

# 《数据结构》PA2辅导课

2018年10月24日

6C300

## 主要内容:PA2辅导

- ❖CST2018 2-1 Meteorites
- ❖CST2018 2-2 Circuit
- ❖CST2018 2-3-2 Mooney
- ❖CST2018 2-4 Sort
- ❖CST2018 2-5 ChromPoly
- ❖CST2018 2-6 Temperature

# **CST2018 2-1 Meteorites**

**计71 戴翼**

## 问题描述

- ◆ 给定  $n$  个数  $a_1 \dots a_n$  的集合， $2 \leq n \leq 2 \times 10^3$ ,  $1 \leq a_i \leq 1 \times 10^9$
- ◆ 每次取出集合中两个数，相乘得出一个新的数，重新加入集合
- ◆ 在每次相乘时，其积计作一次分数

要求：

- ◆ 一个能让每次分数之积最小的相乘（合并）顺序
- ◆ 让总 cost 最小的问题

## 问题分析

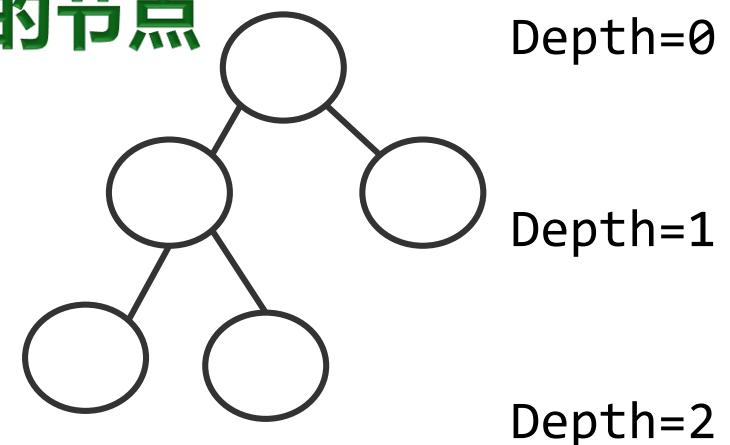
- ❖ 各元素地位相同，忽略位置因素
- ❖ 存不存在贪心算法？
- ❖ 每次把最小的两个合并？需要考虑什么？
- ❖ 如何证明解最小？

## 问题分析

- ❖ 求的解是什么？
- ❖ 分数的乘积就是各元素的乘积
- ❖  $\text{COST} = \prod_1^n (a_i)^k$
- ❖  $a_i$  的  $k$  取决于什么？

## 问题分析

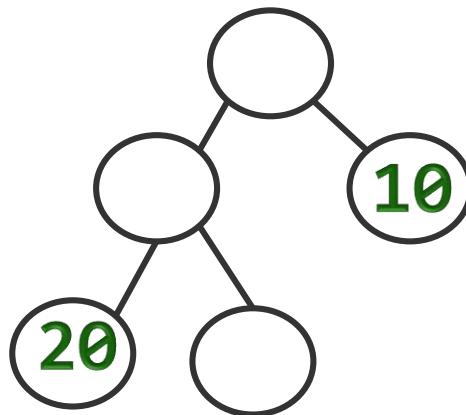
- ❖ K : 被合并的次数
- ❖ 联想到Huffman编码问题：所有叶节点权重\*深度之和最小
- ❖ 等价到二叉树上：  
 $a_i / \text{含 } a_i \text{ 因子的数被乘}$ , COST中 $a_i$ 的指数+1 $\leftrightarrow$   
二叉树上叶节点/叶节点的祖先归属于一个新的节点  
, 该节点深度+1  
乘法 $\leftrightarrow$ 二叉树节点的合并



## 问题分析

❖ 来用huffman的思路证明一下贪心算法

❖ (1) 给定了树的框架，如何往里填数？



原则1：小数深度更高

证明：

假如存在一对  $a_i < a_j$  而  $\text{depth}[i] < \text{depth}[j]$

交换则整体更优

## 问题分析

- ❖ (1) 给定了树的框架，如何往里填数？
- ❖ 原则2：拥有共同父节点的叶节点必然大小顺序相邻
- ❖ 归一化：同一深度的叶节点必然大小顺序连续，否则不符合原则1，那么作出规定，让大小相邻的节点填入同一节点的两个子节点中，不影响整体结果。

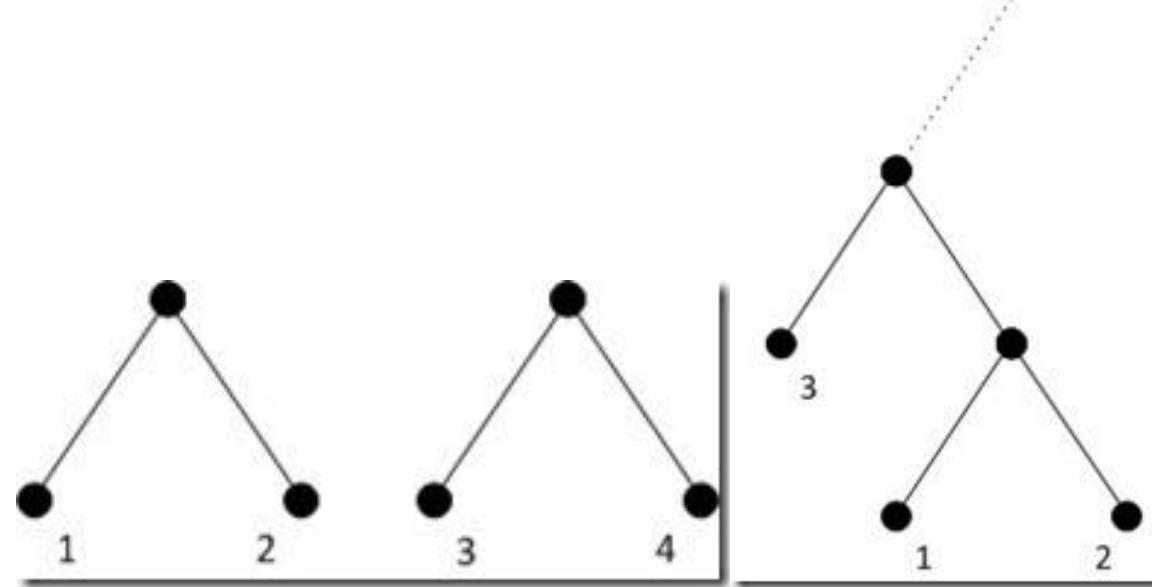
## 问题分析

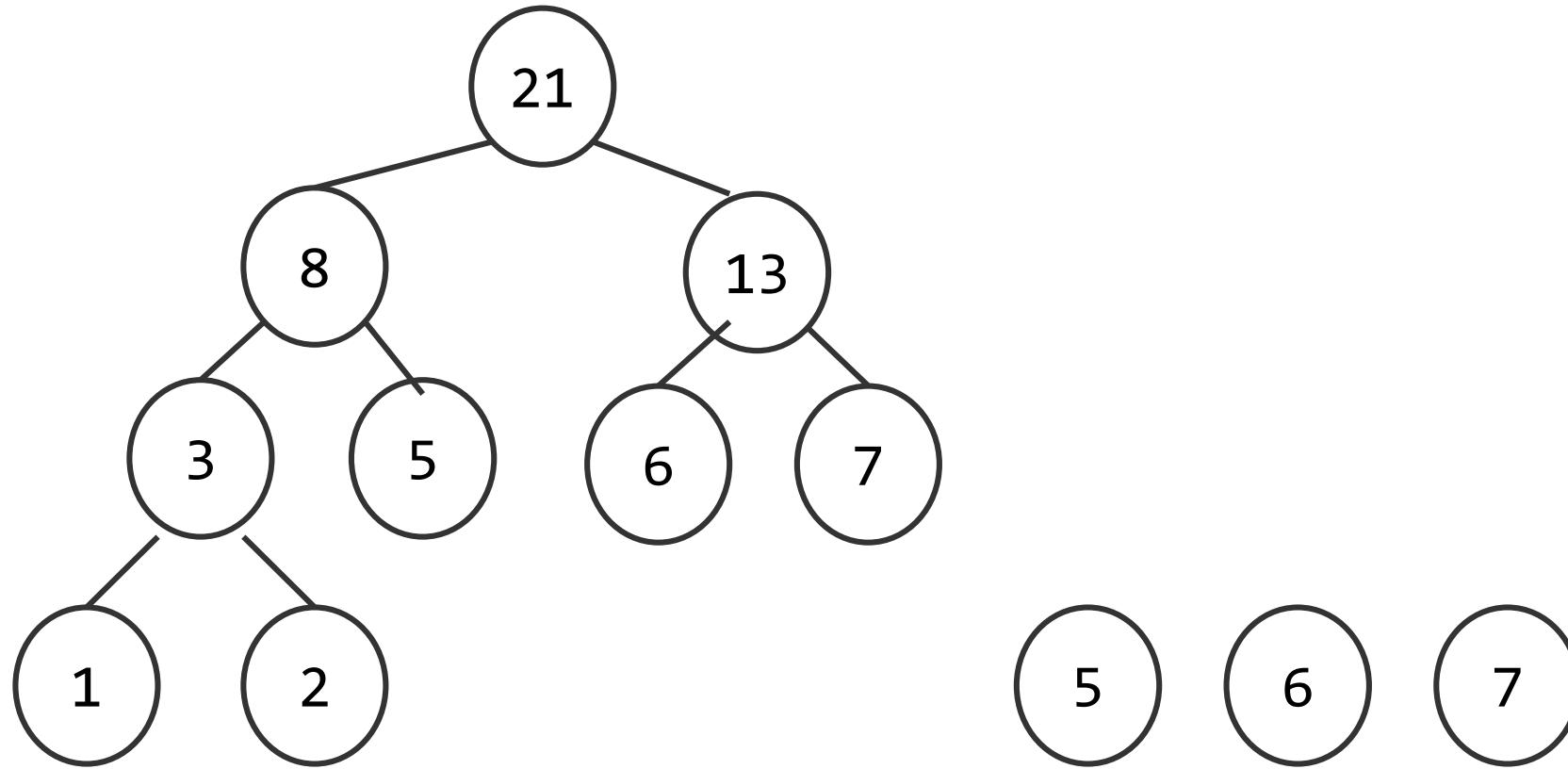
❖ ( 2 ) 如何形成树的框架 ?

