# CF上3000的数据结构题

成都七中 nzhtl1477

# CF526F Pudding Monsters 3000

- 这里k=1...n

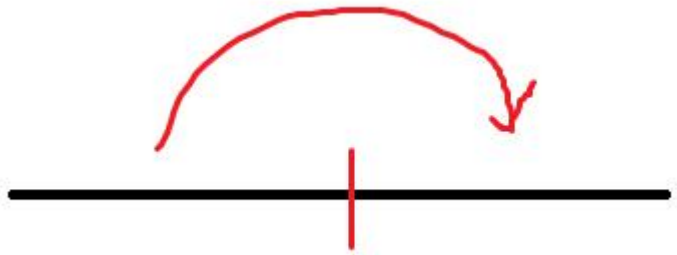
- 给定一个 $n \times n$ 的棋盘，其中有 $n$ 个棋子，每行每列恰好有一个棋子。
- 求有多少个 $k \times k$ 的子棋盘中恰好有 $k$ 个棋子。
- $n \leq 3 \times 10^5$。

# Solution

- 因为这里是k*k的矩形里面有k个，并且每行每列只有1个，所以可以将这个平面在一维上进行投影
- 得到一个排列a，a[x]=y等价于在(x,y)处有一个值
- 区间[l,r]如果满足条件，当且仅当区间内所有数的极差为r-l，不然无法用一个(r-l+1)*(r-l+1)的矩形里面包含这r-l+1个数
- 即全局有多少(l,r)满足[l,r]中的max-min=r-l

# Solution

- 考虑将每个区间表示为二维平面上的点(x,y)
- 首先先初始化(x,y)的权值为y-x：对x=1...n进行矩形减，对y=1...n进行矩形加，即[x,1..n]-=x，[1..n,y]+=y
- 然后对序列进行最值分治，每次分治相当于一个矩形内的点对应的序列上的区间的max都是分治中心，进行矩形加，min同理进行矩形减



# Solution

- 问题即转换为给定一个平面，进行O(n)次矩形加减，问有多少个0
- 由于保证矩形中元素非负，所以可以扫描线+线段树解决
- 类似于矩形面积并，线段树维护区间min和min的个数，即可计算0的个数

- 总时间复杂度O(nlogn)

# CF464E The Classic Problem 3000

- 给定一张n个点m条边的无向图，每条边的边权为pow(2,xi)，求s到t的最短路，答案对10^9+7取模

# Solution

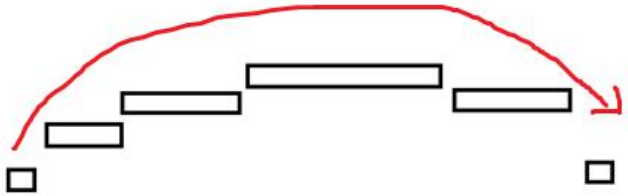
- 考虑直接对这个图跑Dijkstra求最短路
- 边权比较大需要高精度维护
- 如何利用边权的特殊性？
- Dijkstra需要支持什么操作？
- 支持dist[x]=dist[y]+v[x->y]//x->y边权，以及比较dist[x]和dist[y]

# Solution

- 可以发现如果我们对边权开一个值域上的数据结构维护，如01trie
- 则dist[x]是用01trie存的一个二进制数
- dist[x]=dist[y]+v[x->y]等价于将dist[y]复制过来，然后进行修改，想到使用可持久化的数据结构维护

# Solution

- 加法如何实现？
- 这个边权的特殊性导致我们只会发生一段进位
- 进位即在trie上二分出这段进位的区间（这里二分是不多log的）
- 可以维护一下子树内是否全是1，然后用那个向上走然后向下走的二分方法即可找出这个区间
- 然后打一个区间修改为0的标记即可



- 如果觉得可持久化数据结构不能区间修改打标记下放标记的人请仔细想想自己的理由成不成立

# Solution

- 比较如何实现？
- 数据结构如何维护高精度数，支持比大小？
- 区间哈希LCP的方法即可
- 注意到这里比大小是不用外层套二分的，因为trie结构相同，所以可以直接在两个trie上一起二分来找到第一个不相同的位置

- 总时间复杂度O((m+nlogn)logx)

# CF603E Pastoral Oddities 3000

- 给定一张 n 个点的无向图，初始没有边。
- 依次加入 m 条带权的边，每次加入后询问是否存在一个边集，满足所有 n 个点的度数均为奇数。
- 若存在，则还需要最小化边集中的最大边权。

# Solution

- 查询是奇怪的形式，先推性质
- 存在一个边集使得每个点的度数都是奇数的充要条件是不存在奇数个点的联通块
- ->：度数都是奇数，考虑反证法：
- 1. 如果有奇数个点，则所有点度数和是奇数
- 2. 每条边会让两个点度数+1，所有点度数一定是偶数
- 矛盾
- <-：偶数个点的连通块，考虑先找出原图的一个生成树，然后从叶子开始，一个点与其父亲的连边保留当且仅当这个点与其所有儿子的连边数为偶数

# Solution

- 问题转换为，动态加边，维护一个最小的x，使得<=x的所有边构成的连通块中，每个连通块只有偶数个点
- 可以发现加边一定不会变劣，因为只会：
- 1.合并两个大小为偶数的连通块，不变差
- 2.合并大小为奇数和偶数的连通块，不变差
- 3.合并两个大小为奇数的连通块，变优

# Solution

- 由于加边不会变劣，答案单调不增
- 使用数据结构维护当前的森林
- 维护每个连通块的MST，构成的最小生成森林
- 这个最小生成森林是将边从小到大依次加入，第一次满足所有连通块点数为偶数时的森林

# Solution

- 使用LCT维护这个生成森林
- 每次加边
- 如果连通了两个原本不连通的连通块
- 1. 如果加入的边比当前生成森林里的瓶颈路大，若当前所有连通块大小都是偶数则无视，否则加这条边
- 2. 如果加入的边比当前生成森林里的瓶颈路小，则有可能在生成森林中删除边权最大的几条边
- 注意到一条边被删去后不可能被加回来
- 所以可以每个连通块上，LCT维护子树，在LCT上二分出边权最大的边，然后看子树大小是否是偶数，是的话就断开这条边

# Solution

- 如果没有连通两个原本不连通的连通块，则这条边在连通块内部
- 在新产生的环上找最大的一条取代掉
- 注意这个操作之后可能会在生成森林中删除边权最大的几条边
- 这样的森林上的瓶颈路即答案

- 总时间复杂度O((n+m)logn)

# CF1446D2 Frequency Problem 3000

- 给一个序列
- 求最长的子段使得其中有至少两个出现次数最多的元素。
- 输出最长子段长度。
- 我加强一下，n<=5e7

# Solution

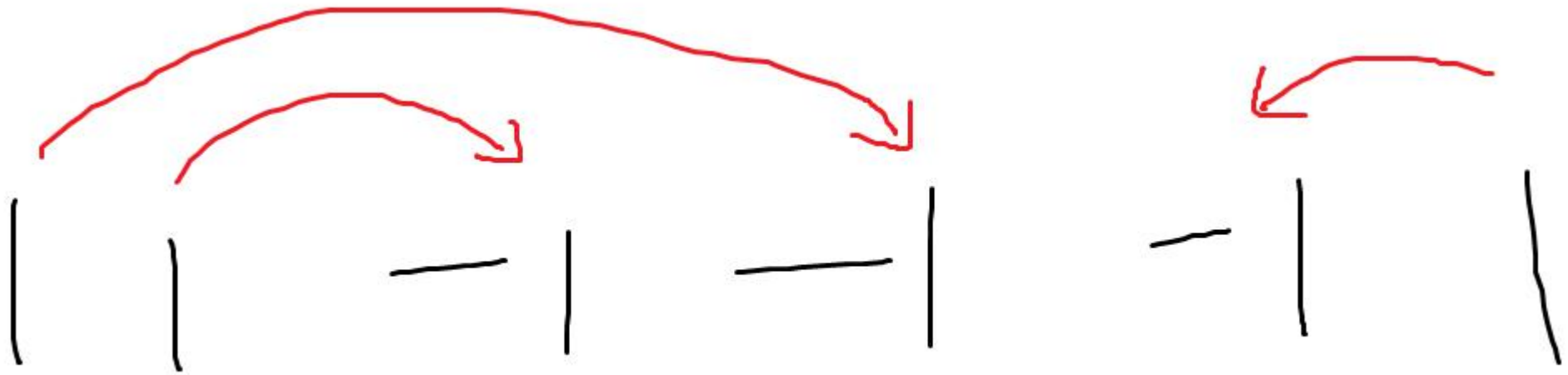
- 可以证明，这两个出现次数最多的元素中，必定有一个是全局的众数
- 考虑答案一定是全局删掉一个前缀和一个后缀
- 删除的过程中，全局的众数一定一直为众数，直到出现了一个与其出现次数相等的数

# Solution

- 假设众数x出现次数为a，我们目前考虑一个值y，计算y与x的答案最大是多少，y出现b次
- 我们如果得到一个O(a+b)的算法，这个题就是根号题了
- 我们要得到一个O(b*polylog(n))的算法

# Solution

- 初始将每个x出现的位置标记为无意义的位置
- 我们枚举y出现的每个位置，然后找离这些位置最近的x出现的无意义位置（左右两边都找），然后将这些位置标记为有意义的位置
- 可以证明标记结束后无意义的位置和答案无关



# Solution

- 一个位置A无意义，即对其前面所有y出现的位置B，[B,A]之间一定x出现次数>y，同理对后面也成立，所以这个位置不可能被包含在答案区间中，因为从这个位置开始，任何前缀和后缀中x出现次数都>=y，所以任何区间中x出现次数都>y
- 于是无意义的位置将序列分为多段不相关的区间
- 所有有意义的位置，与y出现的位置，这些位置左右1的位置可能是答案端点

# Solution

- 所以只有O(b)个可能的答案端点
- 考虑为了找出答案端点，我们需要一个数据结构，支持查询前驱与删点
- 这个可以使用序列线性并查集来做，对每个y的每个位置，预处理出其前面第一个x出现位置挂上去
- 本题还需要支持修改后回退，由于每次修改Ω(logn)次后才有可能进行一次并查集上的合并，所以复杂度正确
- 将x的位置设为1，y的位置设为-1，对有意义的位置跑一个前缀和，维护出和为1,2,...b的最长与最短前缀，这样就可以找出和为0的最长子段，即答案了

# Solution

- 如果区间中出现次数最多的数不是x,y也没关系，因为答案是取max的，这样只是这个区间中x,y的贡献不够优
- 所有数的出现次数和为n，故每次的b和是O(n)的
- 总时间复杂度O(n)

# CF150E Freezing with Style

- 给定一颗带边权的树，求一条边数在 [L,R] 之间的路径，并使得路径上边权的中位数最大。输出一条可行路径的两个端点。

# Solution

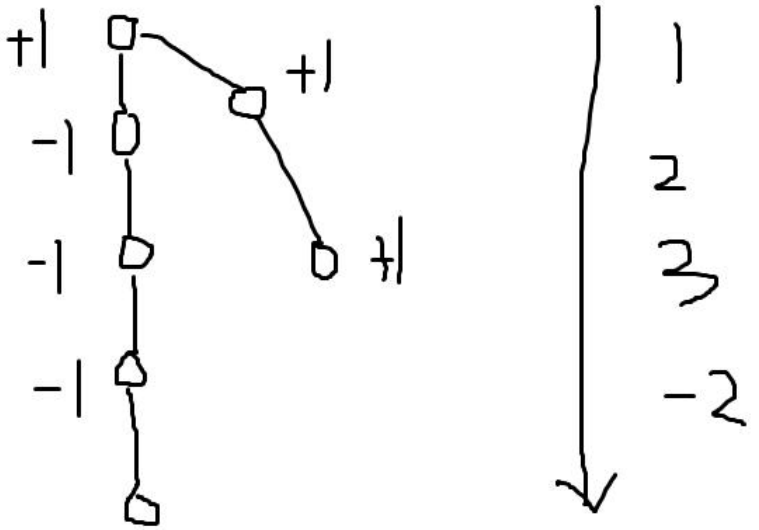
- 二分变成+1-1点权，是否有长度 [L,R] 的路径和>0
- 然后树分治单调队列啥的就做掉了
- O(nlog^2n)

# Solution

- ccz做了一个O(nlogn)的做法，但我觉得不太有用
- CF上他们说他们会1log的做法，我觉得是假的，这个不像是正常算法竞赛的技术
- 他们讨论的帖子里前面说了一堆平凡的东西，然后真正重点的部分直接没说...

# Solution

- 我发现ccz那个1log做法是假的
- 他做了一个：
- 一端插入，删除
- 查询定长区间+1-1rmq（就是这个序列相邻两个位置最多差1）
- 这个可以O(1)
- 删除worst case，插入带均摊
- 但这道题比这个强



# Solution

- 好吧还真的是正常算法竞赛的技术，我们做的问题复杂了
- 设R-L=C
- 这里我们区分一下查询的是前缀还是区间，这个很重要
- 对这两种分别维护

# Solution

- 我们二分答案，然后长链剖分
- 问题变为线性求一棵点权为+1-1的树上，长[l,r]的简单路径，是否有点权和>0的
- 最后的做法和点权无关，我们求长[l,r]的简单路径的点权和的max

# Solution

- 长链剖分后，如果我们可以实现
- 1.O(1)前面插入一个元素
- 2.O(x)代价修改最前面x个元素，注意这里修改只会变大
- 3.O(1)查询一个长C的区间[L,R]的max
- 4.O(1)查询一个前缀的max
- 这样的数据结构，这样长链剖分合并短链，可以O(n)解决这个问题

# Solution

- 对于前缀，我们维护一个单调栈，里面存下每个x，若x位置为前缀最大值，每次最前面插入元素即连续pop一些元素，每个数只会被插入删除一次
- 修改的话也类似，因为我们只改最前面的一些，可以先pop到修改位置，然后把修改的x’个数拿来先内部建立一个这部分修改后的单调栈，这里时间复杂度是O(x’)的，然后用这个单调栈的最大值来弹掉后面的，然后合并起来，这样总复杂度是O(x’)修改了x’个位置的，没问题

# Solution

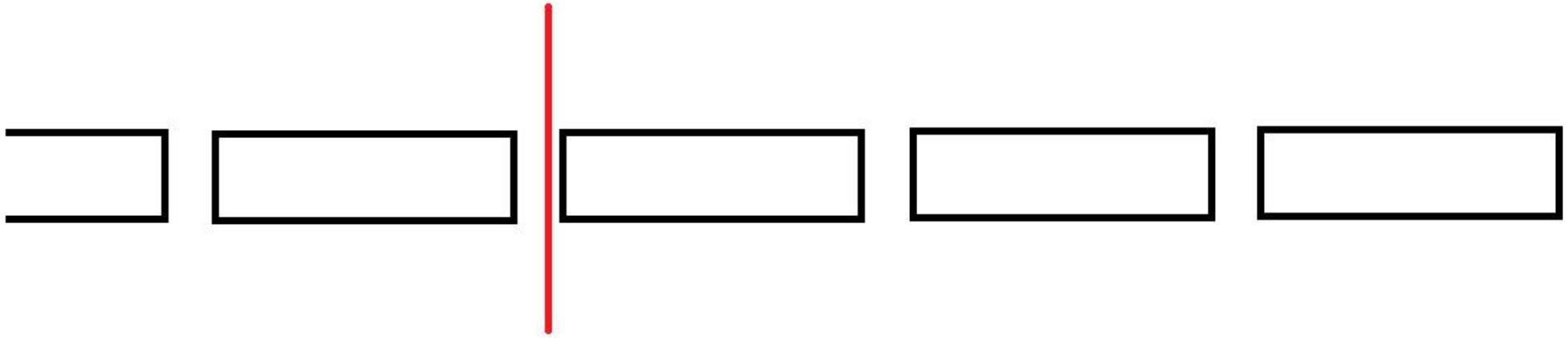
- 考虑用O(1)并查集维护这个序列
- 当一个位置x被弹掉的时候，将x与x-1合并
- 并查集每个连通块维护当前连通块里最小的位置
- 查询x前面第一个在单调栈里的元素，只需要在并查集上查询即可
- 如果只有在前端插入元素，这个听起来很对
- 但是这个修改实际上会破坏性质，因为我们有可能已经合并了x与x-1，然后这时候x被修改成了单调栈中的元素

# Solution

- 我们考虑不用O(1)并查集，用启发式合并，每个连通块开个vector维护，同时维护每个点在哪个连通块中
- 就是我们对序列按logn大小分块
- 如果一个块全被删除了，则合并这个块和前面的块
- 查一个块内的答案，即后面有多少个1这样的，位运算解决
- 然后每次合并两个连通块a,b的时候
- 假设size(a)>size(b)
- 则for b集合中的点，插入a集合中，同时修改所属集合
- 然后开一个数组维护每个集合对应的点编号最小值

