线段树I

任飞宇

延迟标记

1. 线段树

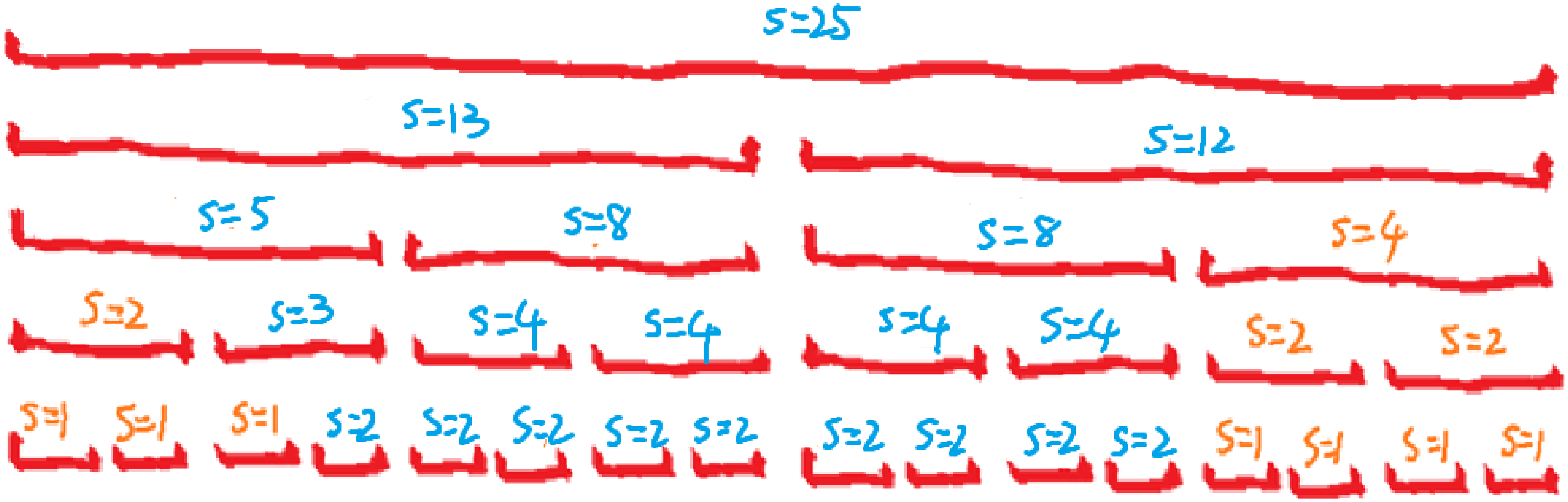
有一个长度为n的数列，在这个数列上进行m次操作：
操作1：1 x y k　含义：将区间 [x,y] 内每个数乘上k
操作2：2 x y k　含义：将区间 [x,y] 内每个数加上k
操作3：3 x y　　含义：输出区间 [x,y] 内每个数的和