❖ 原则 3 :

❖ 父节点等于子节点的权重之和 , 让权重小的节点在深度高的位置

归一法以及交换法证明





## 问题分析

- ❖ Huffman只是直观的形式
- ❖ 回到原题，如何在树上实现这样的算法？
- ❖ 每次获取**最小的2个**，相乘得到新的节点，继续合并
- ❖ 序列上的算法：取出然后插入？ $n^2$
- ❖ 堆
- ❖ ->堆的插入、删除更新

## 难点要点

- ❖ 分解成了两个子问题：乘法+堆维护
- ❖ 堆维护的上移和下移，比较大小？Huffman树的例子能有什么帮助？

## 难点要点

- ❖ PA1的乘法给了什么启发？（空间重复利用...）
- ❖ 和PA1比，多了什么？
- ❖ 总COST和单次相乘不一样！
- ❖ 做个估算，每次合并，COST最多是 $10^{(9*2000)}$ = $10^{18000}$   
而对COST的积： $10^{(10^7)}$ 级

## 难点要点

❖ 让数据对齐

❖ Long long 压位

❖ 中国剩余定理（预处理：找范围和模数，逆元 $t$ ）

$$(S) : \begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases} \quad x = \left( \sum_{i=1}^n a_i t_i M_i \right) \bmod M$$

## 小结

- ❖看上去PA2考察的更多是序列结构之外的知识，比如树，图
- ❖控制常数

# **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  $a_1 \wedge a_2 = 011$
- ◆  $i=2$ ,  $j \in [1, 4] \neq 2$ , 当  $j=4$  时有最大值  $a_2 \wedge a_4 = 110$
- ◆  $i=4$ ,  $j \in [2, 5] \neq 4$ , 当  $j=2$  时有最大值  $a_4 \wedge a_2 = 110$

