线段树的进阶训练

2025 CEIT 算法集训队暑期集训 #1

Orange

沈阳师范大学，人工智能学院，CEIT LAB

2025/07/19

目录

先来一道简单题热身

掉落的方块

在二维平面上的 x 轴上，放置着 n 个方块。
给你一个二维整数数组 positions ，其中 positions[i] $=$ [$left_i$, $sideLength_i$] 表示：第 i 个方块边长为 $sideLength_i$ ，其左侧边与 x 轴上坐标点 $left_i$ 对齐。
每个方块都从无穷高处掉下。方块沿 y 轴负方向下落，直到着陆到**另一个正方形的顶边或者是 x 轴上。一个方块仅仅是擦过另一个方块的左侧边或右侧边不算着陆**。一旦着陆，它就会固定在原地，无法移动。
在每个方块掉落后，回答目前所有已经落稳的方块堆叠的最高高度。
数据范围：$sideLength_i \leq 10^6, left_i \leq 10^8, n \leq 10^5$

先来一道简单题热身

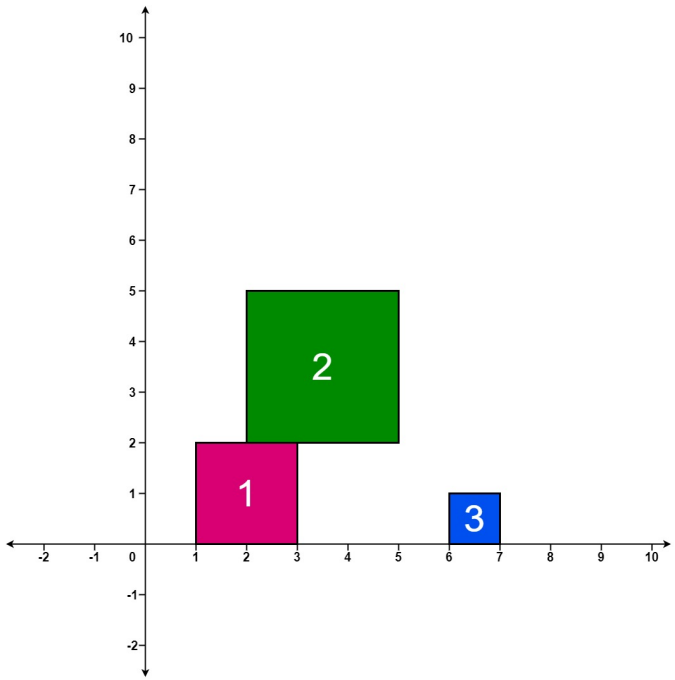


图: 示例图

输入: positions = [[1,2],[2,3],[6,1]]
输出: [2,5,5]

思路

抽象题意：每次给出一个区间 $[l, r]$，首先获取到区间最大值 $V = \max[a_l...a_r]$，然后将区间 $[l, r]$ 全部更新成 $V + x$，在执行完本次操作后，报告全局最大值。

注意！

本题提到了擦边的方块会顺着继续落下而不是停止下落，因此可以考虑将右边界 r 收缩，即变成 $[l, r)$ 也就是 $[l, r-1]$。

相信你很容易发现本题可以使用线段树维护，即实现区间赋值和求区间最值。但是需要注意的是，假设你那样做的话，我们需要维护整个可能被方块落下的数轴部分，即最大可能达到 $10^8 + 10^6$，这显然会 MLE。

思路

注意到：我们对于一个方块，实际上只有 2 个端点是重要的关键点，因此一个方块可以简化成 2 个关键点，我们最多只会有 $2n$ 个关键点，也就是最多 2×10^5 个数需要维护。
考虑对方块的关键点进行**离散化**，再维护即可。
至此，本题很轻松的结束了 $\sim$ 相信你一定从本题上找回了自信 xD。

第二个稍微难一点的热身题

插花

一共 n 个花瓶，一开始所有花瓶都没有花，每个瓶子最多插 1 朵花。维护如下两种操作 m 次：

1. 插花：给定 x 朵花，并给定位置 p，从第 p 个花瓶开始，向右依次插花，遇到有花的瓶子跳过，没有的则插入一朵。若到了最后一个瓶子，花还没用完，则忽略剩下的花。插花后报告第一个插入花和最后一个插入花的瓶子。特别的，若从 p 开始到最后无一个空瓶，则报告-1。
2. 制作花束：给定范围 $[l, r]$，将 $[l, r]$ 中所有的花取出做成一束花束，该操作会清空范围内所有的花瓶。报告花束大小，即包含多少朵花。

