数据结构

2019 年 8 月 3 - 4 日

黄哲威 hzwer

主要内容

- 哈希表
- 并查集
- 字典树

哈希表

- 哈希表通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。
- 这个映射函数叫做散列函数，存放记录的数组叫做散列表。

哈希表

- 单数字散列

- 除留余数法，通常取质数

- 各种运算，如平方取中法、折叠法

- 数列 / 字符串的散列:

- 排序编号（离线情况）

- 边乘边模 (BKDR 算法)

- 自然溢出加速（把保存不了的位丢弃）

哈希表 - 碰撞解决

- 碰撞有多常见？生日悖论

- 怎么解决?

    - 拉链法： 用链表将同一 hash 值的若干元素串联

    - 开放寻址法： 若某元素插入的散列地址已有元素占用，则使用一定方法寻 找下一个可用地址

    - 多重哈希

并查集

- 在一些有 N 个元素的集合应用问题中，我们通常是在开始时让每个元素构成一个单元素的集合，然后按一定顺序将属于同一组的元素所在的集合合并，其间要反复查找一个元素在哪个集合中

- 采用一种全新的抽象的特殊数据结构——并查集来描述

并查集操作

- 初始化：把每个点所在集合初始化为其自身 通常来说，这个步骤在每次使用该数据结构时只需要执行一 次，无论何种实现方式，时间复杂度均为 O(n)

