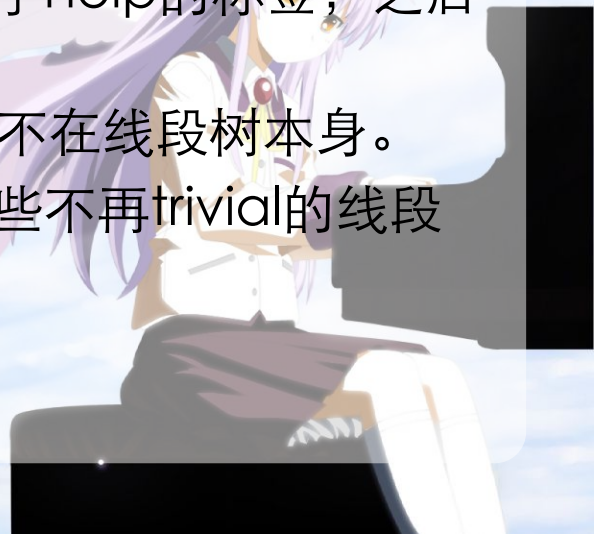

Segment tree
Beats!
C_SUNSHINE
jiry_2
安师大附中 罗哲正
杭州学军中学 吉如一

为什么要做这个PPT

- 今年清华集训，  出了一道线段树题。
- 在讲题的时候pls表示他不会做区间版本。
- 于是本着科学打脸观，我们回去研究了一下这一类问题，于是就有了这个PPT辣。

前（che）言（dan）

- 如今的OI界：
  - 仙人掌当道
  - 玄学题横行
  - 物理题为[和谐]一方
- 相信很多同学在学习了线段树之后，便把它打上了noip的标签，之后便少有研究。
- 的确，在OI中大部分有关线段树的难题，难点都不在线段树本身。
- 那么接下来，让我们回归线段树，来见识一下一些不再trivial的线段树问题。



前（che）言（dan）

- 让我们进入正题→_→
- 在意左下角的都是baka!

picks loves segment tree I

- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数变成$\min(A_i,x)$
- 询问区间[l,r]中所有数的和
- n,m⩽50000
- 我会分块！
- n,m ⩽ 500000
- 线段树?

picks loves segment tree I

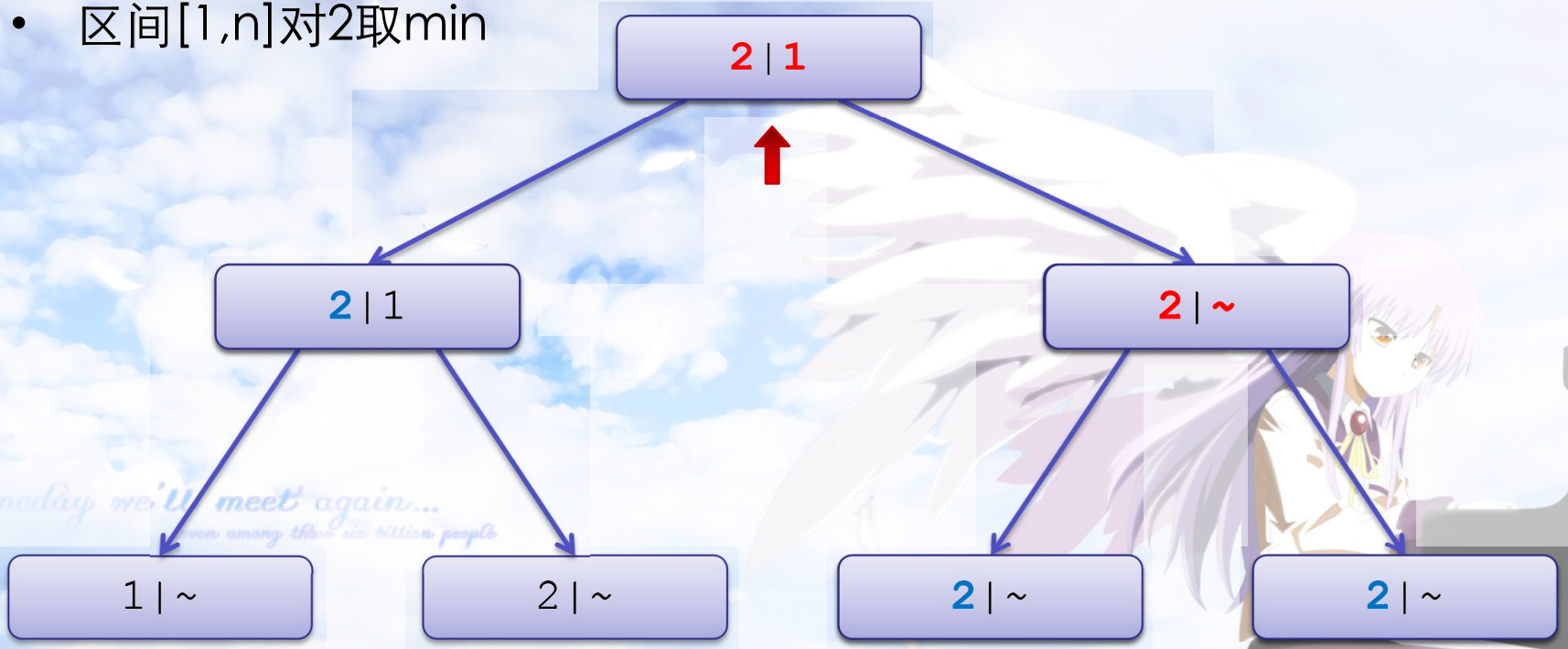
- 最基本的线段树没有办法维护，可以尝试一波乱搞
- 类比某道区间取模的题，我们每一次找出这个区间的最大值mx，如果mx>x，那么暴力修改这个位置的值，否则已经修改完毕，退出。
- 这样我们就得到了一个$O(n^2 \log n)$的优秀解法辣。

