赛格蒙特彼茨 解题报告

杭州学军中学 吉如一

1 试题大意

给出两个长度为n的数组A和B，最开始这两个数组是完全相同的，接下来给出了m个操作，操作有三种类型：

1. 对于所有的$i \in [l, r]$，将A_i变成$A_i + c$。

2. 对于所有的$i \in [l, r]$，将A_i变成$\max(A_i, d)$。

3. 对于所有的$i \in [l, r]$，询问B_i的和。

在每一次操作结束之后，都会进行一次更新：对于所有的$i \in [1, n]$，将B_i变成$\min(B_i, A_i)$。

数据范围如下表所示：

测试点编号	n	m	其他
1	$= 5000$	$= 5000$	
2	$= 100000$	$= 100000$	$t_i \neq 1$
3			当$t_i = 3$时，$l_i = r_i$
4			
5			$t_i \neq 2$
6			
7			
8			
9			
10			

对于所有数据，满足$|c_i| \leq 10^4$，$|A_i|, |d_i| \leq 10^9$

2 算法介绍

2.1 算法一

可以直接暴力模拟整个过程，时间复杂度O(nm)，可以通过第一个测试点。

2.2 算法二

如果不存在操作一，我们可以发现整个过程中A数组都是单调不降的，所以数组B不会发生变化。因此只需要预处理出数组B的前缀和，就能$O(1)$解决每一个询问了。

时间复杂度O($n+m$)，可以通过第二个测试点。

2.3 算法三

考虑三四两个测试点，问题相当于区间加减，区间取max，询问单点历史最小值。

如果没有区间取max，这是一个经典问题，我们只需要对每一个点记录当前的区间加减标记A，以及上一次标记下传后到现在的最小的标记pA。

考虑标记下传的过程，假设从父节点传下来的加减标记是B，历史最小标记是pB，那么更新过程就是$pA = \min(pA, A + pB)$以及$A = A + B$。至于询问我们只要把所有标记给下传到叶子节点，就能得到答案了。

现在，我们考虑如何把区间取max转化成区间加减。

2.3.1 例题一

给出一个长度为n 的数列A。接下来进行了m 次操作，操作有两种类型：

1. 对于所有的$i \in [l, r]$，将A_i 变成$\max(A_i, x)$。

2. 对于所有的$i \in [l, r]$，询问A_i 的和。

数据范围$n, m \le 3 \times 10^5$。

因为标记无法正常合并，所以这个问题用普通的线段树是无法解决的，我们考虑这么一个做法：

我们对线段树中的每一个节点维护它所代表的区间中的数的最小值mi，次小值se，最小值个数t 以及所有数的和ans。

对于每一次修改操作，我们先用普通线段树的方法找到每一个要修改的区间，接着我们从这个区间开始DFS，DFS的时候我们把遇到的所有节点分成三类：

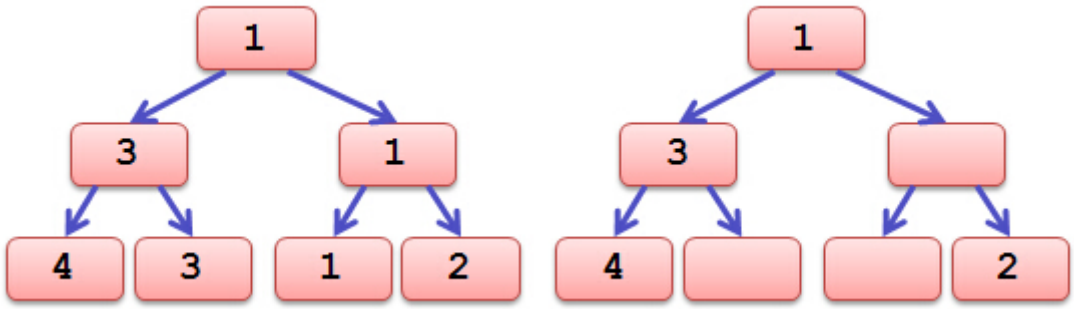
1. $mi \geq x$ 时，显然这一次操作不会对这一个区间中的任意一个数产生修改，所以可以直接退出。