# Solution

- 为了可以修改，我们再维护一个集合数组f，f(x)表示所有以x为最小值的连通块的编号的集合，这个可以在连通块操作的时候以常数代价维护出，因为只涉及到插入删除以及遍历
- 我们预留出当前前缀中在建立的整块，以及除此之外前缀中最靠前的一个整块，为了让<logn长度的修改不至于花费过多代价，查询时我们可以用位运算特判这两个块，是可以维护的



# Solution

- 如果修改位置个数<logn，我们重构出单调栈，把单调栈上的修改施加到这个数据结构的对应块对应位置上即可，复杂度是线性于修改位置个数的，这里有单调栈的均摊的保证
- 如果修改位置个数>logn，我们可能要重构后面的块，后面每重构一个块意味着我们多修改了logn个位置
- 重构块x只需要把f(x)集合中每个连通块的答案进行修改集合，由于是序列，f(x)有O(1)个元素，这里暴力修改即可
- 查询前缀x时，我们先在启发式合并结构中找出其所在的连通块，然后找出最小下标，然后就能找到其前面第一个在单调栈中的元素了

# Solution

- 然后考虑查询的区间是满的长度C的情况
- 考虑维护一个数组b[i]表示[i-C,i]的max
- 每次修改前x个位置，即对x个b[i]进行修改

# Solution

- 这个直接维护的话修改量一次是x^2的
- 但我们可以对修改的这几个位置倒着从后往前跑，维护当前修改的max，然后去更新
- 更新直接取max即可
- 单点插入平凡
- 故这部分总时间复杂度O(n)

- 总时间复杂度O(nlogn)

# Solution

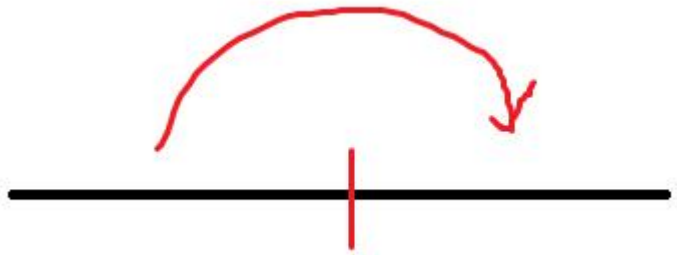
- ccz的做法好像改成log(r-l+1)大小分块就不用做+1-1的问题了
- 不过要依赖于题目性质，因为不能查询过短的前缀
- 我太菜了
- 可能差十个段位吧

# CF997E Good Subsegments 3000

- 给一个排列，查询区间中有多少子区间值域连续
- 注意读题

# Solution

- 考虑将每个区间表示为二维平面上的点(x,y)
- 首先先初始化(x,y)的权值为y-x：对x=1...n进行矩形减，对y=1...n进行矩形加
- 然后对序列进行最值分治，每次分治相当于一个矩形内的点对应的序列上的区间的max都是分治中心，进行矩形加，min同理进行矩形减



# Solution

- 问题即转换为给定一个平面，进行O(n)次矩形加减，问一个矩形内有多少个0
- 先将矩形差分，为3-side矩形，即一维是[l,r]一维是[1,x]
- 然后使用扫描线+线段树沿着[1,x]的维扫这个平面
- 由于保证矩形中元素非负，线段树维护区间min和min的个数，即可计算0的个数
- 问题是3-side矩形在扫描线后，需要查询一个区间在一个前缀时间中的0的个数

# Solution

- 这个其实可以简单维护
- 我们只需要对每个区间记录下多少个0，这个表示为量A
- 然后有一个量B表示累计的历史贡献
- 然后每次扫描线走一步的时候打一个区间B+=A的标记

- 总时间复杂度O((n+m)logn)

# CF319E Ping-Pong 3000

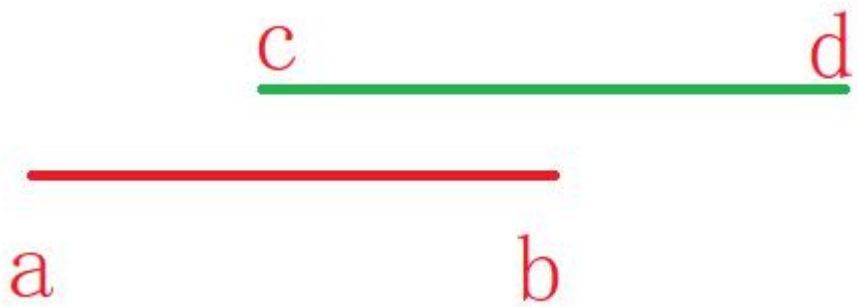
有一个区间的集合，初始为空。当 $c < a < d$ 或 $c < b < d$，且 $(a, b), (c, d)$ 都在集合中时，你可以从区间 $(a, b)$ 移动到 $(c, d)$。你需要支持下面的操作：

- 1 x y，加入一个新区间 $(x, y)$。保证加入的区间长度严格单调递增。
- 2 a b，询问是否能从第 $a$ 个加入的区间移动到第 $b$ 个。

操作数 $1 \leq n \leq 10^5$，保证其他任何数字都是整数且不超过 $10^9$。

# Solution

- 如果两个区间(a,b)和(c,d)不是互相包含，则(a,b)可达(c,d)蕴含了(c,d)可达(a,b)，可达有对称性
- 如果区间(a1,b1)包含了(a2,b2)，(a2,b2)包含了(a3,b3)，则(a1,b1)包含了(a3,b3)，包含有传递性




# Solution

- 每次加入区间(a,b)的时候，直接将有至少一个端点在(a,b)中的区间用数据结构找出，并且将其与(a,b)合并
- 合并后(a,b)可能会扩大，扩大后继续合并，直到不能合并
- 每次合并一个区间则减少一个区间，均摊复杂度正确
- 合并的时候用并查集维护，查询直接在并查集上查询两个区间是否在同一个连通块即可

- 总时间复杂度O(nlogn)

# CF696E ...Wait for it... 3000

给定一颗$N$个点的树，现在有$M$个物品，每个物品一开始都在节点$A_i$上，权值为$i$。现在有两种操作，先读入$Ord$:

$Ord = 1$: 再读入三个整数$u, v, Lim$表示从$u$到$v$这条路径上从小到大把最多$Lim$个物品从树上取出，并且从小到大输出。

$Ord = 2$: 再读入两个数$u, Val$表示给在$u$这棵子树的所有物品加上权值$Val$。

# Solution

- 直接静态Top tree
- Top tree维护每个簇中连接两个end point的路径的max
- 每次1操作的时候就是从O(logn)个Top tree节点的路径中取一个max最大的，递归下去删掉，重复多次
- 叶子上要维护一个堆，因为可能有多个物品

- 总时间复杂度均摊O((n+m)logn)

# CF1163F Indecisive Taxi Fee 3000

- 给你一个 n 个点，m 条边的无向图，每条边连接点 u, v，并且有个长度 w。
- 有 q 次询问，每次询问给你一对 t, x，表示仅当前询问下，将 t 这条边的长度修改为 x，请你输出当前 1 到 n 的最短路长度。

# Solution

- 这一看就是个最短路树的题
- 先建出最短路树，然后从1和n为源跑一遍单源最短路，记下对每个x的dist(1,x)和dist(x,n)

# Solution

- 讨论一下这个修改u <-> v从a改成b，造成的影响：
- 1.a>b且dist(1,u)+a+dist(v,n)==dist(1,n)，即u <-> v可以在最短路上，则答案为dist(1,n)-a+b
- 2.a>b且dist(1,u)+a+dist(v,n)!=dist(1,n)，即u <-> v一定不在最短路上，答案为min(dist(1,n),
- min(dist(1,v)+b+dist(u,n)),
- dist(1,u)+b+dist(v,n)))

# Solution

- 3.a<b且dist(1,u)+a+dist(v,n)!=dist(1,n)，答案不受影响，为dist(1,n)
- 4.a<b且dist(1,u)+a+dist(v,n)==dist(1,n)

- 经过u <-> v：答案是dist(1,n)-a+b

# Solution

- 不经过u <-> v：
- 考虑把最短路树上的那个最短路序列化，然后对于任何一条不在最短路上的边x <-> y，我们记录下x是由最短路上最靠后的哪个点A更新，y是由最短路上最靠前的哪个点B更新，然后1 -> x -> y -> n是一条可行的路径
- 我们要维护出对最短路上每条边，不经过其的最大答案，故上述的路径是对最短路上A -> B这一段取max
- 这里是一个经过O(m)次区间对x’取max，输出每个位置值

# Solution

- 可以发现最优解只可能是1 -> x -> y -> n这样的形式，其中1 -> x是沿着从1开始的最短路树走，y -> n是沿着从n开始的最短路树走，因为这样足以绕开修改的这条不优的边了

- 总时间复杂度O(nlogn+mlogn+q)

- 我感觉最短路树都是奇奇怪怪的东西，不会严谨说明正确性

# CF436F Banners

你要开发一个APP，有付费和免费两个版本。付费版价格为 $p$，免费版则必须看 $c$个广告。你有 $N$个用户，每个用户有两个属性 $a$和 $b$。他们会按照如下规则使用你的APP：

- 若 $b \geq c$，用户会使用免费版
- 否则若 $a \geq p$，用户会使用付费版
- 否则用户不会使用你的APP

对于每个使用了付费版APP的用户，你将获利 $p$。对于每个使用了免费版的用户，你将获利 $cw$， $w$是一个常数。
对于 $[0, \max b_i + 1]$中的每个 $c$的取值，你需要找到一个 $p$，使得获利最大。

# Solution

- 这个应该可以用我不太会的技术做到2log，因为不太会所以先不写题解了

# CF793F Julia the snail 3000

有一只蜗牛在树干上爬，有两种移动方式，沿着某根绳子向上爬，或者顺着树干往下溜。

树干高度为 $n$，有 $m$ 根绳子，第 $i$ 条连接了高度 $l_i$ 至 $r_i$，保证 $r_i$ 互不相同。

有 $q$ 次询问，每次给出两个数 $L, R$，问蜗牛从高度 $L$ 开始爬，只考虑被包含在 $[L, R]$ 间的绳子（即若一条绳子的区间超出询问范围即不能使用），蜗牛能够爬到的最大高度。

$1 \leq n, m, q \leq 10^5$，输入均为正整数。

# Solution

- 这个题有点奇怪
- 因为无修改，考虑扫描线，查询是2-side形式，扫描线扫一个端点，数据结构维护另一个端点的答案
- 如果扫的是x，我感觉不太好维护
- 如果扫的是y，数据结构维护每个x的答案为f[x]
- 扫描线朝着y变大的方向扫，每次可能加入一个[l,r]的绳子
- 因为我们当前维护了每个x的最高答案f[x]，并且扫描线扫了上界
- 所以所有f[x]>=l的x，都可以走到更高的位置r

# Solution

- 由于向下滑是无代价的，所以在上界确定的情况下能贪心走得越高越好
- 操作相当于对全局的x，如果f[x]>=l，f[x]=r，且r一定比l和f[x]大
- 我们可以线段树每个节点维护最大值和严格次大值
- 将值相同的多个数缩起来处理
- 当节点x’被操作时，如果只修改其最大值，打个标记即可

# Solution

- 如果修改了严格次大值和最大值，则将这两个合并，然后递归两个儿子，看是否需要修改
- 1. 如果有一个儿子内含有最大值和严格次大值，则递归这个儿子进行合并
- 2. 否则我们修改两个儿子的最大值，这两个儿子的严格次大值不需要改动，并用来更新父亲的严格次大值

# Solution

- 可以发现这个在每次满足条件1时递归，但没递归一层就会合并掉两个值
- 线段树上节点大小和为O(nlogn)，所以1递归次数为O(nlogn)

- 总时间复杂度O((n+m)logn)

# CF1178G The Awesomest Vertex 3000

给定一棵根为 1 的有根树，每个节点有两个权值 $a[i]$ 和 $b[i]$ 。定义 $R(v)$ 为 $v$ 祖先的集合（包括自己），
则一个节点 $v$ 有多棒取决于其真棒程度，真棒程度是这样定义的：

$$\left|\sum_{w\in R(v)} a_w\right| \cdot \left|\sum_{w\in R(v)} b_w\right|$$

$|x|$ 表示 $x$ 的绝对值。

现在请你支持两种操作：

- $1\ \ v\ \ x$ — 将 $a_v$ 加上 $x$
- $2\ \ v$ — 输出以 $v$ 为根的子树中最大的真棒程度

数据范围：
$1 \le n \le 2 \cdot 10^5, 1 \le q \le 10^5\ -5000 \le a_i \le 5000\ , -5000 \le b_i \le 5000$
$1 \le x \le 5000$

# Solution

- 这个应该可以用我不太会的技术做到2log，因为不太会所以先不写题解了

# CF773E Blog Post Rating 3000

- 有一个博客，初始赞数为0，每个人会浏览这篇博客，如果这个人期望的赞数大于这边博客的赞数，他就会赞这篇博客，如果小于这篇博客，就踩一下（博客赞数-1），相同就不操作。询问对于每个i，1~i的人按照一个顺序浏览，求一种顺序使得博客最后的赞数最大，输出这个最大的赞数。

# Solution

- 有一个贪心的性质，就是我们从小往大枚举每个人是最优的
- 这样博客的赞数先递减，然后到一个值之后开始不降
- 因为如果出现了先增后降，开始增的时候的是a>赞数，开始降的时候是赞数>b，而这段时间赞数在递增，所以a>b，然而我们一定先访问小的a，然后访问大的b

# Solution

- 证明这个方法是优的话，考虑在这个基础上swap两个元素a，b，则会构成一个逆序对先b后a
- 结果本来是在博客rating高的时候访问了a，低的时候访问了b，现在是高的时候访问了b，低的时候访问了a，a<b
- 讨论一下大小关系可以发现这样不会变优

# Solution

- 于是我们要维护一个有序的数据结构，支持插入一个元素，查询最小的x使得这个数据结构中第x小的元素比-x大，这个就是分界点
- 直接维护f[x]表示第x小的元素-x，然后找第一个0，这个维护一个rmq，在平衡树上二分即可
- 大概操作就是插入元素，区间减1，rmq

- 总时间复杂度O(nlogn)

# CF331D3 Escaping on Beaveractor 3000

- 平面上有有些有向线段，平行于轴，当走到有向线段的时候自己的方向就要变成其方向，自己速度1。
- 给定这些有向线段，多组数据，每组给出初始位置和初始方向和时间，问是否会走出边界，如果不，输出最后在哪里，否则输出离开边界时候的坐标。数据都是1e5，时间1e15

# Solution

# CF185E Soap Time! – 2 3000

# Solution

# CF1218B Guarding warehouses 3000

# Solution

# CF765F Souvenirs 3100

- 区间查两个数的差的最小绝对值

# Solution

- 考虑i位置和哪些位置j能形成有意义的二元组
- 有意义的二元组即对答案有影响的(i,j)
- 如果是ai<aj<ak，则(i,k)的意义被(i,j)掩盖

# Solution

- 考虑ai<aj,aj>ak
- 1.ai>ak
- 则|aj-ak|>|ai-ak|，这里有|ai-ak|<1/2|ai-aj|，出现了值域减半
- 2.ai<ak
- 则|ai-aj|>|ai-ak|，这里限制了一个下界，之后再出现ai<ak’的情况可以类比1了
- 总的有贡献的二元组为O(nlogv)个

# Solution

- 将询问看做二维平面上的点
- 每个(a[i],a[j])的pair即对所有l在[1,i]中，r在[j,n]中的询问，其答案对|a[i]-a[j]|取max
- 注意到这里的矩形是2-side的，我们选择合适的方向扫描线，就只会插入不会删除了
- 问题变为矩形max=x，单点值
- 扫描线+线段树，总时间复杂度O(nlognlogv+mlogn)

# CF176E Archaeology 3100

有一棵 $n$ 个点的带权树，每个点都是黑色或白色，最初所有点都是白色的。有 $q$ 个询问：

- 把点 $x$ 从白色变成黑色。
- 把点 $x$ 从黑色变成白色。
- 查询黑点的导出子树 (用最少的边把**所有的**黑点连通起来的树) 的总边权和。

保证 $1 \leq n, q \leq 10^5, 1 \leq x \leq n$。

# Solution

- 这种直接用Top tree重拳出击就行
- 讲个阳间做法
- 一个点集的两两连通最小边集，就是将这个点集所有点按照DFS序排序后，相邻两两点距离和（注意第一个点与最后一个点相邻）
- 这个对应了树的欧拉回路，直接记这个性质就行