picks loves segment tree I

- 我们来试着打一打补丁。
- 对线段树上的每一个区间维护区间最大值mx，这个区间中最大值出现的次数t，区间次大值se，当然还要维护区间和sum
- 现在考虑打上区间取min标记x：
- 如果mx<=x，那么对sum就没有修改。
- 如果se<x<mx，那么sum=sum-(mx-x)×t。
- 如果x<=se<mx，那么…
- 妈呀我不会做
- 如果遇到这种情况，我们分别DFS这个节点的两个孩子，如果当前DFS的过程中遇到了前两种情况，就直接修改打上标记然后退出，否则就继续DFS。

picks loves segment tree I

- 区间[1,n]对2取min



picks loves segment tree I

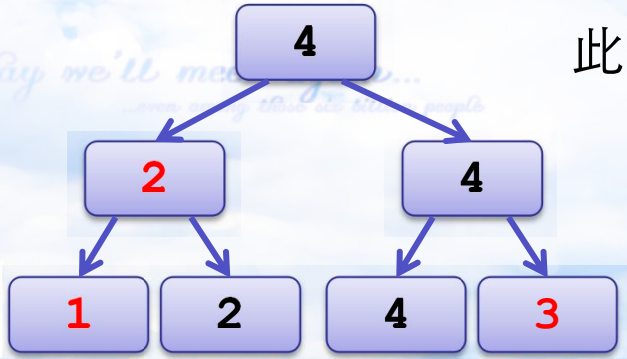
- 我们来试着写一写这玩意的程序，发现跑的飞快。
- 实际上这个做法的复杂度是O(nlogn)的。
- 至于证明嘛…
- 先不告诉泥萌，哼哒。

picks loves segment tree II

- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数变成$\min(A_i,x)$
- 区间[l,r]中的所有数加上x（x可能是负数）
- 询问区间[l,r]中所有数的和
- n,m⩽50000
- 我会分块！
- n,m ⩽ 500000
- 线段树？

picks loves segment tree II

- 可以发现有了区间加减，区间最大值mx，最大值个数t以及次大值se都还是可以维护的。
- 还是考虑沿用刚才的做法。于是我们就得到了一个跑的很快的做法辣！
- 为了避免被打，我们现在有必要来证明一下这题的时间复杂度了。
- 我们把最大值和它的父节点不同的节点称为关键点，令势函数Φ为线段树中的关键点个数，显然有0⩽Φ⩽n 。
- 此时有Φ=3

