极客时间算法训练营
第二十课
树状数组与线段树

李煜东

《算法竞赛进阶指南》作者

目录

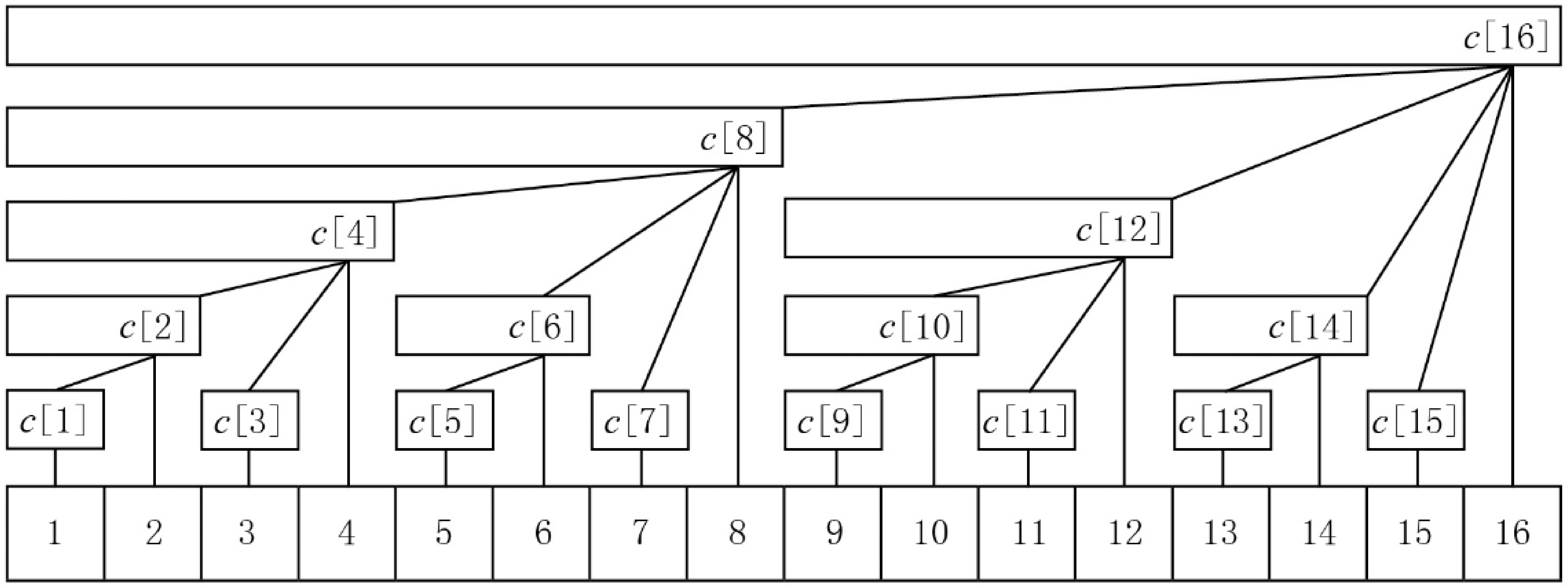
树状数组

树状数组

树状数组（Binary Indexed Tree, or Fenwick Tree）是一种维护数组前缀和、区间和的数据结构

思想和跳表有点类似：

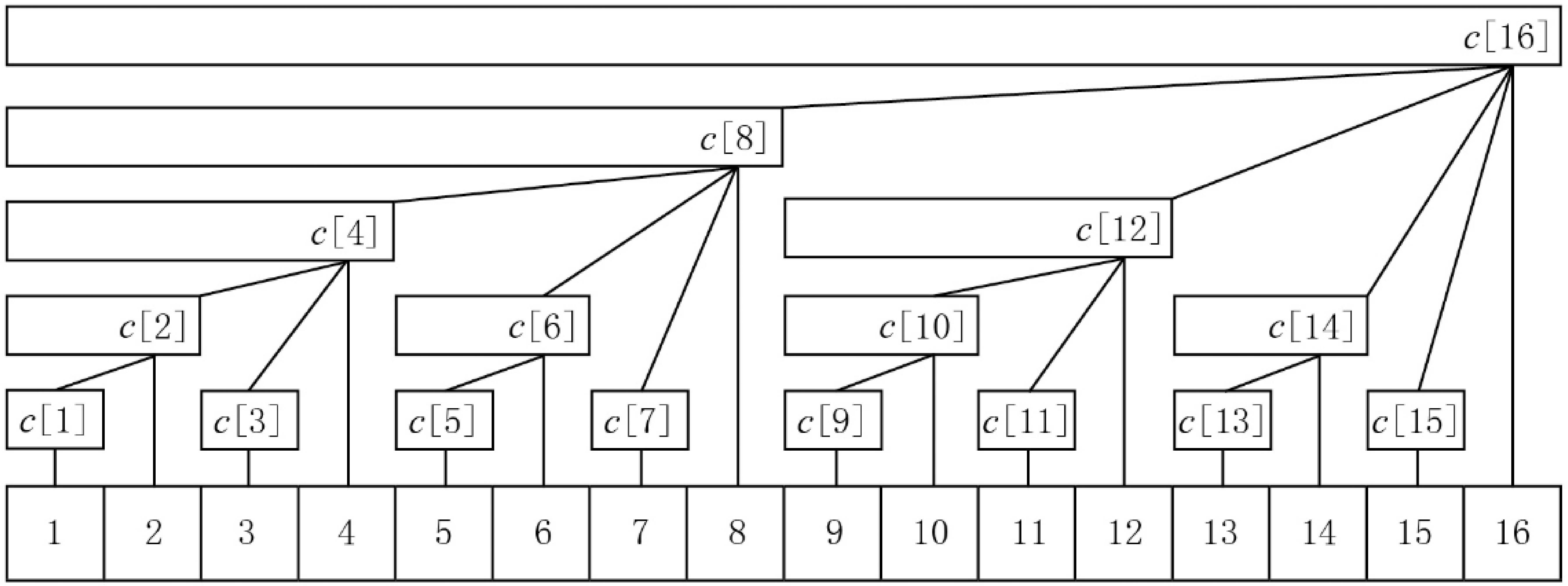
- 跳表：添加索引，高效维护链表
- 树状数组：添加索引，高效维护数组



如何建立索引?

树状数组的一个结点索引的原始数据数量，与该结点编号在二进制下最低位的1有关。

- 1,3,5,7,... 二进制下以1结尾，仅索引1个数据（自身）
- 2,6,10,14,... 二进制下以10结尾，索引2个数据（自身、它前面的那个）
- 4,12,... 二进制下以100结尾，索引4个数据（自身，前面的3个）



二进制分解与lowbit

任意正整数可以唯一分解为若干个不重复的2的次幂之和

- $7 = 2^2 + 2^1 + 2^0$，$12 = 2^3 + 2^2$

lowbit(x) 定义为 x 二进制下最低位的1和后面的0组成的数值（或者说 x 二进制分解下的最小次幂）

- $\text{lowbit}(7) = \text{lowbit}(111_2) = 1_2 = 2^0 = 1$
- $\text{lowbit}(12) = \text{lowbit}(1100_2) = 100_2 = 2^2 = 4$

树状数组 c 的结点 c[x] 存储 x 前面 lowbit(x) 个数据（包括x）的和

- c[7] = a[7]
- c[12] = a[9] + a[10] + a[11] + a[12]