# Solution

- 然后我们实际上可以用一个set来维护
- 插入的时候按DFS序插入，set维护前驱后继
- 每次只需要查O(1)次两点间距离和
- O(n+mlogn)

- 如果使用vEB树来维护前驱后继，因为DFS序值域是n
- 所以总时间复杂度为O(n+mloglogn)

# CF896E Welcome home, Chtholly 3100

- 给定一个长为n的序列
- 1 l r x : 把区间[l,r]所有大于x的数减去x
- 2 l r x : 查询区间[l,r]内的x的出现次数
- 值域为n，n<=1e5

# Solution

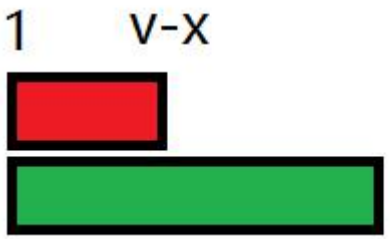
- 这个值域n很明显复杂度和值域有关
- 考虑分块，可以发现每块的最大值总是不增的
- 总的所有块的最大值和当块大小为O( sqrt(n) )时为O( nsqrt(n) )
- 考虑怎么利用这个性质

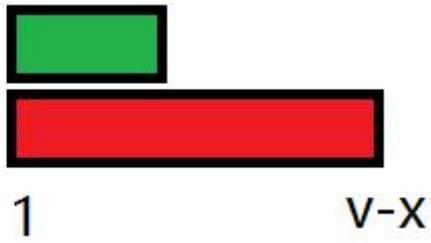
# Solution

- 可以发现：假设区间所有大于x的减x，最大值为v
- 这个操作等价于把区间中所有数减x，之后小于0的再加x
- 当v >= x * 2时，可以把所有[1,x]内的数合并到[x+1,x*2]上
- 当v < x * 2时，可以把所有[x+1,v]内的数合并到[1,v-x]上

# Complexity






- 如果v >= x * 2
- 我们用O( x ) * O( 数据结构复杂度 ) 的代价使得块内的最大值减少了 O( x )
- 如果v < x * 2
- 我们用O( v - x ) * O( 数据结构复杂度 )的代价使得块内的最大值减少了 O( v - x )

# Solution

- 所以要用一个数据结构支持：
- O( 1 )把块内值为x的全部变成x+v或者x-v
- O( sqrt(n) )求出块内经过大量操作后，每个数的值

- 第一种是整块操作的时候要用
- 第二种是重构零散块的时候要用

# Solution

- 维护这个有很多种方法
- 可以每块维护一个值域上的链表
- 定义f[i][j]为第i块里面所有值为j的数的下标的链表
- 区间所有x变成x+v即link( x , x + v )
- 然后维护一下块内可能出现的所有数
- 每次重构的时候即遍历所有可能出现的数，然后遍历其中真正出现的数的链表即可

# Solution

- 也可以用一个并查集维护
- 这个并查集由于只支持：
- 1. merge( x , y )
- 2. for( i = 1 ; i <= sqrtn ; i++ ) find( i )
- 所以复杂度是O( 1 )的并查集
- 本质上和链表的做法是差不多的，不过常数好很多

- 时间复杂度O( (n+m)sqrtn )，空间复杂度O( nsqrtn )

# Solution

- 关于空间比较大的问题
- 由于题目可以离线，所以可以使用离线逐块处理的技巧优化到线性空间
- 大概就是我们对每个块算出其对每次查询的贡献，因为贡献相对独立，注意一个块可能发生零散块重构
- 这样可以重复利用每个块开的O(n)大小数组，做到线性空间

- 总时间复杂度O( (n+m)sqrtn )

# CF679E Bear and Bad Powers of 42 3100

- 定义一个正整数是坏的，当且仅当它是 $42$ 的次幂，否则它是好的。
- 给定一个长度为 $n$ 的序列 $a_i$，保证初始时所有数都是好的。
- 有 $q$ 次操作，每次操作有三种可能：
  - `1 i` 查询 $a_i$。
  - `2 l r x` 将 $a_{l..r}$ 赋值为一个好的数 $x$。
  - `3 l r x` 将 $a_{l..r}$ 都加上 $x$，重复这一过程直到所有数都变好。
- $n, q \leq 10^5$，$a_i, x \leq 10^9$。

- 保证加的数非负

# Solution

- 考虑使用颜色段均摊的trick
- 因为加的数比较小，我们记下每个位置的数，与离其最近的，比其大的42的次幂的差
- 区间染色用缩点平衡树维护
- 区间加相当于这个差进行了一次区间减，递归下去找出所有被减为负数或者0的数，每个数只会被递归O(logv)次

# Solution

- 负数重新计算新的差，0重新进行一次区间加
- 因为颜色段均摊的性质，我们最多有O(n+m)个颜色段，每个颜色段最多导致O(logv)次递归或者区间加，一次递归或者区间加最多O(logn)

- 总时间复杂度O((n+m)lognlogv)

# CF571D Campus 3100

- 有一个长度为 $n$ 的序列，初始全为 $0$。
- 有两类对下标的集合，初始时每一类各有 $n$ 个集合，编号为 $i$ 的集合里有下标 $i$。
- 一共有 $m$ 个操作，操作有五种：
  1. `U x y` 将第一类编号为 $y$ 的集合合并到编号为 $x$ 的集合里。
  2. `M x y` 将第二类编号为 $y$ 的集合合并到编号为 $x$ 的集合里。
  3. `A x` 将第一类编号为 $x$ 的集合中的所有下标在序列中对应的数加上 $x$ 的集合大小。
  4. `Z x` 将第二类编号为 $x$ 的集合中的所有下标在序列中对应的数设为 $0$。
  5. `Q x` 询问序列中下标为 $x$ 的位置上的数。
- $n, m \leq 5 \times 10^5$。

# Solution

- 发现两类集合的修改相对独立
- 我们对每次查询x，如果能离线找出其最后一次x所在的集合，被4操作的时间，则从这个时间开始，累加所有3操作对其的贡献
- 就可以找出这次询问的答案了
- 我们对每个点的第二类编号开一个并查集，每次4操作时更新这个集合最近一次被操作的时间，每次查询x最近被4操作的时间直接用x所属集合最近一次被操作的时间即可

# Solution

- 将询问离线，由上述转换将问题变为只有一类编号的情况
- 1.合并
- 2.编号为x的集合内所有下标y加上一个数z
- 3.下标x在一段时间[l,r]内受到的总修改量

# Solution

- 这个可以使用一个类似可并堆的数据结构维护
- 合并即合并两个数据结构
- 修改即打一个全局加的标记
- 在l时刻时，在x所在的集合中插入下标x，在r时刻时，将x与其上方标记一起合并，如果同时有多个x，加个标号区分一下，即算出x在一段时间中总的修改量

- 总时间复杂度O(n+mlogn)

# CF407E k-d-sequence 3100

- 给一个长为 n 的序列，以及常数 k , d
- 找一个最长的子区间使得该子区间加入至多 k 个数以后，排序后是一个公差为 d 的等差数列。
- N<=2e5,|ai|,|d|<=1e9

# Solution

- 先特判d=0的情况
- 一个区间加入k个数后，排序后是一个公差为d的等差子序列，等价于三条条件
- 1.不能有重复的数字
- 2.区间中所有数模d后都是相同的数
- 3.区间max-min<=r-l+k

# Solution

- 考虑扫描线扫右端点，数据结构维护左端点的答案
- 将序列中的元素按照模d分类，为了满足第一条条件，可以将序列分为一段段的极长的模d相同的连续段，分别处理
- 这样不用考虑第二个条件了
- 问题变为不能有重复数字以及max-min<=r-l+k

# Solution

- 用一个数据结构维护每个左端点到右端点的max-min-r-l-k
- max和-min用单调栈转换为区间加，-l，-k直接当插入时的初始值，差分后-r即区间加
- 问题即维护序列a[l]=max-min-r-l-k，支持区间加，末尾插入，以及求最小的l满足a[l]<=0，以及首端删除（重复数字）
- 可以平凡地维护

- 总时间复杂度O(nlogn)

# CF700D Huffman Coding on Segment 3100

- 给一个长为n的串，有q次询问，每次询问一个区间的最小二进制编码长度，即在可以唯一还原的前提下，将这一段子串转化为长度最小的二进制编码。

# Solution

- 哈夫曼编码就是把每个出现过的元素提出来，按照出现次数加权，做合并果子
- 哈夫曼树在排序后可以做到线性
- 维护两个队列，第一个是初始的所有排好序的元素，第二个保存合成的所有元素，初始为空
- 每次从每个队列中选前2大的元素，找最小的两个合成，合成后放入第二个队列末端

# Solution

- 有一个性质是不同的出现次数是O(sqrtn)的，考虑1+2+...+n=n(n+1)/2
- 考虑使用莫队维护出区间每个数的出现次数，从小到大排序
- 这个有多种维护方法，我之前写过的方法是记录下莫队转移时所有变化，然后到一个查询区间时一起处理掉，用莫队上一个处理的询问维护的区间不同的出现次数，以及记录的转移，得到这个区间的所有出现次数
- 为了得到有序的出现次数，可以使用基数排序，也可以莫队转移时直接维护
- 时间复杂度O(nsqrtm+msqrtn)

# Solution

- 这里考虑可以批处理同一个出现次数出现很多次的情况
- 如果当前最小是相同的x个a，则可以合并出x/2个2a
- 我们边维护两个队列边缩点
- 复杂度分析：
- 每O(1)次合并，下一次可能合并出的元素大小至少变大1
- 经过O(sqrtn)次合并，只可能合并出>sqrtn的元素了
- 而这两个队列如果均>sqrtn，则有O(sqrtn)个元素
- 每次合并减少一个元素，此时合并O(sqrtn)次
- 总时间复杂度O(nsqrtm+msqrtn)

# CF633H Fibonacci-ish II 3100

# Solution

# CF453E Little Pony and Lord Tirek 3100

- 给一个序列
- 每个位置有初值ai，最大值mi，这个值每秒会增大ri，直到mi
- 有m个发生时间依此增大的询问，每次询问区间和并且将区间的所有ai变成0

# Solution

- 这个问题每次查询是区间查询并赋值，可以考虑颜色段均摊的方法
- 假设一段上次修改时间是x，这次修改时间是y，这段的贡献怎么求？
- 将贡献分为这段时间充能满了和没满的分别讨论
- 计算出一个数组a[i]表示i位置从0开始充能充a[i]秒，使得充a[i]+1秒后满

# Solution

- 区间中没充满能的位置即所有i，满足a[i]<y-x，我们要求这些位置的r[i]的和
- 区间中充满能的位置即上述的补，我们要求这些位置m[i]的和
- 即变成区间a[i]<x’的b[i]的和，二维数点即可
- 有初值，颜色段均摊导致查询次数O(n+m)次
- 总时间复杂度O((n+m)logn)

# CF536E Tavas on the Path 3100

- 给定一棵n个节点的树，每条边有边权。
- 有m个询问，形式为(u,v,l)，求u到v的路径，假设长度为p，第i条边权值为xi，构造一个长度为p的01串s，如果xi>=l，那么si=1，否则si=0。
- 对于得到的串s，假设它有k段连续的1，第i段长度为pi，那么要你输出所有f[pi]的和，其中f数组一开始就给出。

# Solution

- 将询问的l和树上权值一起离线
- 问题变为单点0变1，查询链上每个极长1段的f和
- 考虑使用静态LCT，这样只用合并logn段，每段维护出内部的答案以及两端的极长1个数，合并的时候算一下父亲节点除两端以外内部的f和

- O((n+m)logn)

# CF855F Nagini 3100

有$10^5$个集合，编号从1到$10^5$。每一个集合内初始没有元素。你需要支持两个操作：

$1\ l\ r\ k(1 \leq l < r \leq 10^5, k \in [-10^9, 0) \cap (0, 10^9])$：在编号在$[l, r)$的集合中加入元素$k$；

$2\ l\ r(1 \leq l < r \leq 10^5)$：询问编号在$[l, r)$内的集合的权值和。

一个集合的权值定义为：如果集合中同时存在$> 0$的元素和$< 0$的元素，则其权值为最小的$> 0$的元素和最大的$< 0$的元素的绝对值的和，否则为$0$。

# Solution

- 我们存一下每个位置是否有负数是否有正数，分开处理
- 经过平凡转换后问题变为
- 1.区间对k取max
- 2.区间和
- 我们可以线段树每个节点维护最大值和严格次大值
- 将值相同的多个数缩起来处理
- 当节点x’被操作时，如果只修改其最大值，打个标记即可

# Solution

- 如果修改了严格次大值和最大值，则将这两个合并，然后递归两个儿子，看是否需要修改
- 1. 如果有一个儿子内含有最大值和严格次大值，则递归这个儿子进行合并
- 2. 否则我们修改两个儿子的最大值，这两个儿子的严格次大值不需要改动，并用来更新父亲的严格次大值

# Solution

- 可以发现这个在每次满足条件1时递归，但没递归一层就会合并掉两个值
- 发现q比n小，可以离线合并操作的等价类，即序列长度变为q
- 线段树上节点大小和为O(qlogq)，所以1递归次数为O(qlogq)

- 总时间复杂度O(qlogq)

# CF1332G No Monotone Triples 3100

定义一个三元组 $(i,j,k)$ 为单调三元组，当且仅当满足

- $i<j<k$
- $a_i \le a_j \le a_k$ 或 $a_i \ge a_j \ge a_k$

现有一个长度为 $n$ 的序列 $a$，和 $q$ 次询问，每次询问给定 $L,R$，找到一个最长的**子序列** $b$，满足 $|b| \ge 3$，且 $b$ 中不含有单调三元组。
对于每个询问，输出 $b$ 中的每个元素在 $a$ 中的**位置**。

$3 \le n \le 2 \cdot 10^5, 1 \le q \le 2 \cdot 10^5, R-L \ge 2$

- 多解的话输出任意一个

# Solution

- 可以枚举一下大小的排列，发现|b|>4时一定不可行，所以只用考虑|b|<=4的情况
- 先维护|b|=3
- 对每个位置x找出最大的y满足a[y]<a[x]，最小的z满足a[z]>a[x]
- 对每个位置x找出最大的y满足a[y]>a[x]，最小的z满足a[z]<a[x]
- 区间中若存在这样的二元组(y,z)，则可以找到一组|b|=3的解
- 问题即带插入前驱，离线后变带删除前驱可以使用线性并查集O(n)维护
- 直接用set维护是O(nlogn)

# Solution

- 再维护|b|=4
- 考虑枚举最左端的元素a[x]
- 在右端我们需要选出一个a[y]<a[x]，一个a[z]>a[x]，然后一个w，满足a[y]<w<a[z]
- 考虑维护两个单调栈，一个单调不增，一个单调不降
- w一定不在这两个单调栈中，不然要么a[y]>a[w]要么a[z]<a[w]

# Solution

- 于是只需要找到x右边最近的两个不同单调栈中的元素a[y],a[z]，然后找到最近的一个在max(y,z)右边且不在单调栈中的元素a[w]
- 因为a[w]不在这两个单调栈中，意味着我们一定能在(x,w)中找出a[y’]<a[w]，a[z’]>a[w]（这两个不一定是上述的y,z），除非a[w]是单调栈中除了a[x]以外第一个元素
- 而因为已经找出了y,z，所以这个(x,y,z,w)的构造是存在的，w不可能是第一个元素
- 然后也变成区间是否存在二元组问题
- 预处理使用set的时间复杂度为O(nlogn)，vEB树为O(nloglogn)

# Solution

- 预处理后，对每个位置维护其左边最近的二元组另一个端点
- 问题即变为rmq

- 总时间复杂度O(nloglogn+q)

# CF1476G Minimum Difference 3100

有一个长为 $n$ 的序列 $a$。有 $m$ 个查询，每个查询如下：

- `1 l r k`，从 $[l,r]$ 区间取 $k$ 个不同的数 $x_1,\dots x_k$，使得它们之间的数量 $cnt$ 最大差值最小，（也就说，对于任何一种取法，取 $k$ 个数，对每个数出现的次数排序，使得最多出现次数减去最小出现次数最少）
- `2 p x`，修改 $a_p$ 为 $x$。

$1 \leq n,m \leq 10^5$

# Solution

- 有一个性质是不同的出现次数是O(sqrtn)的，考虑1+2+...+n=n(n+1)/2
- 考虑使用莫队维护出区间每个数的出现次数，从小到大排序
- 这个有多种维护方法，我之前写过的方法是记录下莫队转移时所有变化，然后到一个查询区间时一起处理掉，用莫队上一个处理的询问维护的区间不同的出现次数，以及记录的转移，得到这个区间的所有出现次数
- 为了得到有序的出现次数，可以使用基数排序，也可以莫队转移时直接维护
- 时间复杂度O(nsqrtm+msqrtn)