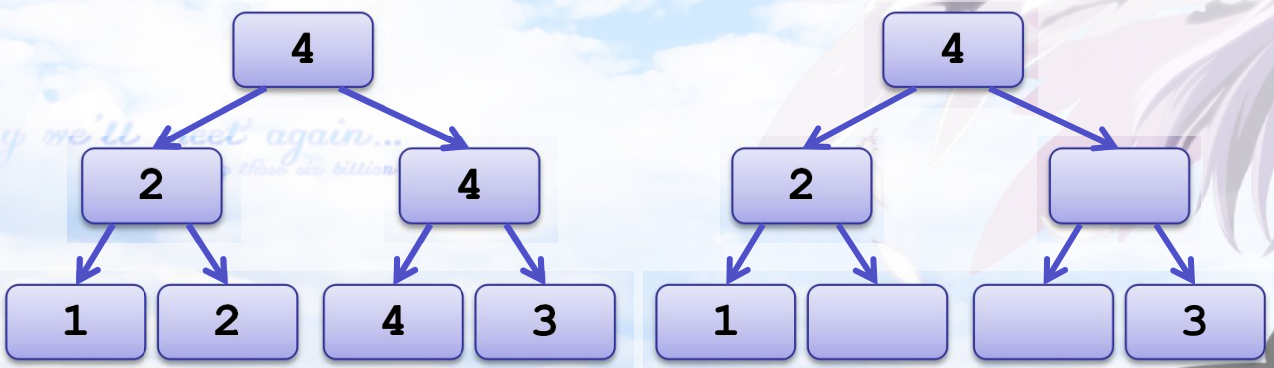


picks loves segment tree II

- 考虑DFS时的终止节点v，设它的父亲为f，那么有$mx_v=mx_f$或者$mx_v<mx_f$，其中第二类点在DFS前是关键点，在DFS后不是关键点。设第二类点数为A，那么访问的总点数一定是O(Alogn)的。此时我们花费的时间为O(Alogn)，而Φ减少了A。
- 考虑所有修改操作，每一次修改的时候影响到了线段树上O(logn)个节点，最坏的情况下每一个受影响的节点都由非关键点变成了关键点，那么每一次修改操作都至多使势函数增加O(logn)
- 因此势函数的总变化量是O(mlogn)的，所以可以得到每一次DFS的均摊复杂度是$O(\log^2 n)$。
- 由此可以得到这个算法的时间复杂度是$O(n\log n+m\log^2 n)$。

picks loves segment tree II

- 虽然目前的复杂度已经比分块优秀许多了，但是依然有些难以接受。
- 经过实现之后，发现这个算法处理500000的数据在UOJ上只需要0.6s，实际的运行效率更像O(nlogn)。
- 我们可以换种方式来考虑：把线段树上每一个节点的最大值看成是区间取min标记，次大值看成是子树中标记的最大值。既然把最大值看成是标记，那么一些无用的标记就可以被删去：



picks loves segment tree II

- 这些标记满足每一个标记的值一定比它子树中的所有标记的值大。因此每一个位置实际的值等价与它到根路径上碰到的第一个标记的值，而上述算法中的DFS过程，相当于是对子树中比当前标记大的标记进行了回收。
- 考虑区间加减对区间取min标记的影响，其实就和普通的线段树一样，我们对所有访问到的节点都进行了一次标记下传。
- 因为回收标记的时间复杂度不会超过打标记和标记下传的时间复杂度之和，所以就有标记回收（即DFS）的时间复杂度是O(mlogn)的。
- 因此，上述算法的时间复杂度为O(mlogn)

picks loves segment tree III

- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数变成$\min(A_i,x)$
- 区间[l,r]中的所有数变成$\max(A_i,x)$
- 询问区间[l,r]中所有数的和
- $n,m \leqslant 50000$
- 我会分块！
- $n,m \leqslant 500000$
- 线段树？

picks loves segment tree III

- 仿照picks loves segment tree I的算法，我们可以得到一个类似的做法：
- 对于线段树上的每一个节点，我们维护一下区间最大值mx，最大值出现次数t1，次大值se1，区间最小值mi，最小值出现次数t2，次小值se2。
- 显然标记是可以合并的，这玩意也是可以维护的。
- 然后对于区间取min操作，我们在打标记的时候就不停的DFS直到se1<x；对于区间取max操作，我们在打标记的时候就不停的DFS直到se2>x。

picks loves segment tree III

- 我们可以发现这玩意的复杂度应该和前一题的时间复杂度是一样的。
- 在分析复杂度的时候可以把两种标记（区间取min，区间取max）分开来考虑，可以发现每一种操作对另外一种标记位置的影响只有标记下传。
- 因此就可以得到两种DFS的均摊复杂度都是每次O(logn)的，即这个算法的时间复杂度是O(mlogn)。

picks loves segment tree IV

- 我们来对比一下第二题和第三题，可以发现在复杂度分析的时候，我们并没有用到区间加或者区间取max这些特定操作的性质，因此无论额外凑上去的是什么操作，这个复杂度分析依然是成立的。
- 因此，如果我们给定了一个操作集合，如果这些操作可以用线段树维护区间最大值，最大值个数和次大值，那么我们就可以加上区间取min这一操作，并询问区间和，时间复杂度依然是O(mlogn)的。
- 现在让我们来出一道题吧，名字就叫做picks loves segment tree IV好了。
- 唔…要出什么样的题呢。我们来看几道范本吧：

picks loves segment tree IV

3337: ORZJRY I

Time Limit: 30 Sec Memory Limit: 512 MB
Submit: 121 Solved: 27
[Submit][Status][Discuss]