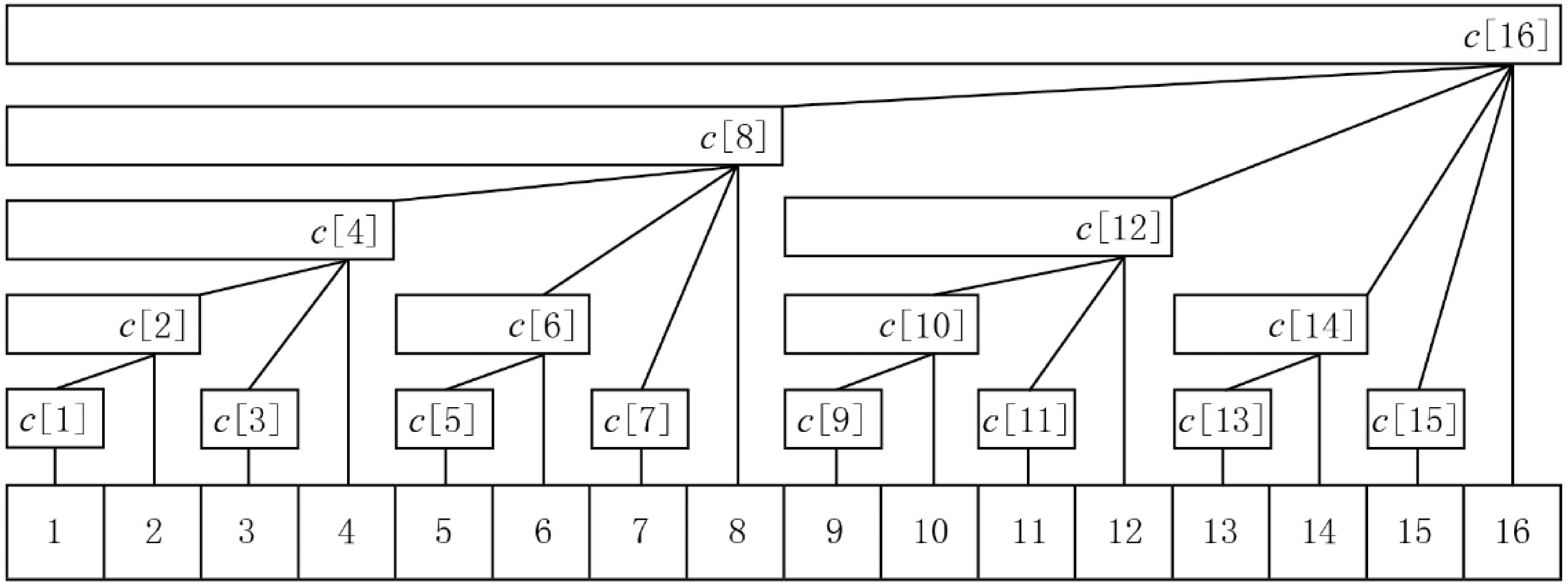
树状数组的性质

每个内部结点 $c[x]$ 保存以它为根的子树中所有叶结点的和。

除树根外，每个内部结点 $c[x]$ 的父亲是 $c[x + \text{lowbit}(x)]$。

树的深度为 $O(\log N)$。

如果 N 不是 2 的整次幂，那么树状数组就是一个具有同样性质的森林结构。



查询

树状数组支持的第一个基本操作 —— 查询前缀和

根据整数的二进制分解，可以把任意区间 [1,x] 拆成 O(logN) 个小区间

- 13 = 8 + 4 + 1，对应二进制 $1101_2 = 1000_2 + 100_2 + 1_2$
- [1, 13] 可以拆成 [1, 8], [9, 12], [13, 13]
- 对应二进制：$[1_2, 1101_2]$ 拆成 $[1_2, 1000_2]$, $[1001_2, 1100_2]$, $[1101_2, 1101_2]$

规律：

- 13 前面的 lowbit(13) = 1 个数，对应区间 [13,13]，再往前一个数是 12
- 12 前面的 lowbit(12) = 4 个数，对应区间 [9, 12]，再往前一个数是 8
- 8 前面的 lowbit(8) = 8 个数，对应区间 [1, 8]，结束

查询

```
int query(int x) {  // 查询前缀和（前x个数据的和）
    int ans = 0;
    for (; x > 0; x -= x & -x) ans += c[x];
    return ans;
}
```

前缀和知道了，区间和（第 l ~ r 个数据的和）可以直接由 query(r) - query(l - 1) 得到

时间复杂度：O(logN) —— 循环次数不超过二进制位数

规律：

- 13 前面的 lowbit(13) = 1 个数，对应区间 [13,13]，再往前一个数是 12
- 12 前面的 lowbit(12) = 4 个数，对应区间 [9, 12]，再往前一个数是 8
- 8 前面的 lowbit(8) = 8 个数，对应区间 [1, 8]，结束

更新

树状数组支持的第二个基本操作是单点增加，即把某个数据 x 增加一个值 delta

需要更新的索引就是 $c[x]$ 以及它的所有祖先结点，至多 O(logN) 个

利用性质：每个内部结点 $c[x]$ 的父亲是 $c[x + \text{lowbit}(x)]$

```
void add(int x, int y) {  // 第x个数据加上y
    for (; x <= N; x += x & -x) c[x] += y;
}
```

时间复杂度：O(logN)