# Solution

- 这道题中直接使用带修改莫队维护区间本质不同出现次数
- 查询时用双指针在不同出现次数的结构上扫即可维护出答案

- 总时间复杂度O(nm^2/3+msqrtn)

# CF418E Tricky Password 3100

# Solution

# CF960H Santa's Gift 3100

给一棵$n$个点的有根树，每个点有一个颜色$a_i$，对于每种颜色有一个权值$b_i$。再给你一个常数$C$。

共进行$Q$次操作，操作分为两种：

- `1 x y` 把节点x的颜色修改为y
- `2 x` 对颜色x进行询问。求$\frac{\sum_{i=1}^{n}(S_i \times b_x - C)^2}{n}$，其中$S_i$是$i$的子树中，颜色=x的点的个数。

$2 \leq n \leq 50000, 1 \leq m, Q \leq 50000$

$1 \leq a_1, a_2, ..., a_n \leq m$

$1 \leq b_1, b_2, ..., b_m \leq 100$

# Solution

- 对每个颜色开一棵Top tree（其实静态LCT维护子树就行）
- 每个颜色x，每个节点i维护子树中x出现次数，每次修改对应于一次链加
- 每次修改颜色的时候对应于O(1)个颜色的Top tree上的O(1)次修改
- 将式子展开后发现需要维护全局平方和，和

- 总时间复杂度O(n+qlogn)

# CF792F Mages and Monsters 3100

- 有初始魔法值m，有两个操作：
- 1.学习一个魔法，该魔法施放时每秒消耗法力值y，造成伤害x。
- 2.出现一个怪物，其生命值为h，有t的时间限制将其击败。在攻击过程中，在同一时刻只能释放一个法术，一个法术可以释放任意时间。可以从已学习的任意法术中选择。询问是否可以击败。每次刚遇见怪物时法力值为满。

# Solution

# CF720D Slalom 3100

# Solution

# CF1344E Train Tracks 3100

给定 $n$ 个点的一棵树，以 $1$ 为根，边有边权。

有 $m$ 辆从 $1$ 开始到 $s_i$ 的火车，每个火车有一个初始时刻 $t_i$，在初始时刻时从根出发，向着目标 $s_i$ 前进，当火车在 $x$ 时刻到达一个点 $u$ 时，假设下一个路径上的点是 $v$，两点间边权是 $d$，则在 $x+d$ 时刻到达 $v$，火车在到达目标点后停止。

由于每个点可能可以到达多个点，每个点有且仅有一个当前时刻可以到的儿子，每秒钟只能切换某一个点可以到的儿子，切换比火车开动先进行，若一辆火车走向了非目标方向的点，则立刻自爆。每个点初始能到的儿子是确定的。

求出能否不发生自爆，如果不能，第一次自爆最晚什么时候发生？在此基础上，至少切换多少次一个点可以到的儿子。

# Solution

- 考虑对每个点，维护出到其的所有火车的时刻
- 对原树进行启发式合并的过程
- 对重儿子维护一个全局标记，将每个轻儿子的数据结构加上一个边权，合并进来
- 如果在最终合并出的数据结构中，有x为y的前驱，且x和y来自于不同的儿子，则在[x+1,y]时刻中必须进行一次切换，不然就会爆炸
- 启发式合并过程的复杂度：
- 使用平衡树是O(n+mlognlogm)，vEB树是O(n+mlognloglogm)
- （实际上我觉得是O(n+mlogmin(n,m)loglogm)）

# Solution

- 可以发现上述的约束条件是充分必要条件
- 启发式合并导致总共有O(mlogmin(n,m))个时间段区间
- 问题转换为给定一个序列，有一些区间，将序列的每个位置分配给包含其的某个区间，使得每个区间均被分配一个位置
- 扫描线从左到右扫序列，使用数据结构维护当前没有处理的区间里最靠左的一个右端点，问题即转换为插入删除查询前驱
- 使用平衡树可以做到1log，有O(mlogmin(n,m))次修改
- 使用vEB树可以做到：
- 总时间复杂度O((n+mlogmin(n,m))loglogm)

# Solution

- 如果这里的时间t=n的话有个更好复杂度的做法
- 可以发现，若最后的时间段数>2n，则一定出现爆炸
- 因为每秒只能切换一次，故总共最多切换n次
- 如果只是验证可行性的话，则时间段数>2n时直接输出NO
- 这样可以限制时间段数O(n)
- 然后我觉得可以用splay启发式合并证O(n+mlogm)的启发式合并的复杂度
- 这样时间复杂度可以优化为O(nloglogm+mlogm)

# CF477E Dreamoon and Notepad 3100

# Solution

# CF720F Array Covering 3100

# Solution

# CF487E Tourists 3200

Cyberland 有 n 座城市，编号从 1 到 n，有 m 条双向道路连接这些城市。第 j 条路连接城市 aj 和 bj。每天，都有成千上万的游客来到 Cyberland 游玩。

在每一个城市，都有纪念品售卖，第 i 个城市售价为 wi。这个售价有时会变动。

每一个游客的游览路径都有固定起始城市和终止城市，且不会经过重复的城市。

他们会在路径上的城市中，售价最低的那个城市购买纪念品。

你能求出每一个游客在所有合法的路径中能购买的最低售价是多少吗?

你要处理 q 个操作:

C a w: 表示 a 城市的纪念品售价变成 w。 A a b: 表示有一个游客要从 a 城市到 b 城市，你要回答在所有他的旅行路径中最低售价的最低可能值。

输入格式

第一行包含用一个空格隔开的三个数，n、m、q。

接下来 n 行，每行包含一个数 wi。

接下来 m 行，每行包含用一个空格隔开的两个数 aj, bj。（1≤aj,bj≤n,aj≠bj1≤aj,bj≤n,aj≠bj）

数据保证没有两条道路连接同样一对城市，也没有一条道路两端是相同的城市。并且任意两个城市都可以相互到达。

接下来 q 行，每行是C a w 或 A a b ，描述了一个操作。 输出格式

对于每一个A类操作，输出一行表示对应的答案。

# Solution

- 对这个图建立其点双连通分量的树形结构
- 当到达一个点双连通分量内部一个点，即可到达其他点
- 由于可能多个点双连通分量共用一个最高点，所以修改复杂度有问题
- 考虑对每个点双连通分量不维护其最高点的最小值

# Solution

- 查询到点双x时，若继续查询x的最高点（即普通的点），则正确性有保证
- 否则x一定是查询的树上路径的LCA，特判即可

- 总时间复杂度O(n+mlogn)

# CF643G Choosing Ads 3200

- 给定一个长度为 $n$ 的序列和一个整数 $p$。
- 有 $m$ 个操作，操作要么是区间赋值，要么是询问区间内出现次数至少占 $p\%$ 的数。
- 输出询问的答案时，可以包含错的数，也可以重复输出，但对的数一定要在答案中，且输出的数的个数不超过 $\lfloor \frac{100}{p} \rfloor$。
- $n, m \leq 1.5 \times 10^5$，$20 \leq p \leq 100$。

# Solution

- 考虑类比p%=50%+eps的情况，此时可以使用线段树维护区间内的答案，每次合并时，若两个相等则出现次数累加，否则互相抵消，信息变为抵消后非负的一个
- 对于这个p为更小的常数的问题，也可以用类似的方法，每个节点维护[100/p]个答案，合并时用类似的方法即可
- 区间修改对于区间值的影响是平凡的
- 查询出一个区间的可能的答案后，需要区间修改，维护区间中x出现次数，这个对每个颜色开一棵平衡树然后颜色段均摊即可

- 总时间复杂度O(n+mlogn)

# CF1017G The Tree 3200

- 给定一棵树，维护以下3个操作：
- 1 x表示如果节点x为白色，则将其染黑。否则对这个节点的所有儿子递归进行相同操作
- 2 x表示将以节点x为root的子树染白。
- 3 x表示查询节点x的颜色

# Solution

- 这个3操作的答案实际上只受到其上方祖先的1,2操作影响
- 若没有2操作，考虑每次查询x时，我们实际上是在找x的祖先中，是否有一个祖先y，满足y到x的链上1操作次数比链长度大
- 记a[x]表示x被1操作了多少次
- 即是否存在x的祖先y，满足x~y的a的和>dep[y]-dep[x]
- 这个可以在静态LCT上二分，维护一下区间中每个后缀的(a[i]-1)的最大值与和即可O(logn)解决，树链剖分是O(log^2n)的

# Solution

- 考虑对子树x进行2操作
- 在子树x内的1操作可以全部进行暴力清空，均摊次数O(m)
- 在清空子树x内的1操作后，第一种可能是x的父亲此时为白色，这种情况则问题已解决
- 第二种可能是x的父亲此时为黑色

# Solution

- 这种情况中，有可能x的祖先中上述后缀最大值y>0，导致子树内深度<=y的点被认为是黑色的，实际上应当是白色的
- 此时子树内的节点查询其上方1操作最深延伸到多少时得到的都是同一个值dep[x]+y，而期望为dep[x]
- 于是求出这个y，进行一次子树减即可

- 总时间复杂度O(n+mlogn)

# CF1019E Raining season 3200

题意：有一棵$n$个点的树，每条边的边权是一个一次函数$a_i \times t + b_i$。求对于所有的$t \in [0, m-1]$的树的直径。

$n \leq 10^5, m \leq 10^6, a_i \leq 10^5, b_i \leq 10^9$。

# Solution

- 树分治，可以将每个条边的边权看做是一个半平面
- 分治后计算出到分治中心的前缀半平面交
- 考虑使用边分治便于分析，合并两个连通块的前后缀半平面交来得到跨过分治中线的半平面交时可以直接进行归并，复杂度线性于连通块总边数
- 然后将跨过分治中线的半平面交与两个连通块的半平面交取个max，因为是凸函数所以直接归并后还是凸函数
- 分治结束后得到O(n)个半平面，单调扫描即可
- 总时间复杂度O(nlogn+m)

# CF1214G Feeling Good 3200

有一个$n \times m$的矩阵$M$，其元素值为$0/1$.

初始所有元素都是$0$

每次操作可以翻转第$a$行的第$i \in [l, r]$的所有元素$M[a][i]$

每次翻转结束后，输出一个矩形左上角为$(x1, y1)$右上角为$(x2, y2)$使得满足：

- $1 \leq x1 < x2 \leq n$ , $1 \leq y1 < y2 \leq m$
- 矩形的四个角，同侧异色，对角同色。

如果没有任何矩形满足条件，则输出$-1$

- n,m<=2000,q<=500000

# Solution

# CF1109F Sasha and Algorithm of Silence's Sounds 3200

给一个$n \times m$的网格图（$n, m <= 2000, nm <= 2 * 10^5$），每个格子上有个权值$f_{ij}$，保证$f_{ij}$构成一个1~$nm$的排列。问有多少区间满足这个权值在这个区间内的格子构成的连通块是一棵树。

# Solution

- 扫描线从1 -> nm扫区间的右端点，数据结构维护最左可行的左端点
- 每次插入一个元素(i,j)，若(i-1,j),(i+1,j),(i,j-1),(i,j+1)在当前可行的区间中，则成环，一直移动左端点直到不成环（即删除环上最小的元素）
- 使用LCT维护，这样就维护出了每个右端点，区间不含有环的最小左端点
- 但是这样只是维护出了构成森林的左端点，还需要是一棵树

# Solution

- 一个无环子图是树的性质等价于点数-边数=1
- 对每个左端点维护其到右端点的点数-边数-1
- 每次右端点移动时，每个点到右端点的点数+1，考虑新增的点r相邻的点x，与其的边在所有左端点<=x的子图中，于是是一个区间-1，保证每个数都>=0，维护区间0的个数
- 每次左端点移动时不需要在线段树上的操作

- 总时间复杂度O(nmlog(nm))

# CF543E Listening to Music 3200

给你一个长度为$n$序列$a_i$，和一个常数$m$，定义一个函数$f(l,x)$为$[l,l+m-1]$中小于$x$的数的个数，有$q$个询问，每次给定$l,r,x$查询$min_{i=l}^{r} f(i,x)$。

# Solution

- 将x从大到小排序后处理，<x的变为1，>=x的变成1
- 每次修改一个位置即进行一个单点修改，影响一个区间内的f进行一次区间加
- 查询即查询对应的x的rmq
- 在线查询用可持久化线段树维护即可

- 总时间复杂度O((n+m)logn)

# CF482E ELCA 3200

有一棵以 $1$ 为根的有根树，第 $i$ 个节点的父亲为 $f_i$，每个节点上有一个数为 $a_i$。

共有 $m$ 个事件：

`P x y` ：若 $x$ 是 $y$ 的祖先，把 $f_y$ 改为 $x$，否则把 $f_x$ 改为 $y$。

`V x v` ：把 $a_x$ 改为 $v$。

求初始和每个事件发生后随机两个点（可以是同一个点）的 LCA 的 $a_i$ 的期望

# Solution

- 考虑i点对答案的贡献，即为(i子树内点数平方-每个儿子子树内点数平方)*a[i]
- 问题即换根，修改点权，维护每个点子树内点数平方，以及每个点每个儿子子树内点数平方
- 前者考虑a[i]x^2与a[i](x+y)^2的差分是2a[i]xy+a[i]y^2，每次修改即链上每个位置的答案加上2y*a[i]x和y^2a[i]，打个标记可以维护
- 如果使用LCT维护子树的方法后者和前者可以类似维护

- 总时间复杂度O(n+mlogn)

# CF1209G2 Into Blocks (hard version) 3200

给你 $n$ , $q$，$n$ 表示序列长度，$q$ 表示操作次数。

我们需要达成这么一个目标状态：如果存在 $x$ 这个元素，那么必须满足所有 $x$ 元素都必须在序列中连续。

然后你可以进行这么一种操作，将所有的 $x$ 元素的变为任意你指定的 $y$ 元素，并且花费 $cnt[x]$ 的花费，$cnt[x]$ 代表 $x$ 元素的个数。

现在有 $q$ 次询问，每次询问单点修改一个位置的值，求修改完之后最小花费使得序列满足目标状态。

注意：更新不是独立的，之前的更新会保留。

# Solution

# CF414E Mashmokh's Designed Problem 3200

给定一棵 $n$ 个节点的有根树，每个点连出的边都有序，共有 $m$ 个操作。 $(n \leq 10^5, m \leq 10^5)$

操作有：

- 1.查询两个点 $u, v$ 的距离
- 2.以 $v$ 为根的子树从树中分开，并添加一条与其第 $h$ 个祖先的连边作为该祖先的最后一个儿子。
- 3.查询从一个点出发，按边的顺序进行 dfs,深度为 $k$ 的最后遍历的点

# Solution

- ETT可以支持换根与维护点深度
- 直接使用ETT维护这棵树
- 3查询则在ETT上二分，每个结点维护内部深度最小和最大的点
- 若右儿子有深度>=k的点，且右儿子有深度<=k的点，则递归右儿子，否则递归左儿子，这个可以用深度最小最大判断出来
- 对于1操作，我们使用LCT直接维护就行了

- 总时间复杂度O(n+mlogn)

# CF1446F Line Distance 3200

## 题目描述

给定一个整数 $k$ 和 $n$ 个在欧几里得平面上的不同点，第 $i$ 个点的坐标是 $(x_i, y_i)$。

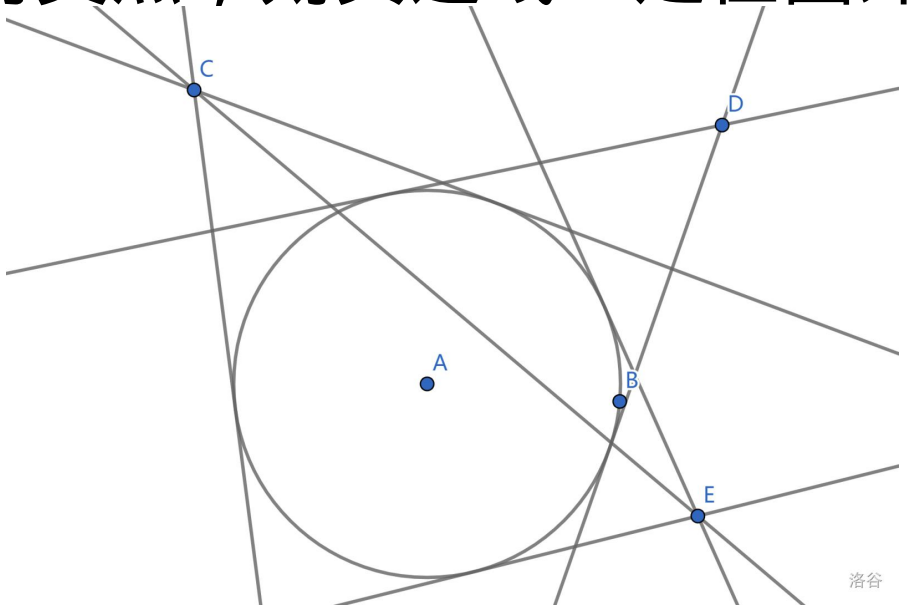
考虑 $\frac{n(n-1)}{2}$ 对坐标 $((x_i, y_i), (x_j, y_j))$ $(1 \leq i < j \leq n)$。对于每对坐标，计算从这两个点连成的直线到原点 $(0, 0)$ 的最近直线距离。输出第 $k$ 小的距离。