1. 线段树

使用线段树，每个节点维护对应区间所有数的和s。设数组一开始全为1，那么线段树如下图所示：

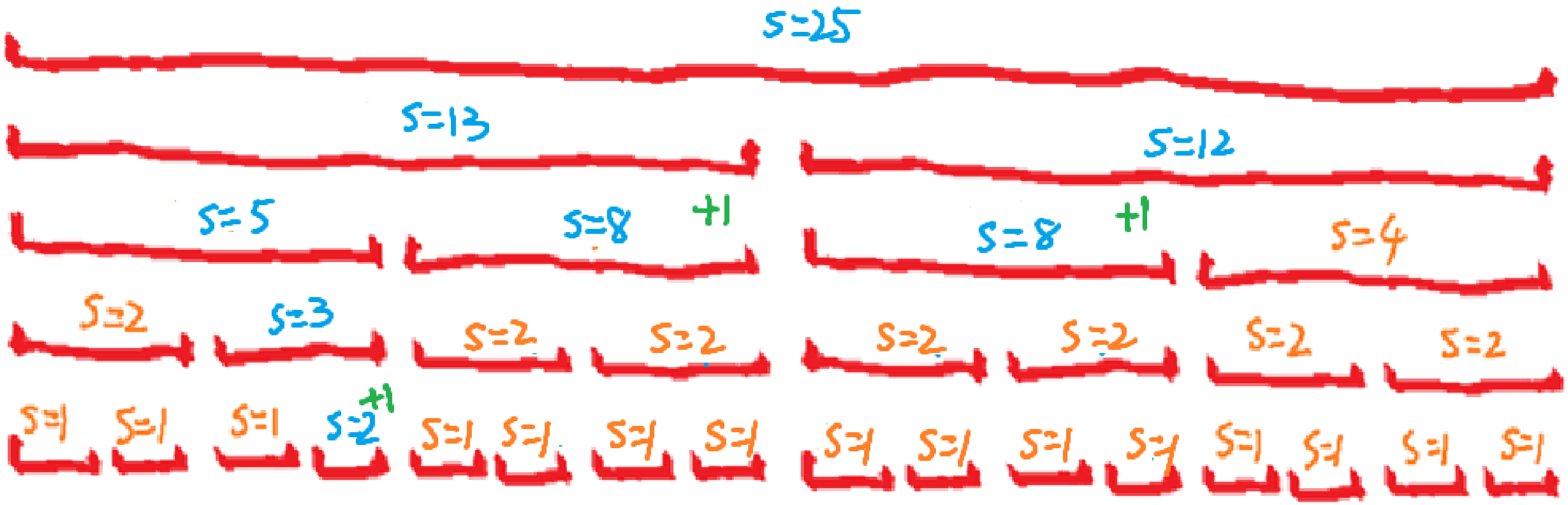
1. 线段树

假设接下来执行修改2 4 12 1，则区间[4,12]中的每个位置都加上1，s的变化如图。



1. 线段树

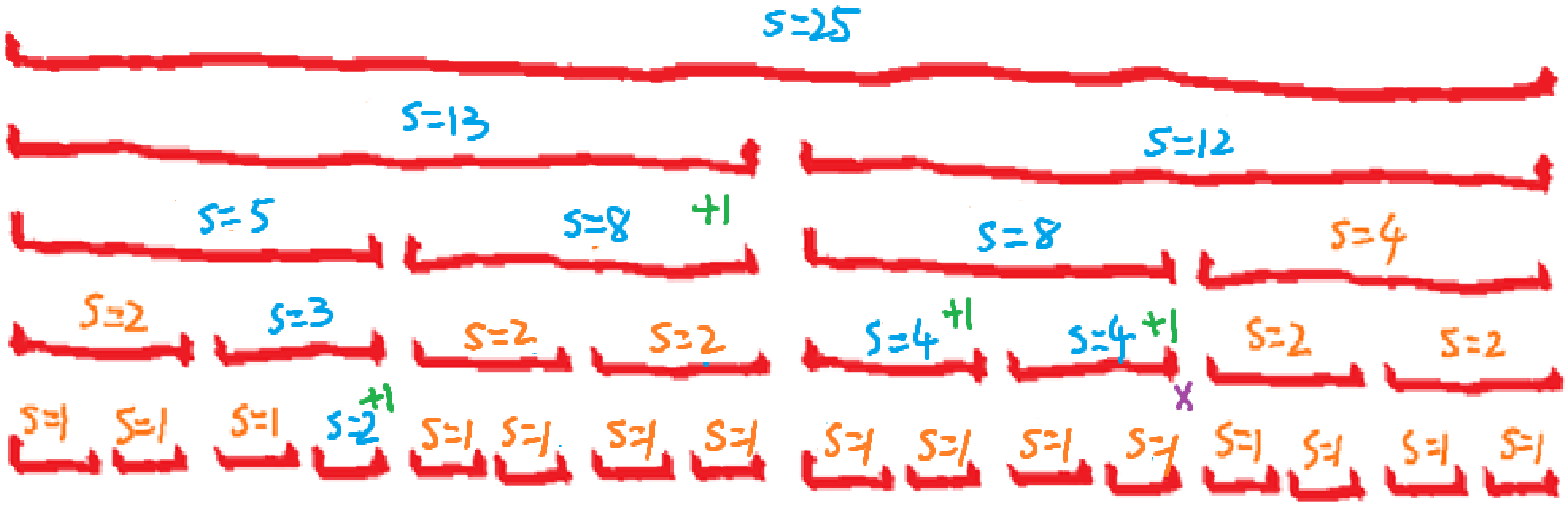
但是需要修改的节点太多了，为了保证效率，我们改为修改logn个节点的信息，并在这些节点上打上**延迟标记**。



此时标记下方的节点维护的信息是不真实的，但没关系，我们可以通过**下传标记**(Pushdown)来得到真实信息。等我们下次要访问下方的节点时再把标记往下传，获取真实的信息即可。