如果要修改一个数据，可以先算出差值，再执行 add 操作

实战

区域和检索 - 数组可修改

https://leetcode-cn.com/problems/range-sum-query-mutable/

模板题

树状数组的局限性

树状数组有实现简单、效率高、省空间等诸多优势

但也有很大的局限性

维护的信息需要满足区间减法性质

- 不然无法通过前缀和相减得到区间和
- 例如无法直接拿来维护区间最值

不能很好地支持修改操作

- 单点修改需要先求出差值，转化为增加操作
- 基本上难以支持区间修改（修改连续的一段数据）

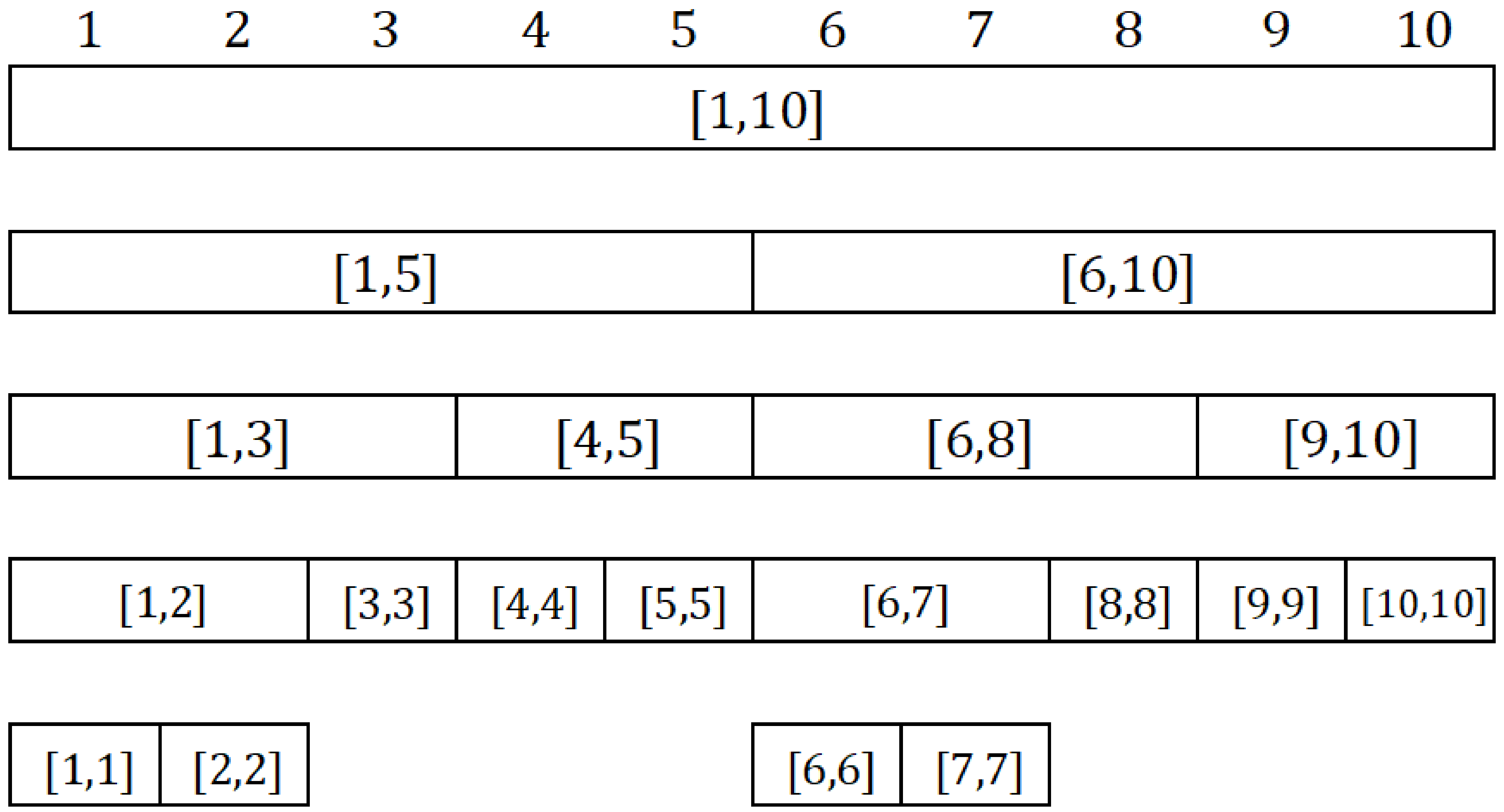