## 数据范围

$2 \leq n \leq 10^5, 1 \leq k \leq \frac{n(n-1)}{2}$

# Solution

- 考虑二分答案，即给定一个圆，求有多少点对之间的连线完全在圆外
- 可以发现，我们对每个点，算出以其为点光源，能照到圆上哪些位置，这个范围是一段圆弧
- 如果两段圆弧有交点，则其连线一定在圆外，否则一定在圆内



# Solution

- 对每个点算出这个圆弧的范围
- 问题即给定一个环，有n个区间，求有多少区间两两有交
- 将环变成序列，问题排序后使用一个数据结构即可维护

- 总时间复杂度O(nlognlog(1/eps))

# CF1056H Detect Robots 3200

有一张 $n$ 个点的图，给你 $m$ 条路径，每条路径中每个点最多出现一次，如果存在两个点 $A$、$B$，他们在两跳不同的路径中均出现，且在两条路经中 $A$ 到 $B$ 之间的点依次有一个点不同，则输出“Human”，否则输出"Robot"。

数据范围为 $3 \times 10^5$。

# Solution

- 题目中的两个A -> B的路径内部的点完全相同可以转化为两个A -> B的路径，这条路径中A的后继结点相同
- 考虑数学归纳，若相同，则递归到串长-1的子问题，若不同，则是Robot，串长为1的子问题平凡
- 考虑对串长度进行根号分治

# Solution

- 若一个串a长度>sqrtn，可以算出其与其他每个串是否是Human
- 记下这个串a中每个点的出现位置
- 对于串b，扫描线扫x从|b| -> 1，考虑b中每个点b[x]=A，找出a[y]=A，则找出是否存在z>x满足b[z]在a中出现过，且出现位置在y之后，若存在，则需要检查b[x+1]和a[y+1]是否相等，否则不检查，这样是完备的
- 维护目前后缀最大的z即可
- 这样对b计算的时间复杂度是|b|的，由于这样的串有O(sqrtn)个，总时间复杂度为O(nsqrtn)

# Solution

- 若一个串a长度<sqrtn，我们对其每两个点(x,y)，记下f(x,y)=a[x+1]，同时检查是否和之前计算过的f(x,y)不同，不同则是Human
- 这样时间复杂度是O(|a|^2)的，总时间复杂度O(nsqrtn)

- 总时间复杂度O(nsqrtn)

# CF576E Painting Edges 3300

- 给定一张 $n$ 个点 $m$ 条边的无向图。
- 一共有 $k$ 种颜色，一开始，每条边都没有颜色。
- 定义**合法状态**为仅保留**染成 $k$ 种颜色中的任何一种颜色**的边，图都是一张二分图。
- 有 $q$ 次操作，第 $i$ 次操作将第 $e_i$ 条边的颜色染成 $c_i$。
- 但并不是每次操作都会被**执行**，只有当执行后仍然合法，才会执行本次操作。
- 你需要判断每次操作是否会被执行。
- $n, m, q \leq 5 \times 10^5$，$k \leq 50$。

# Solution

- 是二分图等价于不存在奇环
- 维护一个点集，支持插入边与维护是否存在奇环，可以使用带权并查集，维护每个点到根的距离，加边(x,y)的时候检查x,y到根的距离奇偶性是否相等即可
- 带修改可以使用对时间的线段树分治解决
- 注意到每次修改作用范围时间最开始只能确定可能为[l,r]（r为下次修改的时间），也可能这次修改没有进行

# Solution

- 当递归到l的叶子时判断修改是否进行
- 若进行，则在线段树区间[l,r]插入这次修改中的插入
- 同时，这次修改中删除的边e，打一个[l,r’]删除e的标记，当一个节点同时有插入和删除某条边的操作时将其视为不存在
- 我认为时间复杂度和k无关

- 总时间复杂度O(qlogmin(n,m)logq)

# CF639F Bear and Chemistry 3300

- 给定一张 $n$ 个点 $m$ 条边的初始无向图。
- $q$ 次询问，每次询问给定一个点集 $V$ 和边集 $E$。
- 你需要判断，将 $E$ 中的边加入初始无向图之后，$V$ 中任意两个点 $x, y$ 是否都能**在每条边至多经过一次**的情况下从 $x$ 到 $y$ 再回到 $x$。
- $n, m, q, \sum |V|, \sum |E| \leq 3 \times 10^5$，强制在线。

# Solution

- 问题即每次给出多条边，问x和y是否在一个边双连通分量中
- 先把原图中的极大边双连通分量缩成一个点，原图即变为一个森林
- 每次查询可以在这个森林上建立类似虚树的结构，然后跑一遍tarjan
- 这里可以先随便规定一下森林中多棵树的顺序，然后就比较好做了

- 总时间复杂度O(n+m+q)

# CF1270H Number of Components 3300

给一个长度为 $n$ 的数组 $a$，$a$ 中的元素两两不同。

对于每个数对 $(i,j)(i<j)$，若 $a_i<a_j$，则让 $i$ 向 $j$ 连一条边。求图中连通块个数。

支持 $q$ 次修改数组某个位置的值，每次修改后输出图中连通块个数。

$n,q\leq 5\times 10^5, 1\leq a_i\leq 10^6$，保证任意时刻数组中元素两两不同。

- 连边是无向边

# Solution

- （这个我感觉挺难的不会做）
- 首先有一个性质，每个连通块在序列上都是一段连续的区间
- 序列上三个连续的位置i<j<k
- 若i与k连通，j不与i,k连通，则a[i]>a[j]>a[k]
- 一定可以找到一个依次连通的节点序列i1,i2,...ix,k1,k2,...ky，满足i1与i连通，ky与k连通

# Solution

- a[i1]<a[i2]<...<a[ix]<a[k1]<a[k2]<...<a[ky]
- i1<i2<...<ix<k1<k2<...<ky
- 因为a[i]>a[j]>a[k]，i+1=j，j+1=k，i与i’连通，k与k’连通
- 所以i1<i2<...<ix<i且k1<k2<...<ky<k且a[ix]<a[k1]<a[k]
- 而ix<j，a[j]>a[k]，故a[ix]<a[k]<a[j]，矛盾
- 故i,j,k一定连通
- 故所有极大连通块一定是序列上的一个连续区间

# Solution

- 考虑一个分界点(p,p+1)的情况，一定是[p+1,n]比左边的都小，[1,p]比右边的都大
- 问题变为给定一个排列，支持单点修改，维护有多少p满足：
- min[1...p]>max[p+1,n]
- 对每个值t，设序列中>t的位置为1，<t的位置为0
- 则若相邻的”10”子段个数为1，则a[p]=t的p是一个分界点
- 注意到特判全局min和max后这个相邻的”10”子段个数永远>=1

# Solution

- 对每个t预处理出这样的”10”子段个数，之后使用线段树维护每个t的答案（线段树维护的是值域）
- 每次单点修改a[p’]=t’，考虑a[p’-1]与a[p’+1]与a[p’]的关系，可以发现影响是线段树上O(1)段区间
- 问题即区间加，维护区间0个数，保证非负
- 预处理可以使用线性并查集做到O(n)

- 总时间复杂度O(n+mlogn)

# CF704E Iron Man 3300

“铁人”yyb在玩游戏。在一个$n$个点的树上，yyb放置了$m$个鸡贼。每个鸡贼有四个整数参数$t_i, c_i, v_i, u_i$，表示这个鸡贼会在$t_i$时刻**出现**在点$v_i$，并以每时刻$c_i$条边的速度向$u_i$点**匀速**移动，到达$u_i$点时**立刻消失**。

如果一个时刻有两个鸡贼在同一位置，它们会立刻爆炸。

（注意，如果一个鸡贼的$u_i = v_i$那么它会在$t_i$时刻出现，此时如果这个点有其它鸡贼一样会发生爆炸）

yyb想知道最早有鸡贼爆炸的时刻。如果自始至终都没发生爆炸输出 `-1` 。

如果你的答案和标准答案的绝对或相对误差不超过$10^{-6}$那么被视为正确。

- 同一位置不一定需要在树的点上，走到一半也算

# Solution

- 考虑对原树进行树链剖分，之后会将每个人放在DFS序上连续的O(logn)个区间上
- 这样一个人相当于O(logn)条线段
- 问题转换为给定O(nlogn)条线段，DFS序为x轴，时间为y轴，找出时间最小的交点
- 对每条链分别计算即可

- 总时间复杂度O(nlog^2n)

# CF983D Arkady and Rectangles 3300

- 按顺序在坐标轴上画n个颜色为1…n的矩形（数字大的颜色覆盖数字小的颜色），问最后能看到多少种颜色。

# Solution

- 扫描线扫一维，一个值为x的矩形被看见当且仅当其第二维的区间在扫描线的过程中有一个时刻min<=x
- 问题变为维护一个线段树，维护子树中值最大的未被看见的颜色max，子树中最小的颜色min，同时每个点维护一个永久化的标记tag表示这个节点上覆盖的颜色，col表示这个节点上未被看见的颜色，为了方便删除，这四者都使用数据结构维护

# Solution

- 更新时
- 若一个节点上的tag比儿子维护的max大，则子树内的颜色无法被看见，否则子树内的颜色在当前节点内是有可见的，继续上传max
- 若一个节点上的col比儿子维护的min大，则这个节点上的颜色可能可以被看到，将其上传为max
- 每次操作后，若根节点里的max是存在的，则其可看见，递归将其删除
- 若使用set，总时间复杂度O(nlog^2n)，使用vEB树的复杂度更优
- 总时间复杂度O(nlognloglogn)

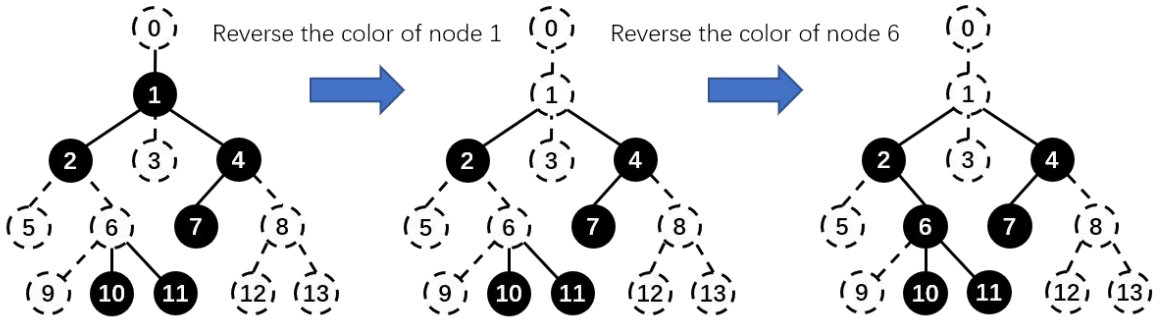
# CF1172E Nauuo and ODT 3300

- 树，点有颜色，带修改，每次修改后输出所有链颜色数的和

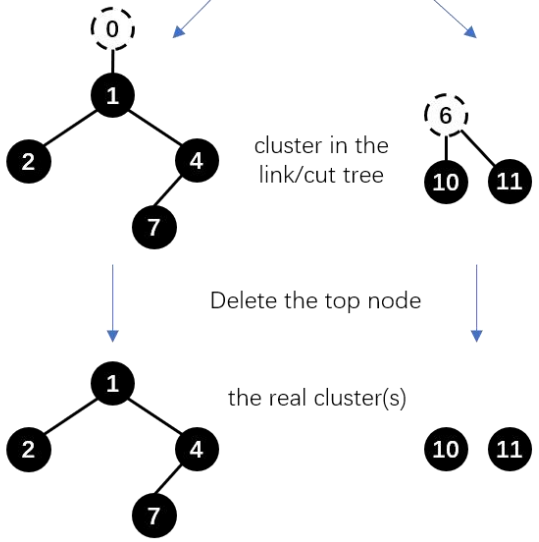
# Solution

- 对每种颜色分别考虑不含该颜色的简单路径条数
- 令当前处理的颜色为 c，把颜色为 c 的视为白色，不是 c 的视为黑色，那么不含 c 的路径条数就是每个黑联通块的大小的平方和，修改就是当颜色是 c 等价于颜色不是 c 时翻转一个点的颜色。所以，问题转化成了黑白两色的树，单点翻转颜色，维护黑联通块大小的平方和
- 为了不可持久化，在处理完一个颜色后需要回退这个颜色的所有修改
- 对每个点维护子树大小，儿子大小平方和，在 link/cut 的时候更新答案即可。可以每个黑点向父亲连边，这样真正的联通块就是 Link/cut Tree 里的联通块删掉根
- 总时间复杂度O((n+m)logn)