- 查找：查找元素所在的集合，即根节点

- 合并：合并将两个元素所在的集合合并为一个集合。通常来说，合并之前，应先判断两个元素是否属于同一集合

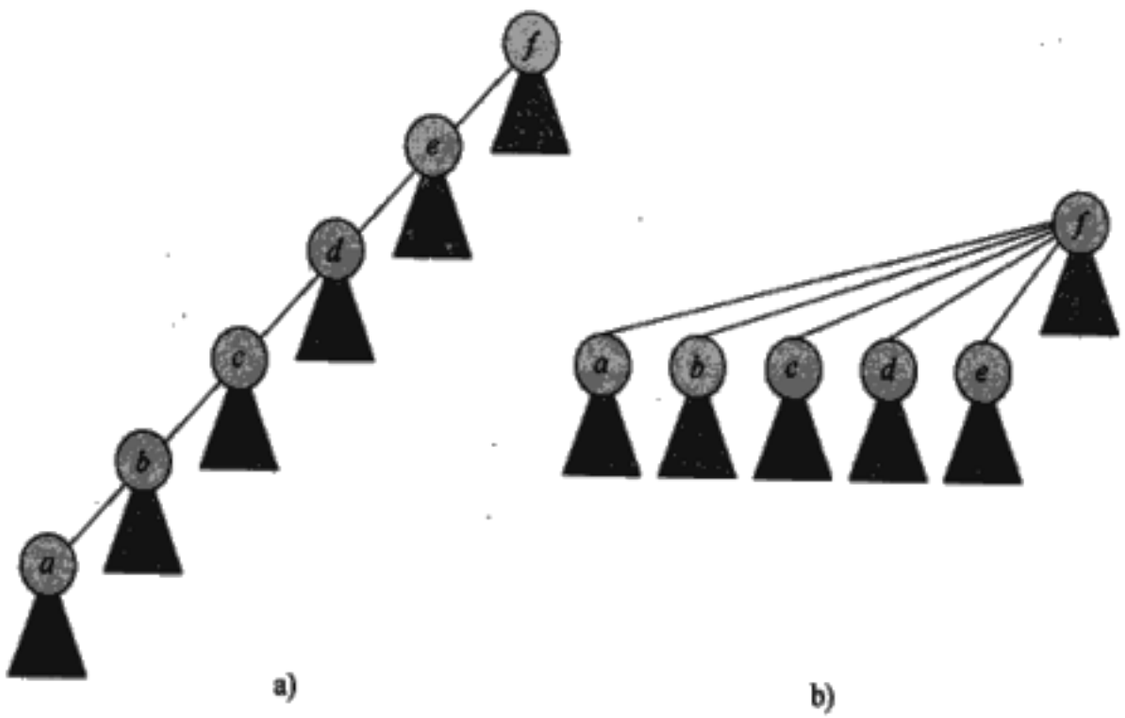
带权并查集

- 维护元素所属集合的同时记录集合的一些信息：
- 维护集合的最大最小值；维护集合的和；维护其它的信息
- 很像树形递推

并查集优化

- 路径压缩：当我们经过递归找到祖先节点后，回溯的时候顺便将它的子孙节点都直接指向祖先。

- 启发式合并（按秩合并）：给每个点一个秩，其实就是树高每次合并的时候都用秩小的指向秩大的。



并查集优化

- 当同时使用按秩合并和路径压缩时，最坏情况运行时间为 $O(m\alpha(n))$，其中 $\alpha(n)$ 是一个增长极其缓慢的函数。

- 在各种实际情况中，可以把这个运行时间看作与 m 成线性关系。

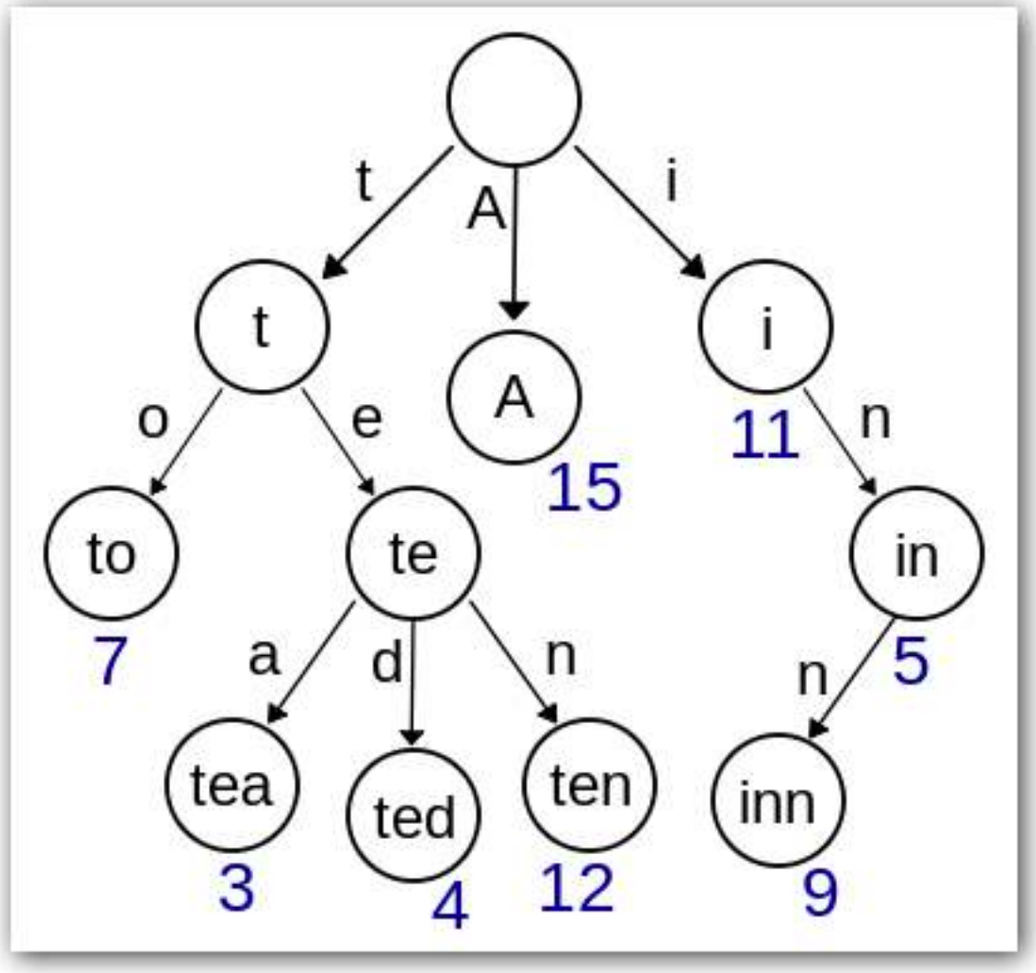
字典树

基本性质：

1、根节点不包含字符，除根节点外每一个节点都只包含一个字符

2、从根节点到某一个节点，路径上经过的字符连接起来，就是该节点对应的字符串

3、每个节点的所有子节点包含的字符都不相同



字典树的应用

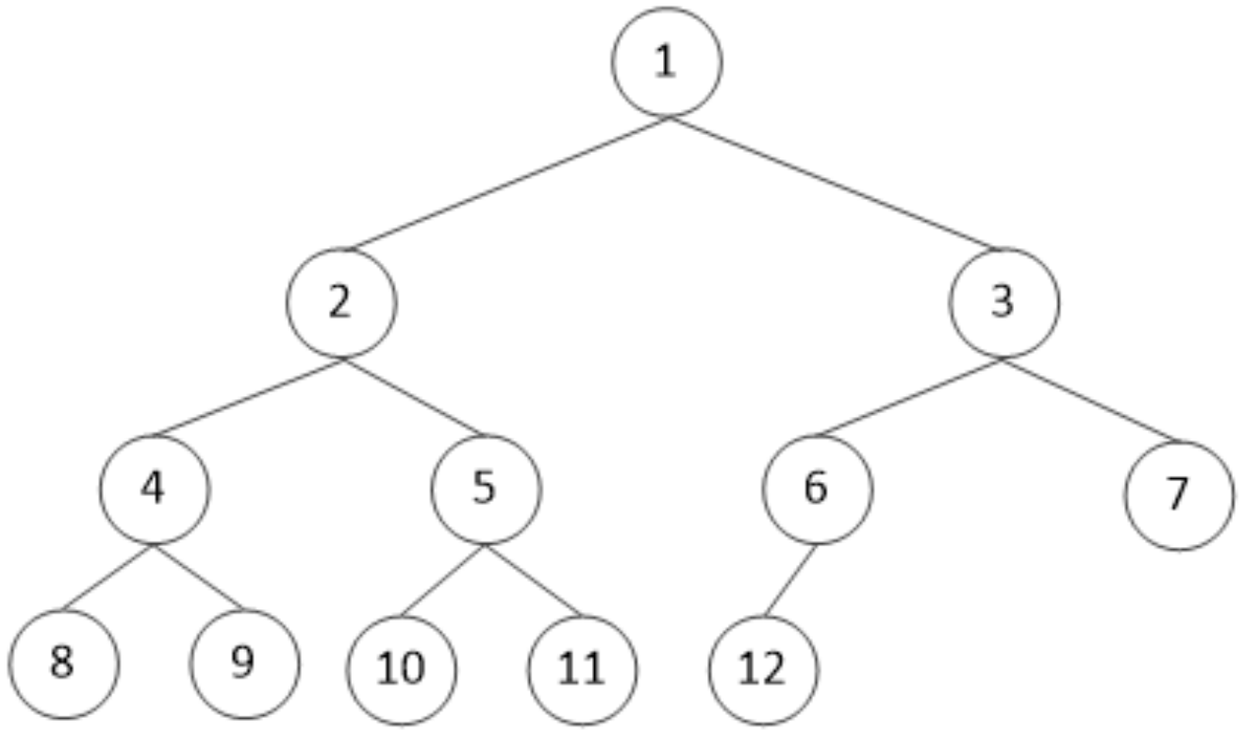
1. 用来排序字符串

2. AC 自动机（NOIP 不要求

3. 两个树的最长公共前缀，转化为求 LCA

LCA

- 最近公共祖先 (Lowest Common Ancestors)
- 对于给定的一棵有根树，给定两个点u，v
- LCA(u,v) 表示一个结点x，这个结点为u的祖先，也为v的祖先，且这个结点深度最大



求 LCA 的方法

- 倍增求LCA：
- 用fa[k][i]表示节点K的第2^i父亲是谁。
- 每次查询两个节点a, b最近公共祖先直接做就可以了。
- 树链剖分求LCA，略

习题第一部分

650A.Watchmen

- 给定二维平面的 n 个坐标，求满足曼哈顿距离等于欧几里得距离的点对数

- 1 ≤ n ≤ 10^5,1 ≤ xi, yi ≤10^9, 可能有重点

650A.Watchmen

- 符合条件的点对在同一行 / 列
- 用 hash (map) 统计

Spy Syndrome 2

- 给定长为(n <= 10000)的主串，给(m <= 100000)个长不超过 1000 的子串，总长不超过 1000000
- 问主串是否能由子串拼出，每个子串可以多次使用

Spy Syndrome 2

- Fi 表示主串的前 i 位能否用子串拼出
- 求每个子串的哈希值，主串每位枚举串长 <= 1000，求出哈希值和子串进行匹配转移

例题. 树的同构

- 给定两棵 n 个点的有根树，求两棵树是否同构 (相同)
- 1 ≤ n ≤ 10^5

例题. 树的同构

- 自下而上哈希
- 对于一个父亲节点，将其所有子树的 hash 值排序，其得到的数组用各种方法 hash，即可计算出父亲节点所代表的整个子树的 hash 值
- 无根树的情况，把重心作为根

团伙

- n 个强盗，强盗之间可能有朋友或者敌人关系。
- 1. 我的朋友的朋友是我的朋友。2. 我敌人的敌人也是我的朋友。 两个强盗是同一团伙的条件是当且仅当他们是朋友。
- 现在给定 m 条强盗之间的关系，问最多有多少个强盗团伙。保证输入的强盗关系的合法性。
- $0 \le n \le 100000, 0 \le m \le 1000000$

团伙

- 将有关系的强盗并集
- 若 2 个强盗是朋友，则连一条边权为 0 的边
- 若 2 个强盗是敌人，则连一条边权为 1 的边
- 若 2 个强盗“有关系”，则这 2 个强盗属于同一个连通图，反之亦然
- 2 个强盗之间的任意一条路径阐述了他们之间的关系，将路径上的所有边权加起来，对 2 取模，为 0 则是朋友，为 1 则是敌人
- 实现合并的时候，要计算两个根之间的关系

CHSEQ22.Chef and Favourite Sequence

- 一长度为 n 的 0,1 序列，初始时所有元素为 0。
- 有 m 个区间 [Li, Ri]，可以任选若干个区间进行区间翻转操作 (0 变 1, 1 变 0)，问可以得到多少种不同的序列
- 答案对 (10^9 + 7) 取模
- 1 ≤ n, m ≤ 10^5

CHSEQ22.Chef and Favourite Sequence

- 考虑将一段区间取反，相当于将它的差分序列的 l 和 r+1 俩个位置取反

- 如果某一项操作可以用其它一些操作替代的话，它就可以直接被舍弃了，可以用一个并查集来维护

- 即把所有操作区间的左右端点在并查集中合并，如果一个区间的左右端点已经并在一起了，它就可以被其它操作区间替代

- 最后剩下 k 个操作，答案就是 2^k

环

- 给出一张 n 个点 m 条边的无向图，依次询问某个点所在连通块是否为环，是否为树?

- 1 ≤ n, m ≤ 10^6

环

- 依次连边，当一条边连接的两个点处于同一连通块时

- 根据出入度判断连通块的形状

Usaco2012Jan.Bovine Alliance

- 给出 n 个点 m 条边的图（不一定连通），现把点和边分组。
- 每条边要和它相邻两点之一分在一组。点不可以和多条边一组，但点可以单独一组，问分组方案数。
- 答案对 10^9 + 7 取模。
- 1 ≤ n, m ≤ 10^5

Usaco2012Jan.Bovine Alliance

- 连通块是独立的，设某个连通块点数为 n，边数为 m
- 若 m > n，边数大于点数，必然有一个点要与多条边分在一组，无解
- 若 m = n，　环 + 外向树，解为 2
- 若 m = n - 1, 树，解为 n
- 若 m < n - 1, 不存在这样的连通块
- 使用并查集维护连通块的边数和点数

UOJ14. DZY Loves Graph

- DZY 开始有 n 个点，现在他对这 n 个点进行了 m 次操作，对于第 i 个操作（从 1 开始编号）有可能的三种情况：

- Add a b： 表示在 a 与 b 之间连了一条长度为 i 的边（注意，i是操作编号）。

- Delete k： 表示删除了当前图中边权最大的 k 条边。

- Return：表示撤销第 i – 1 次操作。保证第 i–1 次不是 Return 操作。

- 1 <= n <= 5 * 10^5，部分数据只有 Add，部分数据没有 Return 操作。

- 每次操作以后，求全图的最小生成树边权和，若不存在输出 0。

UOJ14. DZY Loves Graph

- 只有加边的情况，那么当图连通后第一次有了最小生成树。
- 因为加的边权是单调递增的，最小生成树保持不变直到最后。
- 问题是怎么从并查集删除掉最后添加的 K 条边?

UOJ14. DZY Loves Graph

- 使用按秩合并的并查集，由于一条边被插入后就不会修改，直接模拟加边删边。

- 如何实现 Return?

- 使用类似离线的做法，我们在做第 i 个操作的时候可以知道第 i+1 个操作是否是Return操作。

- 如果第 i 个操作是 Add 操作，那么第 i+1 个操作是否是 Return 并没有太大的影响，因为加入一条边和删除一条边的时间代价都是O(logn)。

- 如果第 i 个操作是 Delete 操作，第 i+1 个操作是 Return 操作，说明其实删边操作是假的，只要求一个答案。我们可以事先存下使用当前权值前 k 小的边时最小生成树大小直接输出即可。

链型网络

- 给定一张无重边，无自环的无向图，n 个点，m 次操作
- 一开始图中没有边，每次操作可以加边，或询问有多少个点满足：将该点删除后，原图的每个连通块都为一条链
- 1 <= n, m <= 10^5

链型网络

- 思考下面一些简单的情况：
  原图为若干条链，则答案为点数 n;
  原图为单个简单环和若干条链，则答案为环大小;
  原图中超过一个连通块有环，答案为 0.
  原图中一个连通块为一个点连接三条链，答案为 4;
  原图中一个连通块为一个点连接三条以上的链，答案为 1;

- 但考虑到环套树这样更复杂的情况，上述分类讨论也无能为力

链型网络

- 我们需要思考链本身的性质，一个图的每个连通块为链，等价于 每个点的度数小于等于 2 且无环

- 转化后的条件明显更有利于解决问题

- 首先考虑图中是否有度数大于等于 3 的点，如果存在一个度数大于等于 3 的点 u, 因为要保证去掉一个点后每个点的度数都小于等于 2，我们要么去掉 u，要么在 u 度数恰好为 3 时，去掉 u 的三个相邻点中的一个，而其他的点均不可能成为答案

链型网络

- 在加边过程中第一次出现度数为 3 的点的时候，对于可能成为答案的 4 个点，分别维护一个去掉该点之后的图

- 然后在 4 个图中分别进行判断

- 只要在加边时判断每个点度数都小于等于 2, 再用并查集判断是否有环即可

链型网络

- 剩下的就是每个点的度数都小于等于 2 的情况，这样每个连通块 只能是链或者简单环，我们只需采用一开始的分类讨论即可.