## 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

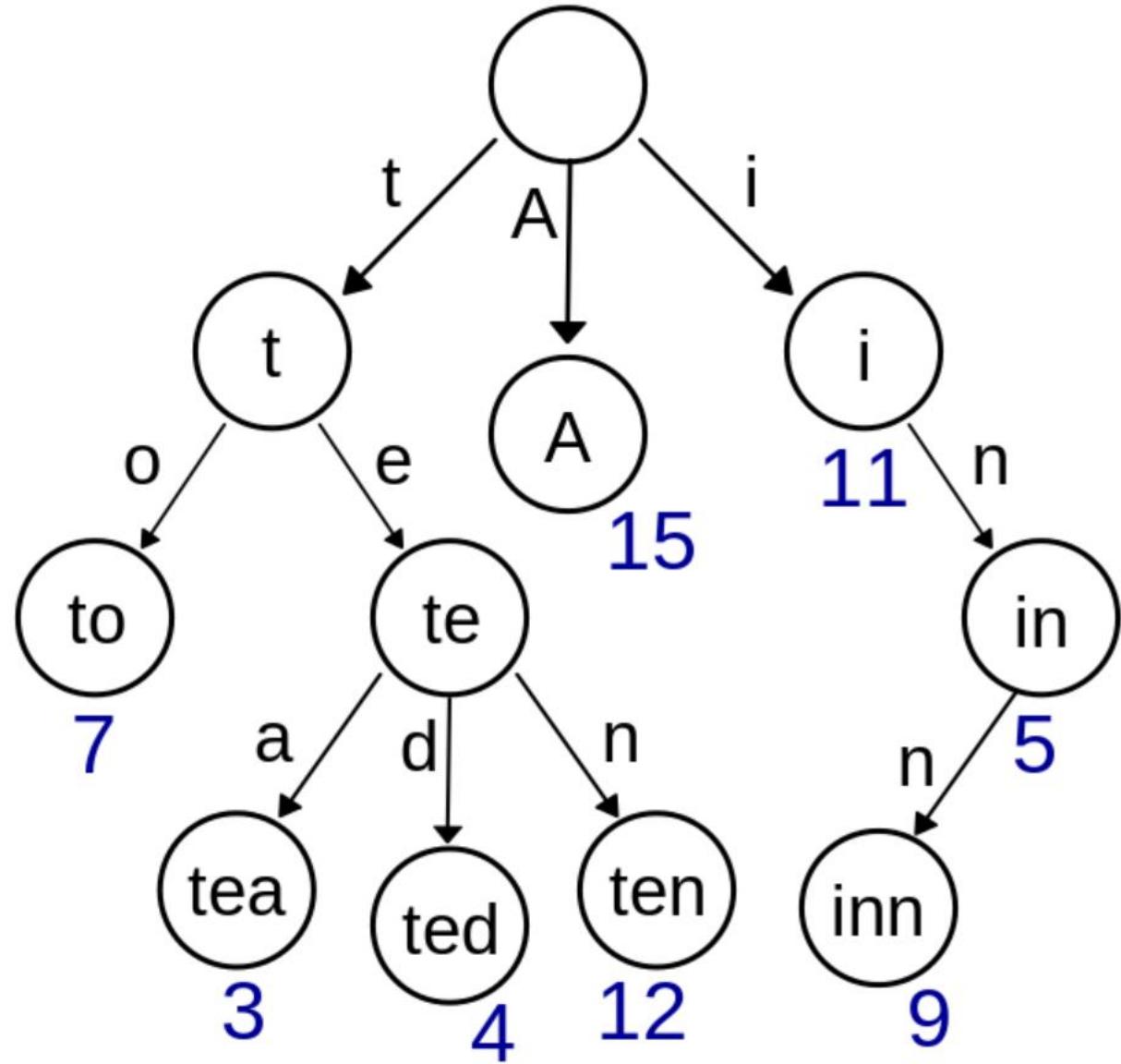
❖Trie树！

# 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

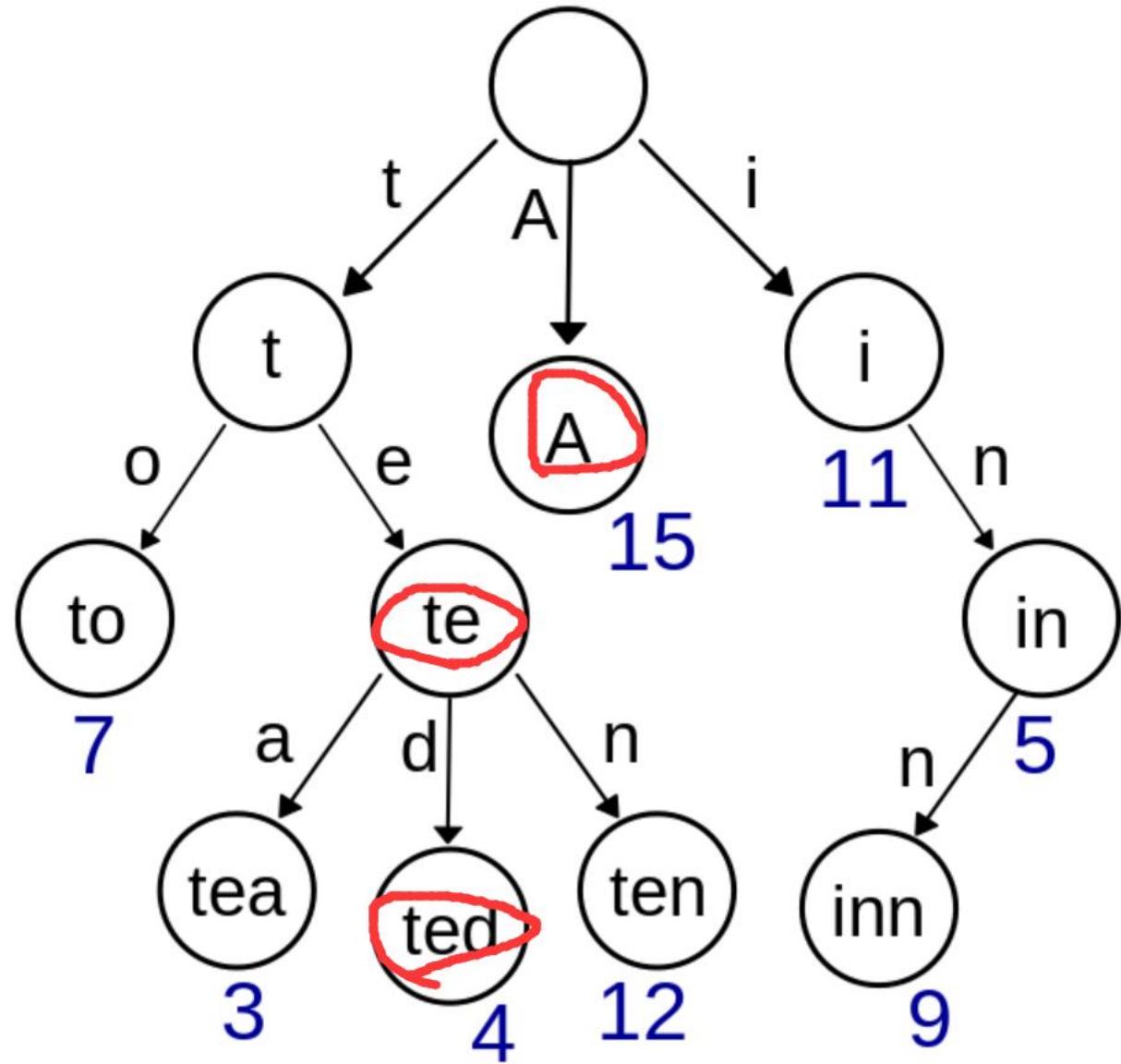
# Trie树

- ❖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



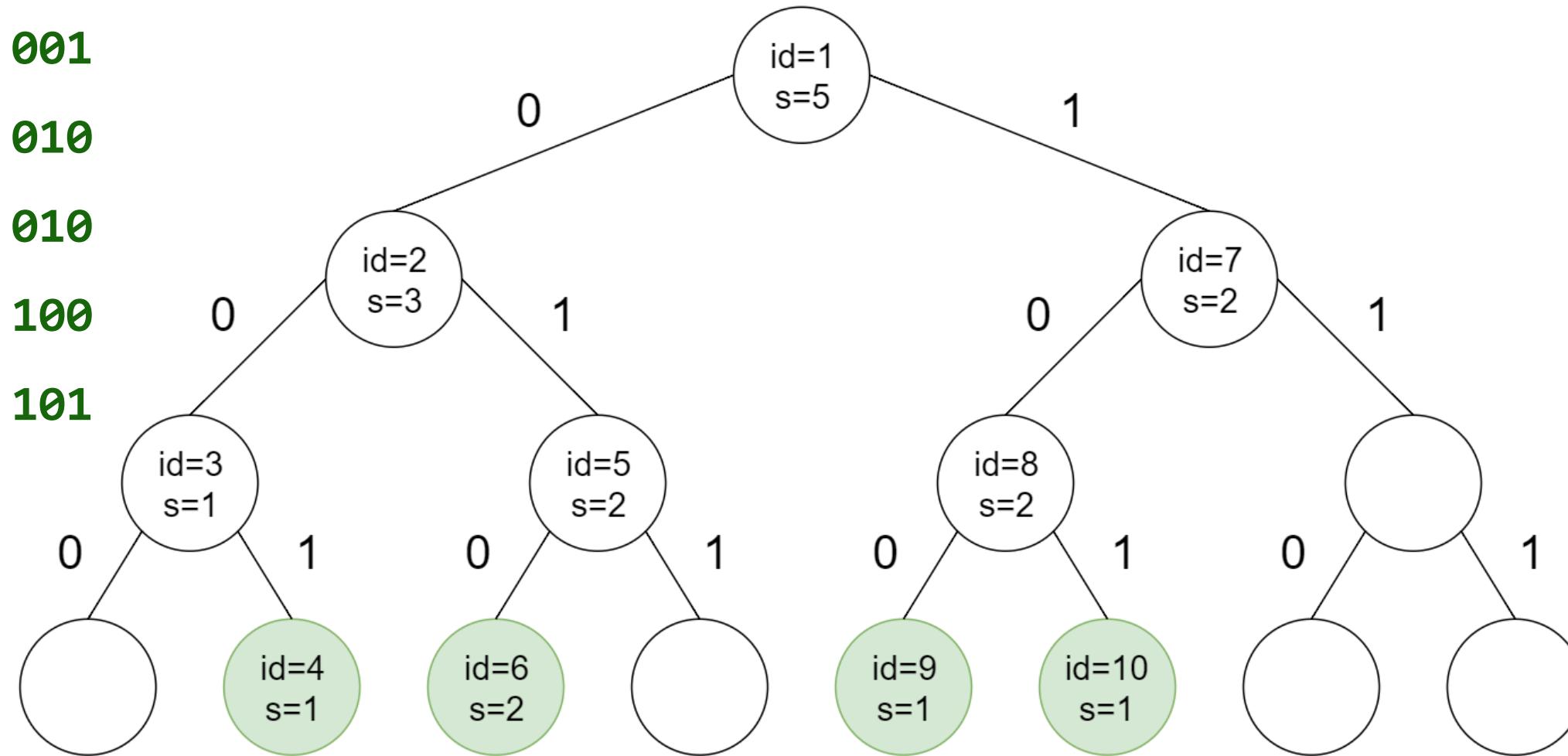
## Trie树

- ❖ 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

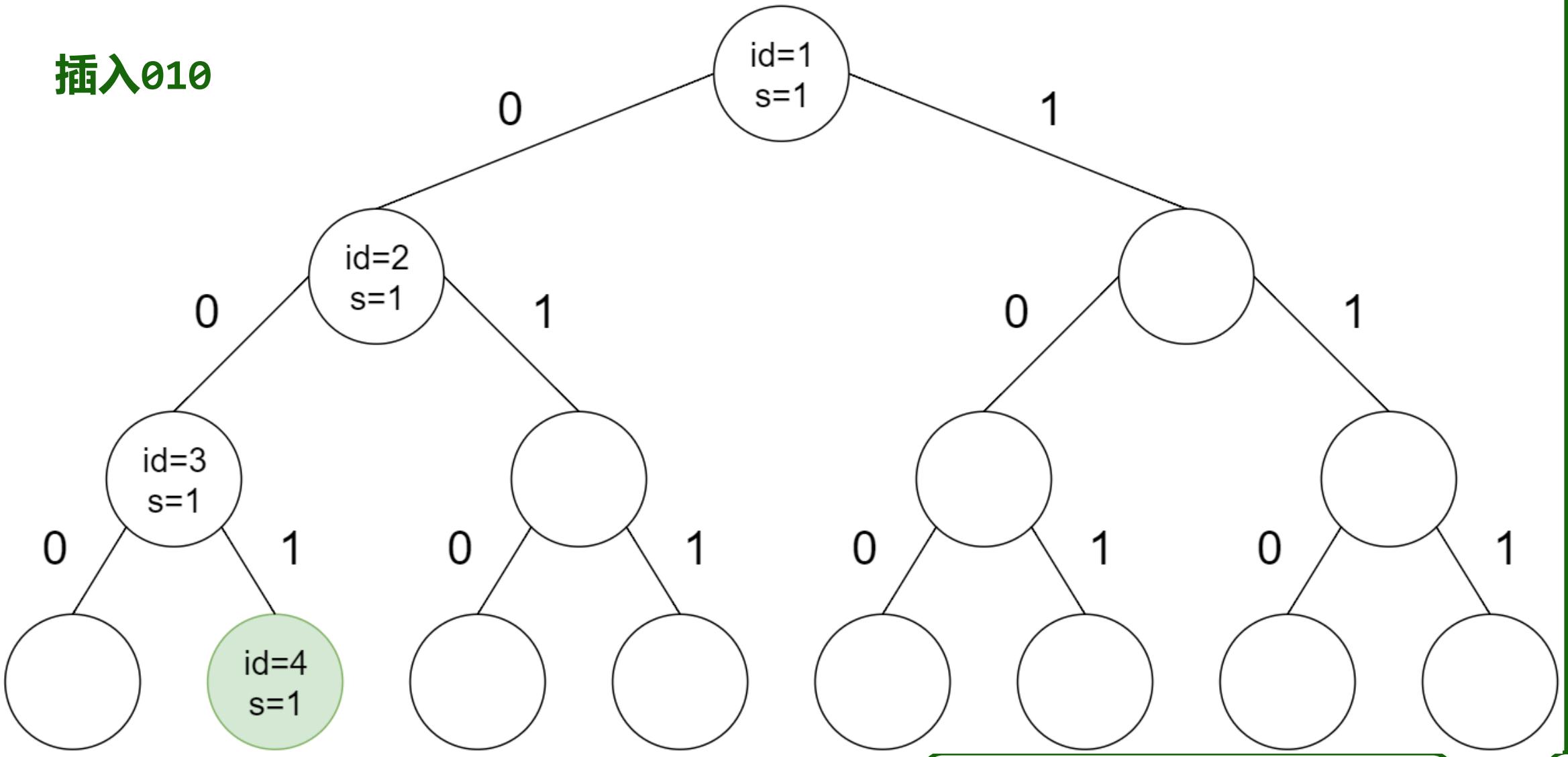


## Trie树

❖ 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