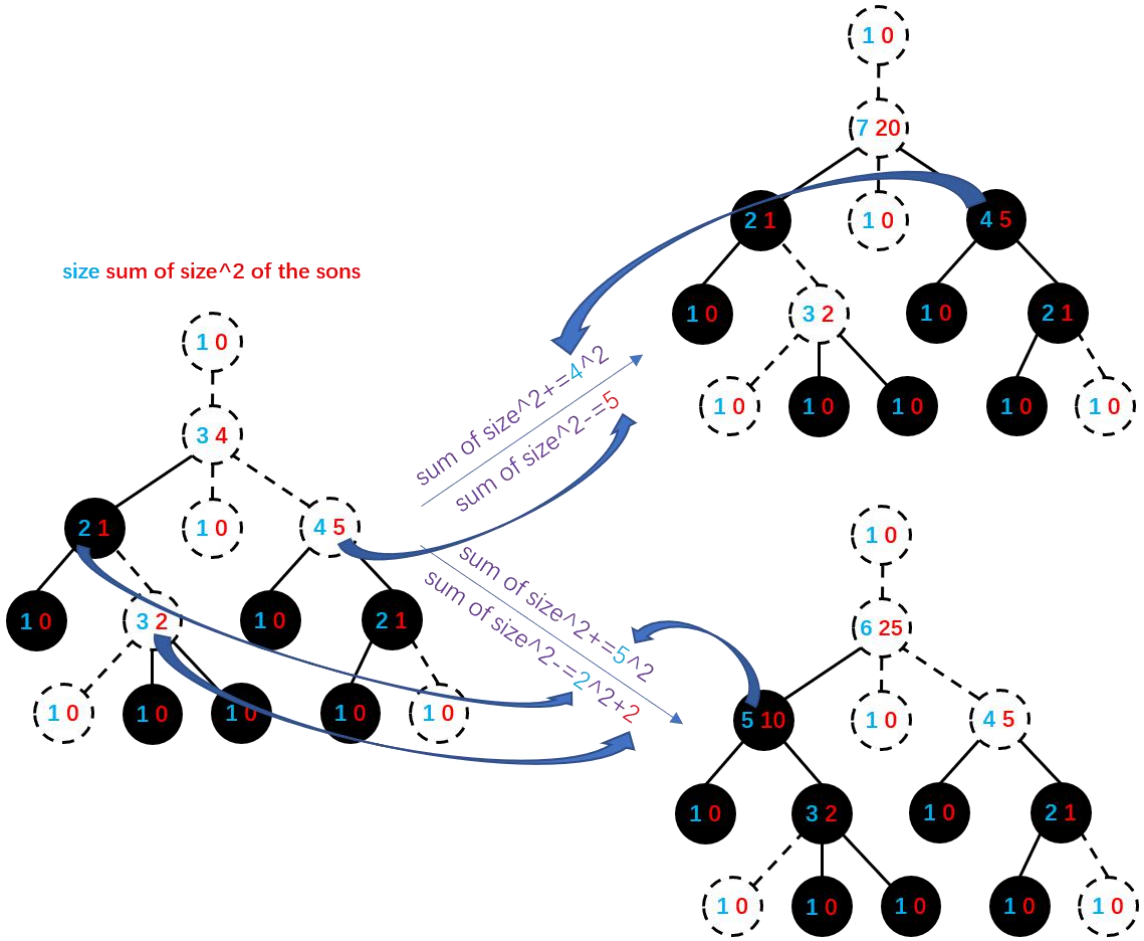
# Solution

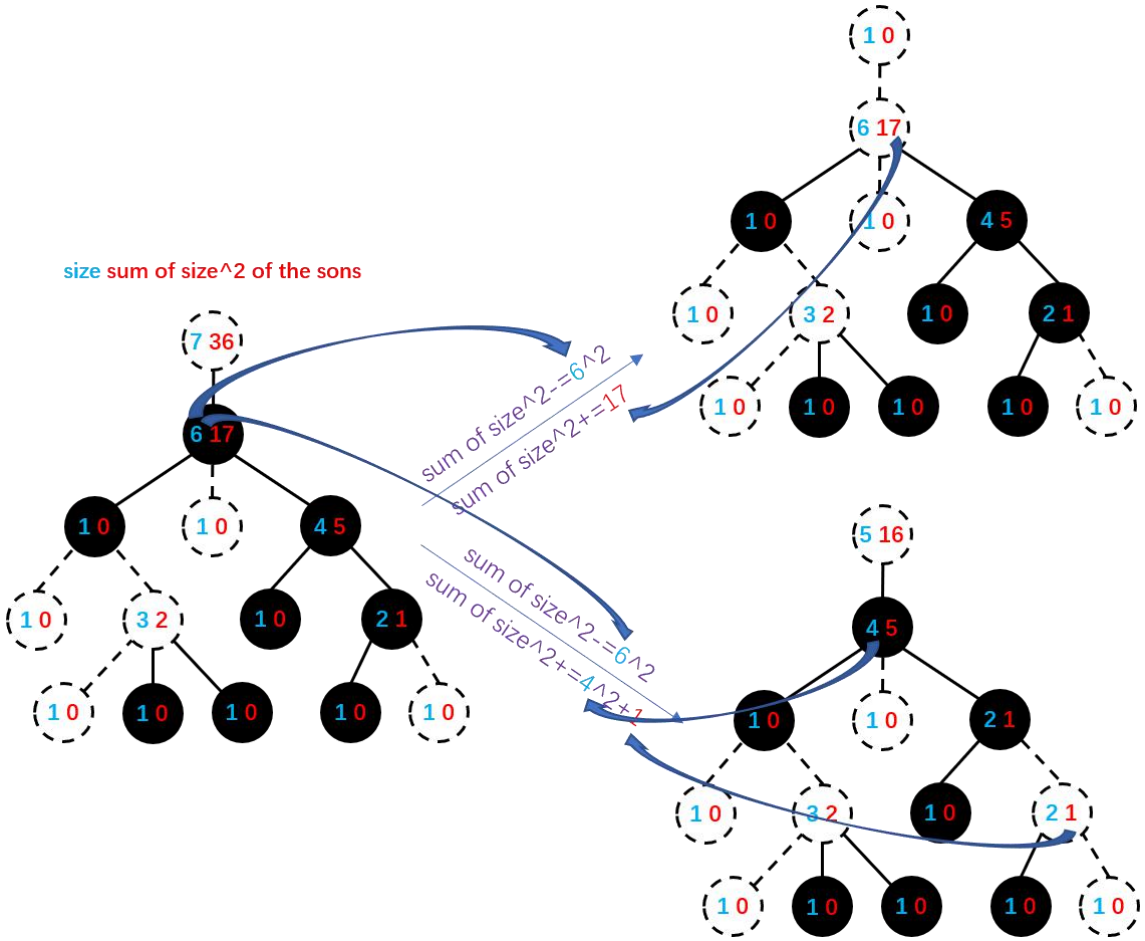



clusters: 1,2,4,7 ; 10 ; 11 2 ; 4,7 ; 10,11 2,6,10,11 ; 4,7



# Solution


size sum of size^2 of the sons
sum of size^2+=4^2
sum of size^2-=5
sum of size^2+=5^2
sum of size^2-=2^2+2


size sum of size^2 of the sons
sum of size^2-=6^2
sum of size^2+=17
sum of size^2-=6^2
sum of size^2+=4^2+1

# Solution

- 可以使用静态LCT维护子树
- 同时也可以用普通的2log的HLD结构
- 都是可以1log解决的

# CF1340F Nastya and CBS 3300

你需要动态维护一个多种括号组成的括号序列。需要支持两种操作：

- $1\ i\ t$：修改 $i$ 位置的括号为 $t$（绝对值表示种类，正数表示是左括号，负数表示是右括号）。
- $2\ l\ r$：查询 $[l,r]$ 区间是否是一个合法的括号序列。

序列长度为 $n$，不同的括号种类数为 $k$，操作次数为 $q$。

一个多种括号组成的括号序列 $S$ 是合法的当且仅当：

1. $S$ 为空。
2. $S$ 可以表示为 $A+B$ 的形式，其中 $A,B$ 都是合法的。
3. $S$ 可以表示为 $(^*+A+)^*$ 的形式，其中 $A$ 是合法的并且 $(^*$ 与 $)^*$ 是一对**相同种类的左右括号**。

例如 `()[{}()]` 和 `{([{}])}` 是合法的，而 `][[` 和 `([)]` 不是。

# Solution

- 考虑线段树维护序列，对一个节点维护其经过括号匹配后的结果，如果一个节点出现了两个不匹配的左右括号挨在一起的情况（如“(]”），则这个节点不可能被成功匹配，可以直接标记匹配失败
- 所以一个节点如果可能和别的节点合并后成功匹配，其不匹配括号一定是一些左括号然后一些右括号
- 为了方便判括号是否可以匹配，使用字符串哈希的方法，考虑我们对每个节点维护了其不匹配括号中左括号与右括号分别的哈希值，如何合并节点信息

# Solution

- 这里用小括号，只表示括号方向
- 如)))(((与))))(合并得到))) )(，可以发现合并时，一定会抵消掉左儿子的所有右括号，或者右儿子的所有左括号，否则匹配失败
- 如上述例子合并时，我们需要知道右儿子这x个左括号的哈希值，这个可以通过在右儿子的子树中递归实现：
- 若右儿子的左儿子的左括号>=x个，我们只需要递归右儿子的左儿子即可
- 否则右儿子的左儿子的左括号一定全部包含在这x个括号中，因为维护了所有不匹配左括号的哈希值，所以只需要递归右儿子的右儿子

# Solution

- 这样合并时构成了一个单侧递归结构，合并两个节点的时间复杂度为O(logn)，总时间复杂度为O(log^2n)
- 建树的时间复杂度为T(n)=2T(n/2)+O(logn)，解得T(n)=O(n)
- 总时间复杂度O(n+mlog^2n)

- P.S 四年前ccz想过区间不匹配括号的维护，他题解给的那个线段树套可持久化平衡树的方法是被完爆的。我们出过区间不匹配括号半群的题P6781，当时设计了一个类top tree结构，实际上也是可以用这个单侧递归方法做的，只是不可减了要麻烦一些

# CF1172F Nauuo and Bug 3300

Nauuo 是一个喜欢编程的女孩子。有一天她在做一道题，要求计算一些数的和对一个数 $p$ 取模的结果。

她写出了如下的代码，然后获得了 WA 的评测结果。

**Function** — a fake way to calculate sum modulo p

```
1:  function ModAdd(x, y, p)
2:      if x + y < p then
3:          return x + y
4:      else
5:          return x + y − p
6:
7:  function Sum(A, l, r, p)
8:      result ← 0
9:      for i ← l to r do
10:         result ← ModAdd(result, A[i], p)
11:     return result
```

她很快发现了 bug——`ModAdd` 函数只对 $[0, p)$ 中的数起作用，但是这个问题中的数有可能不在这个范围之内。她对这个错误的函数很好奇，所以她想知道它的运行结果。

然而，原来的代码运行得太慢了，所以她来找你求助。

给定数组 $a_1, a_2, \ldots a_n$ 和一个数 $p$，有 $m$ 个询问，每次给定两个数 $l$ 和 $r$，你需要计算函数 `Sum(a,l,r,p)` 的返回值。函数 `Sum` 的定义在上面的伪代码中已经给出。

注意，在上面的代码中，整数不会越界。

# Solution

- 可以转换为查询一个初值在区间[l,r]中被减去多少次p
- 记f(x,l,r)表示经过区间[l,r]后，减去了恰好x次p，最小的初始值
- 这个MODADD有单调性，可以发现f(x+1,l,r)>=f(x,l,r)，且[f(x,l,r),f(x+1,l,r))中的值经过区间[l,r]后都被减去了x次p
- 使用线段树维护这个序列
- 每次合并区间[l,mid]和[mid+1,r]时，考虑用f(x,l,mid)与f(y,mid+1,r)合并出f(x+y,l,r)

# Solution

- 单调性的证明：
- 考虑初值为x与x+1
- 找出x与x+1进行相同操作的LCP，下一个位置只可能是x’变成x’+y，x’+1变成x’+1+y-p，然后再从这个位置开始找出进行相同操作的LCP，之后下一个位置只可能是x’’变成x’’+y-p，x’’+1-p变成x’’+1+y-p，于是这一段实际上对x与x+1是完全相同的，并且x+1在任何时刻被减p的次数不少于x
- 对上述过程进行数学归纳，可以发现x+1被减p的次数不少于x

# Solution

- 由于f(x+1,l,mid)-1是恰好减去x次p的最小初始值，若f(x+1,l,mid)-1-xp>=f(y,mid+1,r)，则f(x,l,mid)与f(y,mid+1,r)可以合并出f(x+y,l,r)
- 需要满足
- 1. 这个值>=f(x,l,mid)，这样在左区间才会减去x次
- 2. 这个值-xp+sum(l,mid) >=f(y,mid+1,r)，这样才足够在右区间减去y次
- 故合并时，f(x+y,l,r)与max(f(x,l,mid),f(y,mid+1,r)+xp-sum(l,mid))取min

# Solution

- 可以发现，当f(x,l,mid)中x>mid-l+1时，f的值不会改变，所以对于长度为len的区间，f是一个O(len)段的分段函数，无意义的部分我们不考虑
- 之后枚举x，找出f(x,l,mid)=a与f(x+1,l,mid)=b，[a,b)经过区间操作后变为[a-xp,b-xp)，由f的单调性，使用双指针找出f(y,mid+1,r)，满足f(y-1,mid+1,r)<a，找出f(z,mid+1,r)，满足f(z+1,mid+1,r)>b
- 于是[a,b)与f(y,mid+1,r),f(y+1,mid,r)...f(z,mid,r)构成了O(z-y)段值域上的区间，每一段对应分段函数的一段
- 对于f(x+1,l,mid)=a’与f(x+2,l,mid)=b’，[a’-(x+1)p,b’-(x+1)p)只可能与f(y,mid,r),f(y+1,mid,r)...产生贡献，因为a’-(x+1)p>=a-xp

# Solution

- a’-(x+1)p>=a-xp的证明：
- 考虑初值a-1经过一个区间的操作后每个位置是否减少了p
- 考虑初值a+p-1，假设在位置y前，初值进行的操作与a-1完全相同，在位置y处，一定是a-1没有被减，a+p-1被减了，于是二者在位置y后进行的操作完全相同
- 而由定义，初值a-1在区间操作中总共被减去x-1次p，故初值a+p-1被减去x次p，由之前证过的单调性，[a,a+p-1]做为初值会得到同样的结果，故b>a+p-1，而a’=b，故a’>=a+p，a’-(x+1)p>=a-xp

# Solution

- 由于a’=b，故[a,b)与[a’,b’)在经过[l,mid]的操作后变换为的区间[a-xp,b-xp)与[a-(x+1)p,b-(x+1)p)相交的范围是p
- 而对任意x’,l’,r’，f(x’+1,l’,r’)-f(x’,l’,r’)>=p，代入x’=y,l’=mid+1,r’=r
- 故每个f(x,l,mid)与f(x+1,l,mid)的相交范围中只会有O(1)段f(y,mid+1,r)
- 于是对于长度为len的区间，这里归并的总复杂度可以优化为O(len)

# Solution

- 使用线段树归并维护出区间答案
- 每次查询时在线段树的O(logn)个节点上，从左往右，计算出初值经过节点内操作后得到的值，然后带入下一个节点，依次计算
- 使用二分查找复杂度为O(nlogn+mlog^2n)
- 使用vEB树复杂度为O((n+m)lognloglogn)
- 使用离线基数排序为O(nlogn+mwloglogm)

- 总时间复杂度O(nlogn+mwloglogm)

# CF1290E Cartesian Tree 3300

给定一个 $1 \sim n$ 的排列 $a$ .

对于一个整数 $k \in [1, n]$ , 将排列中 $\leqslant k$ 的项构成的子序列建**大根笛卡尔树**. 这棵笛卡尔树的所有节点的子树大小之和记为 $s_k$ .

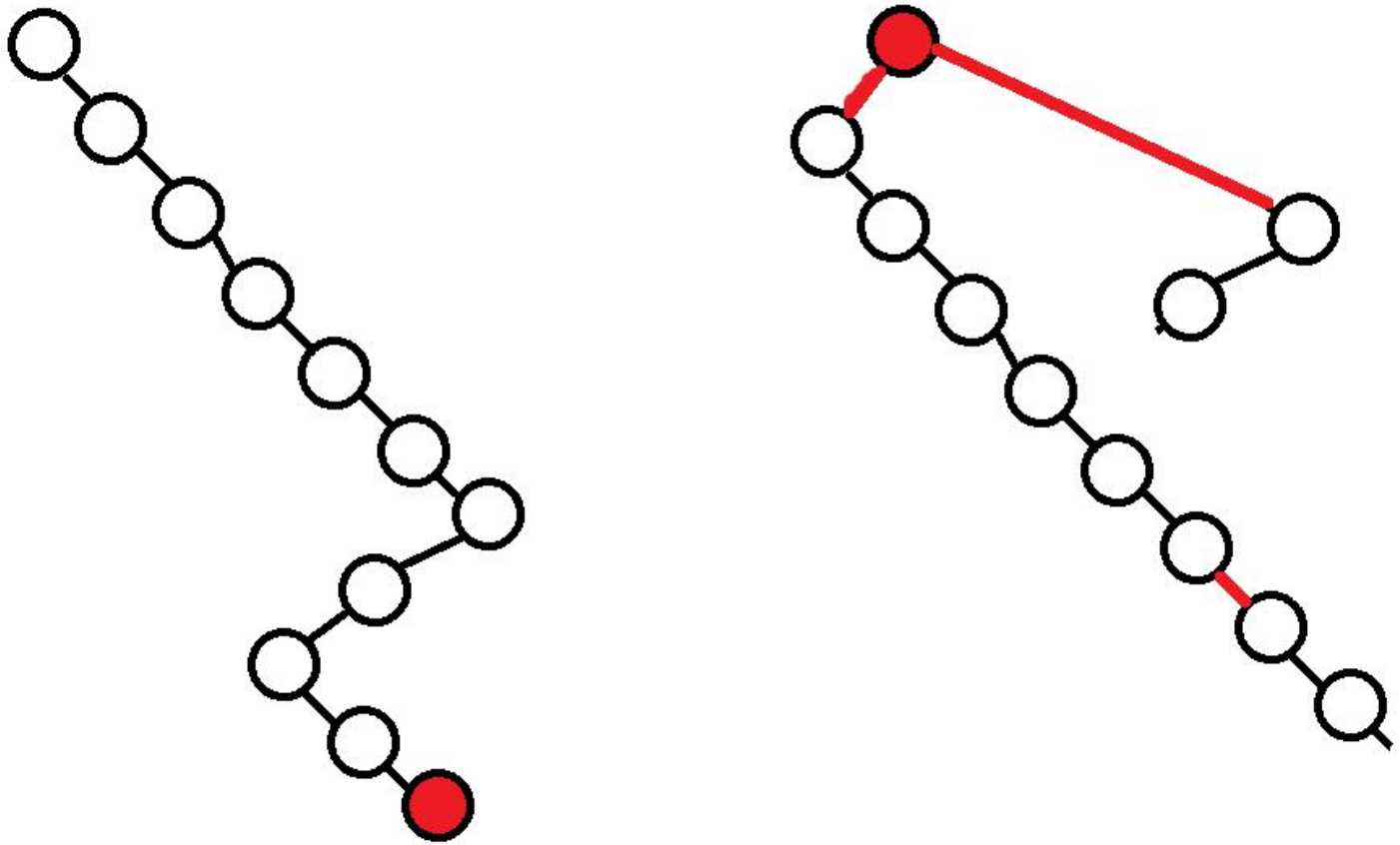
$\forall k \in [1, n]$ , 求 $s_k$ .