- 原图为若干条链，则答案为点数 N;
  原图为单个简单环加若干条链，则答案为环大小;

- 原图中超过一个连通块有环，答案为 0.

- 这只需要维护一个记录集合大小的并查集就能做到.

序列维护数据结构

- 前缀和
- 稀疏表
- 分块，线段树，树状数组
- 二叉搜索树

前缀和和差分序列

- 对于序列 A[i]，如果用 S[i] 表示前 i 个数的和
- sum[l, r] = S[r] - S[l - 1]
- 差分序列可以处理区间加法，单点查询
- 0 3 0 0 -3 0 的区间和是 0 3 3 3 0 0
- 即在差分序列上修改两个数就能实现区间加法
- 可以扩展到二维的情况

RMQ

- RMQ (范围最小值查询) 问题是一种动态查询问题，每次询问数组 A 在区间 [l, r] 中最小的元素值。

- 解决一般RMQ问题的几种方法

稀疏表 (Sparse Table)　O(nlogn) - O(1) （不支持修改）

分块 O(n) - O(√n)

线段树 (Segment Tree)　O(nlogn) - O(logn)

稀疏表

- 稀疏表 (SparseTable) 算法复杂度为：O(nlogn) - O(1)
- ST算法预处理：用dp[i, j]表示从i开始的，长度为2^j的区间的最小值，则有递推式
- dp[i, j] = min{dp[i, j - 1],dp[i + 2^(j - 1), j - 1]}
- 即用两个相邻的长度为 2^(j - 1) 的块，更新长度为 2^j 的块。
- 因此，预处理时间复杂度为 O(nlogn)。

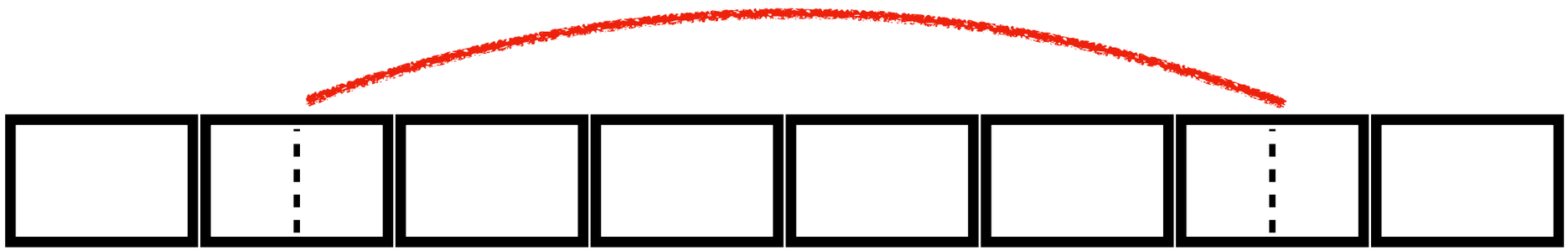
稀疏表

- 对于一段区间 [l, r]
- 我们取 $k = \lfloor \log_2(r - l + 1) \rfloor$
- 查询[l, r]：分成两个区间[l, l + 2^k - 1]和[r - 2^k + 1, r]
- query[l, r] = min(dp[i][k], dp[r - 2^k +1][k])

分块

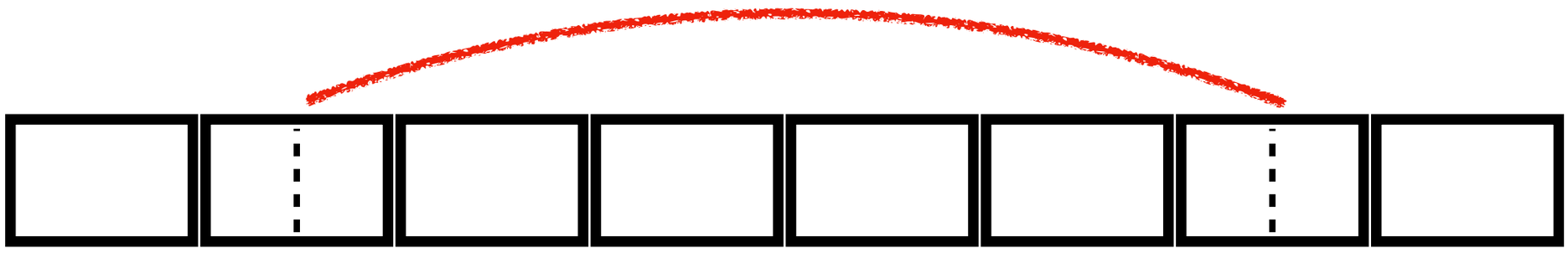
- 我们将数列划分成若干个不相交的区间，每个区间称为一个块，块的大小一般是 $\sqrt{n}$

- 一次区间操作会跨越多个块，其中区间两端点所在的块只有一些部分在区间内



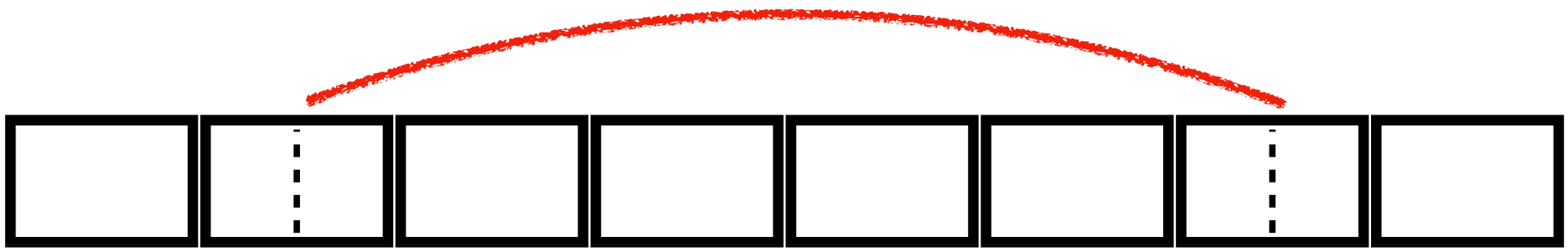
分块

- 每个块内维护最小值
- 对于每个询问，两头的块暴力，中间的块每个只需要 O(1) 查询最小值
- 支持修改，修改一个元素需要重新求它所在块的最小值



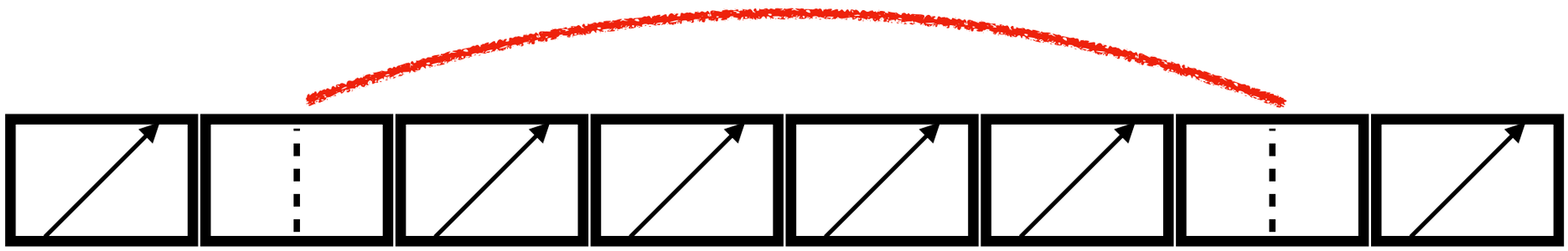
例题. 分块入门 2

- 一个长为 n 的数列 A，有 m 个操作
- 操作分为：
- 1. 把某个区间的元素都加上一个数
- 2. 询问区间内小于某个值的元素个数
- 1 <= n, m <= 50000, 0 <= Ai <= 10^9



例题. 分块入门 2

- 将分块后的序列 A 中每一块的元素排序得到序列 B
- 在每一块里就可以用二分查找查询小于某个数的元素个数
- 注意两头的块要在 A 中查询



例题. 线段树

- 有一个长度为 n 的序列，a1, a2, …, an。现在执行 m 次操作，每次可以执行以下两种操作之一：

- 1. 将数列中的某个数 ai 修改为 v。

- 2. 询问一个下标区间 [l, r] 中所有数的和。

例题. 线段树

- 暴力做法，直接用数组维护

修改：O(1)