2. $mi < x < se$ 时，显然只有所有最小值会被修改，因为我们记录了最小值个数，所以可以直接让 num 加上 $t \times (x - mi)$，并把 mi 赋值为x，然后打上标记就可以直接退出了。

3. $se \leq x$ 时，无法直接更新，所以采取的方案是分别递归左儿子和右儿子，然后回到这个节点的时候从两个孩子处更新信息。

如果进行实现的话，我们可以发现这一个算法的运行效率非常高。实际上我们可以证明它的时间复杂度是$O(m \log n)$的。

首先我们把最小值看成标记，记对线段树每一个节点，记录一个标记值，它的值等于它所控制的区间中的最小值。接着如果一个点的标记值和它父亲的标记值相同，那么我们就把它这个位置的标记删去，大致转化如下图所示（左图记录的是线段树中的最小值，右图是转化后每一个位置的标记）：



显然在转化之后，线段树中最多存在n个标记。这些标记满足每一个标记的值都小于它子树中的所有标记的值。每一个位置实际的值等于从它对应的线段树的叶子节点出发，向上走遇到的第一个标记的值。

观察上述算法，可以发现我们维护的区间次小值，相当于子树中标记的最小值。而暴力DFS的作用，相当于在打上了新的标记后，为了满足标记值随深度递增的性质，在子树中回收掉值小于它的标记，即标记回收。现在我们要证明的是标记回收的复杂度。

首先我们定义标记类的概念：

1.同一次区间取max标记产生的标记属于同一类。

2.同一个标记下传产生的新标记属于同一类。

3.不满足前两个条件的任意两个标记属于不同类。

接着定义每一个标记类的权值等于这一类标记加上线段树的根节点在线段树上形成的虚树大小（即所有子树中存在这一类标记的点的个数），定义势函数$\Phi(x)$为线段树中的所有标记类的权值总和。

考虑一次区间取max操作，我们添加了一个新的标记类，它的权值等于我们打标记时经过的点数，这显然是单次$O(\log n)$的。

考虑一次标记下传，显然我们只让这一类标记的权值增加了$O(1)$。

考虑DFS的过程，我们一旦开始进行暴力DFS，那么说明这个点的子树中一定存在需要回收的标记，而在回收之后这个点的子树中一定不存在这一类标记，所以我们经过每一个节点，一定至少存在一类标记的权值减少了1，那么必定伴随着势函数减少了1。

因为标记下传只存在我们在线段树上确定打标记区间的时候，所以标记下传的总次数是$O(m\log n)$的，而打标记时对势函数产生的总贡献也是$O(m\log n)$的，所以势函数总的变化量只有$O(m\log n)$。

到此，我们就证明了这个算法的复杂度是$O(m\log n)$ 的。

2.3.2 例题二

给出一个长度为n 的数列A。接下来进行了m 次操作，操作有三种类型：

1. 对于所有的$i \in [l, r]$，将A_i 变成$\max(A_i, x)$。

2. 对于所有的$i \in [l, r]$，将A_i 加上x。

3. 对于所有的$i \in [l, r]$，询问A_i 的和。

数据范围$n, m \le 3 \times 10^5$。

因为加上了区间加减，区间最小值、次小值和最小值个数还是可以维护的，所以我们考虑沿用刚才的做法。

接着我们来证明复杂度。因为有了区间加减这一个操作，例题一的证明就难以沿用了，但是大致的思想可以借鉴。

我们可以修改一下标记类的定义：

1.同一次区间取max标记产生的标记属于同一类。

2.同一个标记下传产生的新标记都属于原来的一类。

3.对于一次区间加减操作，对于当前线段树中的任意一类，如果它被修改区间完全包含或者和修改区间完全相离，那么这一类就不变；否则我们就让这一类分裂成两类：一类是被修改的部分，一类是剩下的部分。