Description

Jry最近做(屠)了很多数据结构题，所以想 BS你，他希望你能实现一种数据结构维护一个序列：

输入格式	说明	示例 （例如原序列为 5 2 6 3 1 4）
1 x val	**在第 x 个数后插入一个 val** （0≤x≤序列长度，val>0）	输入：1 2 7 序列：5 2 **7** 6 3 1 4
2 x	**删除第 x 个数** （1≤x≤序列长度）	输入：2 1 序列：**2 6 3 1 4**
3 x y	**翻转第 x 至第 y 个数** （1≤x≤y≤序列长度）	输入：3 3 5 序列：5 2 **1 3 6** 4
4 x y k	**将第 x 至第 y 个数旋转（向右移动）k 次** （1≤x≤y≤序列长度，1≤k≤y-x）	输入：4 1 6 1 序列：**4 5 2 6 3 1**
5 x y val	**将第 x 至第 y 个数加上 val** （1≤x≤y≤序列长度，val>0）	输入：5 3 4 5 序列：5 2 **11 8** 1 4
6 x y val	**将第 x 至第 y 个数都修改为 val** （1≤x≤y≤序列长度，val>0）	输入：6 1 4 7 序列：**7 7 7 7** 1 4
7 x y	**询问第 x 至第 y 个数的和** （1≤x≤y≤序列长度）	输入：7 2 4 输出：**11**
8 x y	**询问第 x 至第 y 个数中最大值与最小值的差** （1≤x≤y≤序列长度）	输入：8 1 3 输出：**4**
9 x y val	**询问第 x 至第 y 个数中与 val 的差的绝对值的最小值** （1≤x≤y≤序列长度，val>0）	输入：9 2 4 5 输出：**1**
10 x y k	**询问第 x 至第 y 个数中第 k 小的数** （1≤x≤y≤序列长度，1≤k≤y-x+1）	输入：10 1 6 4 输出：**4**
11 x y val	**询问第 x 至第 y 个数中比 val 小的数的个数** （1≤x≤y≤序列长度，val>0）	输入：11 2 5 4 输出：**3**

picks loves segment tree IV

Input

第一行是N和M，表示有这棵树有N个点M个询问

然后是N-1行，每行x,y表示x-y有一条边

接下去是N行，每行是一个数字，表示每个点的权值

后面一行表示根

接下来是M行

第一个数字是K

K=0 表示子树修改，后面x,y，表示以x为根的子树的点权值改成y

K=1 表示换根，后面x，表示把这棵树的根变成x

K=2 表示链修改，后面x,y,z，表示把这棵树中x-y的路径上点权值改成z

K=3 表示子树询问min，后面x，表示以x为根的子树中点的权值min

K=4 表示子树询问max，后面x，表示以x为根的子树中点的权值max

K=5 表示子树加，后面x,y，表示x为根的子树中点的权值+y

K=6 表示链加，后面x,y,z，表示把这棵树中x-y的路径上点权值改成+z

K=7 表示链询问min，后面x,y，表示把这棵树中x-y的路径上点的min

K=8 表示链询问max，后面x,y，表示把这棵树中x-y的路径上点的max

K=9 表示换父亲，后面x,y，表示把x的父亲换成y，如果y在x子树里不操作。

K=10 表示链询问sum，后面x,y,z，表示表示把这棵树中x-y的路径上点的sum

K=11 表示子树询问sum，后面x，表示以x为根的子树的点权sum

picks loves segment tree IV

- 用刚才得到的算法，我们可以出出来一道怎样的题目呢：
- 区间[l,r]中的所有数变成$\min(A_i,x)$
- 区间[l,r]中的所有数变成$\max(A_i,x)$
- 区间[l,r]中的所有数加上x
- 区间[l,r]中的所有数的符号取反
- 区间[l,r]中的所有数乘上x
- 区间[l,r]中的所有数都变成x
- 询问区间[l,r]中所有数的和
- 询问区间[l,r]中的最大值，次大值，最小值，次小值，最大值个数，最小值个数
- $n,m \leqslant 500000$

picks loves segment tree IV

- 当然如果你的想象力够好，没准可以再脑补出来一些其他的修改操作上去。
- 线段树大一统?
- 这个时候人群中跳出来了一个picks!