查询：枚举下标求和 O(n)

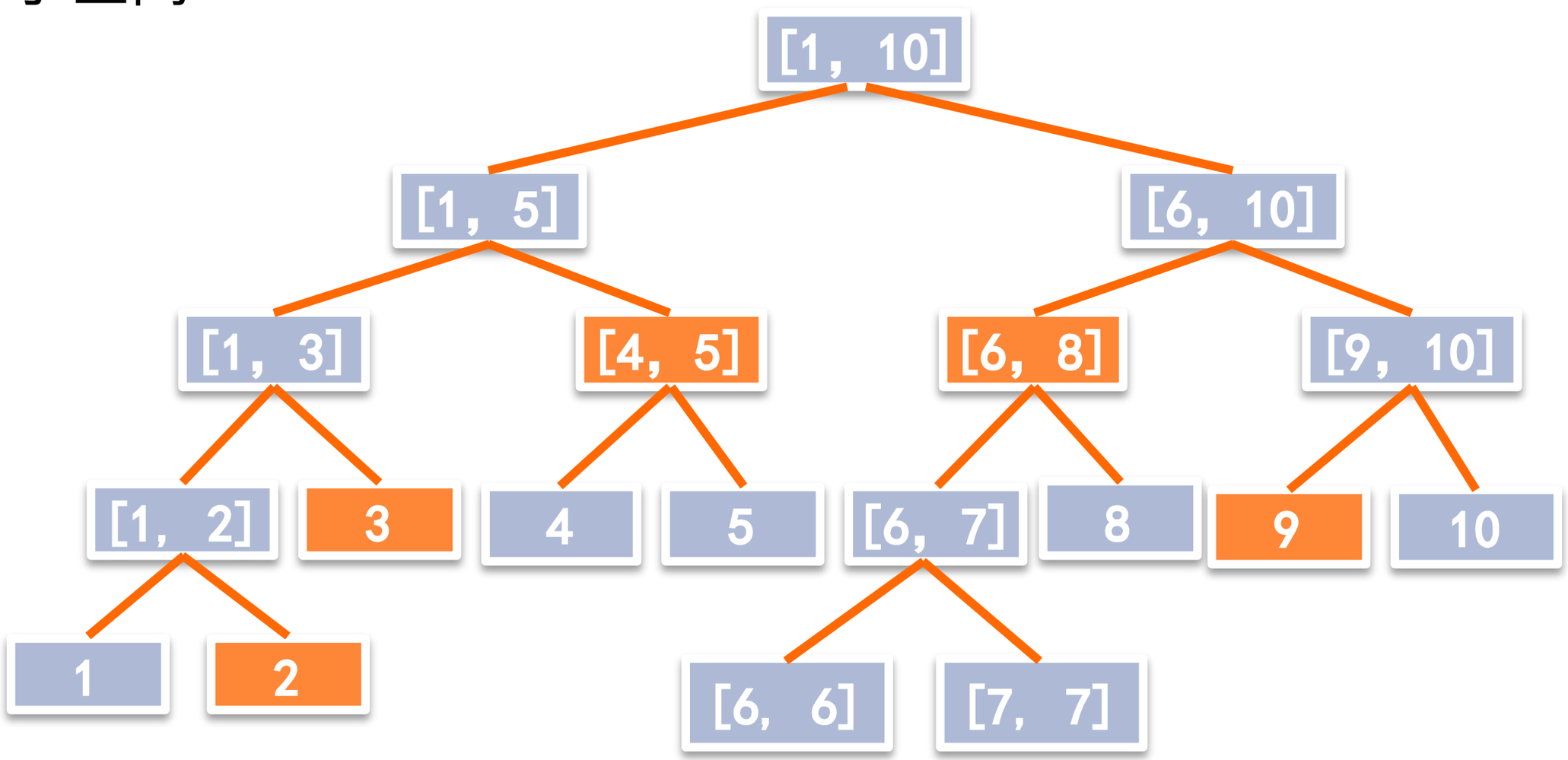
- 如果用分块的思想，维护每一块的和即可

- 进一步，我们引入线段树，通过把区间组织成树形结构来进一步加速

例题. 线段树 - 结构

- 如下是区间为 1 - 10 的线段树

- 当我们需要对一个区间进行操作时，同样将其划分成若干个小区间



例题. 线段树 - 结构

- 每个节点维护一个闭区间[l, r](l <= r)的信息
- 根节点表示[1, n]的信息
- 如果 $\mathrm{l} = \mathrm{r}$ 那就是叶子结点
- 如果 l < r 那就是内部节点，它有两个子节点 [l, mid], [mid+1, r], (mid = (l + r) / 2)

例题. 线段树 - 结构

- 用 1 表示根节点，下标为 x 的点的左子节点下标为 2x, 右子节点 2x+1。

- 用 sum[x] 表示 x 代表的区间里所有数的和。

- 对于叶子结点 x，它代表 [l, r](l = r)，sum[x] = a[l]

```
void update(int x){

  sum[x] = sum[x * 2] + sum[x * 2 + 1];

}
```

例题. 线段树 - 查询

- 假设询问区间是 [A, B]，现在所在的节点表示的区间为[l, r]
- 计算 mid = (l + r) / 2，左子节点的区间为 [l, mid]，右子节点的区间为 [mid+1, r].
- 如果 A <= mid，即询问区间与左子节点有重合，需要递归到左子节点。
- 如果 B >= mid + 1，即询问区间与右子节点有重合，需要递归到右子节点。
- 递归完之后，需要把两个孩子询问的结果加起来作为返回值。

例题. 线段树 - 修改

- 由于修改是对单个元素进行修改。
- 比如修改第 i 个元素。
- 我们先找到 [i, i] 所在的节点，然后修改它的 sum，然后一路向上更新每个祖先的 sum 即可。

树状数组

- 与线段树类似，树状数组也是用于动态查询区间信息、支持修改的数据结构。

- 与线段树相比，它更简洁，实现起来也更方便，但是它能维护的信息比较有限。

- 树状数组是在 1994 年由 Peter M. Fenwick 在他的论文《A New Data Structure for Cumulative Frequency Tables》中率先提出的，所以又称 Fenwick Tree。

树状数组

- 树状数组处理的问题一般有如下形式：
- 给定一个数组 a[1..n]，支持以下两种操作：

1. 修改：给 a[i] 加上 v

2. 查询：询问 a[1] + a[2] + … + a[i]

树状数组的思想

- 每个正整数都可以表示为若干个 2 的幂次之和（二进制分解）。

- 类似的，每次求前缀和，我们也希望将区间 [1,i] 分解成 log2 i 个子集的和。

- 也就是如果 i 的二进制表示中如果有 k 个 1，我们就希望将其分解为 k 个子集之和。

树状数组的思想

基于这种思想，我们可以构造一张表格：

内容代表该“子集”所包含的 a 数组的元素。

例如，下标为 8 的子集包含了 a[1..8]，而下标为 5 的子集只包含 a[5]。

下标	1	2	3	4	5	6	7	8	9	10	11	12
内容	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12