1. 线段树

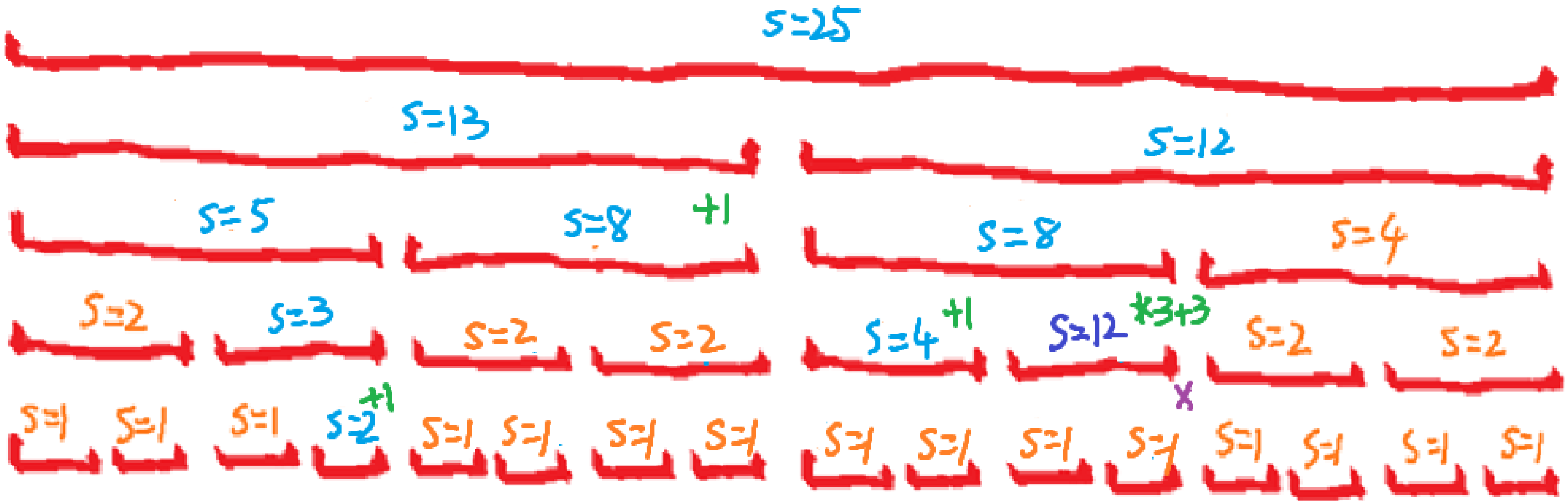
假设接下来执行操作1 11 12 3，由于标记讲究先来后到，所以在访问到节点x之前，我们要把根到x父亲这条路上的标记都做一次下传：



这样，从x的父亲到根的节点都没有标记了，节点x的信息就是真实的。我们可以放心地更新节点x的信息，并在节点x处打上乘标记

1. 线段树

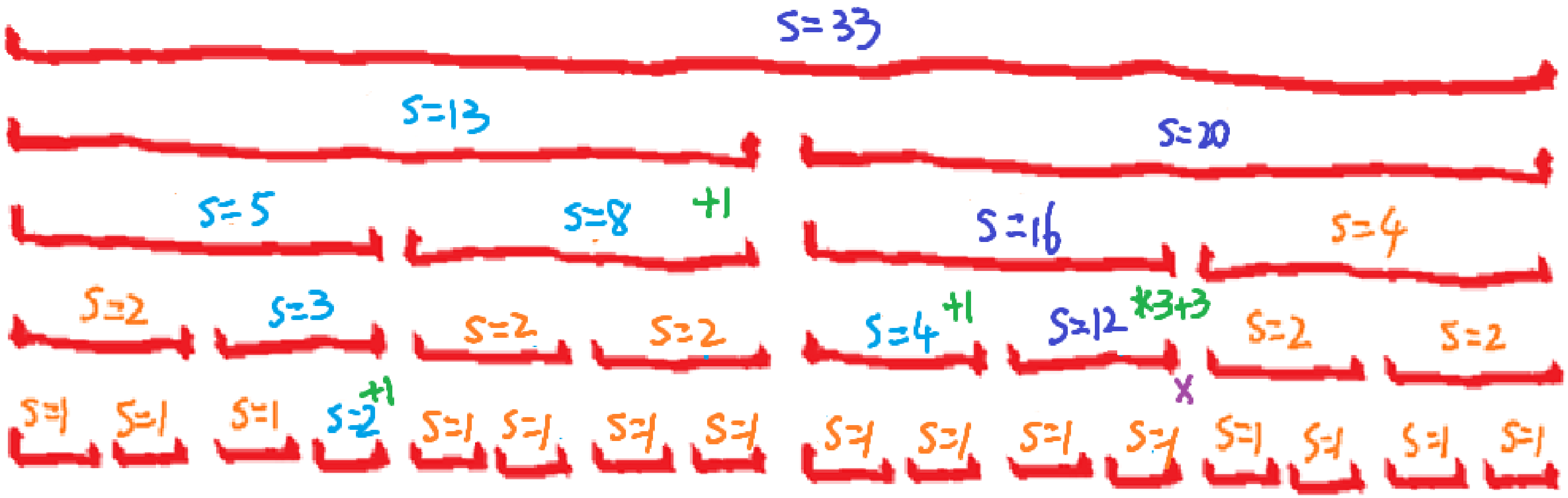
假设接下来执行操作1 11 12 3，由于标记讲究先来后到，所以在访问到节点x之前，我们要把根到x父亲这条路上的标记都做一次下传：



