PTZ summer 2020 Day1 I

显然每次都是吃那些比自己小的鱼中最大的。

那么找到大于等于自己的最小的鱼 i，现在只能吃比它小的那段前缀，考虑在线段树上二分最小的后缀使得吃完这个后缀之后可以吃 i，然后在线段树上对吃掉这段打标记即可。每次这个过程都会让自身重量至少翻倍，所以复杂度是 $O(\log n \log W)$。在询问完之后撤回所有标记即可。

复杂度 $O(q \log n \log W + n \log n)$。

PTZ summer 2020 Day4 E

设 La_i 表示 a 中从左至右第 i 个 1 的位置，Ra_i 表示 a 中从右至左第 i 个 3 的位置，$suma_i$ 表示 a 中前 i 个位置有几个 2。同理有 $Lb_i, Rb_i, sumb_i$。

那么我们枚举 i, j，答案就是 $i + j + \min(suma_{Ra_j - 1} - suma_{La_i}, sumb_{Rb_j - 1} - sumb_{Lb_i})$，其中要满足 $Ra_j > La_i, Rb_j - Lb_j$，那么我们可以在枚举 j 的时候双指针维护合法的 i，然后有树状数组维护上面这个式子。

复杂度 $O(n \log n)$。

CF1656H

一个简单的想法是 $a_i \nmid \operatorname{lcm}\{b_j\}$，那么 a_i 不可能在答案内，删去，反过来一样。一直删到 a_i, b_i 都没有数可以删了，就是答案（如果是空集就无解）。

考虑 $a_i \mid \operatorname{lcm}\{b_j\} \iff \gcd(a_i, \operatorname{lcm}\{b_j\}) = a_i \iff \operatorname{lcm}\{\gcd(a_i, b_j)\} = a_i$。那么对每个 a_i，开一个线段树维护现在还没有被删的所有 b_j，$\operatorname{lcm}\{\gcd(a_i, b_j)\}$ 的值。对于每个 b_j 也类似开一棵线段树。

考虑只有 $O(n)$ 次删数，每次影响 $O(n)$ 个线段树。每次删除看起来是 $O(\log W \log n)$ 的，但是事实上，每次求 $\operatorname{lcm}(a, b)$ 的复杂度为 $O(\log \min(a, b) - \log \gcd(a, b))$，所以每次删除应该是 $O(\log W + \log n)$ 的。

总复杂度 $O(n^2 (\log W + \log n))$。

PTZ winter 2020 Day5 D

首先我们枚举一条线段 p 并钦定它是我们选出方案里最短的那条，那么我们就可以忽略掉那些严格在它内部的线段以及与它无交的线段，然后对于剩余的线段，如果一条线段同时包含了 p 的两个端点，那么它加入方案一定更优，我们记录下这样的线段的数量然后忽略它们。

剩下的线段只有那些恰好包含 p 一个端点的线段。我们设 A 为包含 p 左端点的线段的集合，B 为包含 p 右端点的集合。那么我们只需要保证，对于选出的线段 a, b，假如 $a \in A, b \in B$，那么 a, b 必须有交。设 len_i 为线段 i 的长度，in_i 为线段 i 和 p 交的长度，out_i 为 len_i 减去 in_i，那么就是要保证 $in_a + in_b > len_p \vee out_a + out_b > n - len_p$。把一个 A 中的线段看作黑点 (in_a, out_a)，B 中的线段看作白点 $(len_p - in_b, n - len_p - out_b)$。那么就是要选出一个点集，要保证不存在一个黑点处于某一个白点的左下方。

那么就是选择一条单调不升的折线，把折线下方的白点都加入答案，折线上方的黑点都加入答案，这一步可以直接 `dp`。

需要一个支持前缀加后缀加，全局取后缀max的数据结构。

复杂度 $O(n^2 \log n)$。

CF1270H

发现所有连通块都是在原序列上的一段区间。所以考虑用线段树维护，维护区间最大值，区间最小值，区间连通块数，跨过中线的连通块的信息以及这个连通块覆盖了左子区间原来的连通块个数，覆盖了右子区间原来的连通块个数。