树状数组的思想

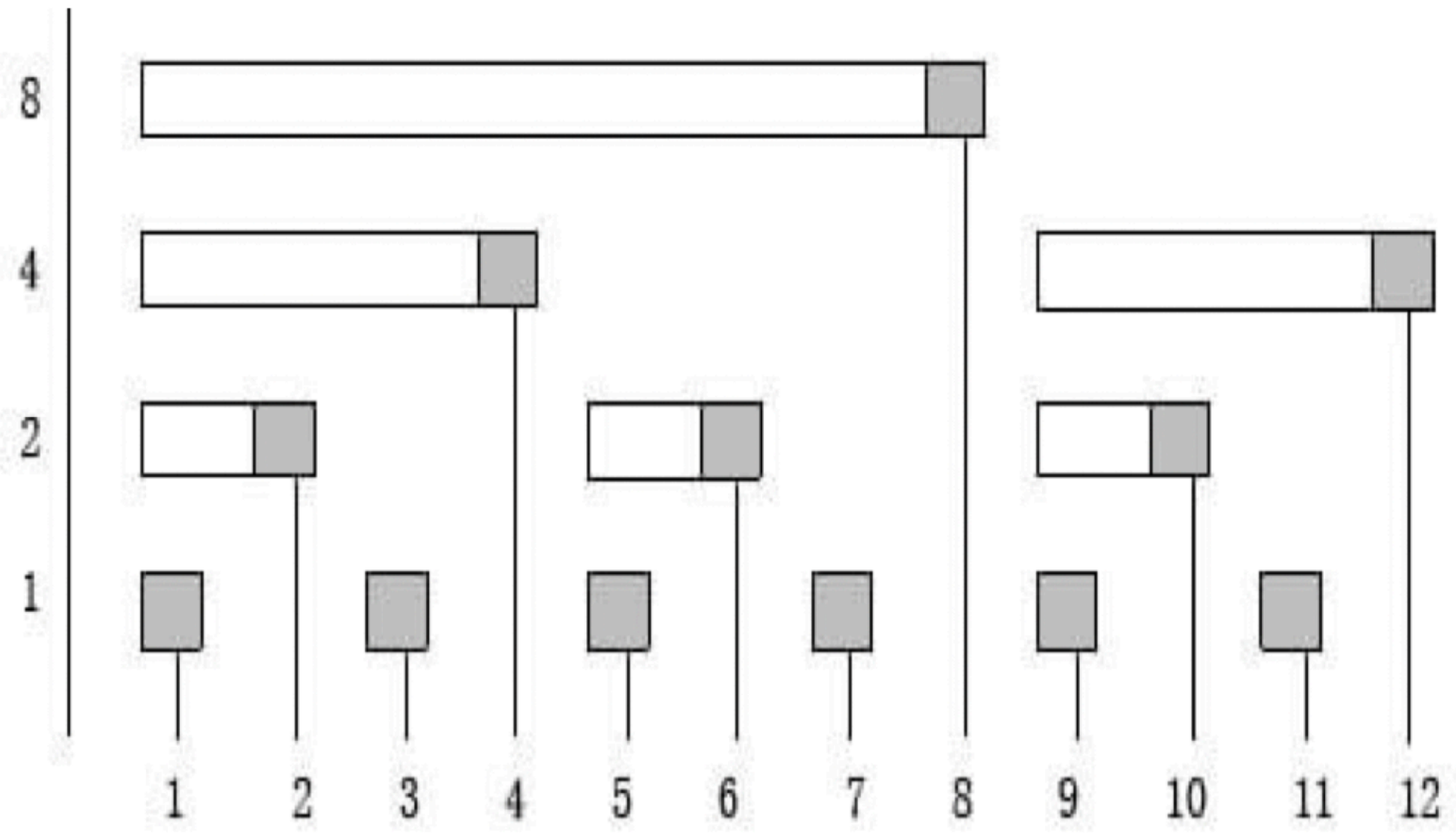
- 下表就是一个实际的例子：
- 用 sum 来表示子集和。
- 比如求 7 的前缀和，只需要计算 sum[7] + sum[6] + sum[4]
- 求 10 的前缀和，只需要计算 sum[10] + sum[8]

下标	1	2	3	4	5	6	7	8	9	10	11	12
a数组	1	2	0	1	0	0	2	2	1	2	1	3
前缀和	1	3	3	4	4	4	6	8	9	11	12	15
sum	1	3	0	4	0	0	2	8	1	3	1	7

树状数组的思想

- 我们可以用下图来更直观的理解：

- 深色方块代表自己下标对应的值 a[i]，浅色代表还要维护的别的下标对应的值 a[k .. i-1]



树状数组的思想

- 现在留下的问题就是，子集要如何划分，观察下表：

下标	1	2	3	4	5	6	7	8
下标的二进制	1	10	11	100	101	110	111	1000
内容	1	1..2	3	1..4	5	5..6	7	1..8
元素个数	1	2	1	4	1	2	1	8
元素个数的二进制	1	10	1	100	1	10	1	1000

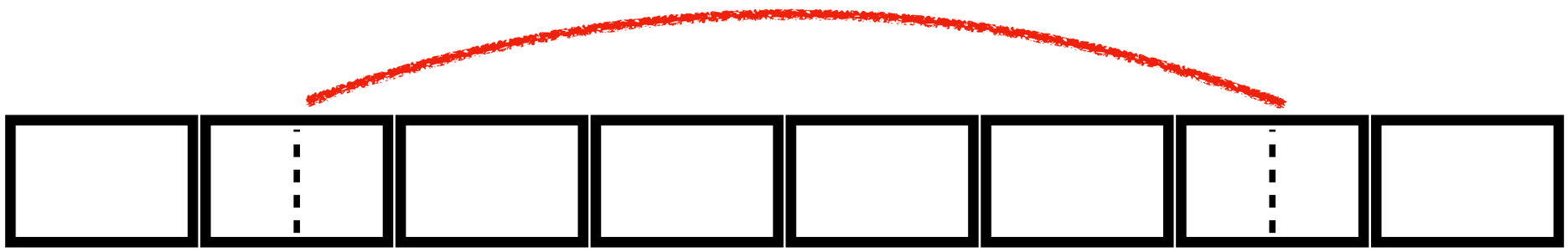
- 可以发现，元素个数的二进制就是下标的二进制表示中最低位的1 所在的位置对应的数。

树状数组的思想

- 如何求一个数 x 的二进制最低非0位对应的数?
- Lowbit(x）函数!
- C(x) = x - (x and (x - 1))
- x and (x - 1) 将 x 最低位的 1 以及后面所有的 0 都变成了 0, 然后再被 x 减去，这就是我们要的数。

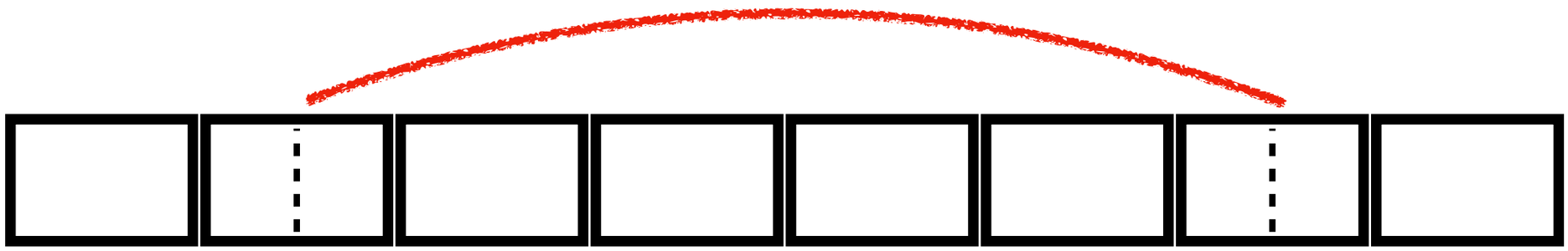
例题. 分块入门 4

- 一个长为 n 的数列 A，有 m 个操作
- 操作分为：
- 1. 把某个区间的元素都加上一个数
- 2. 询问某个区间的元素和
- 1 <= n, m <= 50000, 0 <= Ai <= 10^9