重新定义每一类标记的权值：等于这一类标记在线段树上所有节点形成的虚树大小（在这儿不包含根节点）。定义辅助函数$\Phi(x)$等于所有标记类的权值和。

这样一次区间加减操作对辅助函数函数的影响只有单次$O(\log n)$的标记下传（分裂标记的时候势能函数只会减少）；一次标记下传只会让辅助函数增加$O(1)$；一次打标记只会让辅助函数增加$O(\log n)$。

考虑DFS的过程，我们先通过线段树定位了若干个节点，然后从这些节点开始DFS。我们对每一个开始节点考虑每一类会被回收的标记：如果这一类标记只有一部分在这个子树中，那么我们每一次回收都会让这一种标记的权值减一；而如果这一类标记全部都在子树中，我们回收的过程相当于先花了若干的代价定位了这一类标记的LCA，然后再在标记的虚树上进行回收。

所以我们可以把复杂度分成两部分，第一部分是减少辅助函数时花费的时间复杂度，这是$O(m\log n)$的；第二部分的复杂度是定位某一类标记的LCA时产生的复杂度，每一类标记都可能对这一部分的复杂度产生$O(\log n)$的贡献，因为标记的总数只有$O(m\log n)$，所以标记的类数也是$O(m\log n)$的，因此这一部分的复杂度的一个上界是$O(m\log^2 n)$的。

到此我们就证明了这一个算法解决这个问题时的复杂度是$O(m\log^2 n)$的（当然，如果使用标记深度和作为势函数可以更方便的得到这个上界，这儿主要是沿用例题一的证明）。

实际上这个上界是比较松的，对目前我构造的数据来说，这个算法的运行效率都更接近与$O(m\log n)$。因此我猜想在算法二的证明中标记的总类数是线性的，即这一个算法的时间复杂度是$O(m\log n)$的，然而到目前为止还没有好的方法能够证明，也无法构造出卡到$O(m\log^2 n)$的数据。在接下来的讨论中，我们就使用$O(m\log^2 n)$来表示这个算法的复杂度。

回归到这道题目的三四两个测试点，我们继续考虑例题二中的算法。我们发现在DFS到终止节点的时候，我们相当与给这一个节点中的所有最小值加上了一个特定值——这就等价于一个特殊的加减操作。

我们可以把线段树中的每一个节点中的数都拆成两类数——最小值和非最

小值，并且使用两套标记来进行维护。因为当我们对最小值进行加减的时候，保证了修改后最小值依然要小于次小值，所以标记是可以进行下传和合并的。

用这种方法我们就能够把区间取max操作给转化成区间加减操作啦。然后就只需要对每一个点的最小值和非最小值维护加减标记以及标记的历史最小值就可以了。

时间复杂度$O(m\log^2 n)$，结合算法一和算法二期望得分40分。

2.4 算法四

考虑五六七这三个测试点，我们需要解决区间加减，询问区间历史最小值的和这一个问题。

这个问题相对比较困难，因为直接对这个问题使用懒标记并且维护的话是做不到的，所以我们需要进行一定的转化。

考虑定义一个辅助数组$C_i = A_i - B_i$，我们考虑给A_i加上x时对C_i的影响。

首先如果B_i没有变化，那么就相当于给C_i加上了x。然而实际上加上了x后，可能B_i会被更新，这时满足$A_i + x < B_i$。从C_i 的角度来说，这种情况发生当且仅当$C_i + x < 0$，在更新之后C_i的值变成了0。

所以区间加减操作，相当于把区间中的所有C_i变成$\max(C_i + x, 0)$。

至于询问B_i的区间和，我们可以把它转化成$\sum_{i=l}^{r} A_i - \sum_{i=l}^{r} C_i$，这两个都是可以维护的（$A_i$的区间和就是经典的线段树问题，$C_i$的区间和可以直接使用算法三中例题二的方法）。