这样，从x的父亲到根的节点都没有标记了，节点x的信息就是真实的。我们可以放心地更新节点x的信息，并在节点x处打上乘标记

1. 线段树

最后，我们需要从下往上更新一遍节点的信息：



1. 线段树

修改操作：

```cpp
// C++ Version
void update(int l, int r, int c, int s, int t, int p) {
  // [l, r] 为修改区间, c 为被修改的元素的变化量, [s, t] 为当前节点包含的区间, p
  // 为当前节点的编号
  if (l <= s && t <= r) {
    d[p] += (t - s + 1) * c, b[p] += c;
    return;
  }  // 当前区间为修改区间的子集时直接修改当前节点的值,然后打标记,结束修改
  int m = s + ((t - s) >> 1);
  if (b[p] && s != t) {
    // 如果当前节点的懒标记非空,则更新当前节点两个子节点的值和懒标记值
    d[p * 2] += b[p] * (m - s + 1), d[p * 2 + 1] += b[p] * (t - m);
    b[p * 2] += b[p], b[p * 2 + 1] += b[p];  // 将标记下传给子节点
    b[p] = 0;                                // 清空当前节点的标记
  }
  if (l <= m) update(l, r, c, s, m, p * 2);
  if (r > m) update(l, r, c, m + 1, t, p * 2 + 1);
  d[p] = d[p * 2] + d[p * 2 + 1];
}
```

在访问子节点之前需要进行Pushdown操作，Pushdown需要完成：

1. 获取两个子节点真实的信息
2. 下传标记，跟子节点的标记合并

Pushup需要根据两个子节点的信息得到当前节点的信息

1. 线段树

查询操作类似，只是不需要Pushup操作：

```cpp
// C++ Version
int getsum(int l, int r, int s, int t, int p) {
  // [l, r] 为查询区间, [s, t] 为当前节点包含的区间, p 为当前节点的编号
  if (l <= s && t <= r) return d[p];
  // 当前区间为询问区间的子集时直接返回当前区间的和
  int m = s + ((t - s) >> 1);
  if (b[p]) {
    // 如果当前节点的懒标记非空,则更新当前节点两个子节点的值和懒标记值
    d[p * 2] += b[p] * (m - s + 1), d[p * 2 + 1] += b[p] * (t - m),
        b[p * 2] += b[p], b[p * 2 + 1] += b[p];  // 将标记下传给子节点
    b[p] = 0;                                     // 清空当前节点的标记
  }
  int sum = 0;
  if (l <= m) sum = getsum(l, r, s, m, p * 2);
  if (r > m) sum += getsum(l, r, m + 1, t, p * 2 + 1);
  return sum;
}
```

2. hdoj6967 I love data structure

序列中每个点有个(a_i, b_i)，每次操作为：

1. 给$[l, r]$中的a或者b同时加上一个数
2. 给$[l, r]$中的a和b交换
3. 将$[l, r]$中的a, b变成$3a + 2b$和$3a - 2b$
4. 求$[l, r]$中的$\sum_{i=l}^{i=r} a_i * b_i$

注意到2操作和3操作都是对向量(a, b)的线性变换，不妨统一表示成矩阵的形式。
那么这题的标记就有两种：
1. 乘上一个矩阵（优先）
2. 加上一个向量

线段树的节点要维护的信息为$\sum(a_i * b_i)$，考虑到修改操作的影响，还需要维护$\sum a_i^2$，$\sum b_i^2$，$\sum a_i$，$\sum b_i$的值。