线段树

线段树

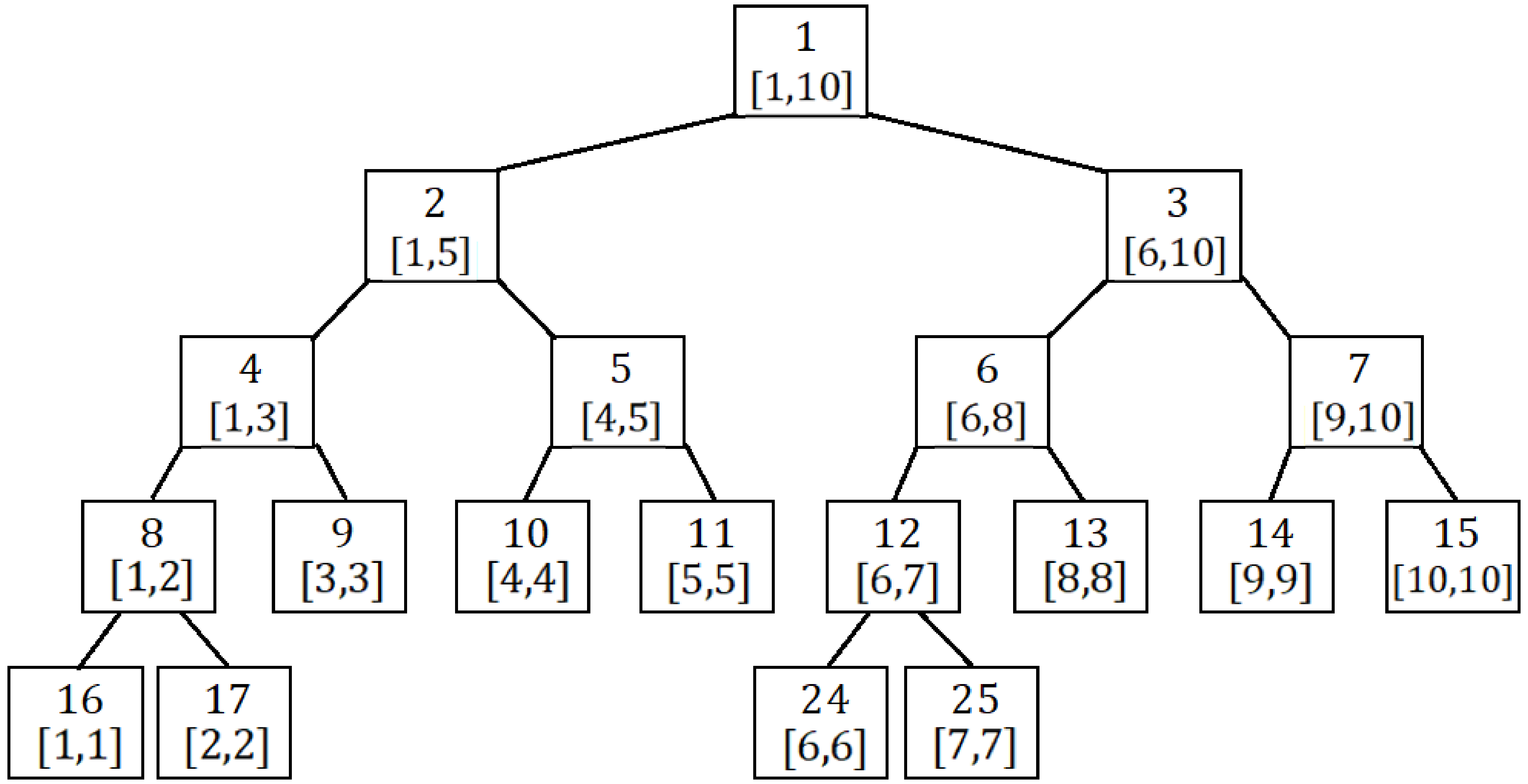
线段树（Segment Tree）是一种基于分治思想的二叉树结构，用于在区间上进行信息统计。

- 线段树的每个节点都代表一个闭区间。
- 线段树具有唯一的根节点，代表的区间是整个统计范围，如 $[1, N]$。
- 线段树的每个叶节点都代表一个长度为1的元区间 $[x, x]$。
- 对于每个内部节点 $[l, r]$，它的左子节点是 $[l, mid]$，右子节点是 $[mid + 1, r]$，其中 $mid = (l + r)/2$（向下取整）