时间复杂度$O(m\log^2 n)$，结合算法一和算法二期望得分70分。

2.5 算法五

对于全部的数据，我们发现就只是比五六七这三个测试点多了一个区间取max操作，而在算法三中我们已经得到了一种把区间取max操作转化成加减操作的方法。

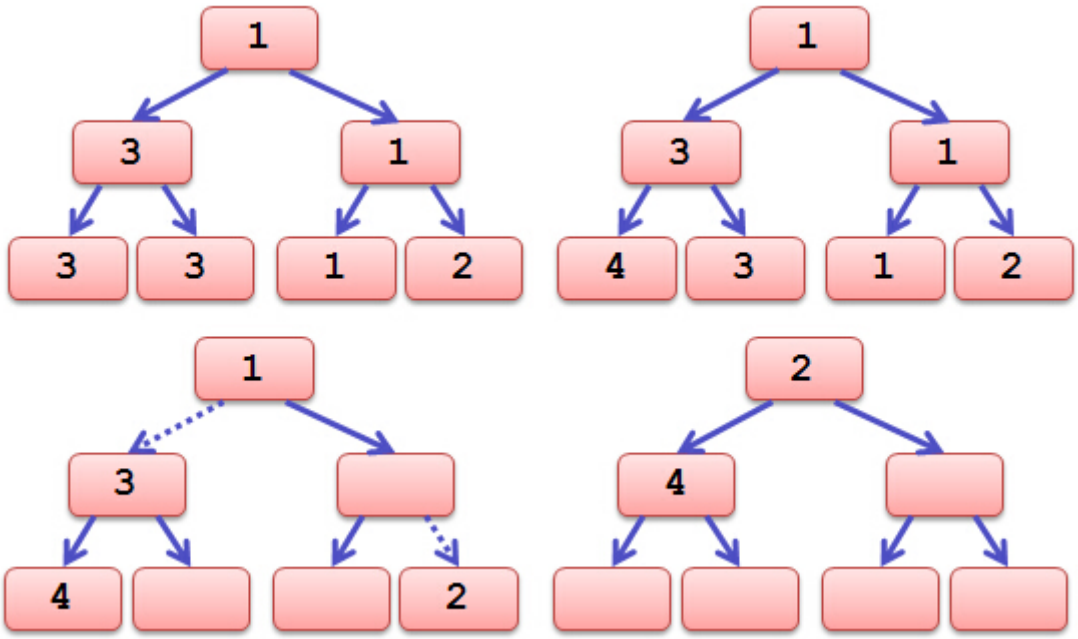
所以最终的算法就出现了。最开始和算法四一样，把询问转化成维护A数组和C数组的前缀和。接着我们用算法三中的方法，对数组A 维护区间最小值和次小值，从而把区间取max操作转化成对区间最小值的加减操作，这样就能维护数组A的区间和了。

然后，我们来考虑怎么对C维护区间和，影响到C的操作有：对所有$i \in [l,r]$，把C_i变成$\max(C_i + x, 0)$（接下来称为第一类操作）；对所有$i \in [l,r]$，如果A_i是这个区间中A的最小值，那么把C_i变成$\max(C_i + x, 0)$（接下来称为第二类操作）。

所以还是使用算法三的方法，我们对线段树的每一个节点维护所有A的值是最小值的位置中C_i的最小值、次小值、最小值个数以及所有A的值不是最小值的位置中C_i的最小值、次小值、最小值个数，然后在对C进行修改的时候分别进行暴力DFS就行了。

最后我们来证明这个算法的时间复杂度。我们把A数组的部分称为第一层，C数组的称为第二层。首先第一层的复杂度和算法三的时候一样，我们能够轻易的证明出它是$O(m \log^2 n)$的，所以我们接下来只考虑第二部分的复杂度。

仿照算法三中的方法，首先我们先把第二层的最小值给转化成标记。可以对A的值等于最小值的所有位置以及A的值不等于最小值的所有位置分别转化成标记（接下来我们把这两个分别称为第一棵树和第二棵树），转化之后的树如下图所示（上方是原树，两侧分别为A和C的最小值，下方左图是第一棵树，右图是第二棵树）：