3.砖墙

给定一个长度为 n且初始值全为 0的序列。你需要支持以下两种操作：

- Add L, R, h：将序列 $[L, R]$内所有值小于 h的元素都赋为 h，此时不改变高度大于 h的元素值
- Remove L, R, h：将序列 $[L, R]$内所有值大于 h的元素都赋为 h，此时不改变高度小于 h的元素值

你需要输出进行 k次上述操作之后的序列。

3.砖墙

给定一个长度为 n 且初始值全为 0的序列。你需要支持以下两种操作：

- Add L, R, h：将序列 $[L, R]$ 内所有值小于 h 的元素都赋为 h，此时不改变高度大于 h 的元素值
- Remove L, R, h：将序列 $[L, R]$ 内所有值大于 h 的元素都赋为 h，此时不改变高度小于 h 的元素值

你需要输出进行 k 次上述操作之后的序列。

每个节点维护标记L，R，表示跟L取max，跟R取min。
某个节点x的标记下传的时候，用 L_x 和 R_x 来更新子节点的标记范围，即

$$L_{son} = \max(\min(L_{son}, R_x), L_x),$$
$$R_{son} = \max(\min(R_{son}, R_x), L_x),$$

4. Beautiful Subsequences

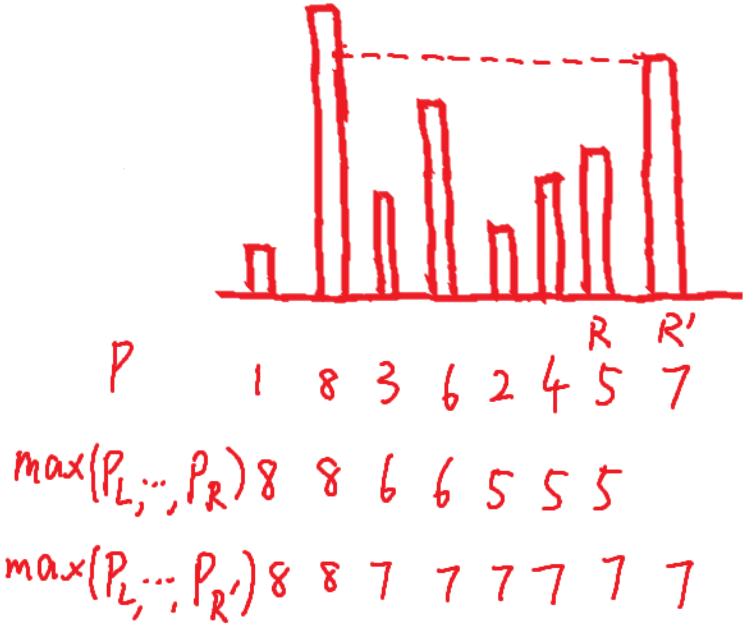
给定一个排列$P=(P_1,\dots,P_N)$和一个整数$K(0\le K\le 3)$，请找出满足以下条件的(L,R)的数量：

1. $1\le L\le R\le N$
2. $\max(P_L,\dots,P_R)-\min(P_L,\dots,P_R)\le R-L+K$

4. Beautiful Subsequences

尝试枚举R，然后统计有多少L满足条件，即$\max(P_L,\dots,P_R)-\min(P_L,\dots,P_R)-(R-L)\le K$ 。

在R从左往右枚举的过程中，我们维护每个L对应的左式的值。考虑R每右移一位，对左式的值的影响。



把$\max(P_L,\dots,P_R)$的变化视为区间覆盖操作的话不好处理，因为min和R的值也在变。

我们把它视为多个区间加，这个复杂度和单调栈相同。

$\min(P_L,\dots,P_R)$的变化同理。

于是问题变为区间加减，询问多少数字$\le K$。 $(0\le K\le 3)$

线段树似乎没法维护这个信息，但一个重要的观察是左式一定≥ 0

线段树可以实现区间加，维护区间最小值和最小值的数量，这里我们再维护第2/3/4小值和对应的数量，就能支持查询了。

线段树二分

有时我们需要找一个最小（大）的满足某个条件的位置，我们可以在线段树上找这个位置。
通过线段树节点维护的信息，快速判断要找的位置在左区间还是右区间。

例如可以用线段树维护一个集合，支持加入/删除一个数，询问第k小的数。

5. Siano

农夫 Byteasar 买了一片 n 亩的土地，他要在这上面种草。

他在每一亩土地上都种植了一种独一无二的草，其中，第 i 亩土地的草每天会长高 a_i 厘米。

Byteasar 一共会进行 m 次收割，其中第 i 次收割在第 d_i 天，并把所有高度大于等于 b_i 的部分全部割去。

Byteasar 想知道，每次收割得到的草的高度总和是多少，你能帮帮他吗?

