F The Happy Prince and Other Tales

tag : **二维树状数组/二维线段树**

找到 $L \leq l, R \geq r, U \geq sum$ 的 sum 最大的区间 $[l, r]$。

- 首先对题面进行分析，总共只有 $O(n^2)$ 个区间，每个区间有三个信息：(l, r, sum)。
- 题目要求找到 $L \leq l, R \geq r, U \geq sum$ 的区间。
- 三维偏序？！！
- 赶快 cv R 题代码，小改一下，hahaa，一血是我的了！！！
- T了

以上是某人拿到这道题之后的完整心路历程，我不说是谁。

本题正解：二维树状数组

首先仍然需要排序降掉一维，不过这次我们按照 sum 从小到大排序。然后依次将 sum 值加入二维树状数组中，然后寻找 $L \leq l, R \geq r$ 这个范围的最大值即可。

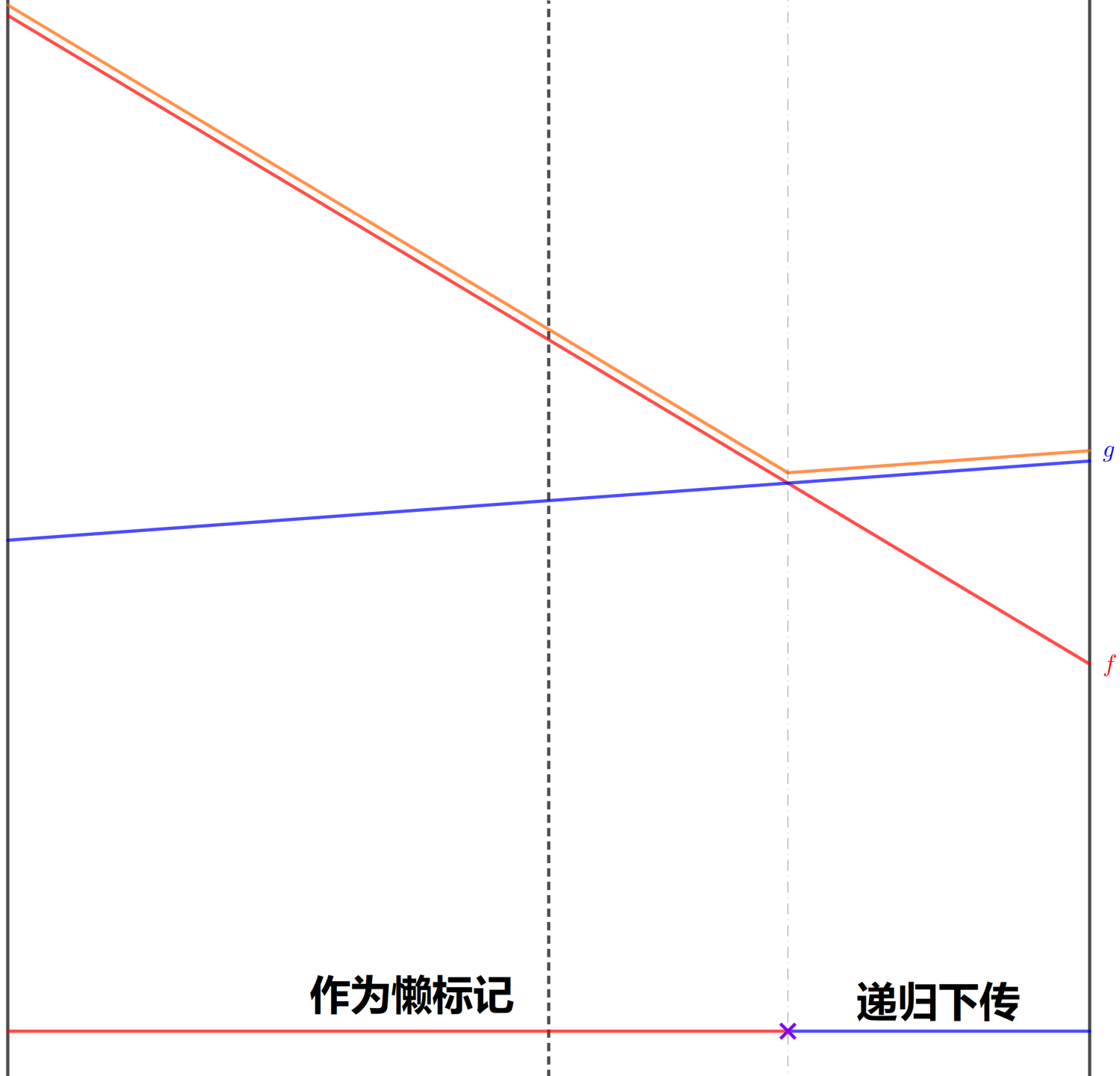
S 末日下的恋歌II

tag : 李超线段树

维护以下操作

1. 加入一条线段。
2. 查询在横坐标为 x 的竖直直线与所有有交点的线段中最靠上的一条。

在李超线段树的节点中，我们需要维护所有覆盖整个区间的线段中，在区间中点处最靠上的那一条线段的编号。查询某个横坐标 x 时，我们遍历线段树中所有包含 x 的区间，共有 $O(\log n)$ 个这样的区间，其构成从根节点到叶子节点的一条链。只有这 $O(\log n)$ 个线段可能成为最大值。计算这些线段在 x 位置的坐标值即可。