可以发现任何时刻，任何一个出现在第二棵树中的标记一定在第一棵树中

出现过，所以两棵树上出现过的标记总数的数量级和第一棵树是相同的，所以我们只需要对第一棵树进行分析就行了。

因为第一棵树的定义是依赖数组A的，所以如果在原树中如果某一个节点的最小值和它的父节点不同，那么在第一棵树中这个位置到它父亲的转移是不存在的，我们需要删掉这条边（在图中用虚线表示）。因此在转化后第一棵树被剖成了若干个联通块，每一个联通块都至少存在一个叶子节点且满足标记值随着深度的增加而增加的性质。

因为直接对第一棵树进行标记回收以及区间加减的时候和A数组并没有什么区别，所以我们只考虑A数组中进行暴力DFS的时候产生的影响。在暴力DFS的时候，我们不仅对第一棵树进行了若干次子树加减操作，而且还合并了若干个联通块。不过如果我们在DFS的同时对标记进行下传，那么在合并两个联通块的时候，处在下方的联通块的根节点和上方联通块的根节点之间的路径上应该没有任何标记，所以这时直接把两个联通块合并起来，是不影响标记随着深度递增的限制的。

在算法三中，我们证明了暴力DFS的时候经过的节点的数的一个上界是$O(m\log^2 n)$，因此在第一棵树中标记下传的次数的上界是$O(m\log^2 n)$的，因此标记总数是$O(m\log^2 n)$的。沿用算法三中的证明，我们可以得到这个算法的一个复杂度上界$O(m\log^3 n)$。

不过不难发现这个上界是非常松的，换句话说常数非常小，实际上在目前构造出来的数据中，暴力DFS的次数都远远没有达到$O(m\log^3 n)$的级别，因此我猜想应该存在更低的上界的证明（实际上如果算法三中我的猜想得到了证明的话，沿用上述证明就可以得到一个$O(m\log^2 n)$的上界），但是因为目前我还没有得到更好的证明方法或者极限数据构造方法，所以目前只能用$O(m\log^3 n)$ 这样一个比较松的上界来衡量这个算法的运行效率。

时间复杂度$O(m\log^3 n)$，空间复杂度$O(n)$，期望得分100分。

2.6 数据生成方法

数据显然不能随机，因为当数据随机的时候，取max操作的修改次数期望大约是$O(m\log n)$级别的：因为区间加减的影响很小，可以忽略，所以一个位置被修改的概率当且仅当这次取max的数是影响到它的数中最大的，因此第i次取max操作它被修改的概率是$\frac{1}{i}$，对n和m求和后可以得到期望修改次数

是$O(m \log n)$ 的。

所以可以把每一次数据的长度划分成三段：第一段构造一些修改次数以及历史最小值修改次数尽可能多的数据，第二段放一个范围比较大的随机数据，第三段并列的放很多范围比较小的随机数据。这样数据的强度就基本够了（不过没有考虑一些奇形怪状的算法还是有可能被水过去的）。

2.7 得分估计

这是一道难度不是很高的题目，主要考察了有关线段树这一基础数据结构的使用技巧。因为我在冬令营的营员交流上已经大致介绍过了算法三中提到的算法，选手不必从头开始对算法进行思考，这在一定程度上降低了这题的难度。因此这题的难点主要在于：想到对辅助数组C 的构建，举一反三想出这一算法的嵌套方法以及具体实现出这一算法（这一题的代码量不算小，也有一定的细节，但是因为比赛时间很充裕，只要想出算法的话大部分选手应该都能成功的实现出来）。

因此在我看来，这题应该在比赛中起到送分送温暖的作用，我觉得在比赛中应该会有一到两名选手获得高于70 分的分数，另外有一到两名选手获得70分，大部分选手应该都能获得40分及以上的分数:)。