# Solution

- 这题直接动态笛卡尔树就行了，用科技碾了
- 因为每次加入的数都是最大的，等价于将一个点单旋到根
- 考虑所有点的子树size和，等价于所有点的深度和

# Solution

- 动态笛卡尔树把一个点单旋上去的过程：



# Solution

- 可以发现对这条路径，以每个拐点为划分，会把每条极长右链都拼到左边去，每条极长左链都拼到右边去
- 拼完之后拐点变成O(1)个
- 所以我们可以考虑O( 拐点个数 )地维护这样的结构，以拐点个数为均摊
- 这里HLD一下，轻边导致的拐点是平凡的，类似access的切换，重边导致的拐点可以均摊分析出
- 拐点个数变化量O( (n+m)logn )

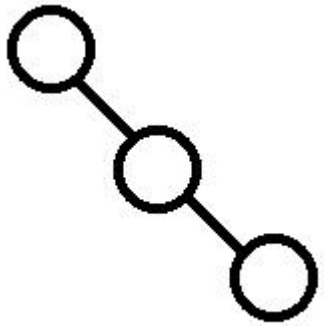
# Solution

- 于是我们均摊维护拐点，考虑直接用LCT维护整棵树，上旋x->y的时候，就提取出x到y的路径。
- 然后通过在splay上二分来将x到y的路径分割成若干个权值单调的连续段。
- 将这些连续段分别拆开来后，按照一定顺序合并起来即可。
- 这样时间复杂度可以证明O( (n+m)log^2n )

# Solution

- 使用“极长直链剖分”的结构，即把每个x，假设父亲是y，父亲的父亲是z，如果三者方向相同，则放到同一个极长直链中
- 按这样的树链剖分方法可以简单地均摊维护拐点
- 这个方法是1log的，关于LCT上的势能分析比较麻烦，暂时略过

- 总时间复杂度O( (n+m)logn )



# CF1083F The Fair Nut and Amusing Xor 3300

对于两个等长的非负整数序列，我们定义这两个序列的相似度为将其中一个序列转化为另一个序列所需的最小操作次数。一次操作定义如下：

- 选择序列中的一个长度为 $k$ 的子段，将子段内的所有元素异或上一个相同的值 $x$，$x$ 可以任意决定。

例如，当 $k = 2$ 时，将序列 $\{0, 0, 0\}$ 的子段 $[1, 2]$ 内的所有元素异或上 $1$ 得到序列 $\{1, 1, 0\}$，之后将子段 $[2, 3]$ 内的所有元素异或上 $2$ 得到序列 $\{1, 3, 2\}$，因此序列 $\{0, 0, 0\}$ 与序列 $\{1, 3, 2\}$ 的相似度为 $2$。

特殊地，若其中一个序列无论如何都不能转化到另一个序列，那么这两个序列的相似度为 $-1$。

给定两个长度为 $n$ 的序列 $\{a_i\}$ 与 $\{b_i\}$，你需要求出它们的相似度。这之后，将会有 $q$ 次修改操作，单次操作格式如下：

- `s p v`：其中，$s$ 是一个小写字符，且要么为 `a`，要么为 `b`。若 $s =$ `a`，则表明将 $a_p$ 的值修改为 $v$；若 $s =$ `b`，则表明将 $b_p$ 的值修改为 $v$。

在每一次修改操作结束后，你也需要求出两个序列的相似度。输出最开始及每次修改后的答案。

$1 \le k \le n \le 2 \times 10^5, 0 \le q \le 2 \times 10^5, 1 \le p \le n, 0 \le a_i, b_i, v < 2^{14}$，输入保证给出的两个序列长度均为 $n$。

# Solution

- 定义c[i]=a[i]^b[i]，问题即将c转换为全0数组的最小代价
- 如果不带修改，可以贪心解决：
- 令i=1->n，若c[i]!=0，则对c[i]...c[i+k-1]异或上c[i]，c[i]!=0的次数即为答案
- 定义f[i]为对[i,i+k-1]进行了异或f[i]的操作
- 考虑位置i与i+1，对于f[i-k+2]...f[i]，i与i+1都进行了相同的异或操作
- 只有i受到f[i-k+1]的影响，i+1不受到，而计算到i+1时，c[i]经过修改已经变为0了

# Solution

- i处初值为c[i]，受到了f[i-k+1]...f[i]的影响，现在i处值为0
- i+1处初值为c[i+1]，受到了f[i-k+2]...f[i]的影响
- 故c[i]^f[i-k+1]^...^f[i]=0
- i+1处受到的影响为f[i-k+2]^...^f[i]=0^c[i]^f[i-k+1]
- c[i+1]此时的值为c[i]^f[i-k+1]^c[i+1]
- 此时i+1处应该进行的修改为f[i+1]=c[i]^c[i+1]^f[i-k+1]
- 定义d[i]=c[i]^c[i+1]，即f[i+1]=d[i]^f[i-k+1]
- f[i]=d[i-1]^f[i-k]

# Solution

- 可以发现对于f[i]，其递推式展开后为
- f[i]=d[i-1]^d[i-k-1]^d[i-2k-1]^...^d[i-xk-1]，i-(x+1)k-1<=0
- 可以将下标i按照模k意义进行划分
- 每次修改只影响O(1)个d[i-1]，设值从A改为B，等价于是对f的一个后缀异或上了A^B
- 全局f[i]=0的位置即不需要进行修改的位置
- 若f[n-k+2]...f[n]中有非0的位置，则无解

# Solution

- 按对k取模后分类，问题变为给定一个序列
- 1.区间对x做xor
- 2.后缀0元素个数
- 序列分块，每个块开一个桶表示每个值出现次数（哈希表也行）
- 然后直接维护即可
- 为了卡空间可以离线逐块处理

- 总时间复杂度O(n+qsqrtn)

# CF1322E Median Mountain Range 3300

长度为$n$的序列$a_i$,每次把$[2, n-1]$之间的数变成$\text{median}(a_{i-1}, a_i, a_{i+1})$，其中median表示三个数中的中位数。

问多少次操作之后，序列就不再发生变化了。输出序列发生变化的次数，以及最终的序列。

$n \leq 500000, 1 \leq a_i \leq 10^9$

# Solution

# CF1361F Johnny and New Toy 3300

给定一个序列 $P_1W_1P_2W_2\cdots W_{n-1}P_n$

是一个 $1$ 到 $n$ 的排列，$W$ 是一个 $1$ 到 $n-1$ 的排列，定义 $W_0=W_n=0$

每次可以选择一对 $1\le l\le r<n$ 满足 $\min_{i=l}^{r}W_i>\max\{W_{l-1},W_{r+1}\}$

设 $W_{l\cdots r}$ 中的最小值为 $W_m$，把 $P_l$ 到 $P_m$ 组成的段和 $P_{m+1}$ 到 $P_{r+1}$ 组成的段互换位置

这种操作可以进行任意多次，有 $q$ 次修改，每次交换 $P$ 中的两个数，修改后求 $P$ 能达到的逆序对个数最小值

# Solution

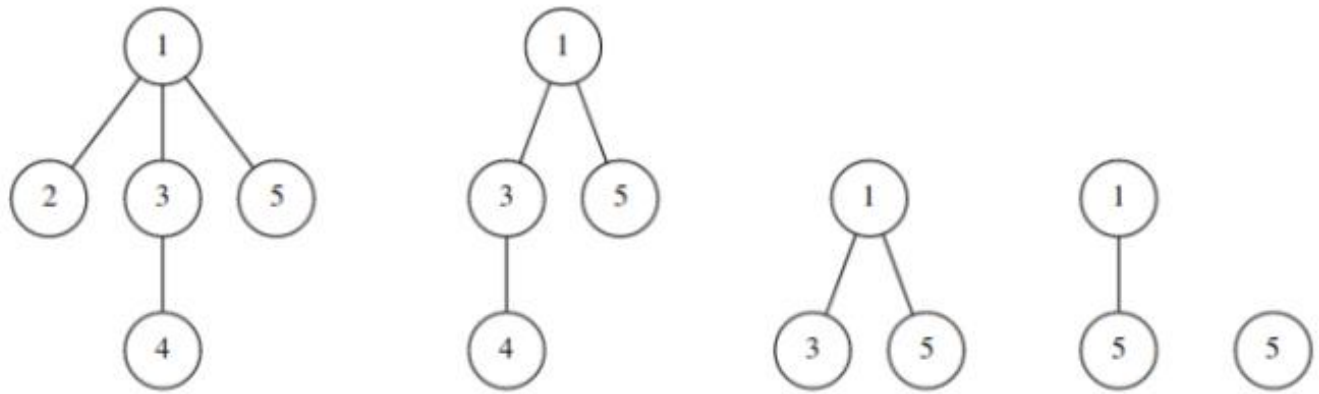
# CF1477E Nezzar and Tournaments 3300

- 等一手中文题意

# Solution

# CF1137F Matches Are Not a Child's Play 3400

我们定义一棵树的删除序列为：每一次将树中编号最小的叶子删掉，将该节点编号加入到当前序列的最末端，最后只剩下一个节点时将该节点的编号加入到结尾。




例如对于上图中的树，它的删除序列为：2 4 3 1 5

现在给出一棵$n$个节点的树，有$m$次操作：

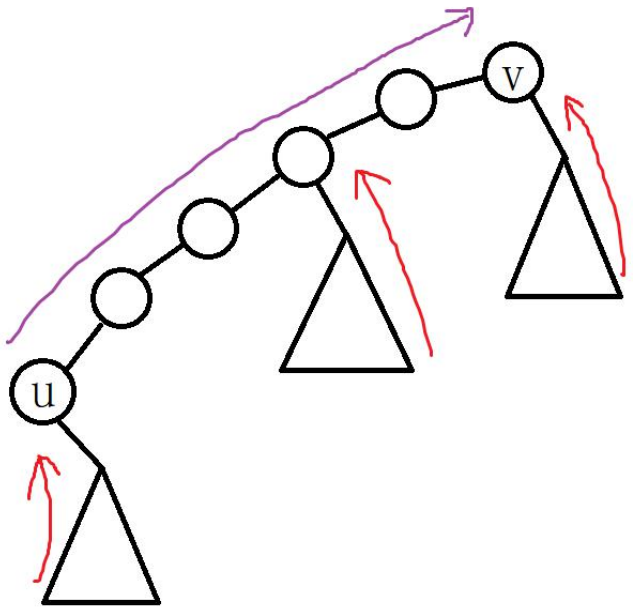
`up v`：将$v$号节点的编号变为当前所有节点编号的$\max+1$

`when v`：查询$v$在当前树的删除序列中是第几号元素

`compare u v`：查询$u$和$v$在当前树的删除序列中谁更靠前

# Solution

- 3操作可以由两次2操作代替
- 由于这个修改只能将一个点修改为最大值max+1
- 考虑之前最大值a[u]=max，修改后a[v]=max+1
- 则这次修改的影响是，先烧完了除u到v路径上的所有点，然后烧u，沿着u到v的路径烧到v
- 考虑除了这条路径以外的点
- 可以发现这些点被烧的相对顺序不发生变化



# Solution

- 考虑LCT上的access均摊，每次修改即
- makeroot(u)，然后access(v)，之后u到v的路径按u到v的顺序处于删除序列的末端，即[n-x+1,n]，x为路径上点数
- 使用一个数据结构维护这样的连续段，支持access时将一些连续段删除，之后加入一个新的连续段
- 查询一个点在删除序列的位置时，在LCT上找到其属于哪个连续段，然后通过维护的数据结构查询其排名即可

- 总时间复杂度O((n+m)log^2n)

# CF757G Can Bash Save the Day? 3400

一棵 $n$ 个点的树和一个排列 $p_i$，边有边权，支持两种操作：

- `1 x x`，询问 $\sum_{i=l}^{r} dis(p_i, x)$。
- `x`，交换 $p_x, p_{x+1}$。

$n, q \leq 2 \times 10^5$，强制在线。

# Solution

- 点到集合的dist和可以用点到根加点到根和的方法处理，可以使用静态LCT做到1log，不过一般用2log的树链剖分+线段树也可以
- 不带修改的话就可持久化一下线段树，每次询问差分[l,r]=[1,r]-[1,l-1]，然后在两个可持久化线段树的位置上查询
- 这个swap相邻的操作如何处理？
- 直接换成修改？
- 但是这样会多一个log

# Solution

- 注意到我们只swap相邻的两个位置
- 比如swap了x和x+1的位置，那对1,2,...x-1的可持久化数据结构没有影响，由于有交换律，所以对x+2,x+3,...n的可持久化数据结构也没有影响
- 可以把x-1的可持久化数据结构插入p[x+1]，作为x的可持久化数据结构
- 树链剖分复杂度为O((n+m)log^2n)，使用静态LCT可以做到O((n+m)logn)
- 总时间复杂度O((n+m)logn)

# CF1039E Summer Oenothera Exhibition 3400

- 给定长度为 n 的序列，常数 w 。
- 接下来 q 组询问，每次给出一个常数 k，求最少把序列分为多少段使得每段序列中数的极差不超过 w-k，输出最小的段数 -1。

# Solution

- 考虑从小到大对询问排序，设当前处理的询问极差为x
- 极差随着加入的数变多而单调不降
- 维护出f[i]表示当前从i开始，区间[i,f[i]-1]的极差<=x，[i,f[i]]的极差>x
- 答案即每次从i开始，i <- f[i]，这样迭代的轮数
- 对每个i记录g[i]表示其f[i]变大需要询问的极差变大多少
- 当极差变化时，枚举当前g最小的位置，看其f是否需要变化
- 当f变化时，可以二分之后f的位置，并用ST表求出rmq

# Solution

- 根号分治，令B为阈值
- 对每个i，若f[i]-i<B，使用LCT维护f[i]构成的树，每次f[i]发生变化时即进行O(1)次link cut，查询答案即查询i到根的路径
- 若f[i]-i>=B，则在这里将其标记为动态树森林中的一个根节点，并不再更新其f值
- 每次查询答案的过程最多跳过O(n/B)棵动态树
- 每次f发生变化时代价为O(logn)，共有O(nB)次变化

- 总时间复杂度O(nsqrtmlogn)

# CF936E Iqea 3400

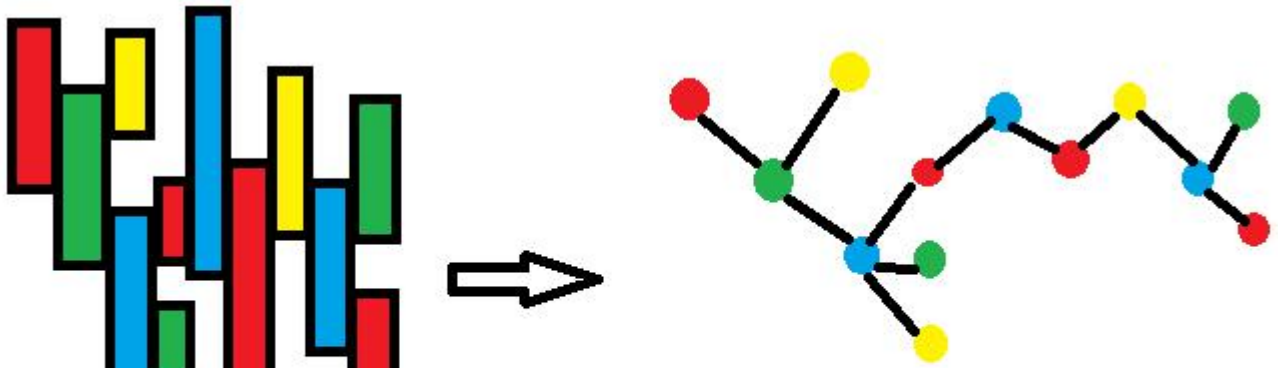
在无限的二维网格上，你需要维护两种操作。

1. 在 $(x, y)$ 处新开一个商店。
2. 询问离 $(x, y)$ 最近的商店的距离。

唯一能够通过的路径，是 $n$ 个给定的点，若两个点的位置四相邻，那么可以直接通过。保证这些长条连通且二维网格的剩余部分也连通。路径的长度定义为走过的网格数。

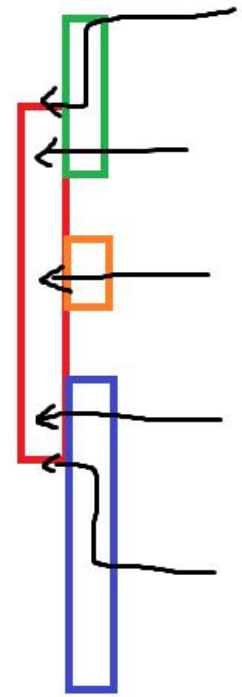
# Solution

- 长条连通，剩余部分也连通，说明构成一个类似于树结构的东西
- 求出每一列每个极长的给定点区间
- 对这样的区间，将其当做树上的一个节点，变成一棵边权为1的树



# Solution

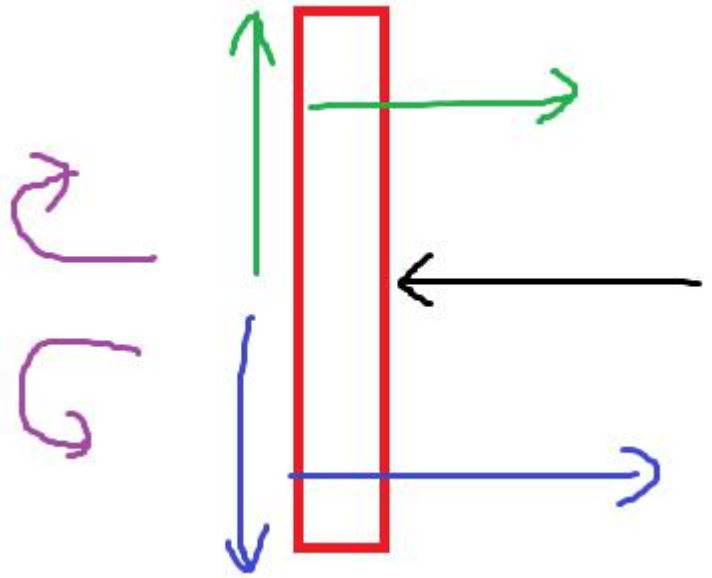
- 上述的建树过程是只考虑x轴距离，没有考虑y轴距离的
- 进行点分治，每个分治中心维护一个数据结构
- 这里的分治中心指的是一个极长的列，我们将其当做区间建立一棵线段树，支持查询一个区间中算上了y轴距离的商店
- 修改时沿着点分治的树形结构向上爬
- 每次更新当前分治中心当前爬到的位置上的值