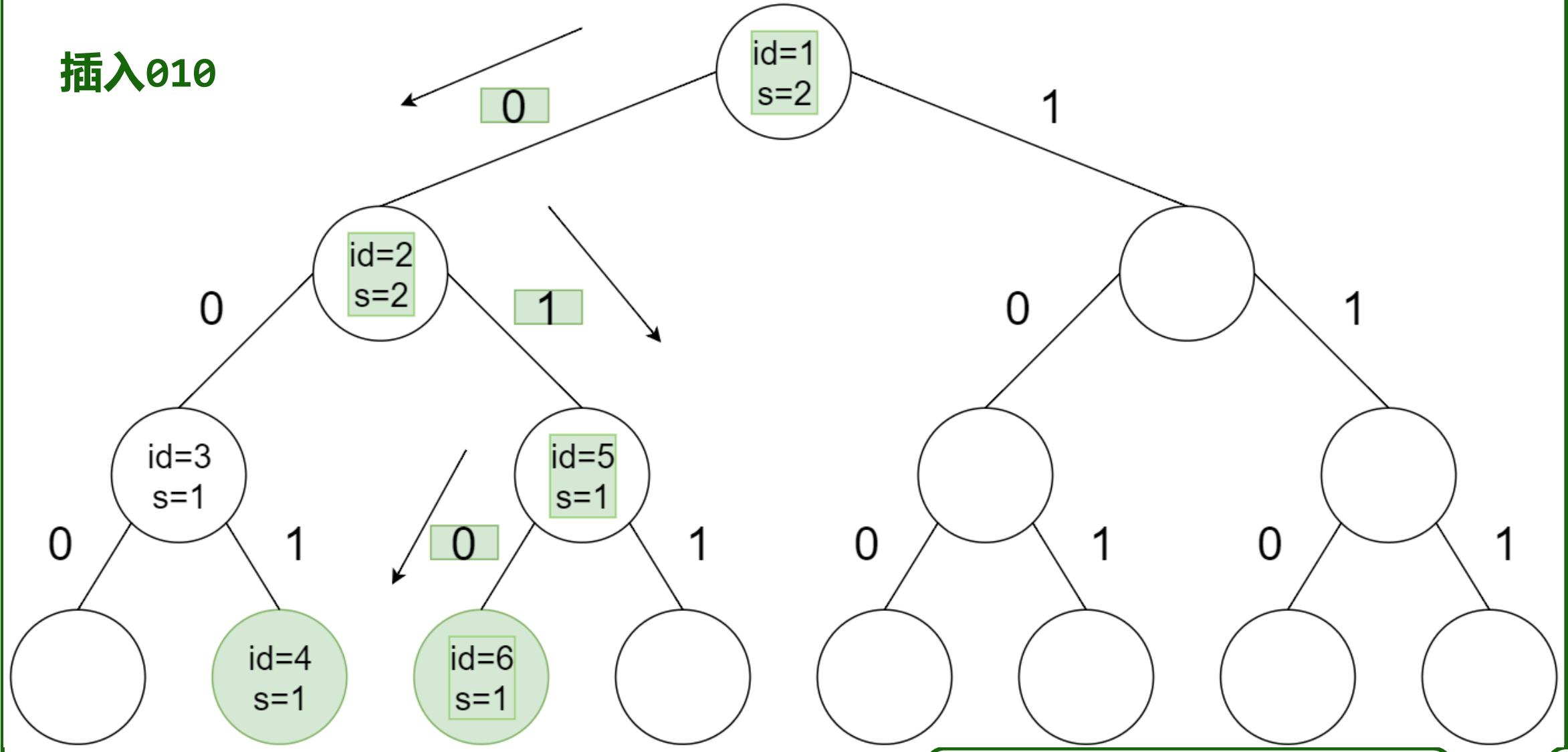
# Trie树

插入010



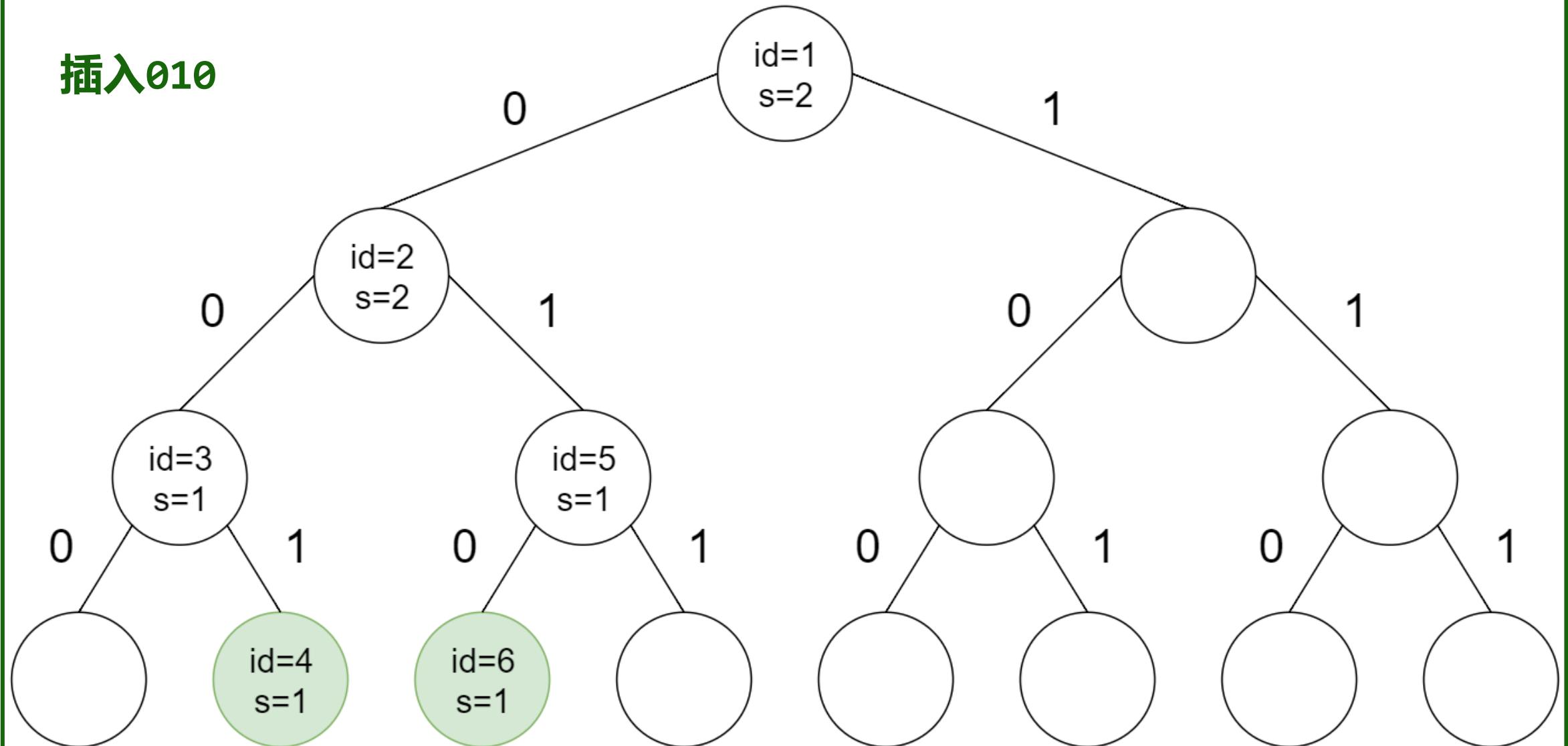
# Trie树

插入010



# Trie树

插入010

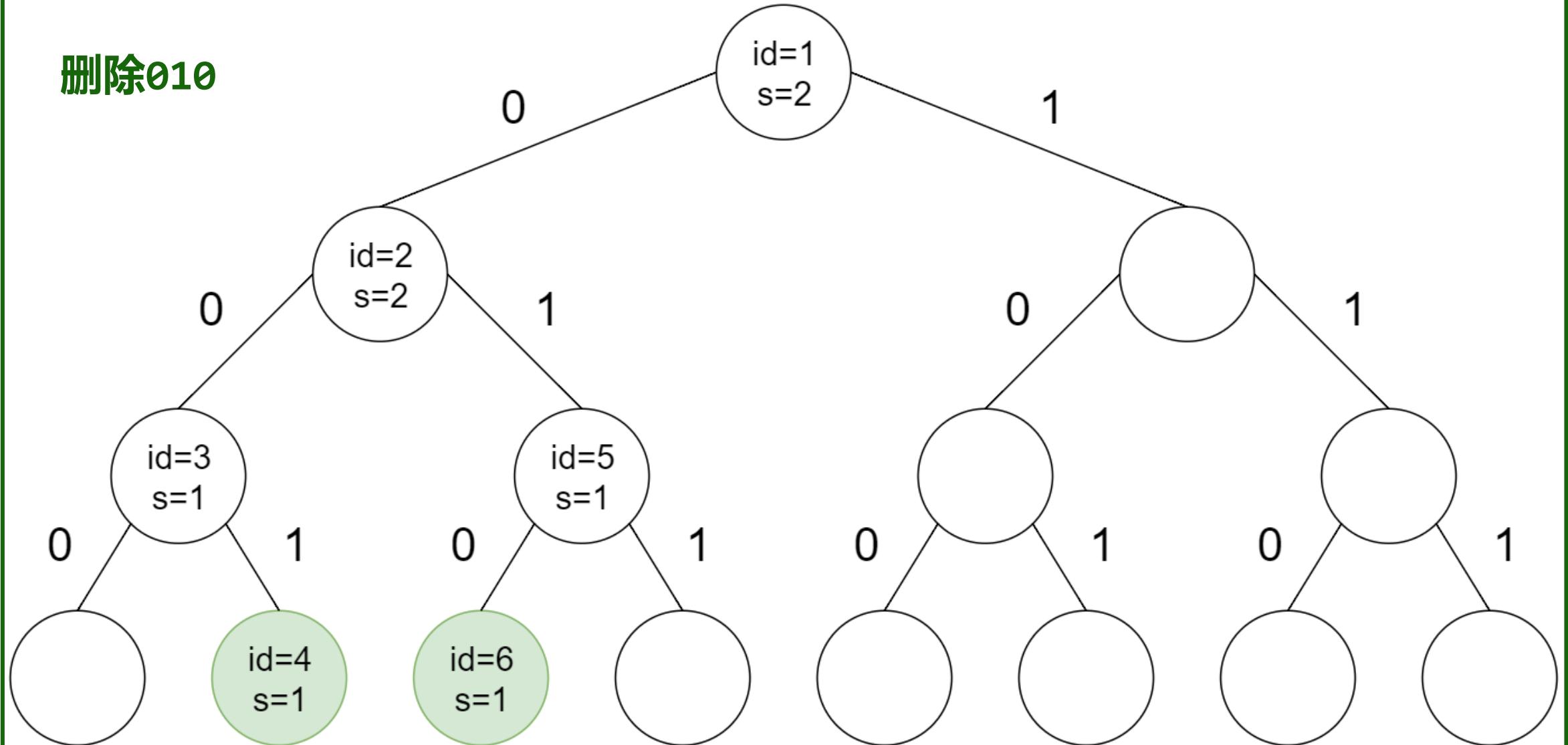


## Trie树

- ❖ 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

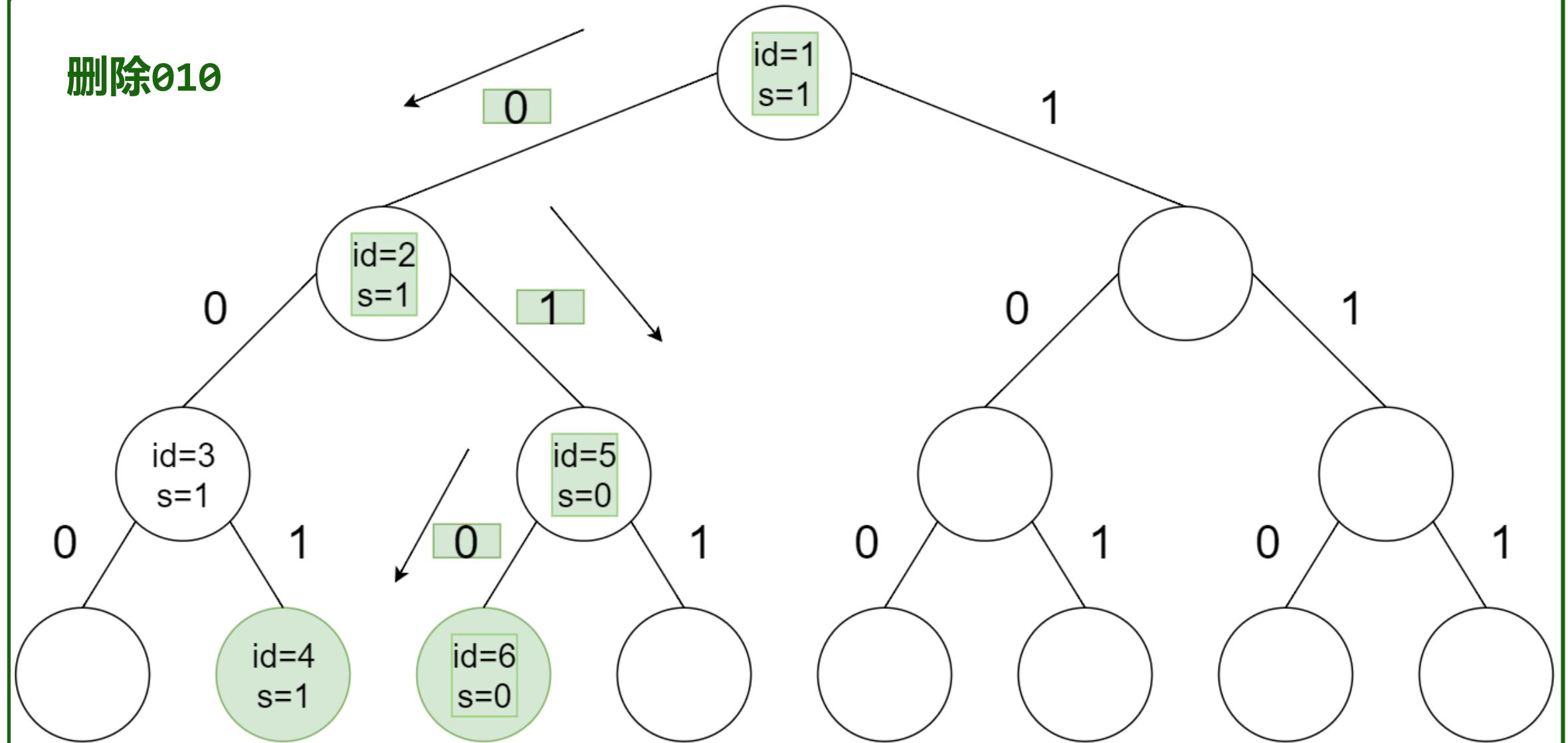
# Trie树

删除010



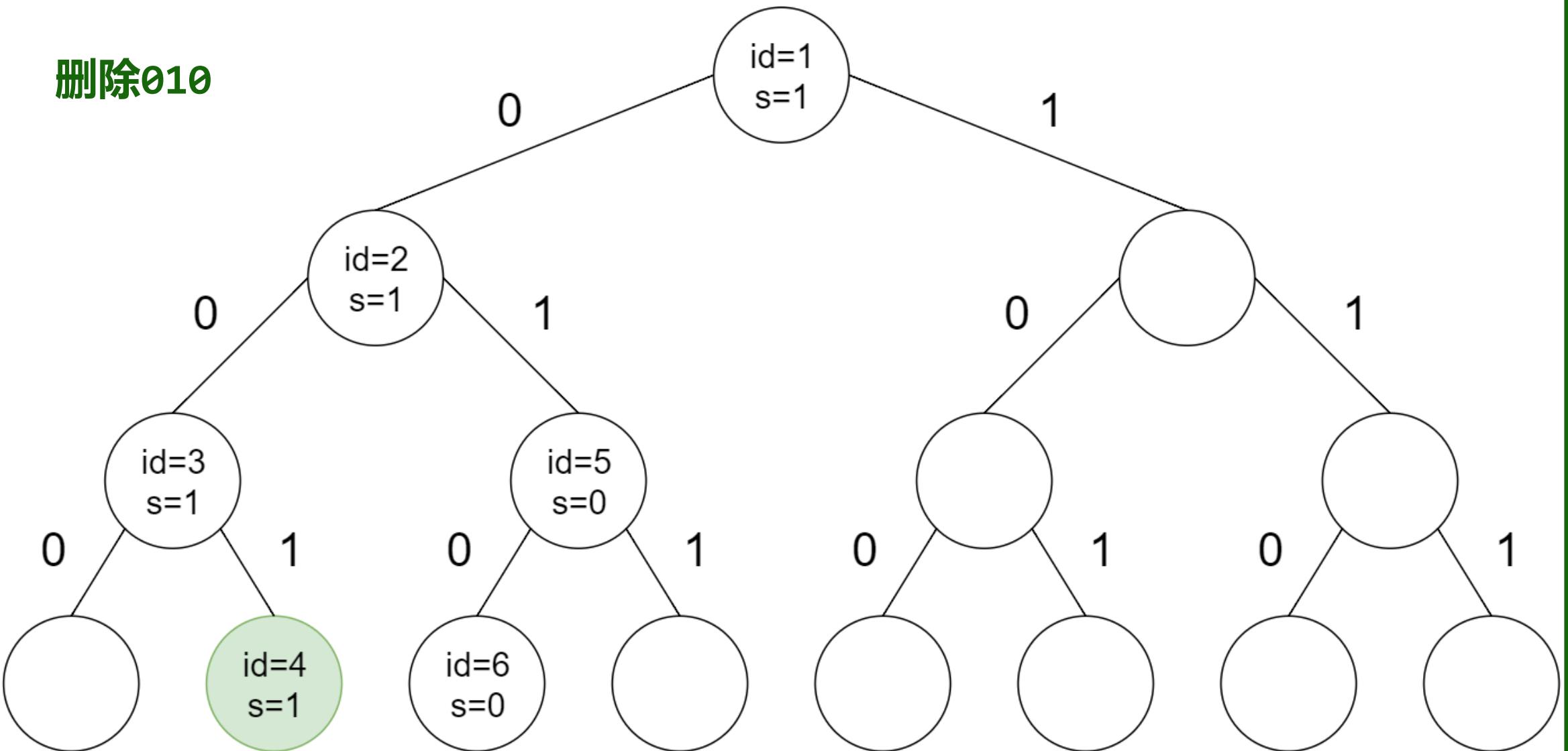
# Trie树

删除010



# Trie树

删除010

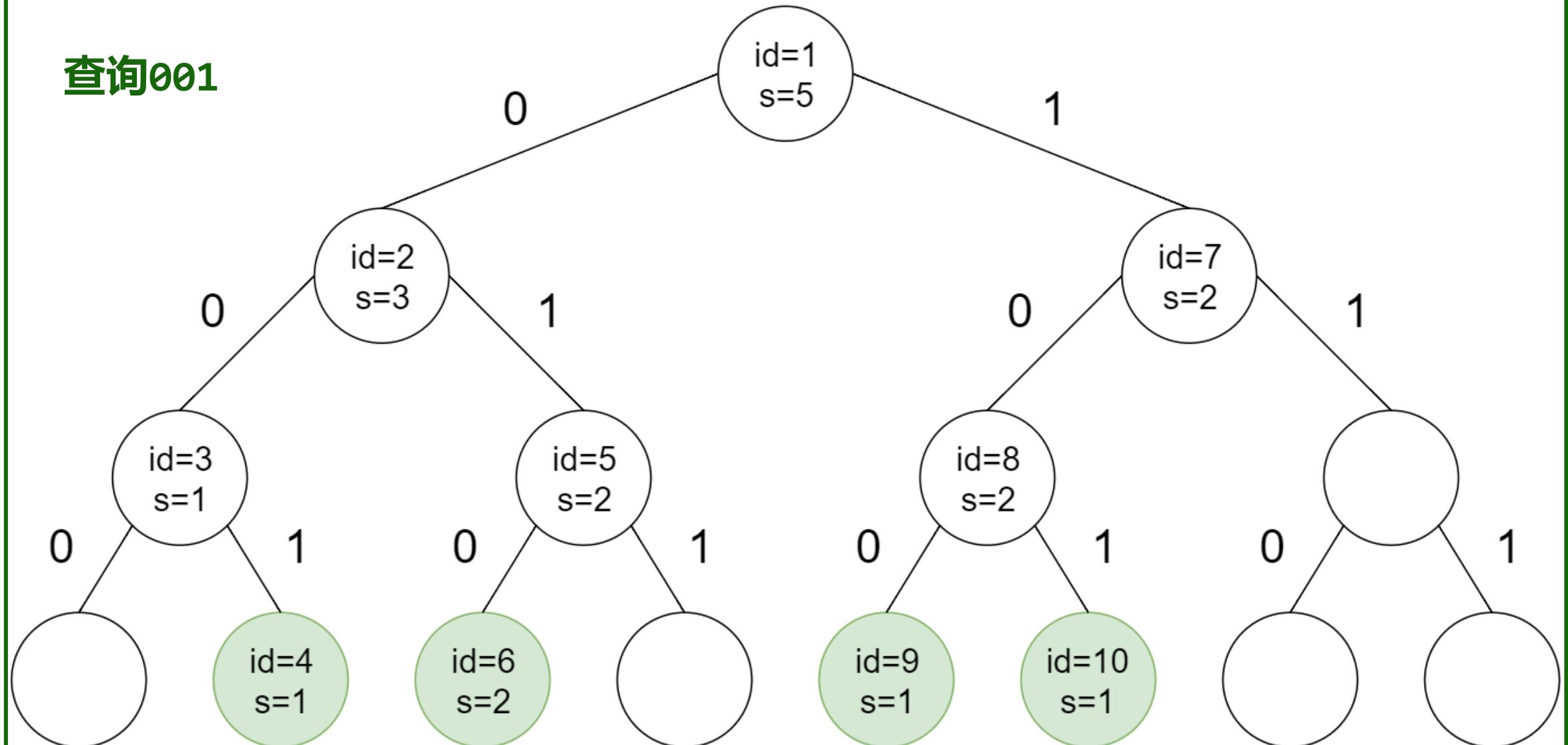


## Trie树

- ❖ 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。