线段树



区间视角



二叉树视角

线段树

除去树的最后一层，整棵线段树一定是一棵完全二叉树

树的深度为 $O(\log N)$

可以按照与二叉堆类似的“父子2倍”节点编号方法：

- 根节点编号为1。
- 编号为 p 的节点的左子节点编号为 $p * 2$，右子节点编号为 $p * 2 + 1$。

这样一来，就能简单地使用数组来保存线段树

由于最后一层不一定是连续的，保存线段树的数组长度不要小于 4N

区间最值问题（Range Maximum Query）

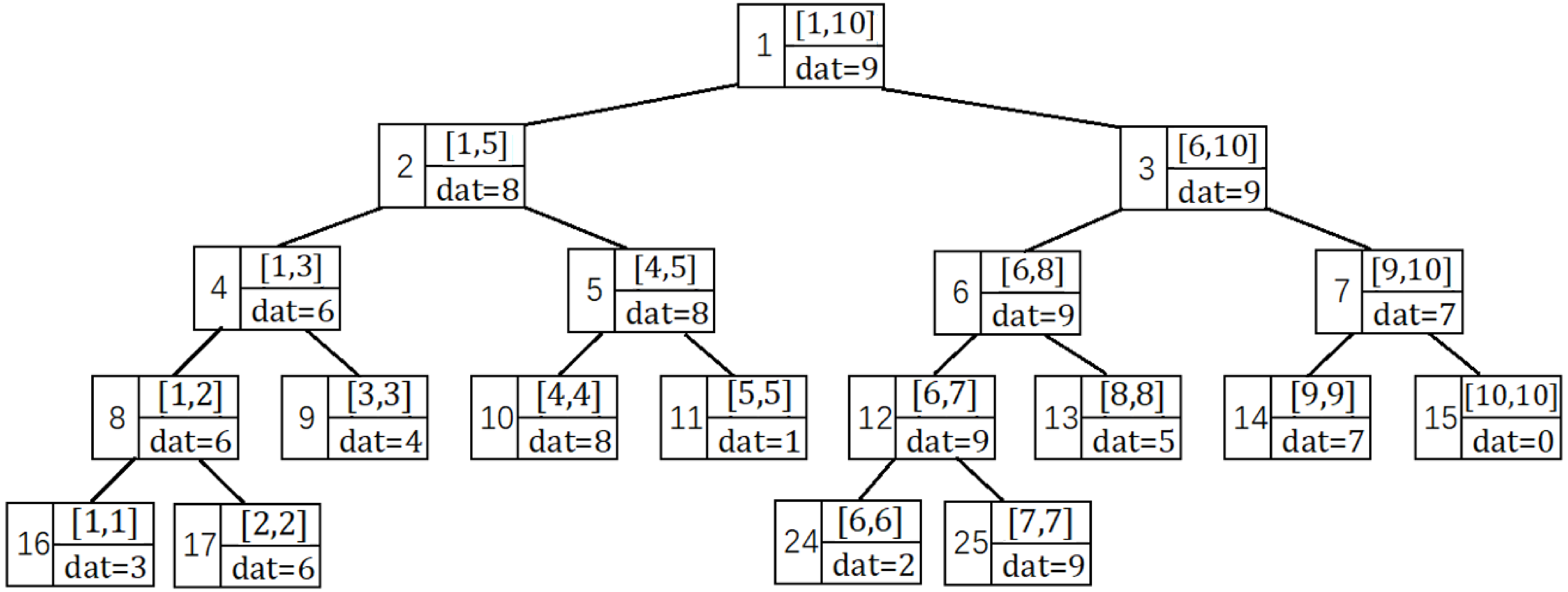
维护一个序列，支持：

- 查询区间最值（第 l 个数到第 r 个数的最大值）
- 单点修改（更新第 x 个数据）
- *（选做）区间统一修改（把第 l 个数到第 r 个数都置为 val）

建树

Build(1, 1, n)