对于 100% 的数据，$1 \le n, m \le 5 \times 10^5$，$1 \le a_i \le 10^6$，$1 \le d_i, b_i \le 10^{12}$。

数据保证 $d_1 < d_2 < ... < d_m$，并且任何时刻没有任何一亩草的高度超过 10^{12}。

5. Siano

不妨把草按照生长速度从小到大排好，任何时刻草的高度都是单调不降的，
那么每次割草的范围一定是一段后缀。

我们需要实现生长操作和割草操作，每种操作使用一个标记。
考虑维护区间最右边的数用来查找割草的范围。
知道了每次割草的范围，那么维护区间和就可以回答每次割了多少草。

所以需要两个标记和两种信息。割草标记（区间覆盖）优先。

```
void cut(int k,int l,int r,ll val){//割
    tree[k].tag=tree[k].maxn=val;
    tree[k].sum=1ll*(r-l+1)*val;
    tree[k].day=0;
}
```

```
void grow(int k,int l,int r,ll day){//生长
    tree[k].day+=day;
    tree[k].sum+=1ll*(sum[r]-sum[l-1])*day;
    tree[k].maxn+=1ll*a[r]*day;
}
```

6. Hotel

奶牛们正在前往苏必利尔湖度假，它们准备入住一家旅馆。这家旅馆共有一排n个房间，一开始所有房间都是空的。接下来会有m个事件发生：

1. 一队奶牛希望入住长度为x的**连续**空房间，请给出连续x个房间最左边的房号，这个房号越小越好。如果找不到输出0，如果找到了，这些房间会变成入住状态。
2. 某个区间的房间全部退房，变成空房间。

尝试在线段树节点上维护一些信息用来二分。

为了判断要找的段是否全部在左区间，我们可以每个节点维护区间最长空段。

要找的段也有可能横跨左右两个区间，我们要知道左区间的最长空后缀和右区间的最长空前缀

所以每个节点维护三个信息，修改操作为区间覆盖，使用一个标记。

7. 列队

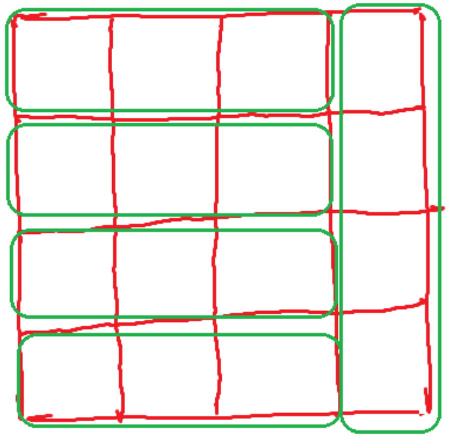
在一个$n*m$的方阵中，每个位置有一个数字，一开始i行j列的数字为$(i-1)*m+j$。
接下来发生q次事件，第i次事件先取出位置(x_i,y_i)上的数字，然后所有数字向左填补空缺，再向上填补空缺，取出的数字放到空出来的位置(n,m)上。请输出每次取出的数字。
$n,m,q \le 3*10^5$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	3	4
6	7	8	12
9	10	11	16
13	14	15	5

7. 列队

根据题目的操作，我们需要维护每一行和最后一列。
向左填补相当于去掉某一行中的某个数字，然后在末尾加入一个数字，
向上填补相当于去掉最后一列中的某个数字，然后在末尾加入一个数字。