# Solution

- 每次查询即分别查询前后缀算上y轴在当前分治中心的序列上移动的距离的贡献后的最小商店
- 使用普通的树分治方法时间复杂度O(nlogn+mlog^2n)
- 可以建立top tree做到O(n+mlogn)

- 总时间复杂度O(n+mlogn)



# CF833E Caramel Clouds 3400

天上有$n$朵云，每朵云$i$会在时间$[l_i, r_i]$出现，你有$C$个糖果，你可以花费$c_i$个糖果让云i消失，同时需要保证你最多让两朵云消失．现在有m个独立的询问，每次给你一个需要让阳光照k时间的植物，问你从时刻0开始，这个植物最快什么时候能长成．$n, m \leq 3 \times 10^5$

# Solution

# CF1060G Balls and Pockets 3400

给出一个无限长的序列 $p_0, p_1, p_2, \ldots$，初始 $p_i = i$。

给出 $n$ 个互不相同的整数 $a_1, a_2, \ldots, a_n$，可以对序列 $p$ 做以下操作若干次：

- 将序列 $p$ 的第 $a_1, a_2, \ldots, a_n$ 项从序列 $p$ 中删掉，然后将剩余的数字按照原来在序列 $p$ 中的顺序重新排列作为新的序列 $p$。

举个栗子：初始序列为 `0 1 2 3 4 5 6 7 8 9...`，$a = \{1, 3, 4\}$，那么

- 经过 1 次操作后序列变为 `0 2 5 6 7 8 9 10 11 12...`；
- 经过 2 次操作后序列变为 `0 5 8 9 10 11 12 13 14 15...` .....

给出 $m$ 组询问，每组询问给出两个整数 $x, k$，询问进行 $k$ 次操作后的序列 $p$ 中 $p_x$ 的值。

**输入格式**

第一行两个整数 $m, n (1 \leq m, n \leq 10^5)$,

第二行 $n$ 个整数 $a_1, a_2, \ldots, a_n (0 \leq a_1 < \ldots < a_n \leq 10^9)$。

接下来 $m$ 行每行两个整数 $x_i, k_i (0 \leq x_i, k_i \leq 10^9)$ 描述一组询问。

# Solution

# CF853E Lada Malina 3400

# Solution

# CF1491H Yuezheng Ling and Dynamic Tree 3400

给定一棵大小为 $n$ 的 $1$ 为根节点的树，树用如下方式给出：输入 $a_2, a_3, \ldots, a_n$，保证 $1 \leq a_i < i$，将 $a_i$ 与 $i$ 连边形成一棵树。

接下来有两种操作：

- `1 l r x` 令 $a_i = \max(a_i - x, 1)(l \leq i \leq r)$。
- `2 u v` 查询在当前的 $a$ 数组构成的树上 $u, v$ 的 LCA。

两种操作共有 $q$ 个。

$2 \leq n, q \leq 10^5$，$2 \leq l \leq r \leq n$，$1 \leq x \leq 10^5$，$1 \leq u, v \leq n$

# Solution

- 按sqrtn为块大小对序列分块，对每个点i，维护f[i]表示i跳父亲，第一个跳到的块外节点
- 可以发现，如果一个块被作为整块修改了sqrtn次，则这个块中每个元素i，其f[i]=fa[i]，之后对f的修改可以直接通过打标记实现
- 零散块中的f直接O(sqrtn)从左往右重构即可

# Solution

- 查询x,y的LCA时，类似树链剖分跳重链
- 若当前x,y不在同一个块中，设f[x]<f[y]，则y=f[y]
- 若当前x,y在同一个块中：
- 1. x!=y，则在这之前的过程中没有错过二者的LCA，继续x=f[x],y=f[y]
- 2. x=y，则之前的过程中可能错过二者的LCA，找到现在的x和y上一次跳块的位置，从该位置开始暴力跳O(sqrtn)次可以找到LCA
- 总时间复杂度O((n+m)sqrtn)

# CF1515H Phoenix and Bits 3500

维护大小为 $n$ 的集合 $a$，$q$ 次操作：

- `1 x y z`：将大小在 $x$ 到 $y$ 之间的数按位与 $z$。
- `2 x y z`：将大小在 $x$ 到 $y$ 之间的数按位或 $z$。
- `3 x y z`：将大小在 $x$ 到 $y$ 之间的数按位异或 $z$。
- `4 x y`：查询在 $x$ 到 $y$ 之间有多少个不同的数。

$1 \leqslant n \leqslant 2 \times 10^5$，$1 \leqslant q \leqslant 10^5$。

# Solution

- 3500就这？
- 建一棵trie树
- 每次修改拆出logn个子树
- xor操作可以打标记维护
- 查询即和普通线段树类似，递归找到logv个区间表示查询的区间

# Solution

- and和or操作等价于将某些位不变，其他位都强制设为同一个数
- 可以通过维护对于子树内每个位，是否存在这个位为0或1的元素来实现，如果这个位同时有0,1，则暴力递归进子树合并点，否则对于这个位来说，and和or的操作等价于xor操作
- 初始有O(nlogv)个trie上的节点
- 每次递归进子树合并会减少1个节点，并且代价为O(logv)
- 故总时间复杂度O((n+m)log^2v)

# CF799F Beautiful fountains rows 3500

- 在一个长度为 m 的数轴上，有 n 种球，每种球会出现在区间 [li,ri] 中。
  一个合法的区间满足：这个区间里有球，并且每种出现过的球都出现了奇数次
  求所有合法的区间的长度之和。

# Solution

- 扫描线扫右端点，数据结构维护左端点
- 分别维护当前还会向右延伸的区间，以及当前已经完全在左侧的区间
- 假设当前扫描到i，找出还会向右延伸的区间中，最小的x，满足x是这个区间的左端点，[x,i]长度为偶数，则x+1是左端点的一个下界
- 找出完全在左侧的区间[y,z]，满足[y,z]的长度为偶数，则y+1是左端点的另一个下界
- 将这两个下界取max

# Solution

- 对于一个已经在左侧的区间[a,b]，其对答案的限制为
- 所有[x,i]，满足a<=x<=b，且x-b为偶数的x是不受[a,b]限制的
- 所有[x,i]，满足a<=x<=b，且x-b为奇数的x是被限制的，即这样的x不对答案造成贡献
- 将奇偶分别使用一个线段树进行维护，右端点为i时，即查询i的奇偶性对应的线段树的答案
- 上述限制当一个区间完全在左侧时发生，可以用一个区间修改为0表示
- 所有可以延伸的区间只影响答案的下界，因为右端点一致
- 总时间复杂度O(nlogn)

# CF1034D Intervals of Intervals 3500

- 有 n 个区间 [ai,bi] 定义区间的区间 [l,r] 的价值是第 l 个区间到第 r 个区间的并的长度，找出 k 个不同的区间的区间，使得总价值最大。

# Solution

# CF1083D The Fair Nut's getting crazy 3500

给定一个长度为 $n$ 的序列 $\{a_i\}$。你需要从该序列中选出两个非空的子段，这两个子段满足：

- 两个子段非包含关系。
- 两个子段存在交。
- 位于两个子段交中的元素在每个子段中只能出现一次。

求共有多少种不同的子段选择方案。输出总方案数对 $10^9+7$ 取模后的结果。

需要注意的是，选择子段 $[a,b]$、$[c,d]$ 与选择子段 $[c,d]$、$[a,b]$ 被视为是相同的两种方案。

$1 \leq n \leq 10^5, -10^9 \leq a_i \leq 10^9$。

# Solution

- 我感觉Owen_codeisking的题解写的好就直接复制了

首先枚举交集 $[l,r]$，然后计算每个交集的答案。

定义

$\mathrm{pre}_i$ 为数值和 $a_i$ 相同，最靠近 $i$ 且 $< i$ 的位置，若不存在为 $0$。

$\mathrm{suc}_i$ 为数值和 $a_i$ 相同，最靠近 $i$ 且 $> i$ 的位置，若不存在为 $n+1$。

记

$$f(i,j)=\max(\mathrm{pre}_i,...,\mathrm{pre}_j)+1$$

$$g(i,j)=\min(\mathrm{suc}_i,...,\mathrm{suc}_j)-1$$

则

$$ans=\sum_{i=1}^{n}\sum_{j=i}^{n}[i\geq f(i,j)][g(i,j)\geq j](i-f(i,j))\times(g(i,j)-j)$$

考虑怎么算这个答案。

我们先枚举 $i$。这个 $f(i,j)$ 不降，$g(i,j)$ 不升，一定有一个最大的 $j$，使得 $[i,i],...,[i,j]$ 计入答案。我们发现对于 $i_1<i_2$，$j_1<j_2$，这样就可以双指针计算 $j$。

# Solution

现在的问题就是对于每个 $i$，怎么快速计算

$$\sum_{k=i}^{j}(i-f(i,k))\times(g(i,k)-k)$$

现在把括号拆开

$$=\sum_{k=i}^{j}g(i,k)-i\times\sum_{k=i}^{j}k-\sum_{k=i}^{j}f(i,k)\times g(i,k)+\sum_{k=i}^{j}f(i,k)\times k$$

这些东西可以在线段树上搞。每次 $i$ 左移时，把所有值取 $\max$ / $\min$ 的操作换成区间赋值。

现在你需要维护几种操作：

1. 对 $a/b$ 序列区间赋值
2. 求 $a/b$ 序列区间和
3. 求 $a/b$ 序列区间 $(a/b)_i\times i$ 和
4. 求 $a_i\times b_i$ 和

时间 $\mathcal{O}(n\log n)$

# CF1336F Journey 3500

- 给定一棵树和 m 条链，求多少对链的交中包含的边数 $\geq k$。

# Solution

# CF1148H Holy Diver 3500

给定一个最初为空的数组，需要支持以下操作：

- 给定 $a, l, r, k$，在数组末尾插入 $a$，然后查询有多少数对 $(i, j)$ $(l \leq i \leq j \leq r)$，满足 $\operatorname{mex}(\{a_i, a_{i+1}, a_{i+2}, \ldots, a_j\}) = k$。

强制在线。

$\operatorname{mex}(S)$ 表示集合 $S$ 中最小的未出现的**自然数**。

# Solution

- 这个强制在线，在末尾插入我感觉实际上是降低题目难度的
- 因为这个暗示了可能是扫描线，并且扫描线很可能是从左往右扫序列维，并且知道扫描线做法后可以平凡地带后端插入
- 扫描线扫序列右端点，数据结构维护每个mex=i时，左端点可能有多少个，记为f[i]
- 维护last[i]表示i最后的出现位置
- 可以发现对mex=i，可能的左端点是一段连续区间，min(last[j])(j=1->i-1)为这个区间右端点，min(last[j])(j=1->i)为这个区间左端点

# Solution

- 当从i扫描到i+1时，last[a[i+1]]=i+1，即单点修改为全局最大值
- 记f1[i]= min(last[j])(j=1->i-1),f2[i]=min(last[j])(j=1->i)
- 有f[i]=f1[i]-f2[i]
- 可以发现实际上相当于一棵动态笛卡尔树，每次last[a[i+1]]=i+1即将a[i+1]单旋到根，维护f1,f2需要维护最左链
- 直接套用动态笛卡尔树是O((n+m)logn)的
- 讲个阳间点的做法，只考虑维护f1，每次修改时，若a[i+1]-1的前缀min与a[i+1]相同，则这个修改不对f1造成任何影响
- 若f1[x']==f1[x'+1]，则认为x'与x'+1构成连续关系，序列变为若干连续段

# Solution

- 若不同，则找到最小的x，满足a[i+1]<x，且last[x]=min(last[j])(j=1->x)
- 则a[i+1]-1,a[i+1]...x-1构成连续关系，继续考虑x与后面的位置构成的连续关系
- 总的连续段变化次数为O(n)
- 因为修改是将一个位置修改为全局最大值，当x与x+1不连续（即f1[x]!=f1[x+1]）后，1...x的修改只会让f1[x]变小，x+2...n的修改对f1[x+1]无效，所以只有x+1位置的修改可能让x与x+1连续，而每次修改只修改一个位置，故每次最多减少1个连续段，但可能将一个连续段拆为多个连续段

# Solution

- 接下来解决区间询问
- 询问[l,r]内所有子区间的mex=k的答案，等价于询问扫描线右端点在l...r中的每个位置i时：
- 若f1[i]<l，则贡献为0
- 若f2[i]>l，则贡献为f[i]
- 否则贡献为f1[i]-l+1

# Solution

- 数一下维度，l...r一维，f1/2一维，而l...r是在线向后插入的维，这里实际上是一个带可持久化的一维问题
- 故对每个mex=k开一棵可持久化缩点平衡树（将连续段差分后线段树也可以）维护，讨论一下即可，毕竟我们有连续段均摊了可以为所欲为
- 至于后端插入，我们扫描线的顺序是对的，天然支持插入

- 总时间复杂度O((n+m)logn)
- （感觉做复杂了，不过这题感觉平凡没有深究的价值，就不管了）

# CF1545F AquaMoon and Potatoes 3500

- 给你三个长为n的序列a,b,c
- 有两种操作：
- 1.a[i]修改为x
- 2.查询前缀[1,r]中有多少(i,j,k)满足i<j<k<=r且b[a[i]]=a[j]=c[a[k]]

# Solution

- 有好多种做法，在线空间带根号，离线空间线性
- 考虑每次处理根号次操作
- 我们每次处理先把这个时间块内被修改的点设为-1，然后就变成有一些新加的点，一些已经存在的点，要计算答案
- 讨论一下每个点是不是新加的点

# Solution

- 1. i,j,k都是新点
- DP一下，每次f1[c[a[i]]]++，然后f2[a[i]]+=f1[c[a[i]]]，然后ans+=f2[b[a[i]]]
- 2. i,j,k都是旧点
- 我们每个时间块内重新DP一下整个序列，这次DP可以求出每个前缀的答案，顺便维护这个就行

# Solution

- 3. i,j是新点，k是旧点
- 枚举每个新的j，for出其前面对应的i个数，则需要查询O(sqrtn)次区间中有多少k满足b[a[k]]=a[j]，将这样的询问离线下来
- 4. j,k是新点，i是旧点
- 与3一样

# Solution

- 5. j是新点，i,k是旧点
- 对每个j查左边b[a[i]]=a[j]的i个数，查右边c[a[k]]=a[j]的k个数
- 6. i,k是新点，j是旧点
- 枚举k，三元组个数即前缀中

# CF1208H Red Blue Tree 3500

有一棵 $n$ 个节点的树，然后给定一个数 $k$，

每个叶子结点都有一个固定的颜色（红或蓝），

其中红色用 $0$ 表示，蓝色用 $1$ 表示

对于每个非叶子节点 $u$ 的儿子 $v$，如果 $\sum(color_v = blue) - \sum(color_v = red) \geq k$

那么这个节点就是蓝色的，否则就是红色的

现在要求支持三种操作：

1 v 输出 $v$ 节点的颜色

2 v c 把 $v$ 节点的颜色改为 $c$

3 h 把当前 $k$ 的值改成 $h$

# Solution