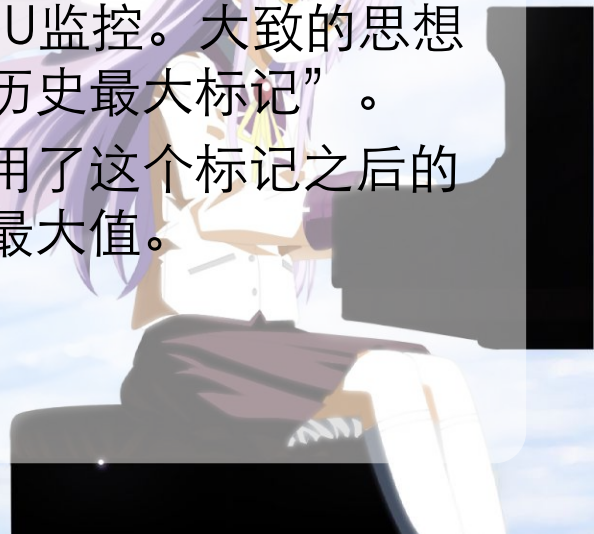
- “你能支持查询历史最大值吗？”

V——清华集训2015 D4T1

- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数加上x
- 区间[l,r]中的所有数变成$\max(A_i-x,0)$
- 区间[l,r]中的所有数变成x
- 询问A_i的值
- 询问A_i的历史最大值
- $n,m\leqslant 50000$
- 我会分块！
- $n,m \leqslant 500000$
- 线段树？

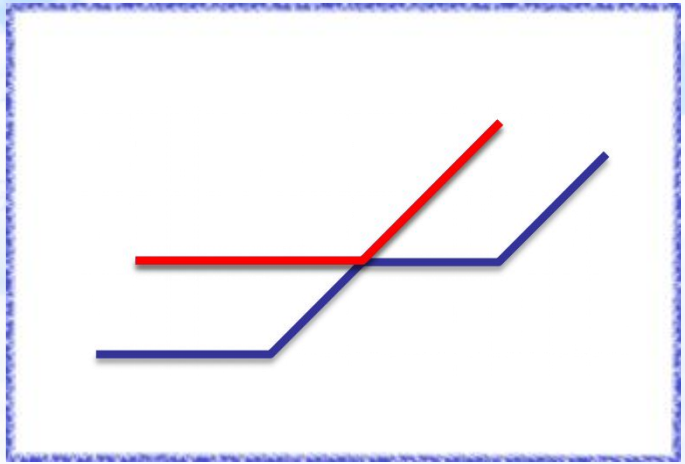
V——清华集训2015 D4T1

- 分析之后发现，我们要支持的其实就只有一个操作：给区间中的所有数加上x再对y取max，我们把这个标记用(x,y)来表示
- 那么这三个修改操作分别是(x,inf)，(-x,0)，(-inf,x)。
- 考虑合并两个标记(a,b)与(c,d)，那么得到的就是(a+c,max(b+c,d))
- 这样我们就可以非常trivial地来支持单点询问辣。
- 至于询问历史最大值，可以参考一道题目叫做CPU监控。大致的思想就是我们对每一个线段树节点，额外记录一个“历史最大标记”。
- 所谓历史最大标记，即区间中的每一个节点在作用了这个标记之后的值都是上一次标记下传到当前状态这一时间段的最大值。



V——清华集训2015 D4T1

- 怎么维护历史最大标记呢，我们可以用每一时刻的标记来更新历史最大标记。即在标记下传的过程中，设父节点的历史最大标记为p，当前节点的标记为q，历史最大标记为w，那么我们相当于要用p+q去更新w。
- 更新标记的过程相当于两个分段一次函数取max的过程：



- 蓝色部分分别为p+q和w，那么更新之后得到的标记就是红色部分。其中横坐标为原来的值，纵坐标为作用了这个标记之后得到的值。（注意情况不止这么一种）

V——清华集训2015 D4T1

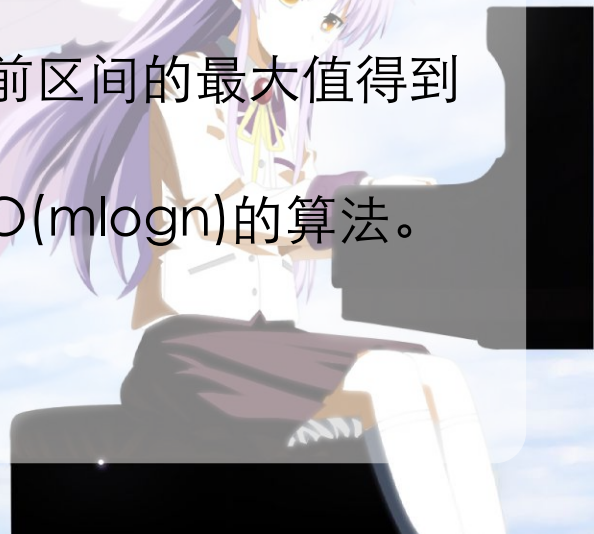
- 通过这种方法我们就能维护历史最大标记了，单点查询的时候只需要把当前叶子到根的标记全部下传到底，叶子节点上的历史最大标记就是答案了。
- 时间复杂度O(mlogn)
- 询问只有单点未免有点小家子气了，我们试着把询问给推到区间上去。