g
f
作为懒标记
递归下传

如何进行修改操作

首先，我们按照普通线段树的方式向下遍历。直到这条线段覆盖某个完整的线段树区间。

计算新插入的线段 f 与原来该区间维护的线段 g，分情况进行讨论：

- 如果 f 在中点处的 y 值大于 g 那么我们交换 f 和 g 即可。以下的讨论中 f 的中点值小于 g。
- 如果在区间的左端点处 f 高于 g，这说明 f 和 g 在左半个区间存在交点，f 可能时左半区间某些 x 的最大值，所以我们递归更新左区间。
- 如果右端点处 f 高于 g，同理，递归更新右区间。
- 如果两个端点处 f 都低于 g，说明 f 全面的不如 g，不可能成为答案，停止更新。

进行一次修改操作，首先按照普通线段树的方式，找到 $O(\log n)$ 个完整覆盖的区间，对于每个区间，只会递归的进入左右区间中的一个，所以每个区间会递归 $O(\log n)$ 次，总复杂度 $O(n \log^2 n)$。

T QHJ全家桶

tag : 吉司机线段树

维护下列六种操作：

1. 区间加
2. 区间取 $\max$
3. 区间求和
4. 区间变化次数
5. 区间历史最大值
6. 区间历史最小值

经典模板题，可以将其分成区间取 $\max$ 和区间历史最值两个部分看待，实战中基本只会出现其中一个部分，真的需要两个部分同时维护的极其少见。本ppt会稍微详细讲一下两个被拆开的部分。

区间取 max

同时维护操作1、2、3。

将某个数与另一个数取 $\max$，实际上等价于给这个数加上一个值。这个操作主要的难点在于给每个位置所加的值都是不同的，这样，我们不得不遍历到线段树的叶子节点，这样单次修改复杂度是不可接受的 $O(n)$。

然而，考虑一种特殊情况：将某个区间对 x 取 $\max$ 时，如果 x 大于这个区间的最小值 mn，小于这个区间的次小值 se_mn，如果想要同时维护区间和 sum 的话，我们只需要再一下维护区间最小值的数量 cnt_mn。这样就相当于 sum 增加了 $(x-mn)*cnt$的值了。

改变次数可以在这里同时维护。

因此，我们可以向下遍历，当遍历到一个满足条件$x > mn$，$x < se_m n$的区间时进行修改，否则继续向下遍历。通过势能分析可以证明，这样的复杂度是均摊 $O(n \log^2 n)$ 的。

区间历史最值

以历史最小值，同时维护操作1、5为例。

我们可以在一个线段树节点处维护当前区间最小值 mn 与历史区间最小值 his_mn，但当进行区间加操作时，我们要如何维护 tag 才能正确的向下传递呢?

假设我们不停的对某个线段树区间进行区间加操作，那么这个区间的懒标记会不停变化。如果我们在每一个时刻都尝试进行下传操作，那么只有其中 tag 值最小的时刻，mn_his_tag 的值才会最终对子树的 his_mn 产生影响。

下传的方式如下（打 $'$ 的代表子节点中的变量）：

$$his\_mn' = \min(his\_mn', mn' + mn\_his\_tag),$$

$$mn\_his\_tag' = \min(mn\_his\_tag', tag' + mn\_his\_tag).$$

询问中的历史最大值和历史最小值都可以这样维护。

然后是将这两部分合并起来，由于要维护历史最值，导致很多变量都不得不同时存一份历史最小值，最终一个线段树节点中存储的变量达到了令人恶心的16个之多（视写法不同可能会更多）。可以去洛谷的吉司机线段树板题的题解区中仔细学习如何将这两个部分合并，这里略过。

U silent wind bell

tag : **哈希/平衡树**

维护下列四种操作：

- 插入：在 S 的第 p 个字符后面插入一个新的字符 c。
- 删除：删除 S 中第 p 个字符。
- 区间翻转：将 S 中 $[p,q]$ 区间内的字符翻转。
- 求后缀LCP：求出后缀 $S_{p\sim L}$ 和后缀 $S_{q\sim L}$ 的LCP的长度。

只有求LCP的操作怎么处理?