时间复杂度 O(n) —— 不超过结点数 4n



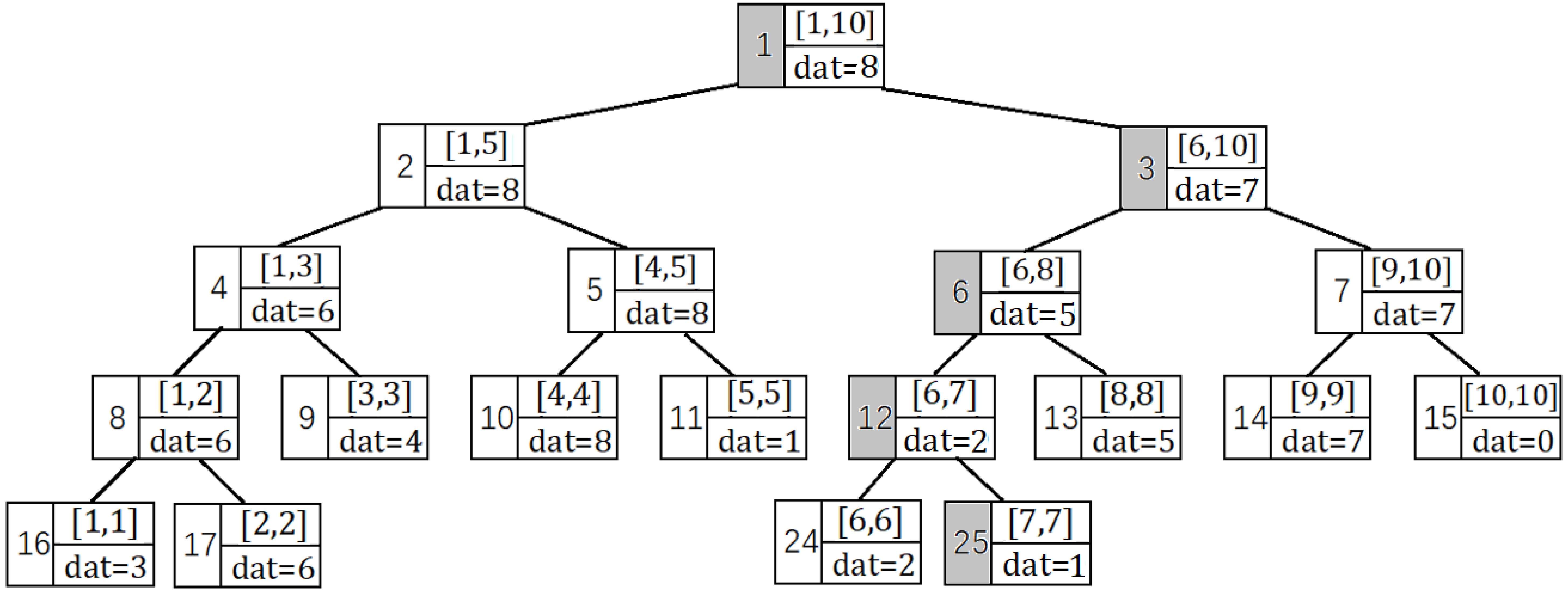
Build(1,1,10)　　A={3,6,4,8,1,2,9,5,7,0}

单点修改

Change(1, x, v)

- 从根（1号）结点出发，递归找到代表区间 [x, x] 的叶子结点
- 自底向上更新保存的信息

时间复杂度 O(log(n)) —— 每层一个结点，更新总次数不超过树高



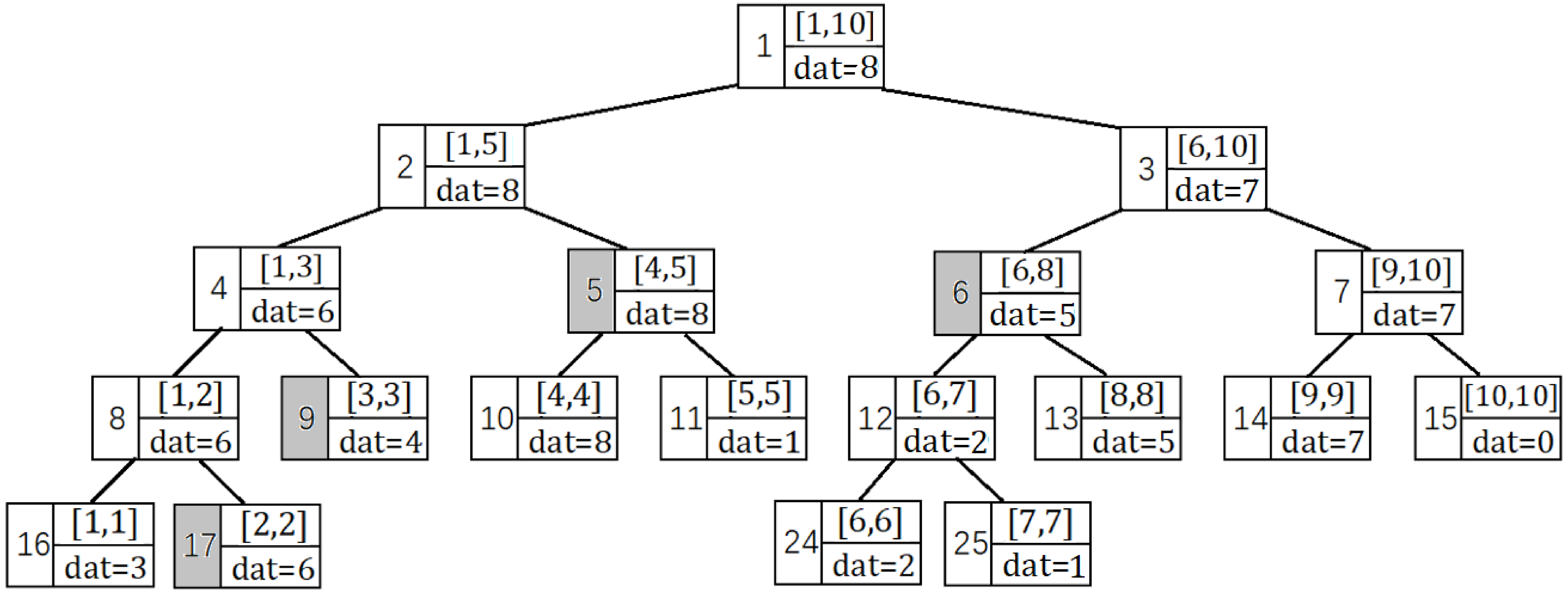
Change(1,7,1)

