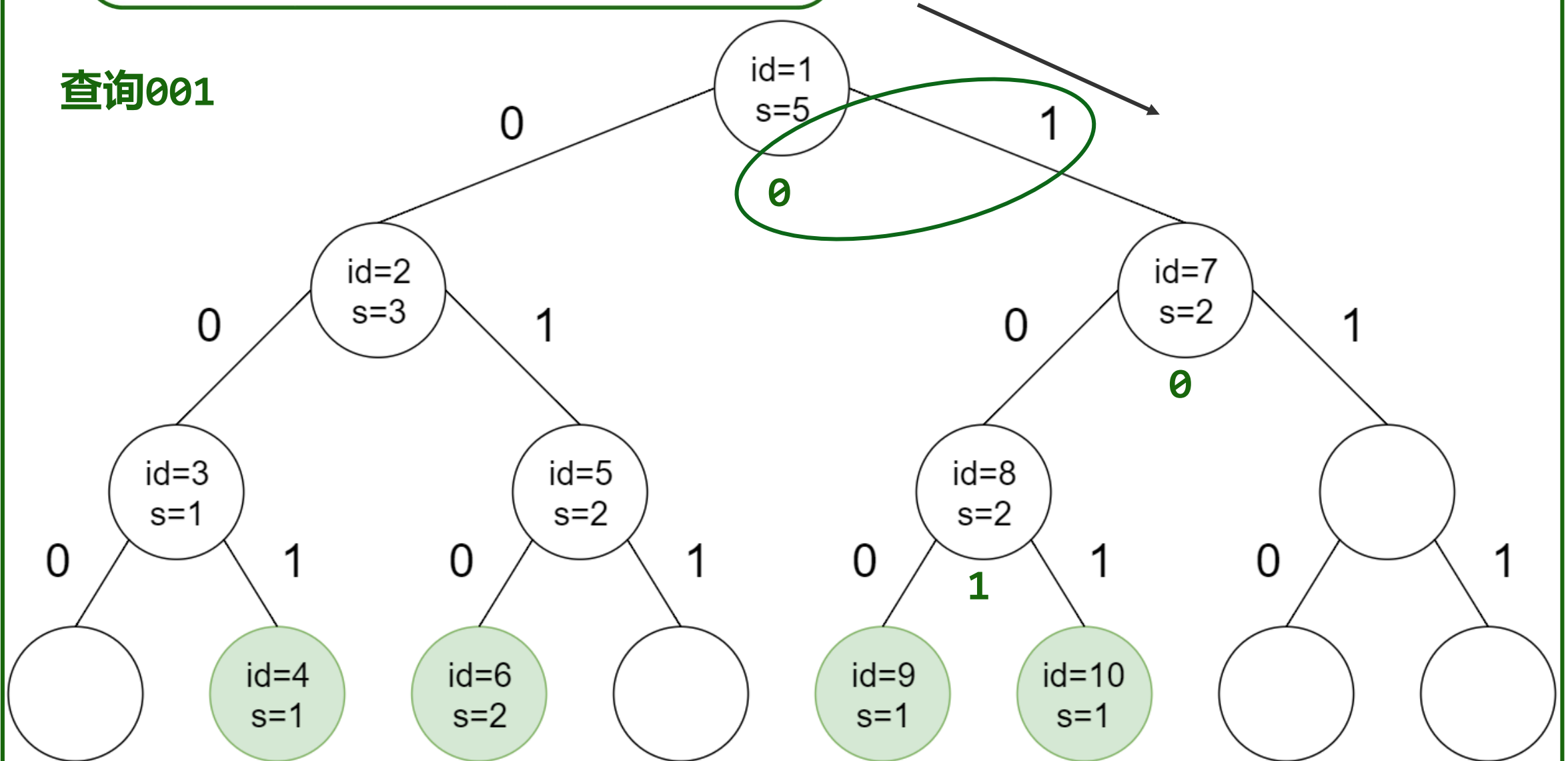
Trie树

查询001



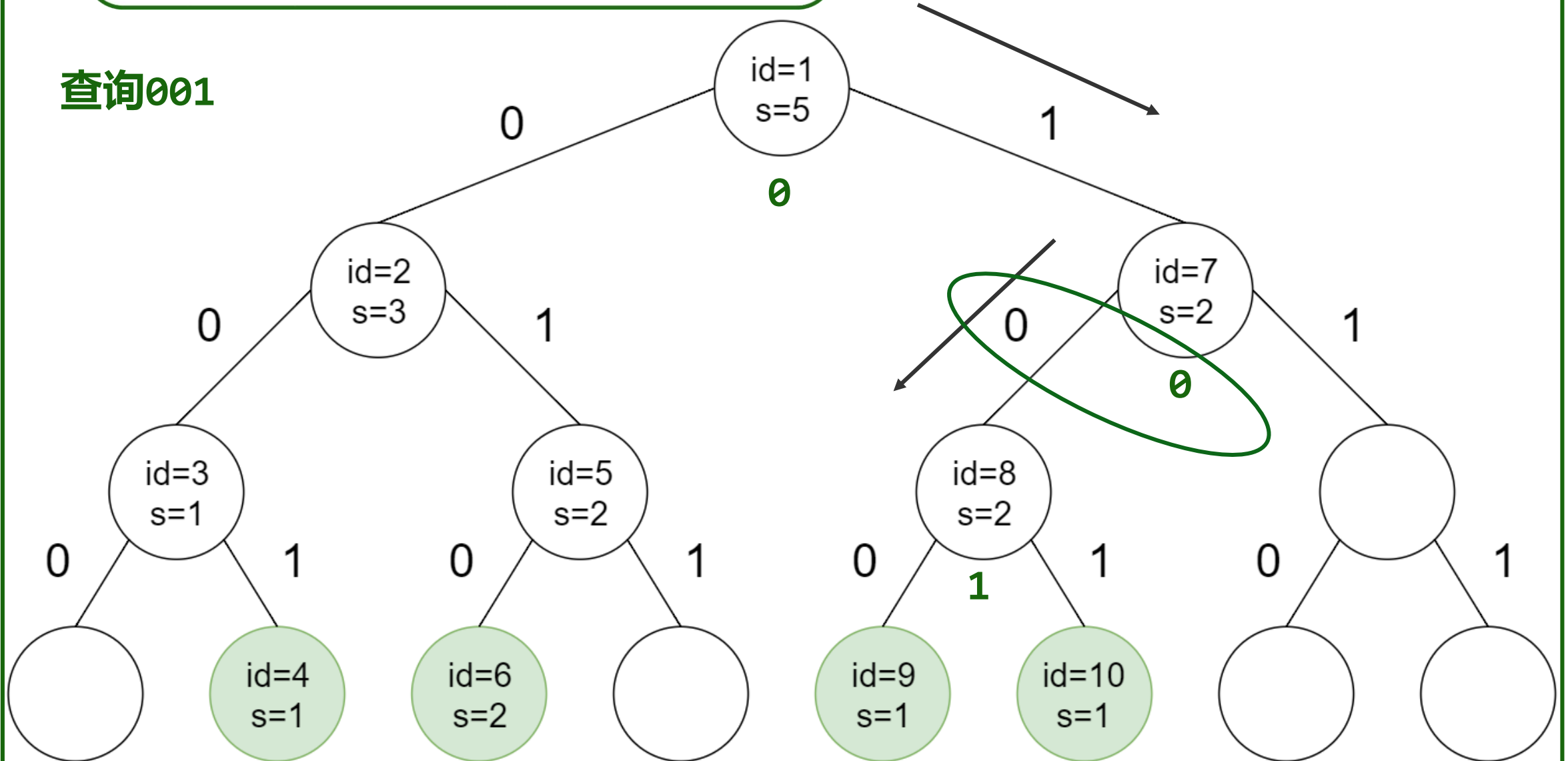
Trie树