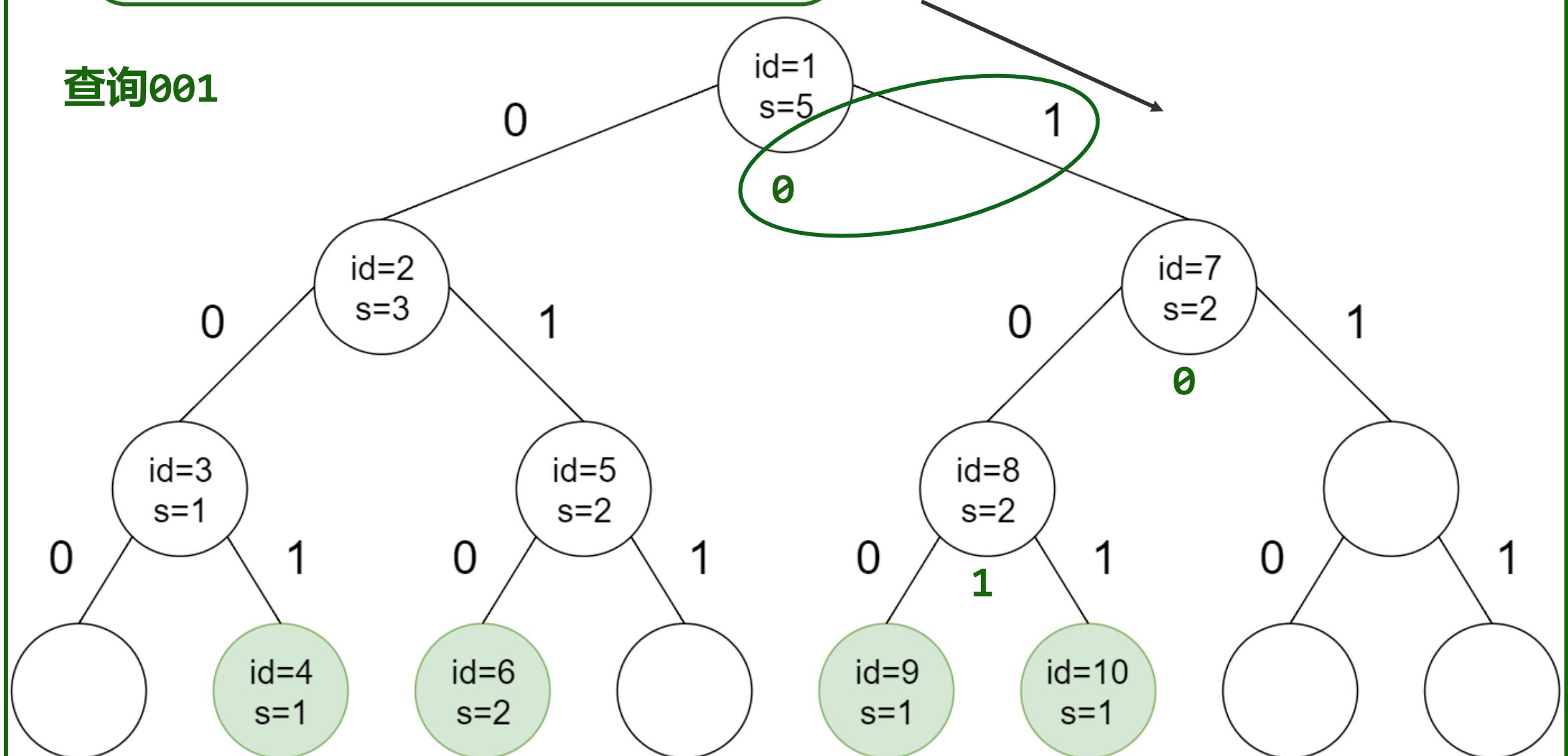
# Trie树

查询001



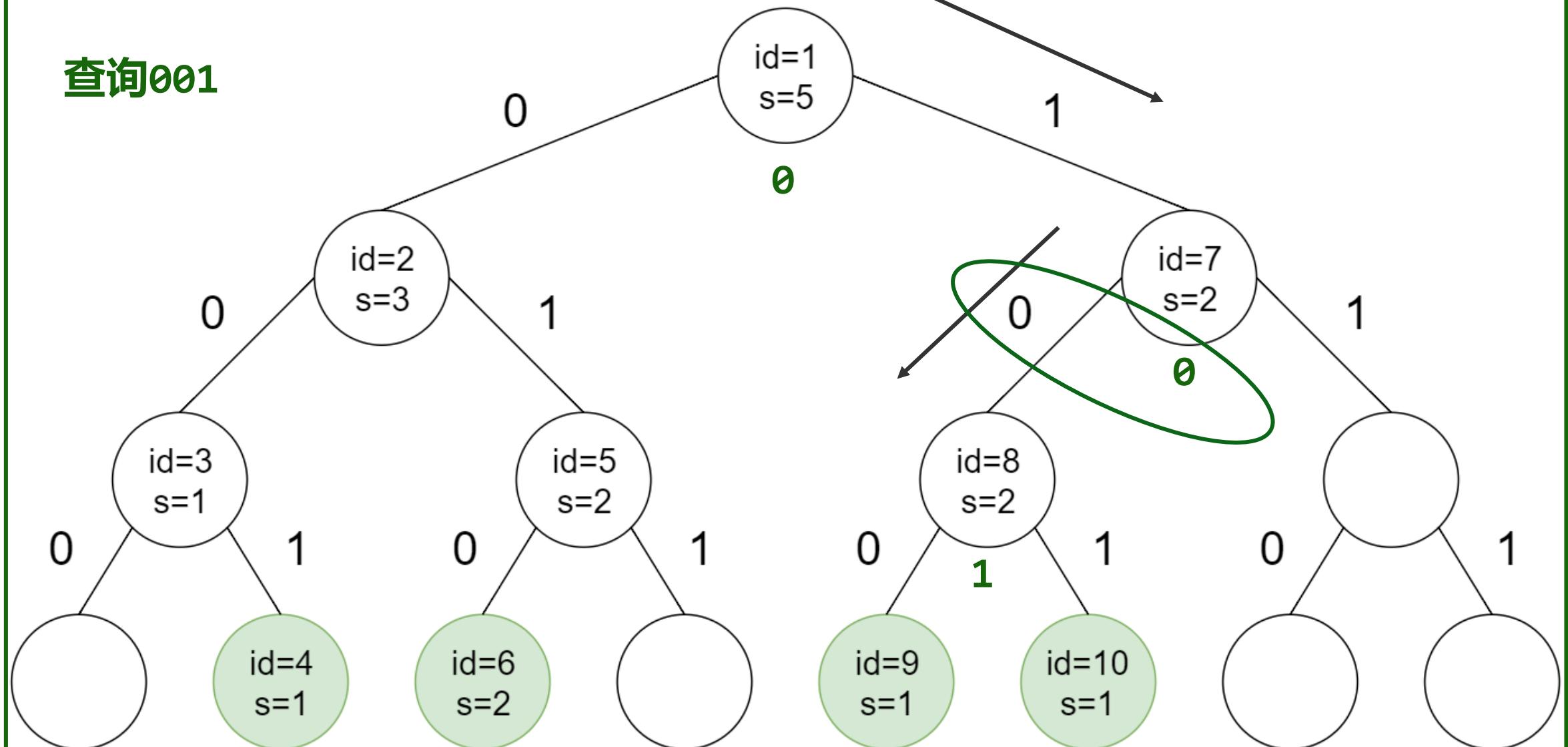
# Trie树

查询001



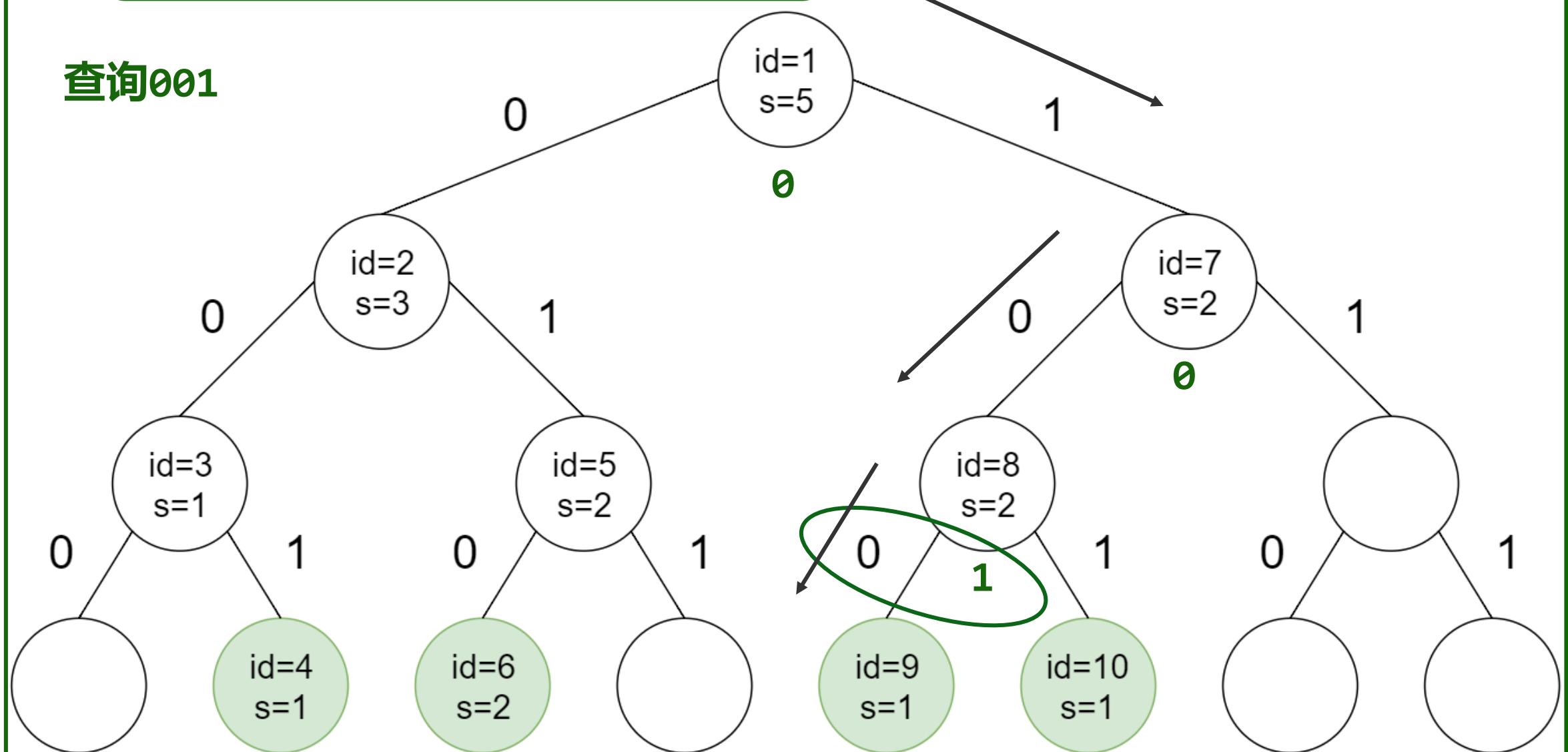
# Trie树

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 $O(\log ai)$ ，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 $O(n)$

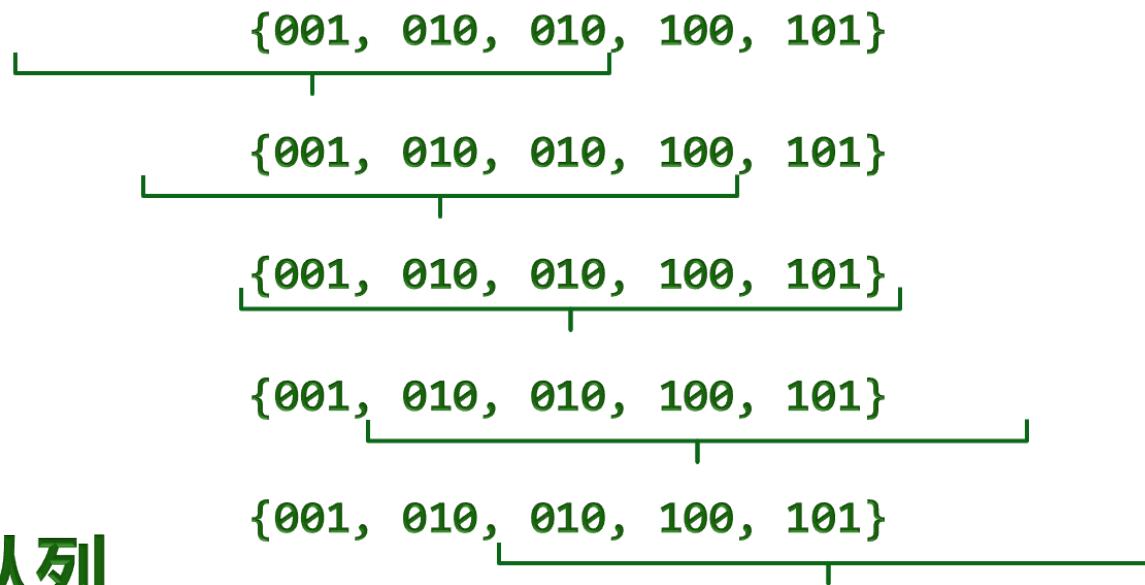
### ❖ 其实...这题更重要的是 $O(n)$ 里的常数...不能忽略掉

## 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



## 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！