区间查询

Query(1, l, r)，从根结点（1号结点）开始递归查询：

- 若 $[l, r]$ 完全覆盖了当前结点代表的区间，则立即回溯，并且该结点的 dat 值为候选答案。
- 若左（右）子结点与 $[l, r]$ 有重叠部分，则递归访问左（右）子结点。

时间复杂度 O(log(n)) —— l, r 各在树上划分出一条边界，最多形成 2logn 个候选区间



Ask(1,2,8)=max{6,4,8,5}=8

实战

区域和检索 - 数组可修改

https://leetcode-cn.com/problems/range-sum-query-mutable/

模板题

一个简单的整数问题 2（选做）

https://www.acwing.com/problem/content/description/244/

带区间修改的模板题

区间修改（选修）

区间查询与修改的区别：

- 区间查询遇到完全覆盖的区间，可以直接返回，最多处理 2logn 个结点
- 对于完全覆盖的区间，区间修改会影响它的所有子结点，时间复杂度最坏 O(n)

解决方案：懒惰标记（又称延迟标记）

- 回想二叉堆的懒惰删除
- 遇到完全覆盖的区间，先打一个修改标记，只要不到子结点中查询，就不往下继续修改
- 在以后的递归查询/修改中遇到了标记，再往下传递一层

时间复杂度优化到 O(log(n))

思想：我mark一下这个bug要改，你们谁以后看见了记得先改一下再run哈，碰不到就不管了

场景：线段覆盖

设计实现一个 class，支持以下调用：

- cover(l, r, color)，把数轴上区间 [l, r] 染成 color 颜色
- query(x)，实时查询数轴上 x 坐标的颜色

10万次调用

坐标范围 0~1e5

离散化

场景：线段覆盖（批处理 + 无穷坐标）

批处理执行一系列操作：

- cover(l, r, color)，把数轴上区间 [l, r] 染成 color 颜色
- query(x)，实时查询数轴上 x 坐标的颜色

10万条操作

坐标范围：任意实数（double 范围内）

离散化

离散化就是把无穷集合中的若干个元素映射为有限集合以便于维护的方法

最常见的场景：坐标压缩

- 题目的坐标范围 -1e9~1e9、任意实数 等
- 其中只有 N 个坐标有用（在数据中出现了）

可以把出现过的坐标按大小顺序映射为 1, 2, 3, ..., N

可用算法：

- 排序去重 + 二分
- 有序集合（sorted set）、有序映射（sorted map）

场景：线段覆盖（批处理 + 无穷坐标）

cover(-1e9, 700, red)
cover(-6.01, 21.627, blue)
cover(98, 700, yellow)
query(12.345)

出现过的坐标（排序去重）：{-1e9, -6.01, 12.345, 21.627, 98, 700}

映射为：{1, 2, 3, 4, 5, 6}

cover(1, 6, red)
cover(2, 4, blue)
cover(5, 6, yellow)
query(3)

Homework

掉落的方块

https://leetcode-cn.com/problems/falling-squares/

实战

区间和的个数（选做）

https://leetcode-cn.com/problems/count-of-range-sum/

区间和 = 前缀和相减，即 $s[r] - s[l] \in [lower, upper]$，其中 $0 \le l < r$

枚举 r，需要统计 r 前面有几个 l 满足 $s[r] - lower \le s[l] \le s[r] - upper$

从前往后扫描：

- “s[l]”是需要插入的东西
- 查询 [s[r] - lower, s[r] - upper] 中有几个

离散化 + 线段树

各种树形数据结构的对比

数据结构	用途	n 次操作时间复杂度
并查集	关系维护、图（连通性）、利用路径压缩等	$O(n\ \alpha(n))$
Trie 树	维护字典（多个字符串的存储和查询）	$O(\text{total length of string})$
二叉堆	最大、最小值的维护与查询	$O(n \log n)$
树状数组	区间信息的维护与查询（需要区间减法性质）	$O(n \log n)$
线段树	区间信息的维护与查询	$O(n \log n)$
平衡二叉搜索树	实现有序集合/映射 区间的信息维护与查询（更加灵活）	$O(n \log n)$

THANKS

极客时间 | 训练营