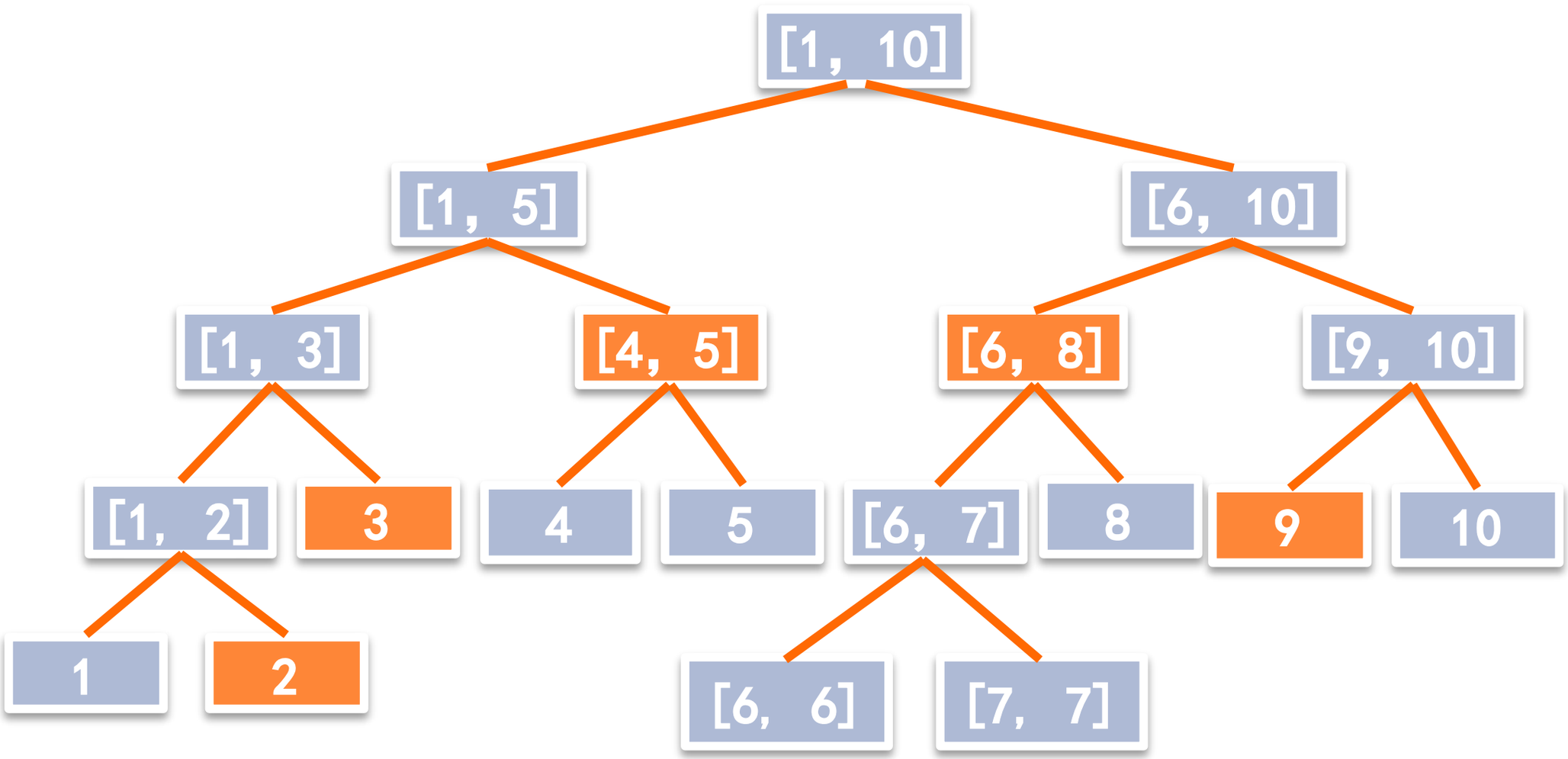
例题. 分块入门 4

- 怎么处理区间修改?
- 两头的块直接修改，并更新元素和
- 完整的块打上一个 +x 标记，在询问的时候考虑一下即可



例题. 分块入门 4 - 线段树

- 线段树也可以引入懒标记的做法
- 修改区间 [2, 9]，则在橙色结点打标记



例题. 分块入门 4 - 线段树

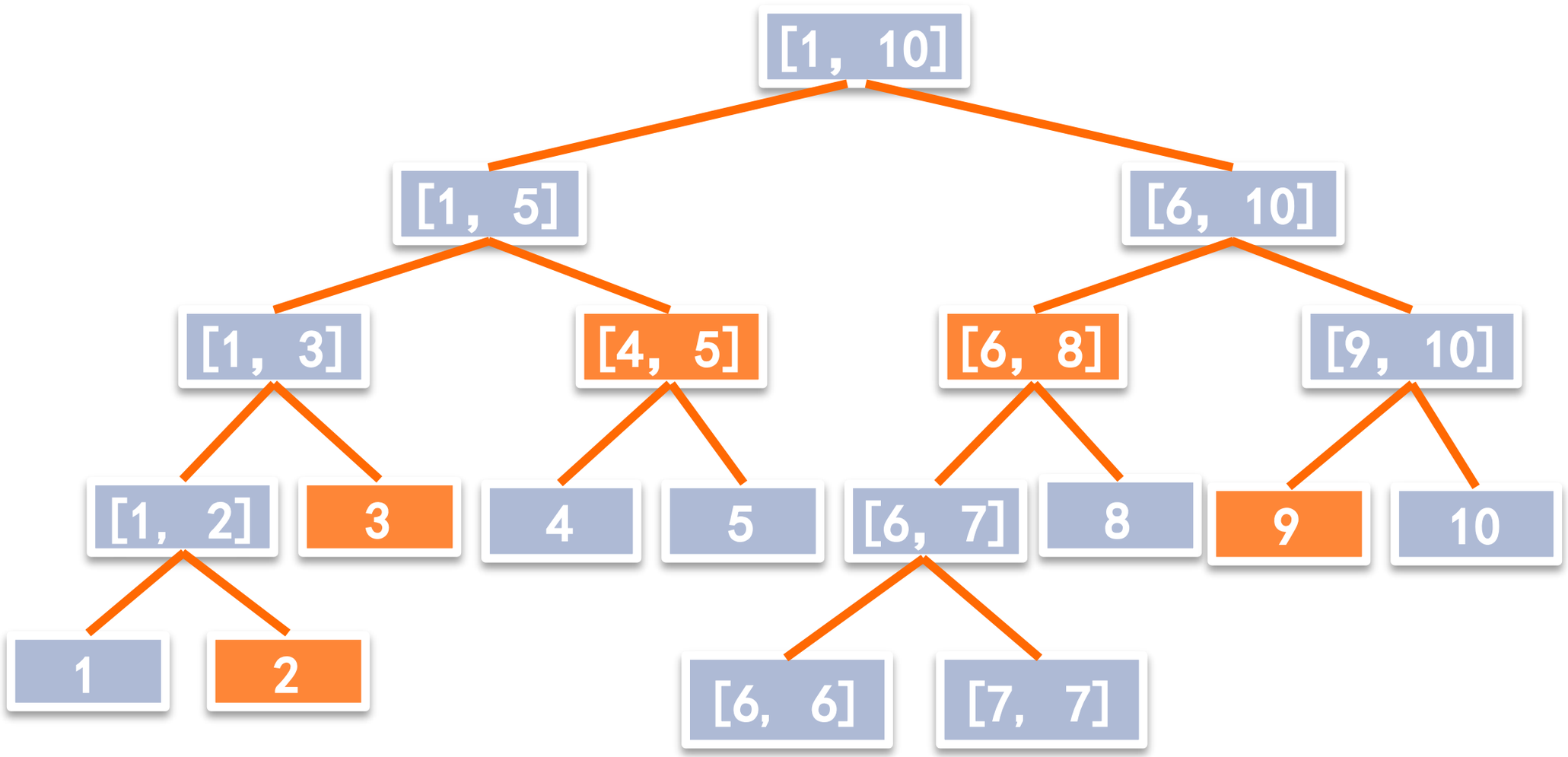
- 需要注意的是，如果有查询操作进入了一个有标记的块
- 有两种解决方法

标记永久化：查询的过程中，记录前驱结点的标记

标记下传：把标记分裂下传至结点的两个儿子

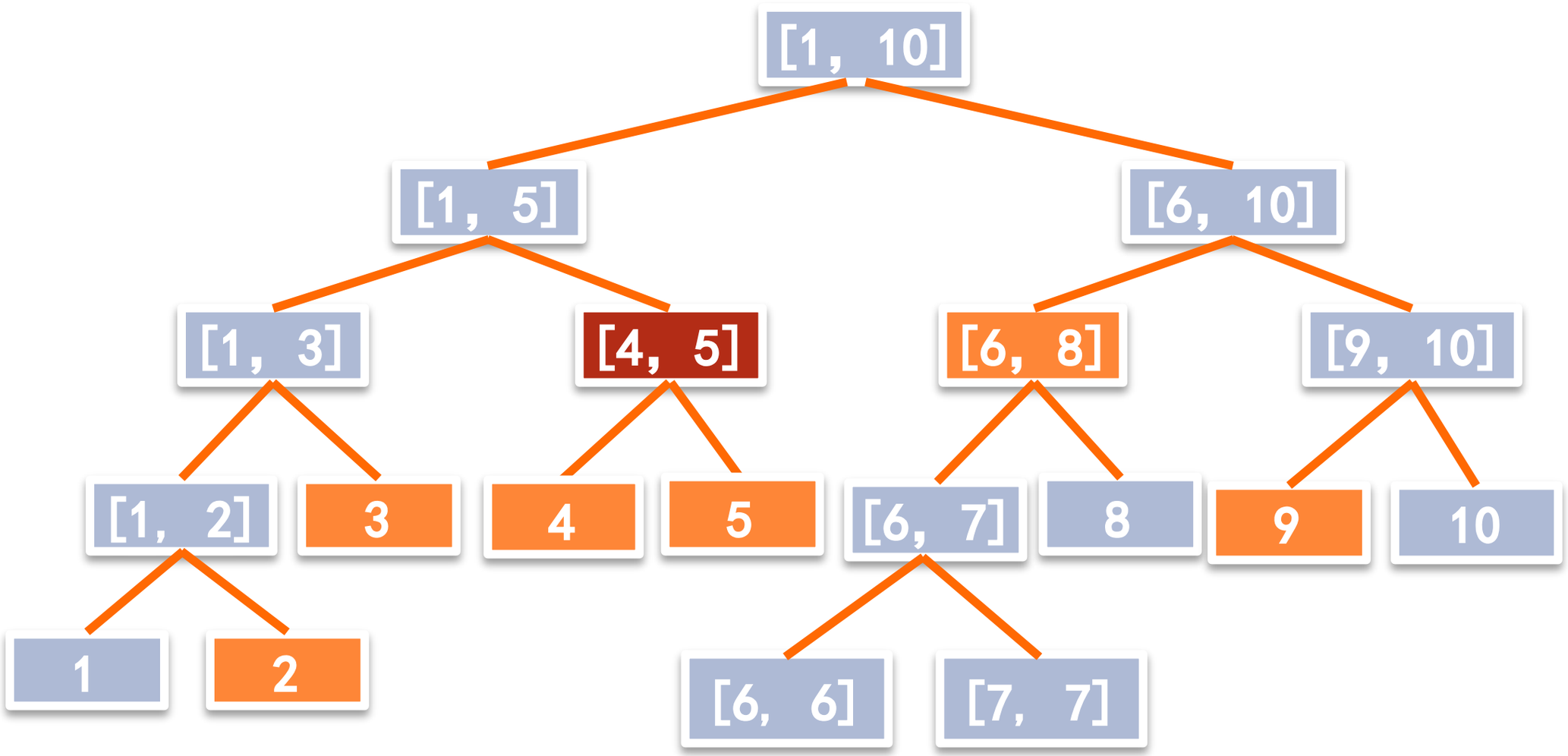
例题. 分块入门 4 - 线段树

- 只要查询不到 [4, 4], [5, 5], [6, 7], [8, 8]， 我们就不用递归下去修改它们的 tag 和 sum。



例题. 分块入门 4 - 线段树

- 如果查询 [5, 5]，需要把 [4, 5] 的标记清空，传给 [4, 4] 和 [5, 5]。



搜索树介绍

- 二叉搜索树(BinarySearch Tree，简称 BST)是具有下列性质的二叉树

1) 若它的左子树不为空，则左子树上所有结点的值均小于它的根结点的值;