VFlea

- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数加上x
- 区间[l,r]中的所有数变成$\max(A_i-x,0)$
- 区间[l,r]中的所有数变成x
- 询问区间[l,r]的和
- 询问区间[l,r]中所有数历史最大值的最大值
- $n,m\leqslant 50000$
- 我会分块！
- $n,m \leqslant 500000$
- 线段树？

VFlea

- 其中询问区间和的部分和pls loves segment tree II相同，这儿就不再赘述。
- 因为要询问区间历史最大值，所以对每一个区间我们都需要维护一个新的值表示在更新这个区间时产生的历史最大值，设为pre。
- 考虑标记(a,b)，可以发现在作用了这个标记之后，整个区间中权值的相对大小顺序不变。
- 所以我们可以直接用下传的历史最大标记作用当前区间的最大值得到的答案来更新历史最大值。
- 于是我们就得到了这个问题的一个时间复杂度为O(mlogn)的算法。



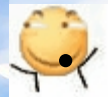
VFlea

- 到此我们就解决了V这题的区间版本辣..

- 但是只有这么一点操作，显然是不够（毒瘤）的，于是我们来接着加强→_→

VFleaKing

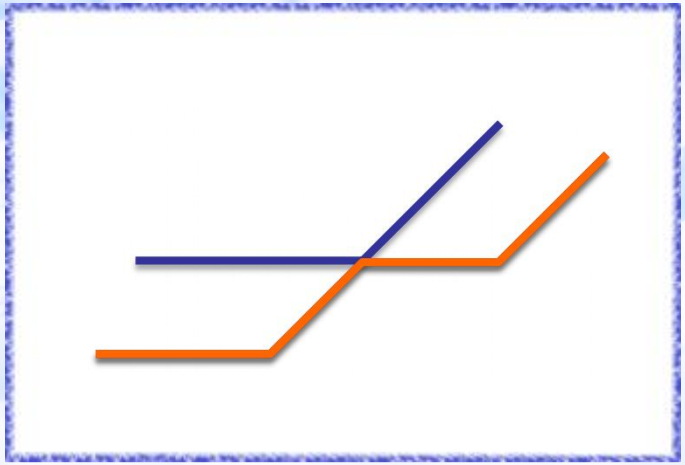
- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数加上x
- 区间[l,r]中的所有数变成$\max(A_i-x,y)$
- 区间[l,r]中的所有数变成x
- 询问区间[l,r]的和
- 询问区间[l,r]中所有数历史最大值的最大值
- 询问区间[l,r]中所有数历史最小值的最小值
- $n,m\leqslant 50000$
- 我会分块！
- $n,m \leqslant 500000$



- 线段树？

Vfleaking

- 我们只考虑增加的询问：区间历史最小值查询。
- 实际上这个问题用V和Vflea的方法是难以处理的。其中比较突出的一个问题是标记没法合并：



- 因为现在我们需要维护的是历史最小标记，所以在更新的时候有可能得到如图中橙色部分的标记。这样在反复更新之后标记就有可能变成含O(n)段的分段函数。这是难以接受的。

VFleaKing

- 我们来尝试把标记(a,b)在不影响历史最小值的情况下拆成区间加操作和区间取max操作。
- 当a>0时，标记(a,b)等价于先加上a，再对b取max；当a<0时，标记(a,b)等价于先对b-a取max，再加上a。
- 于是问题就转化为了：区间加减，区间取max，询问区间历史最小值。
- 因为标记无法合并，所以这个问题不能用传统的线段树来维护，我们继续考虑picks loves segment tree系列题目的做法。
- 之前，我们把这个做法中，应用操作区间取max的标记称呼为“区间取max”标记。诚然这个标记的确起到了这个作用，但是不难发现这个标记还有一些特殊的性质：一个区间打上这个标记之后，只有区间内所有值等于区间最小值的节点发生了变化。（因为打标记的节点都满足有se>x）