数据范围：$n, m \leq 10^5$

第二个稍微难一点的热身题

Input	Output
10 5 1 4 5 2 5 6 1 2 8 2 4 7 1 9 8	4 8 2 2 10 4 -1

表: example

思路

题意：我们用 0/1 表示花瓶内是否有花，现在题目转化成：

1. 插花：从 p 开始，找到最小的左边界 l，使得 $[p, l]$ 内 0 的个数等于 1；找到最小的右边界 r，使得 $[p, r]$ 内 0 的个数大于 x，并将 $[l, r]$ 区间赋值为 1。
2. 制作花束：求 $[l, r]$ 的区间和，并区间赋值为 0。

区间赋值和区间求和都是线段树的常见操作，本题主要在于如何维护插花中的查找 0 的左右边界。
注意到：0 的个数 = 区间长度-区间求和，且 0 的个数随区间长度具有单调性，即固定左边界 p，对应一个右边界 q，$[p, q]$ 中 0 的个数与 q 呈单调递增关系，即 q 是可二分的。因此可以通过二分来解决 q 的求解，至此本题的所有 point 均被解决。
也许稍微难一点的题，也没那么难，对吧 xD

线段树维护复杂信息

线段树维护平均数与方差

维护一个长度为 n 序列 a_i，支持 3 种操作共 m 次:

1. 给定区间 $[l, r]$ 以及 x，给 $[a_l...a_r]$ 都加上 x
2. 给定区间 $[l, r]$，求出区间 $[a_l...a_r]$ 的均值 $\bar{a} = \frac{1}{r-l+1}\sum_{i=l}^{r} a_i$
3. 给定区间 $[l, r]$，求出区间 $[a_l...a_r]$ 的方差 $D = \frac{1}{r-l+1}\sum_{i=l}^{r}(a_i - \bar{a})^2$

数据范围: $n, m \leq 10^5, x \leq 10^9$, 结果对 $10^9 + 7$ 取 mod

思路

首先对于区间均值，线段树可以用区间和/区间长度轻松维护，即只需要维护区间和即可。

考虑如何维护方差：现在，我们可以通过上面的方法很轻易地求出区间均值 $\overline{a}$，我们回来看方差的公式，考虑做出如下变形：

$$D = \frac{1}{r-l+1}\sum_{i=l}^{r}(a_i-\overline{a})^2 \tag{1}$$

$$= \frac{1}{r-l+1}\sum_{i=l}^{r}(a_i^2-2a_i\overline{a}+\overline{a}^2) \tag{2}$$

$$= \frac{1}{r-l+1}\sum_{i=l}^{r}a_i^2-2\overline{a}\frac{1}{r-l+1}\sum_{i=1}^{r}a_i+\overline{a}^2 \tag{3}$$

$$= \frac{1}{r-l+1}\sum_{i=l}^{r}a_i^2-\overline{a}^2 \tag{4}$$

思路

$$D = \frac{1}{r-l+1}\sum_{i=l}^{r} a_i^2 - \bar{a}^2$$

我们得到了一个关于方差计算如此简单的柿子，观察其可以发现，我们最终只需要维护一个 a_i^2 的和即可，也就是每个项的平方和。
我们现在已经知道了需要维护哪些信息以及如何计算答案，那么如何让信息去适应我们的修改操作呢？

维护修改操作

考虑修改对我们维护信息的影响：
我们维护了区间和

$$sum = \sum a_i$$

以及区间平方和

$$sumQ = \sum a_i^2$$

对于一次修改，给区间所有的数加上 x，则有下面的式子：

$$sum' = \sum (a_i + x) = \sum a_i + len \times x = sum + len \times x$$

$$\begin{aligned} sumQ' &= \sum (a_i + x)^2 = \sum (a_i^2 + 2a_i x + x^2) \\ &= \sum a_i^2 + 2x \sum a_i + len \times x^2 \\ &= sumQ + 2x \times sum + len \times x^2 \end{aligned}$$