2) 若它的右子树不为空，则右子树上所有结点的值均大于它的根结点的值;

3) 它的左、右子树也分别为二叉查找树。

搜索树介绍

- 二叉搜索树(BinarySearch Tree，简称 BST)是具有下列性质的二叉树

1) 若它的左子树不为空，则左子树上所有结点的值均小于它的根结点的值;

2) 若它的右子树不为空，则右子树上所有结点的值均大于它的根结点的值;

3) 它的左、右子树也分别为二叉查找树。

搜索树的构建

- 对已经排序的数快速递归构建二叉搜索树， 可以使用分治

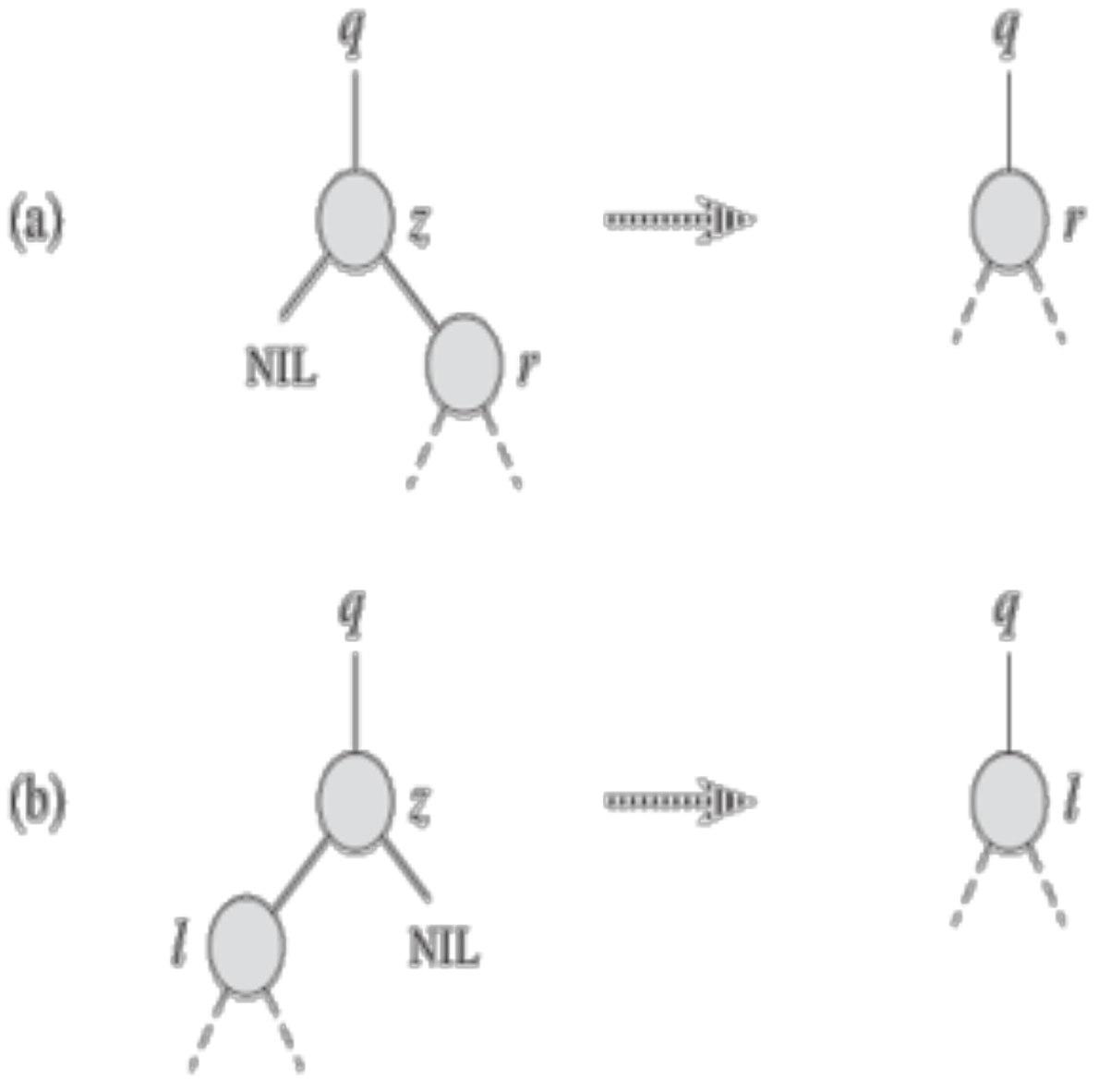
- 在二叉查找树中插入新结点，要保证插入新结点后仍能满足二叉查找树的性质)