VFleaKing

- 因为一个区间的最小值在打标记的时候我们是知道的，最小值变化后的值我们也是知道的，所以就可以避免使用“区间取max”这一个笼统的、变化量不明确的、难以处理的标记。
- 在这儿，我们引入一个全新的操作：区间最小值加减。操作定义为：对于区间[l,r]，设这个区间范围内A_i的最小值为mi，那么我们把这个区间中所有满足A_i=mi的位置都加上x。
- 在这个问题中，我们可以用前几道例题的方法，把区间取max完全转化为区间最小值加减操作。而且在这个转化之后，标记具有一个更加优美的性质：设区间[l,r]的次小值为se，那么必然有mi+x<se。
- 有了这一个性质，我们就可以在打上这类标记的同时，维护区间的最小值和次小值。

VFleaKing

- 经过了这几步思考，算法已经大致成型了。我们对每一个节点记录并维护以下几个量：
  - 区间最小值加减标记A，表示给区间中所有最小值加上A
  - 区间其他值加减标记B，表示给区间中所有值不为最小值的位置加上B
  - 区间最小值加减历史最小标记preA，表示上一次标记下传到当前时刻A的最小值
  - 区间其他值加减历史最小标记preB，表示上一次标记下传到当前时刻B的最小值
  - 区间历史最小值num，区间最小值mi和区间次小值se
- 维护这些量，我们就可以愉快的维护区间历史最小值啦。
- 因为这一部分写的比较抽象，所以就附上一段我的代码吧。代码风格很丑求各位菊苣轻喷。

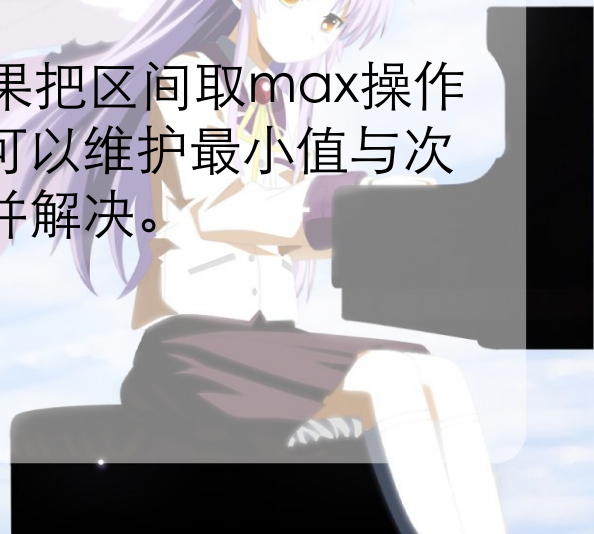
VFleaKing

```
void addall(int k1,int a,int prea,int b,int preb){
    num[k1]=min(num[k1],mi[k1]+prea);
    preA[k1]=min(preA[k1],prea+A[k1]);
    preB[k1]=min(preB[k1],preb+B[k1]);
    A[k1]+=a; B[k1]+=b; mi[k1]+=a; se[k1]+=b;
}
void pushdown(int k1){
    int now=min(mi[k1*2],mi[k1*2+1]);
    if (mi[k1*2]==now) addall(k1*2,A[k1],preA[k1],B[k1],preB[k1]);
    else addall(k1*2,B[k1],preB[k1],B[k1],preB[k1]);
    if (mi[k1*2+1]==now) addall(k1*2+1,A[k1],preA[k1],B[k1],preB[k1]);
    else addall(k1*2+1,B[k1],preB[k1],B[k1],preB[k1]);
    A[k1]=0; preA[k1]=0; B[k1]=0; preB[k1]=0;
}
```

- 因为整个算法的核心就是标记下传，所以就只截取了这一部分。代码中k1×2和k1×2+1分别代表左儿子和右儿子，其余变量含义和上述的定义相同。

VFleaKing

- 到此我们就得到了解决这道题的一个时间O(mlogn)，空间O(n)的优秀算法。
- 我们来回忆一下我们干了些什么：我们利用了前几道的做法，通过维护区间最小值，次小值，把每一个线段树节点下的数拆成了最小值、非最小值两类分别维护，从而把区间取max操作成功转化成了容易处理的加减操作。
- 于是对于一个包含区间取max的线段树问题，如果把区间取max操作替换成区间加减的基础上，原问题的询问可做并可以维护最小值与次小值，那么我们几乎都可以用上述的方法来转化并解决。
- 以下是几个栗子：