如果你一个节点用3个int（左右孩子+s），64层的话直接开一个 $3 \times 64N$ 个int来存这个Trie，会超内存！
- ❖ 这就需要高超的卡常技巧了（像我这么菜才不会卡）
- ❖ 一个最省事的方法就是，你代码的内存开到接近256M就好了！
- ❖ （毕竟这是正解对吧，出题人既然开到那么多说明这么多就够用（逃）
- ❖ （当然有一个黑魔法是路径压缩，我就不介绍了）

## 小结

- ❖ <del>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(╥﹏╥)o </del>

谢谢！！！

❖完

❖谢谢谢谢谢谢！！

**CST2018 2-3-2 Mooney**

**计76 徐天行**

## 问题描述

- ◆ 给定图  $G(V, E)$ ,  $1 \leq |V| \leq 5 \times 10^5$ ,  $1 \leq |E| \leq 1.2 \times 10^6$
- ◆ 每个点上有一个字符  $m/M$
- ◆ (1) 0 点到  $n-1$  点最少经过多少次  $M$  字符
- ◆ (2) 0 点到  $n-1$  点最多经过多少次  $m$  字符

## 问题分析

- ❖ 首先我们注意到，对于第一问，最优结果一定存在一种情况是一条初级通路
- ❖ 想一下？
- ❖ 然后，该问题就简化为点上有代价的单源最短路径问题
- ❖ 鉴于点数和边数不友好的规模，建议使用堆优化的dijkstra

## 问题分析

- ❖ 然后，我们来考虑第二问，第二问的最优答案可能是一般的道路
- ❖ 思考下面一件事情，如果整张图任意两点均可互相到达（强连通图），那
- ❖ 显而易见，答案就是整张图中 $m$ 的个数

## 问题分析

- ❖ 回到本题，如果存在AB两点可互相到达，当某一步走到A/B，如果另一点
- ❖ 将整张图使用Tarjan算法求出所有强连通分量，对强连通分量建立新图G
- ❖ 每当我们走到一个强连通分量上，就会取走属于该分量上所有的m
- ❖ 在拓扑图上使用dp即可

## 难点要点

- ❖ 问题一难点主要在于heap优化dijkstra的代码
- ❖ 问题二难点主要在于要分析出“同一连通分量上所有m一次性拿走的性质”
- ❖ 剩下的就没啥难写的了