是时候 Ac 了！

现在，根据上面我们推导的到公式，我们很容易即可维护所有的信息，包括修改产生的影响以及答案的计算，本题至此结束。
通过上面的过程，不难发现，线段树本质上只是提供了一个快速合并区间的功能，区间信息的设计与合并才是题目真正的难点，也就是其实**线段树的题目本质上还是推公式**。

经典问题的引入

区间最大子段和

给定一个序列 a_i，维护如下两种修改：

1. 单点修改：$a_i \leftarrow x$
2. 区间查询 $[l, r]$ 的区间最大子段和 $\max_{l \le i \le j \le r}(\sum_{k=i}^{j} a_k)$。

对于上面这个经典问题，我们考虑维护每个区间的：区间和 sum，区间最大前缀 pre，区间最大后缀 suf，以及区间最大子段和 tmax。

也就是说，我们通过类似 dp 的思想，从区间合并的角度，考虑 tmax 可能会从哪些可能的答案转移过来，然后再维护转移过程需要用到的中间变量，最后完成信息的维护。

我们把上述问题总结称为 **线段树的区间合并问题**。

一个之前题目的变种

010101 II

给定一个 01 序列 a_i，维护如下两种修改：

1. 区间翻转：对于 $a[l...r]$，$a_i \leftarrow a_i \oplus 1$。
2. 区间查询 $[l, r]$ 的最大交替子串长度。

数据范围：$n \leq 10^5$

分析

首先，我们按照区间合并问题的标准思路，对于某个性质，肯定需要维护：

1. 当前区间的性质
2. 当前区间前缀的性质
3. 当前区间后缀的性质
4. 其他辅助中间变量...

按照上面的思想，考虑分别维护：tmax 当前区间最大交替子串长度,pre 当前区间最大交替前缀长度,suf 当前区间交替后缀长度。

如何进行向上合并信息 (pushup)？

我们依次考虑每一种信息的维护情况 (以下假设 L,R 分别代表左右子树，不带前缀的变量则表示当前区间的变量)：
- 对于 tmax, 很显然，其可以由三种情况得到：左子树的 tmax，右子树的 tmax 以及左边的 suf+ 右边的 pre... 打住！这里似乎出现了问题？对于跨区间的新的答案，我们似乎得考虑左区间的最后一个字符 tail 以及右区间的最开头一个字符 head。如果其不相同，说明左右的后缀和前缀可以”交替地”连接在一起。这一点很重要，我们用一个变量 diff，记 L.tail 是否不等于 R.head。
- 对于 pre，由于其是前缀，因此显然可以由左子树的前缀 pre 得到，此外这里有一个很重要的发现，若左子树的序列不是完全交替，说明 pre 一定在左边就截断了，因此不必考虑跨区间的 pre，即 pre=L.pre，倘若左边是完全交替的，再考虑 dif，若 dif 为真，则 pre=L.len+R.pre，否则 pre 仍然为 L.pre。同理，对于 suf 同样有上面的讨论。

如何维护修改操作？

假设要反转当前区间，不难观察到：**反转不影响交替性**。
因此，事实上，tmax,suf,pre 都仅需要维持不变即可。只有 head 和 tail 需要取反。
最后使用一个 rev 的懒标记维护反转操作即可。
至此，本题也很容易的结束了。

总结

对于线段树维护区间合并类型的题目，其均有固定套路：即维护 suf，pre 和 nowans。我们考虑使用 dp 中的状态转移思想，根据转移需要设置合理的辅助变量完成答案的转移并维护好所有信息，即可轻松解决这类题目。

什么是动态开点？为什么需要动态开点？

在某些题目中，我们要使用线段树维护的区间操作，会遇到要维护的范围很大 (大多发生在权值线段树中，即线段树维护值域) 的情况。

此时我们考虑上面掉落方块一题的思想，我们只维护**关键点**即可，因为关键点之外的其他节点都是未被访问过的，默认为 0，有限的操作次数意味着关键点是有限的。我们考虑**线段树上只保留关键节点**，每次操作引入新的关键节点时，我们再实时的建立这个节点。

一个题目

线段树模板题 · 改

维护一个序列 a_i，最开始所有位置均为 0，支持如下两种操作:

1. 区间修改：给 $a[l \sim r]$ 全部加上 x。
2. 区间求和：求出 $\sum a[l \sim r]$。

数据范围：$n \leq 10^9$,$m \leq 10^4$

input	output
100 5	60
1 1 100 1	40
1 1 60 2	
1 51 80 3	
2 51 60	
2 76 100	

细节

本题在经典线段树的模板上引入了动态开点，我们考虑什么时候才需要动态开点：

- 每次修改操作时，若需要修改的点不存在，则建立
- 每次下传懒标记时，若子节点不存在，则建立

那么，代价呢？

动态开点线段树就是为了解决传统线段树无法应付如此庞大的范围而被提出的。那么动态开点线段树的实际空间需要多大？

$$2M\log_2 L$$

其中 M 表示操作次数，L 表示实际值域的长度，2 是因为我们每次修改线段树时，都是沿着区间的最左边界和最右边界一直到整个区间被完美覆盖，因此本质上就是修改了树上的两条链。

总结

开点线段树是对传统线段树的一种基于线段树特性的空间优化，由于其依赖于每次线段树操作中关键点的建立，并且对查询次数敏感，因此实际上即使优化了空间，其依旧很容易 MLE(当然现在的主流比赛内存都非常宽裕)，或者由于大常数而 TLE。
包括上述的例题的这类题显然可以用 **离散化**的思想，并且这种方法 **常数更小，内存开销也小**。上动态开点线段树，既代码难写，又在时间和空间上不讨好。

那么，动态开点有何意义？

动态开点是一种重要的编程思想，也为我们后续的题目做了铺垫。常规的大值域题目一般都用离散化可以轻松解决 (除非 sb 出题人卡强制在线)。动态开点线段树的真正目的是作为解决后续的**可持久化线段树**，**线段树的合并与分裂**以及**树链剖分**等更高难度问题的基础，因此掌握动态开点的原理，实现以及思想是很有必要的。

引入

覆盖点的最短区间

给定数轴上若干条线段，每条线段给出端点 $[l_i, r_i]$，并给出若干个询问，每次询问一个点 p，回答包含点 p 的最短的线段的长度。数据范围：$n, m \leq 10^5$ $l_i, r_i, p \leq 10^9$

input	output
4 4	3
1 4	3
2 4	1
3 6	4
4 4	
2 3 4 5	

思路

很显然，这个题不需要用线段树来维护。我们可以从本题的思路中学习到扫描线的思想：**考虑从左到右扫描数轴，随着扫描的推进不断同步线段信息并计算答案**。我们先离线所有的线段和询问，按左端点/询问点排序，利用堆维护每个线段的长度和右端点。根据此时扫描到的点 i，首先处理堆，对于每个堆顶，若 i 已经超出了其右边界，说明这个线段已经失效，则将其弹出。此时考虑当前是否有询问问的刚好是 i 这个点，若有，则堆顶就是答案。

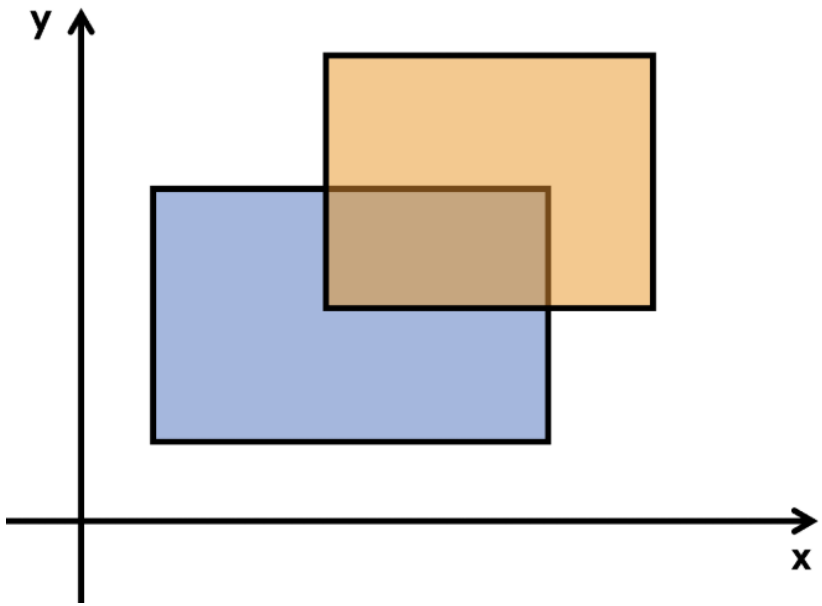
扫描线的本质

扫描线本质上是**处理区间的边界合并问题，即考虑边界的影响**。本题较为简单，即可以直接利用堆来维护边界和答案。堆本质上其实是利用到了**懒删除**的思想，因为答案只跟当前含有某种性质的区间有关，所以我们只需要维护这种性质的区间的边界即可，即该区间何时失效或者何时引入新的区间。而更为复杂的边界处理就需要引入线段树来维护了。