查询001



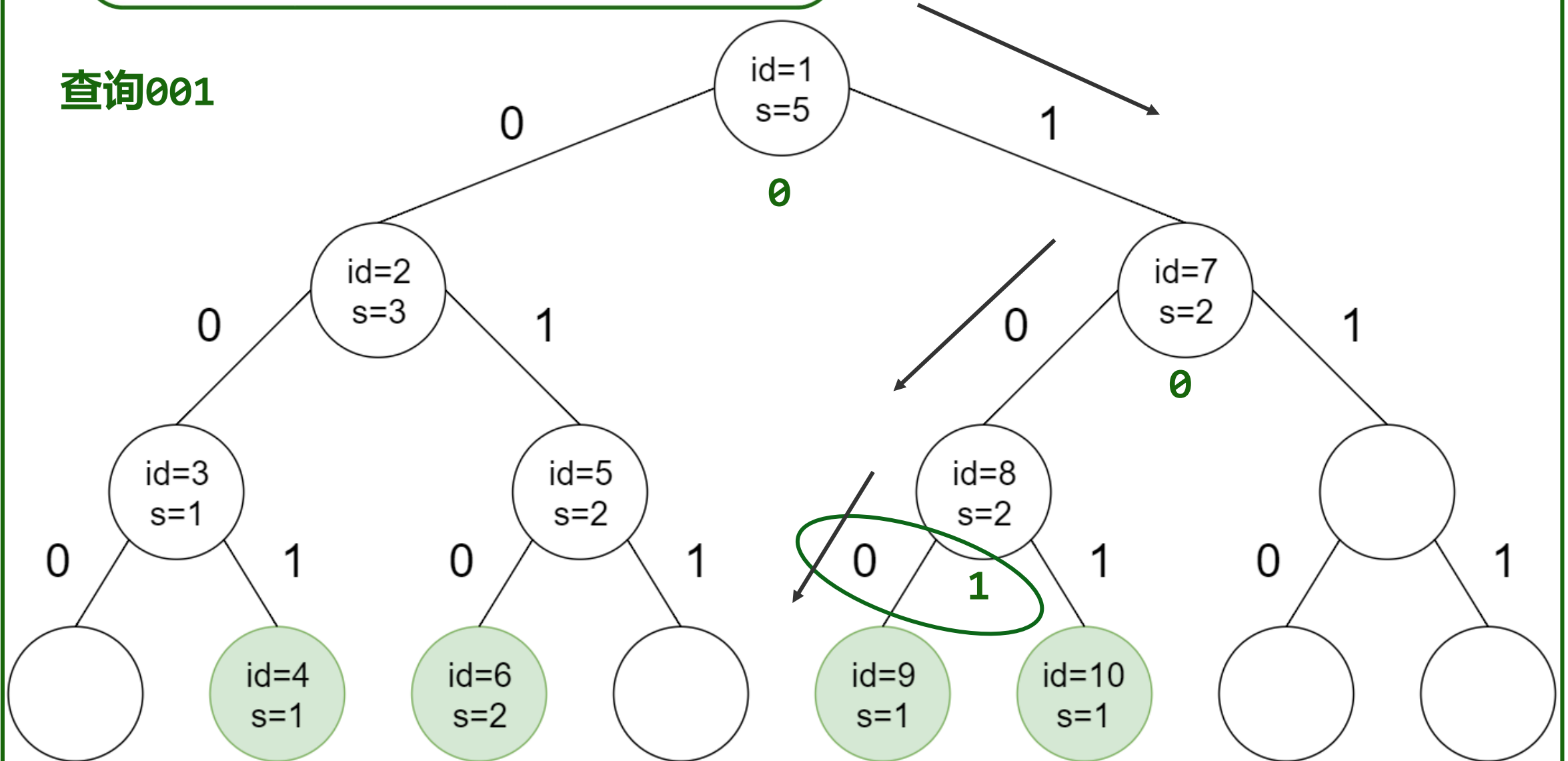
Trie树

查询001



Trie树

查询001



Trie树

- ❖ 每个节点记录一个值 s ，表示该节点以及子树终止节点的个数
- ❖ 插入操作：从根走到新建的终止节点，并且路径上所有点 s 加1
- ❖ 删除操作：从根走到新建的终止节点，并且路径上所有点 s 减1
- ❖ 查询操作：在每个节点判断是往左走还是往右走