## 小结

- ❖ 题目属于入门级别，思维难度一般，比PA1的dp简单多了
- ❖ 主要考察各位有没有备个板子

**CST2018 2-4 Sort**

**张晨**

## 问题描述

- ❖ 有一个比较器，可以每次比较三个数，并返回最小数和最大数
- ❖ 使用这个比较器，将n个互不相同的数从小到大排序
- ❖ 使比较器的使用次数k尽可能小
- ❖ 交互题
- ❖ 最大的测试点  $N=10^6$   $K=1.03*10^7$

## 问题分析

- ❖ 比较器返回的是三个数的大小关系：
- ❖ A, B, C中最大是A，最小是C  $\Leftrightarrow C < B < A$
- ❖ 需要尽可能地利用比较器给出的信息

# 用什么算法

- ❖ 学过的排序算法
- ❖  $O(n^2)$ ：插入排序，选择排序，冒泡排序.....
- ❖  $O(n \log n)$ ：归并排序，快速排序、（堆排序）.....
- ❖ 如果比较器的信息能完全充分利用，一次比较相当于原来的三次比较
  - $\text{Compare}(A, B, C) \Leftrightarrow \text{Compare}(A, B) \text{ Compare}(B, C) \text{ Compare}(A, C)$
- ❖ 所以对  $O(n^2)$  的算法，使用比较器最多只能将比较次数除以3
  - 仍是天文数字

## 快速排序

- ❖任选一个轴点，把比轴点小的数放到轴点左侧，比轴点大的数放到轴点右侧，递归处理轴点左右的两个序列
- ❖使用比较器可以一次将两个数与轴点比较
- ❖期望次数=快排的期望次数/2 $\approx(1.39 * n \log_2 n)/2 \approx 1.39 * 10^7$
- ❖最优次数 $\approx \frac{1}{2} * n \log_2 n \approx 9.97 * 10^7$ （假设每次划分都是均匀的）

## 优化方向

- ❖ 1. 能否在不增加比较次数的前提下，让划分更加均匀？
- ❖ 2. A, B两数与轴点比完后，可以得知AB之间的大小关系，后续处理能否利用上这个关系？

## 小结

- ❖ 充分利用比较信息是解决问题的关键
- ❖ 深入思考排序算法

# **CST2018 2-5 ChromPoly**

**罗凌骁**

## 问题描述

- ❖ 给一张无向图  $G$ ，求其色数多项式  $CP(G, t)$
- ❖ 色数多项式  $CP(G, t)$ ：对图  $G$  使用  $t$  种颜色的着色的方案数， $t$  足够大
- ❖ 限制：同一条边的两点颜色不能相同
- ❖  $n < 30, m < 70$

## 问题分析

- ❖ 根据提供的资料以及维基百科 Chromatic Polynomial 页面
- ❖ 得出一个结论：
- ❖ 这道题没有多项式做法（#P 问题），只能暴搜
- ❖ 但是对于很多情况，可以做一些优化，使得实际效果得到改善

## 问题分析

- ❖ 如果是非连通图，由于每部分的染色是独立的，显然只需要求出每个连通支的色数多项式，再求乘积，即：
- ❖ 设图 $G$ 有 $k$ 个连通支 $G_1, G_2, \dots, G_k$ ，则

$$CP(G) = \prod_{i=1}^k CP(G_i)$$

- ❖ 下面只需要考虑连通图

## 问题分析

❖ 对于特殊的连通图，其色数多项式可直接由公式求得

❖  $CP(K_n, t) = \prod_{i=0}^{n-1} (t - i)$

❖  $CP(T_n, t) = t(t - 1)^{n-1}$

❖  $CP(C_n, t) = (t - 1)^n + (-1)^n(t - 1)$

## 问题分析

- ❖ 策略1：
- ❖ 对于一个连通图，求其色数多项式时，首先判断其是否为特殊图，
- ❖ 如果是则直接返回结果，否则，则考虑进行传统的减边算法：选择一条边进行切断及合并，递归求解后返回两者之差。
- ❖ 能过4个点（50分测试）

## 问题分析

- ❖ 考虑这样一种情况
- ❖ 如果一个连通图的某个点度数为1，称其为一个“伪叶子”
- ❖ 设 $u$ 为一个伪叶子，设 $G'$ 为 $G$ 关于 $V - \{u\}$ 的导出子图（即将 $u$ 去除），
- ❖ 可先对 $G'$ 染色。设颜色数目为 $t$ ，显然无论 $G'$ 如何染色， $u$ 都可使用
- ❖  $t - 1$ 种颜色，故 $CP(G) = CP(G') \times (t - 1)$

## 问题分析

- ❖  $CP(G) = CP(G') \times (t - 1)$
- ❖ 根据上述式子，在策略一的基础上，可以得到策略二：
- ❖ 对于连通图，如果有伪叶子，则将其去掉，递归求解剩下的图的色数多项式，返回乘 $(t - 1)$ 的结果
- ❖ 能过6个点（50分测试）

## 问题分析

- ❖ 策略二还可再做优化
- ❖ 考虑某个点 $u$ 及其邻域 $\Gamma(u)$ 的导出子图 $G_u$ ，如果 $G_u$ 是一个完全图，
- ❖ 那么同前述分析，可以先将 $G' = G - \{u\}$ 染色， $u$ 应有 $t - |\Gamma(u)|$ 种颜色可用，即 $CP(G) = CP(G') \times (t - |\Gamma(u)|)$
- ❖ 很容易可以得到策略二的修改版：
- ❖ 尝试在 $G$ 中找到点 $u$ ，满足 $\Gamma(u)$ 的导出子图为一个团（即完全子图），
- ❖ 缩去 $u$ 递归求解，返回乘 $(t - |\Gamma(u)|)$ 的结果

## 问题分析

- ❖  $\Gamma(u)$  为完全图  $\Rightarrow CP(G) = CP(G') \times (t - |\Gamma(u)|)$
- ❖ 很容易可以得到策略二的修改版：
- ❖ 尝试在  $G$  中找到点  $u$ , 满足  $\Gamma(u)$  的导出子图为一个团（即完全子图），
- ❖ 缩去  $u$  递归求解，返回乘  $(t - |\Gamma(u)|)$  的结果
- ❖ 能过 7 个点（50 分）

## 问题分析

- ❖ 优化仿佛遇到了瓶颈，很难再找到减少递归分支的方法
- ❖ 实际上除了减边还可加边（正如《图论与代数结构》介绍的办法）
- ❖ 加边算法：对于两个不相连的点，其是否同色构成互斥两种情况，对应了
- ❖ 连边两种情况，递归求解并返回相加结果
- ❖ 既然可以加边了，可以尝试“凑”出一些团出来，套用策略二

## 问题分析

- ❖ 可以考虑选择一些度数不太大的点（以免添加过多的边）补充其邻域内缺少的边，以期望得到团。于是在策略二的基础上得到策略三
- ❖ 策略三：
- ❖ 找到当前度数最小的点，若其度数不超过 $k$ ，则在其邻域中使用加边
- ❖ 算法
- ❖ 经过测试， $k = 4$ 的时候可以通过本题， $k$ 在5~6取到极小值
- ❖ 可以合理地认为，图稀疏时应删边，稠密时应加边

## 难点要点

- ❖ 理解色数多项式
- ❖ 推导出特殊图的公式
- ❖ 推出剪枝技巧
- ❖ 编程难度

## 小结

- ❖ 本题主要考察了学生的剪枝能力和编程能力，需要我们有一定的
- ❖ 耐心和勇气。
- ❖ 同时，组合分析时应做到不重不漏，也是很容易错的地方

谢谢观看！

- ❖ 感谢郑林楷同学对我莫大的帮助！
- ❖ 有问题欢迎与我联系：[function2@qq.com](mailto:function2@qq.com)

# CST2018 2-6 Temperature

万峰源

## 问题描述

- ❖ 某气象台每天都要采集气温数据，并提供统计查询服务。其中最常见的一类查询是，根据用户指定矩形区域内所有观察站的观测值计算出平均气温。
- ❖ 原始数据本身的规模急剧膨胀，查询的频率也日益激增，传统蛮力算法的效率已无法满足要求，气象台只好请你帮忙提高查询的效率。
- ❖ 观察站的坐标呈矩阵排列，气象台可以随时修改某一个观测站的气温。用户可以指定某矩形区域来获得该区域内所有观察站的平均气温。
- ❖ 矩阵大小 $\leq 1200 \times 1200$ ，查询数 $\leq 2 \times 10^6$ ，气温范围 $[-1000, 1000]$ .....

## 问题分析

- ❖ 设矩阵大小为 $n$ 和 $m$ ，操作数为 $k$ 。
- ❖ 那么该问题就是对这个矩阵的单点修改与矩形内和查询。考虑到 $k$ 较大，我们要尽可能的控制每次操作的时间复杂度。
- ❖ 矩形查询问题很类似于课上讲过的Range树，它时间复杂度也很好，足够通过此题。
- ❖ 区别是什么呢？支持了单点修改，并且 $n$ 和 $m$ 足够小。
- ❖ 单点修改很好处理，而 $n$ 和 $m$ 较小时就可以用更简单的实现来解决问题了。

## 难点要点

- ❖ 交互库的使用与调试
- ❖ Range树的实现
- ❖ 可能需要用较好的实现方法来得到常数级的优化
  - 内层线段树也可以换用树状数组来解决

## 小结

❖一道看起来很经典的数据结构题，可以通过灵活应用经典数据结构来解决