子串LCP是一个经典问题。

1. 敬请期待字符串专题。
2. 字符串哈希（本题做法）。

字符串哈希

可以将整个字符串视作一个 k 进制数，每个位置上维护一个后缀，这样便可以方便的算出一个区间的哈希值。

通过比较两个区间的哈希值是否相同，便可以 $O(1)$ 比较两个子串是否相等。

想要求出两个子串的 LCP，只需要二分找出最长的使两个子串相同的长度 x 即可。

平衡树

由于本题还涉及区间翻转操作，常见的思路便是使用平衡树来维护。

完成本题需要支持分裂与合并操作，可以自行学习 splay 或 FHQ-Treap 中的一种。

具体来说，平衡树的一个节点代表一个位置，其左子树中是所有在该位置之前的节点，右子树中是所有在该位置之后的节点，一整棵子树就代表原序列中的一段区间。同时，在每个位置的代表节点处存储其子树所代表的一整段区间的哈希值。根节点的存储的哈希值可以通过左右子树的哈希值 $O(1)$ 算出。

如何维护区间翻转?

可以给平衡树上的节点像线段树一样打上懒标记。

在合并和分裂操作时下传标记，同时交换左右子树即可。

翻转之后要如何更改该节点对应区间的哈希值呢？我们可以直接将其反串的哈希值一并维护出来，在涉及区间翻转操作时交换这两个值即可。

在寻找LCP时，为了比较两个区间的哈希值是否相等，我们需要先使用分裂操作裂出相应的区间，来获取其对应的哈希值。之后再合并回去。

二分需要 $O(logn)$ 次，每一次还需要进行分裂合并操作，总复杂度 $O(n\log^2 n)$。

V Way to home

tag : **树套树/整体二分**

进行 m 次操作，每次操作形如：

- `1 l r c`，在编号为 $[l, r]$ 的火堆中**各加入一个**值为 c 的元素。
- `2 l r k`，查询当前在编号为 $[l, r]$ 的火堆中的**所有元素**中，第 k 大的值是多少。

树套树

可能的套法包括但不限于以下这些：

- 线段树套平衡树：外层线段树表示位置，内层平衡树表示所放元素。修改操作按照正常线段树的方法，$O(\log^2 n)$。查询操作先二分阈值 x，查询是否存在 k 个大于 x 的元素，$O(\log^3 n)$。
- 值域线段树套线段树：外层值域线段树每一个点代表每种元素，内层线段树表示对应元素在区间上的分布。修改操作外层单点修改，内层区间加，$O(\log^2 n)$。查询操作先在外层线段树上线段树二分，然后内层区间求和，$O(\log^2 n)$。
- 树状数组套值域线段树：外层树状数组表示位置，由于涉及区间加，内层值域线段树表示每个点的的不同元素数量的差分。修改操作按照上述定义的意义修改，$O(\log^2 n)$。查询操作在树状数组对应节点上的 $O(\log n)$ 棵线段树上二分，O(\log^2 n)。

整体二分

下面介绍整体二分的做法。

首先，如果只处理一个询问区间，那我们可以怎么处理呢？

可以将“查询区间中第 k 大的元素”，转化为“判断区间中是否存在 k 个大于 x 的元素”，从而用二分找到最小的不满足条件的 x 解决这个问题。

可以通过遍历这个询问操作前面所有的修改操作，维护一棵线段树记录某个位置大于 x 的元素数量。如果这个修改所加入的元素大于 x，则对线段树进行区间加操作，否则直接放弃这个区间。

这样处理一个区间的复杂度为 $O(q \log n \log A)$，然而，当处理完一个区间之后，之前维护的线段树信息对处理下一个区间也是有用的，我们没有必要抛弃这些值

具体操作如下：

记录两个操作的数组，分别代表需要向大/小的方向二分的操作。

还是按照顺序扫描所有操作，如果遇到了修改中，操作中加入的元素 w 大于等于这一次二分我们设定的阈值 x 的修改，以及查询操作中，满足有 k 个元素大于阈值 x 的询问，放到同一个操作数组中。因为这个数组中的询问满足了条件，所以我们需要进一步把阈值 x 调高，所以 $w < x$ 的修改操作是没有用的。扫描完所以操作后继续递归处理这半边即可。

遇到$w < x$ 的修改和不满足阈值的的查询类似处理，不过在将询问操作放到另一个操作数组时，为了消除 $w \geq x$的操作的影响，我们需要将这一次询问的 k 减去查询出来 $w \geq x$的数量 cnt，再加入操作数组中，同样递归处理这半边。

复杂度 $O(n \log^2 n)$。

感谢观看