求每一个位置的答案

❖ 注意到 k 是固定的，也就是第 $i+1$ 个位置的查询区间

$$[(i+1)-k-1, (i+1)+k+1]$$

是第 i 个位置的查询区间

$$[i-k-1, i+k+1]$$

向右移动了1个位置

❖ 也就是说，在 i 递增枚举的时候，删除前一个查询区间的左端点，添加右端点的后一个位置

求每一个位置的答案

- ❖ {001, 010, 010, 100, 101}
- ❖ {001, 010, 010, 100, 101}
- ❖ {001, 010, 010, 100, 101}
- ❖ {001, 010, 010, 100, 101}
- ❖ {001, 010, 010, 100, 101}

复杂度分析

❖ 时间复杂度

- Trie的插入、删除、查询操作均为 $O(\log a_i)$ ，在这题里是 $O(64)=O(1)$
- 总共会用到 $O(n)$ 次插入、删除、查询，因此总复杂度是 $O(n)$

❖ 空间复杂度

- 插入一个数字会用到64个节点，每个节点大小为 k ，故Trie树总共为 $O(64kn)=O(n)$
- 其余的空间？存储所有数字、临时变量总共 $O(n)$

❖ 其实...这题更重要的是 $O(n)$ 里的常数...不能忽略掉

难点要点

❖ 其实...这道题，是细节题！

❖ 1. Trie树里终止节点的 $s > 1$ 怎么办？也就是有多个位置

❖ 2. 如何从查询区间 $[i-k-1, i+k+1]$ 里去掉 i 本身？

❖ 3. 内存只有256M！ $O(n)$ 常数决定一切！

链表

❖ 终止节点里 $s > 1$ 的情况

❖ 也就是我们如何找到“位置最前”的那一个呢？

❖ 注意到我们在整个算法里，插入和删除的位置都是递增的，因此在终止节点里写一个队列即可

➤ 插入是插入到队尾

➤ 删除是删除掉队头

➤ 查询答案就是队头

❖ 由于内存少，所以用链表实现队列

{001, 010, 010, 100, 101}

{001, 010, 010, 100, 101}

{001, 010, 010, 100, 101}

{001, 010, 010, 100, 101}

{001, 010, 010, 100, 101}

去掉i本身

- ❖ 在查询的时候，如果到达的叶子大小大于1，找到和查询位置不同的位置就行了（不是队头就是队头后一个）
- ❖ 不用考虑叶子大小等于1且恰好就是查询位置这种情况

内存危机

- ❖ 256M内存！
- ❖ 最坏情况下，考虑在第 k 层就有 n 个Trie节点，那么 k 层(k 从1开始)之后就都是链(长度为 $64-k$)，则节点不超过 $n \times (64-k) + 2^k - 1$
- ❖ 当 $n=500000$ 时， $k=20$ ，最坏情况下节点数有2300W，假如一个Trie节点你用3个int存，爆炸。故你不能存之前说的s
- ❖ 你只需要维护左右孩子是否存在就行了，删除的时候相应的维护下
- ❖ (还有一个黑魔法是路径压缩，我就不介绍了)

小结

- ❖ 我猜时间肯定不够了，所以就假装这里有小结吧
- ❖ 所以这里放一点奇怪的东西吧，我猜也不会放这一页很久
- ❖ 还需要小结这种东西吗，听得懂的肯定都听懂了吧
- ❖ 听不懂的看小结也不会听懂的吧，还是下载ppt来看吧
- ❖ 万一讲的时候跳出来一个人说了一个吊打一切的解法呢
- ❖ 万一被按在地上锤了该怎么办，我好方啊 $o(\pi \sim \pi)$

谢谢！！！！

❖完

❖谢谢谢谢谢谢！！