一个更加复杂的题目

面积求并

在一个无限大的二维平面上，存在 n 个矩形，给出每个矩形的四个顶点坐标，求出这些矩形面积的并。
数据范围：$x, y \leq 10^9, n \leq 10^5$



图：示例

思路

还是扫描线的思路，由于给定的矩形是一个二维的信息，我们可以考虑顺着一个维度进行扫描，用数据结构来维护另一个维度上的信息。
考虑沿 x 轴进行扫描，那么其实每个矩形就变成了根据 x 坐标结构而成的两条 y 轴上的直线，这就是需要维护的关键点，其中一条直线表示开始，另一条表示结束。我们考虑用线段树维护 y 轴，每次加入一条新的矩形边，如果是开始边，那么就对应的区间上 $+1$，如果是结束边则-1. 而且每次我们都可以将当前区间贡献累加到总答案中，即两个关键点之间 x 的坐标之差乘上 y 轴实际上非 0 区间所代表的实际覆盖长度。这样，本题即可结束。

一些实现上的细节

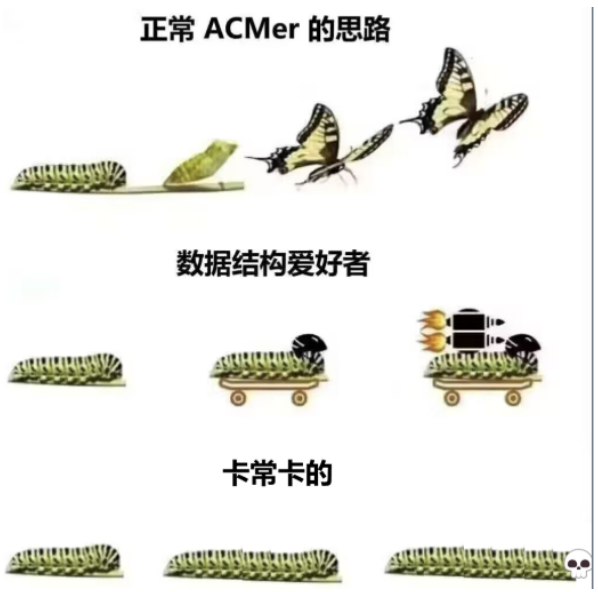
首先，本题范围很大，因此需要考虑对所有的 y 坐标进行离散化。如果你直接打线段树区间修改 + 区间查询非 0 元素，这样不仅难以维护信息，同时对离散化部分的处理更是需要进行较为细致的分讨，还需要维护懒标记的下传。

考虑线段树维护区间总长度 L，区间被覆盖长度 w 以及区间被覆盖次数 cnt，每次修改时，若区间已经被完全覆盖，则当前区间被覆盖次数 $+1$，此时 $w \leftarrow L$，若当前区间没有被完全覆盖，则递归处理左右子节点区间，同时当前的 w 由子节点 pushup 得到。

上述维护，不需要维护懒标记，因为每一个修改都一定对应一个撤销操作，其完全对称，因此无需维护懒标记，只需要维护次数即可。

All in all

线段树实际上只是一个区间维护的工具，而不是一类题型/思想，在遇到实际题目时，我们实际上应该考虑到的是题目究竟需要维护什么信息/如何计算答案，在实际需要区间维护时，才使用线段树来完成。学习线段树主要是学习用到线段树的题目中，答案的思想和算法的设计，而不是线段树实现本身。**不要看到一个题目上来就无脑线段树，否则下场一定很惨**。



希望本课能对你的算法学习提供帮助 ~

Thank You!

Reference

1 掉落的方块
2 插花
3 线段树维护平均数与方差
4 覆盖点的最短区间
5 矩形面积并