1) 若二叉查找树 root 为空，则使新结点为根

2) 若新结点的关键字小于插入点的关键字，则将新结点插入到插入点的左子树中，大于则插入到插入点的右子树中

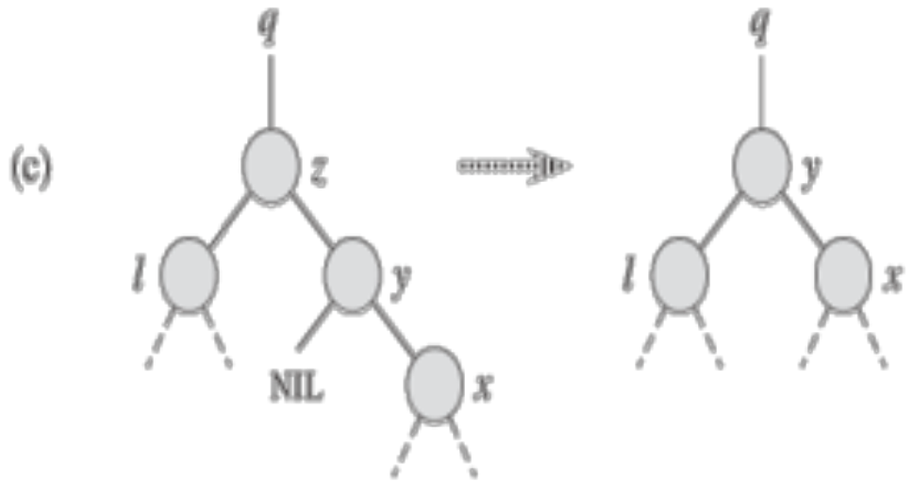
搜索树的删除

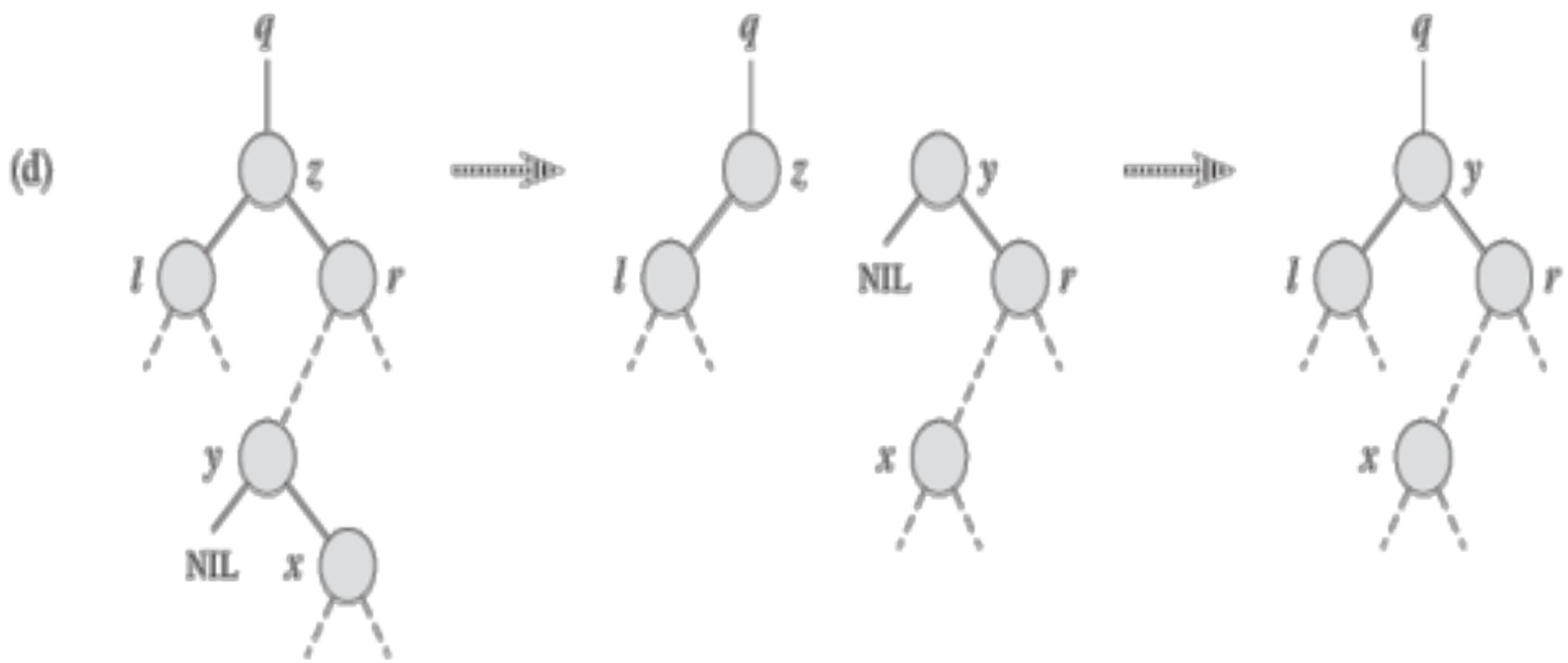
考虑懒惰删除，或者分类讨论



搜索树的删除

考虑懒惰删除，或者分类讨论





搜索树的效率

- 我们已经知道，二叉搜索树上的各基本操作的运行时间都是O(h)，h为树的高度。

- 但是随着元素的插入或删除，树的高度会发生变化。

- 例如，如果各元素按严格增长的顺序插入，那么构造出的树就是一个高度为 n - 1 的链。

- 如果各元素按照随机的顺序插入，则构造出的二叉查找树的期望高度为 O(log n)。

- 当看见题目中出现“数据是随机构造的”时，要能够记起这个结论哦！

平衡树

二叉搜索树会有退化问题，我们希望树尽量平衡，不要左偏或者右偏。

介绍一种简单的平衡树：朝鲜树，其特色就是使用者可以指定一个值，当整棵树的深度大于 K 时就重建这颗树，一般可以把 K 取为√n，这样 n 次插入的复杂度是 O(n√n)。

如果再引入局部重构，就可以得到一个标准的替罪羊树，略。

一般情况下，用 set 可以应付大部分题目。

习题第二部分

466C.Number of Ways

- 给定长为 n 的序列 ai，将其分成 3 个连续子串，要求每个子串的和相同，求划分的方案数

- $1 \le n \le 10^5, -10^9 \le a_i \le 10^9$

466C.Number of Ways

- 求出前缀和，后缀和，整个序列的和 sum
- 找出等于 sum / 3 的前缀和，求其后有多少个 sum / 3 的后缀和
- 可以通过预处理或者，或树状数组 / 线段树在线维护查询

带修改最大字段和

- 给定长为 n 的序列，有 m 次操作，每次操作可以改变序列中某个数字的值，让你在每次操作后输出最大子段和。

- 1 <= n, m <= 10^5

带修改最大字段和

- 假设区间1，分成两段区间2和3。如果 2 和 3 的答案已经算出来了，则 1 的答案为 2 或者 3 的最大答案；或者 2 区间从右边开始的最大子段和，加上 3 从左边开始的最大子段和。
- 这样每个线段树节点就只需要维护三个值

1：最大答案 ans

2：从右边开始的最大答案和 ansl

3：从左边开始的最大答案和 ansr

528A.Glass Carving

- 给一块 w * h 的玻璃，m 次操作
- 每次操作，给出一条平行于坐标轴的直线切割玻璃，询问切割后剩下的最大面积的完整玻璃

- 1 ≤ m, w, h ≤ 10^5

528A.Glass Carving

- 行列分开，用 set 维护横纵坐标差值的最大值

CTUO2015.Lunch Menu

- 你有 L 块钱，要去某食堂买 4 道菜，一个主菜，一个甜品，一个饮料，一个汤，每类都各有 n 种，价格不同。
- 你每天都不能和之前某一天吃的完全一样，问你最多能去这个食堂吃几天?
- 1 ≤ n ≤ 2000, 1 ≤ L ≤ 10^7

CTUO2015.Lunch Menu

- 枚举前两个数，得出 O(n^2) 个数字后排序

- 后两个类似，然后两个序列 A B，排序后扫一下，即对于 B 序列的每一个元素，A 的一个前缀可以和它匹配

2017 乌鲁木齐 G.Query on String

字符串 S，T，有 n 个操作

1. 询问 S[l...r] 中有多少个 T （重叠的也算）
2. 修改 S 的某一位1 ≤ |S| ≤ 100000, 1 ≤ |T| ≤ 10,

2017 乌鲁木齐 G.Query on String

- 预处理后变成区间和查询，修改至多影响 10 个值，树状数组维护。

CF497C. Distributing Parts

- 有 n 个曲子，每个曲子的音域范围为 [ai, bi]。有 m 个演奏家，每个演奏家能演奏的音域范围为 [ci, di]，可以出演 ki 次。

- 如果 ci ≤ ai ≤ bi ≤ di，则说明该曲子可以由演奏家演出 。

- 构造一个方案使得所有曲子都被演奏，无方案输出”NO” 。

- 1 <= n, m <= 10^5

CF497C. Distributing Parts

- 把演奏家和歌曲一起按右端点排序，用一个 set 维护所有歌曲。
- 在某个时刻，考虑演奏家演奏哪一首歌曲，贪心策略是他选择最“难”的那一首，即左端点最左的那一首歌。

339D.Xenia and Bit Operations

- 给定长度为 n = 2^k 的序列，第一次操作将相邻的 n / 2 对数字进行或运算，保留这 n / 2 个结果

- 第二次操作将相邻的 n / 4 对数字进行异或运算，以此类推

- 有 m 次询问，每次修改原序列一个数字，问进行操作以后剩下的最后一个数字

- $1 \le k \le 17, 1 \le ai \le 2^{30}, 1 \le m \le 10^5$

339D.Xenia and Bit Operations

- 线段树模拟
- 每次询问可以自底向上修改

BZOJ 2120. 数颜色

- n 的小球排成一排，第 i 个颜色为 ci。
- m 个询问，每次询问从第 L 支画笔到第 R 支画笔中共有几种不同颜色的画笔。
- n, m <= 50000

BZOJ 2120. 数颜色

- 用 pre[i] 记录前一个和 i 相同颜色的球的所在位置
- 询问 l 到 r 时，如果 pre[i] < l 说明在 l 到 r 这一段没有和 i 颜色 相同的球，则 ans++
- 即每次在 pre 序列中询问 [l, r] 有多少小于 l 的元素
- 利用这种思路我们可以套用之前介绍的分块

BZOJ2002. HNOI2010. Bounce 弹飞绵羊

- 有 n 个装置，第 i 个装置的弹力系数是 k_i，即从 i 会弹射到 i + k_i，如果 i + k_i > n，称绵羊被弹飞。
- 有 m 个操作：

修改某个装置的弹力系数。

询问从某个装置出发，要几次弹射后被弹飞?

- n, m <= 50000

BZOJ2002. HNOI2010. Bounce 弹飞绵羊

- 弹力装置组成一个树结构，题目中要求维护一个动态的树结构。
- 每个点记录跳出分块的步数以及跳到下一分块的哪个点。
- 询问暴力一块一块跳，修改就把所在块内暴力一下。

CF718D. Andrew and Chemistry

- 给定一棵树，每个结点的度数小等于 4
- 要在树上加一个结点，加完后度数依然小于 4
- 问有多少不同构的加法
- $1 \le n \le 10^5$

CF718D. Andrew and Chemistry

- 树的同构有经典的哈希做法，我们设法通过预处理，快速求得（在每个结点上加上一个结点后，新树的哈希值）

- 在树形递推的时候加上记忆化，mp(x,y) 表示 y 的子树 x 的哈希值

- 由于树的度数很小，可以直接把每个结点 x 的儿子的哈希值一起放在 vector 里排好序，再把 vector 放到 map 里编 x 的哈希值

212D.Cutting a Fence

- 给定长为 n 的序列 Ai，求长度为 i 的区间的最小值的平均值，i 从 1 到 n。

- 1 ≤ n, Ai ≤ 10^6

212D.Cutting a Fence

- 维护不同区间长度的答案

- 枚举 i，设 a[i] 与其左边第一个小于它的数的距离为 l，与其右边第一个小于它的数的距离为 r，则 i 对各种长度的区间的答案的贡献是分段函数

- 可以拆分为 2 个等差数列和一个常数数列，使用 2 次前缀和统计即可

- 时间复杂度 O(n)

小奇的糖果

- 有 n 个彩色糖果在平面上。小奇想在平面上取一条水平的线段, 并拾起它上方或下方的所有糖果。

- 求出最多能够拾起多少糖果, 使得获得的糖果并不包含所有的颜色。

- $1 \leq n \leq 10^5$

小奇的糖果

- 维护树状数组实现查询横坐标在一段区间内的点的个数。 维护双向链表实现查询一个点左 (右) 边横坐标最大 (小) 的与它颜色相同的点。

- 不妨只考虑取线段下方的点的情况

- 考虑没有取到的颜色 x, 找出所有不包含这种颜色的区间, 更新答案 ans[x]。

- 按照纵坐标从大到小枚举所有的点, 分别在树状数组和双向链表中删除当前点, 并利用这个点左右两边和它颜色相同的点之间的区间内点的个数更新答案。