Picks loves segment tree V

- 给定一个长度为n的数列A和B，接下来有m次操作：
- A数组区间[l,r]中的所有数变成max(Ai,x)
- A数组区间[l,r]中的所有数加上x（x可能是负数）
- 询问区间[l,r]中Ai-Bi的最大值与最小值
- n,m⩽50000
- 我会分块！
- n,m ⩽ 500000
- 线段树？

Picks loves segment tree V

- 按照刚才的步骤来，如果去掉了A数组区间取max操作，那么就是经典的区间加减询问区间最大值最小值了，相信在座的各位都会做。同时维护A数组的最小值和次小值也是可以时限的。
- 因为，对线段树的每一个节点，我们都可以分别维护这个节点中所有Ai等于这个区间内的最小值时和Ai不等于这个区间内的最小值时所对应的Ai-Bi的最大值与最小值（当然还有加减标记）。
- 如果要再加上询问Ai-Bi的历史最大值和历史最小值自然也是手到擒来。
- 是不是已经感觉到了这个算法的威力了呢？(/≥▽≤)/~
- 当然我们还可以接着加操作：

Picks loves segment tree VI

- 给定一个长度为n的数列A和B，接下来有m次操作：
- A数组区间[l,r]中的所有数变成max(Ai,x)
- B数组区间[l,r]中的所有数变成max(Bi,x)
- A数组区间[l,r]中的所有数加上x（x可能是负数）
- B数组区间[l,r]中的所有数加上x（x可能是负数）
- 询问区间[l,r]中Ai-Bi的最大值与最小值
- n,m≤50000
- 我会分块！
- n,m ≤ 500000
- 线段树？

Picks loves segment tree VI

- 其实做法也很简单，依然考虑通过对这些数进行分类来去掉区间取max操作。
- 我们可以把一个节点中的所有下标分成四类：A值B值同时为最小值，A值为最小值但B值不是，A值不是最小值但B值是，A值B值都不是最小值。
- 我们维护一下区间中A值和B值的最小值和次小值，然后对这四类下标用四套标记来进行处理，就可以非常愉快的解决辣owo。
- 当然对数列更多的情况这个做法依然适用，比如有ABC三个数列维护Ai-Bi+Ci的最大值最小值。只不过如果数列个数是K，这个算法就需要$O(2^K mlogn)$的时间复杂度和$O(2^K n)$的空间复杂度。如果有老司机能把这个K从指数上拽下来欢迎来找我讨论owo
- 最后让我们回归这一个part最开始的问题：

Picks loves segment tree VIII

- 给定一个长度为n的数列A，接下来有m次操作：
- 区间[l,r]中的所有数变成min(A_i,x)
- 区间[l,r]中的所有数变成max(Ai,x)
- 区间[l,r]中的所有数加上x（x可能是负数）
- 询问区间[l,r]中所有数历史最大值的最大值
- 询问区间[l,r]中所有数历史最小值的最小值
- n,m⩽50000
- 我会分块！
- n,m ⩽ 500000
- 线段树？
- （在意题目编号的都是baka！）

Picks loves segment tree VIII

- 本来UR11的C我是想这么出的，但是因为我比较懒再加上数据很难造，所以就删掉了一个操作…然后好像题目难度就打了一些折扣。
- 做法和VFleaKing类似，只不过在这题中，我们需要把所有数给分成三类：最大值，最小值和其他数，然后用三套标记来进行维护。只不过在边界情况下最大值和最小值可能会重合，下传的时候要小心。
- 既然都讨论到这儿了，询问历史最值就可以自然而然地加入picks loves segment tree IV的众多操作中啦。[手动撒花]
- 但是因为历史最值询问对常数和代码复杂度的增幅实在太大，再加上营员交流的时间有限，所以终极版本的大杂烩就留给有兴趣的同学自己脑补辣。[滑稽]

感谢

- 感谢党和国家，感谢CCF提供了这次交流的机会。

- 感谢徐寅展学长的帮助和指导。
- 感谢吕凯风学长提供题目名称。
- 感谢彭雨翔学长提供Idea和照片。
- 感谢曾国洋同学帮忙分析复杂度。
- 感谢王蕴韵同学提出PPT美化建议。

- 感谢大家的倾听。

题外话

- 但是线段树的题目怎么出呢，昨天某位同学的营员交流介绍了一个好方法。

  - 总是有出题人喜欢把序列上用线段树解决的题目出到树上，让选手强行写个树链剖分。
  - 这种行为已经很无趣了。
  - 所以我们想让大家知道，不光可以放在树上，仙人掌也是可以的。

仙人掌也是可以的。

- 下面该Po姐表达一下对这种题的看法了……


The End!
祝大家在冬令营中取得好成绩！
Thanks For Watching!