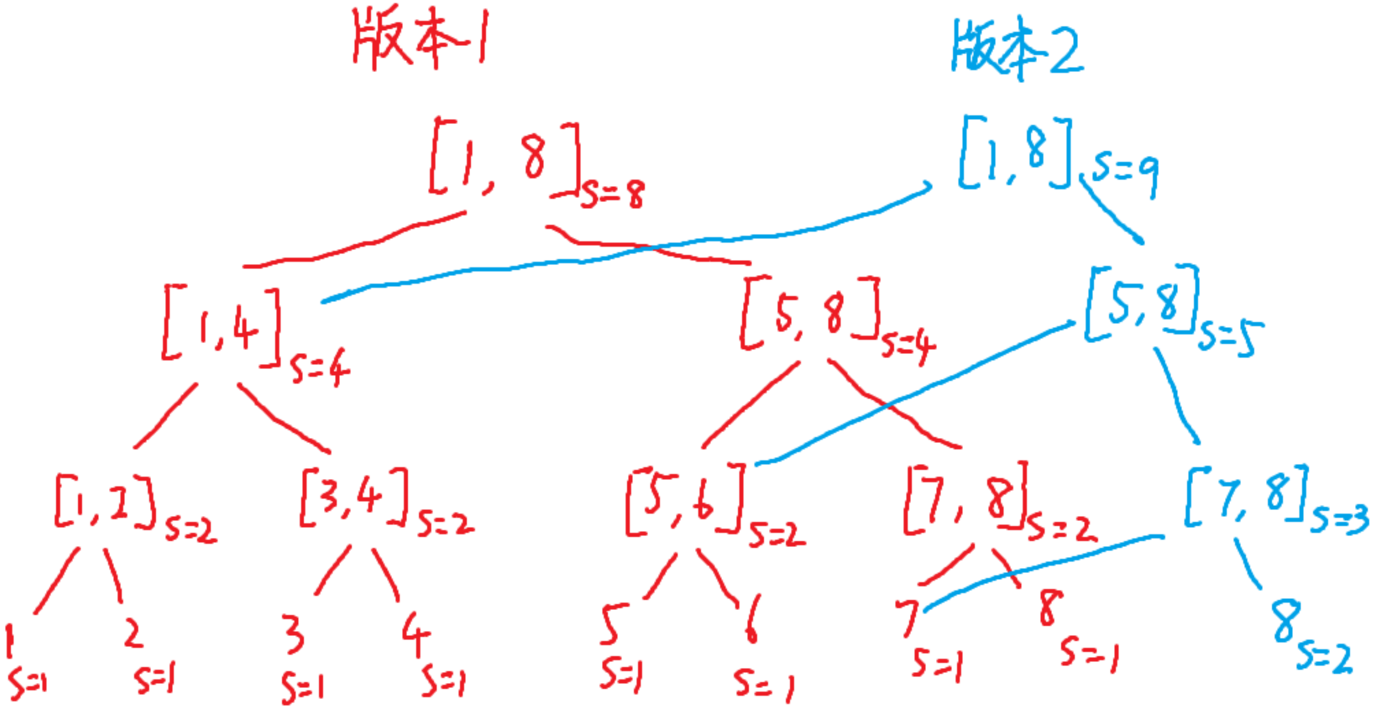
考虑用n+1颗线段树实现这些操作，由于要在末尾加数，所以线段树长度需要$\max(n, m) + q$。
去掉一个数的时候我们不需要真的移动来填补空位，而是在线段树中把这个位置标为空。
取数时找第k个非空的位置即可，线段树节点记区间内非空位置的数量，就可以二分查找了。

n, m的范围很大，不能直接开一个完整的线段树，所以需要使用动态开点。

持久化

随着不断的修改，线段树会不断变化，有时我们想获取之前的信息，这时就需要持久化。
每一次修改会产生一个新版本的线段树，但是我们仍然要保留旧版本的线段树。

可以复制整颗线段树，然后在复制出来的线段树上修改，但是这样效率太低。
好在修改操作只涉及较少的节点，所以不修改的节点是可以复用的，只需要对每个修改的节点建一个新节点即可。



8. K-th Number

给定一个数列，多次询问$[L_i, R_i]$中第K_i大的数是什么？

建立以权值为下标的线段树，从左往右依次添加数字，每添加一个数a_i就产生一个版本T_i，那么询问[L, R]时，$T_R - T_{L-1}$就代表了这个区间的权值线段树，二分就能找到第K大的数。

9. 棒棒糖

Coffee 的世界里也是有棒棒糖卖的，Coffee 买了 n 只连着的棒棒糖。这 n 只棒棒糖包裹在小塑料袋中，排成 一列，相邻的两只棒棒糖的塑料袋是接起来的。为了方便，我们把棒棒糖从左到右编号为$1\cdots n$。

每只棒棒糖有一种口味。第 i 只的口味是 c_i。两只棒棒糖 i,j 的口味相同，当且仅当 $c_i=c_j$。Coffee 对 m 只棒棒糖总体口味的评价比较奇怪。如果这 m 只棒棒糖中，有一种口味 c_0 的数量严格大于总数的一半，那么 Coffee 认为这 m 只棒棒糖主要是 c_0 口味的。Coffee 知道，这里的 c_0 如果存在就一定是唯一的。而当 c_0 不存在时，Coffee 认为这 m 只棒棒糖是混合口味的。

Coffee 暂时舍不得吃棒棒糖，它在想一些好玩的问题。如果考虑棒棒糖序列的一个连续子序列 $s\cdots t(1\le s\le t\le n)$，包括棒棒糖 s 和 t。那么这 $t-s+1$ 只棒棒糖的总体口味是什么呢?

Coffee有一堆这样的问题，一共 m 个。第 i 个问题是棒棒糖子序列 $s_i\cdots t_i$ 的总体口味，请你帮忙解决。

对于 100% 的数据，$1\le n,m,c_i\le 5\times 10^4$。

9. 棒棒糖

建立以权值为下标的线段树，从左往右依次添加数字，每添加一个数a_i就产生一个版本T_i，那么询问[L, R]时，$T_R - T_{L-1}$就代表了这个区间的权值线段树。

因为要找的是严格众数，所以可以在线段树上二分查找。如果答案在左区间的话，左区间的数字个数一定过半，右区间同理。

10. Till I Collapse

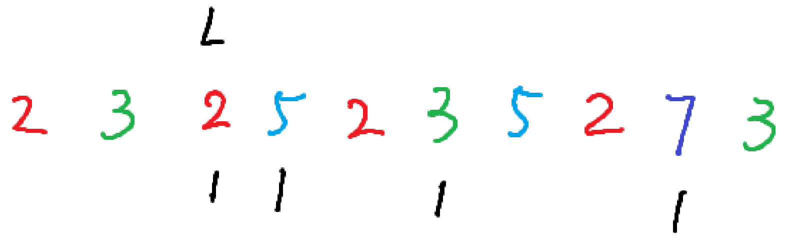
给定长为n的数列，将这n个数划分成m段使得每段中数字的种类数$\leq k$。
对于每一个满足$1 \leq k \leq n$的k，都需要求出一个最小的m。
$1 \leq n \leq 10^5, 1 \leq a_i \leq n$

10. Till I Collapse

贪心地从左边开始，每次取尽可能长的段。注意到每段的长度至少为k，所以段的数量不超过n/k。所以一共向右跳$nlogn$次，我们需要快速找出当前位置向右最远跳到哪里。

每一次要找的段长是可以二分的，直接二分的话多带来一个log 的复杂度。

尝试在线段树上二分。设当前段的左端点为L，为了维护种类数这一信息，我们把从L开始第一次出现的数标为1，那么位置1到位置R的前缀和就是区间$[L,R]$的种类数。



为了找从某个位置L出发跳的长度，我们需要一颗线段树T_L，为了处理全部$nlogn$次跳跃，我们就需要n颗线段树$T_{1..n}$。

使用持久化的技巧，从线段树T_i进行修改得到线段树T_{i-1}，就可以求从任意位置起跳的距离了。

线段树分治

某些离线问题有插入和删除操作，但是删除不好处理。

只有插入很好处理，插入加撤销也比较好处理。

这时候就可以搞出来每一个元素的存在时间，在线段树上 dfs，插入删除就变成了插入撤销。

11. 二分图

给出一张 n 个节点 m 条边的图。
其中每条边 e_i 都只在某一个给出的时间段 $[s_i, t_i]$ 内存在。
对于 0 到 T 的每个时刻，判断其是否为二分图。

$n, m, T \leq 10^5$

11. 二分图

原图为二分图，等价于可以进行 01 染色。如果只有加边操作，那么可以用带权并查集简单实现。

但是现在每一条边都有一个存在区间，这怎么解决呢？

在时间轴上建一棵线段树，然后每一个点开一个 vector。对于一条边，我们直接找到线段树上对应的节点，把这条边挂到上面。

接下来我们发现，要想得到在某个时间点有哪些边，我们只需要从根节点到叶节点，把路上挂的那些边加进去就行了！

这样我们就有了一个思路：我们在这棵时间线段树上进行dfs，每次把对应的边插入，并且判断是否是二分图。

由于 dfs 需要回溯，我们的并查集需要支持撤销（注意撤销和删除不一样），因此只能按秩合并不能路径压缩。