每次合并就是把左区间的一段后缀的连通块和右区间的一段前缀的连通块合成一段区间，考虑在线段树上二分即可。

这个做法甚至可以区间询问。

复杂度 $O(n\log n+q\log^2 n)$。

PTZ winter 2021 Day9 A

我们硬点 1 作为根，那么操作一可以看作子树加（或者全局加然后子树减）。

$\sum dis(v,i)A_i=\sum dep_vA_i+dep_iA_i-2dep_{lca(v,i)}A_i$。

前两项非常容易维护，最后一项可以看作，i 到根的路径加上 A_i，然后询问 v 到根的和。

树剖之后设 $B_i=\sum\limits_{v\in subtree_i}A_v$，那么子树 A_i 加 1，就变成了对子树 B_i 加 sz_i。

链加就相当于对 B_i 链加一次函数。

复杂度 $O(n\log^2 n)$。

Yandex Algorithm 2018 Round 2 A

考虑如何判定 s 是好的，我们希望找到三个不同的元素 a,b,c 使得 $a^2=b^2=c^2=1$。那么我们只需要用 a,b,c 替换 `a`,`b`,`c` 之后计算所有元素乘起来是否为 1 即可。

我们这里可以元素为 3×3 的矩阵，其中 a,b,c 分别为三维坐标系中，点 (x,y,z) 对某个过原点的平面 $Ax+By+Cz=0$ 作对称的坐标变化矩阵，具体的：

$$D=2.0/(A^2+B^2+C^2)$$
$$\begin{bmatrix}A^2D-1 & ABD & ACD\\ BAD & B^2D-1 & BCD\\ CAD & CBD & C^2D-1\end{bmatrix}$$

然后考虑交换两个 s_i,s_j 怎么做，我们可以分治，并钦定 i,j 分别在分治中心两边。然后我们可以枚举 s_i,s_j 分别为什么，然后 `meet in middle`。

复杂度 $O(n\log n)$，常数略大。

CF1548E

考虑一个点不能走到另一个 a_i+b_j 比它小（如果相同，先比较 a_i，如果还相同，比较 i）的点，那么我们称它为代表点。显然一个联通块只有一个代表点，所以数代表点的个数即可。

那么考虑一个点是否可以走到比它小的点，设 pra_i,sfa_i 为
$\max\{j|a_j<a_i,j<i\},\min\{j|a_j<a_i,j>i\}$，$prb_i,sfb_i$ 类似。

那么 (i,j) 如果可以走到比它小的点，那么一定可以走到 $(pra_i,j),(sfa_i,j),(i,prb_j),(i,sfb_j)$ 其中一个。

设 $na_i=\min(\max\{a_{pra_i},\dots,a_i\},\max\{a_i,\dots,a_{sfa_i}\})$，$nb_i$ 类似。

所以 a_i,b_j 为代表点，当且仅当：

$$a_i + b_j \le x$$
$$na_i + b_j > x$$
$$a_i + nb_j > x$$

求出 na, nb 后二维数点即可，$O(n \log n)$。

CF1209G2

考虑每个颜色最左边出现的位置为 L_i，最右边的位置为 R_i，那么在最后的序列里，对于 $j \in [L_i, R_i), a_j = a_{j+1}$。

所以我们可以给每个 $[L_i, R_i)$ 区间给 c_i 加一，那么 c_i 中每个极长非 0 段的答案就是独立的。我们需要让这一段最多不需要改变的颜色最多，那么就是这个区间出现次数最多的颜色。

那么我们把每个颜色的个数放在 L_i 维护，记为 b_{L_i}，那么每次就是询问每个 c_i 中非 0 段中 b_i 的最大值的和。

那么每次修改，我们可以用 `set` 维护每个颜色的 L_i, R_i。每次先去除该颜色的贡献，然后修改后再加上它的贡献。

我们使用线段树来维护，发现用 0 分割区间很难维护，考虑使用 c_i 的最小值来分割，注意到 c_n 一定是 0，最后 $[1, n]$ 的答案不会变。

于是线段树上每个节点记录这个区间中，b_i 的最大值，c_i 的最小值，最左边一段不包含 c_i 最小值的极长段中 b_i 的最大值 Lmx，同理最右边一段 Rmx，以及除了左右两端之外的每一段的 b_i 的最大值和。合并显然。

复杂度 $O(n \log n)$。

PTZ summer 2021 Day3 I

考虑容斥，答案是 $\binom{n}{3}-$ 不合法的对数。考虑不合法的三元组只有三种：两个有交且都和另一个无交，一个和两个都有交且另两个无交，三个两两有交。这三种的个数分别记为 c_1, c_2, c_3。

我们考虑 d_i 为与第 i 个矩形有交的矩形个数，那么考虑 $\sum d_i(n-2) = 2c_1 + 4c_2 + 6c_3$，$\sum \binom{d_i}{2} = c_2 + 3c_3$。那么我们就得到了 $c_1 + c_2$，接下来我们只需要算出 c_3。

考虑枚举哪个矩形的 u_i 是三个矩形里最小的，那么剩下的矩形需要和这个矩形有交，即保证 $d_j < u_i < u_j$，我们可以看作在 u_j 的位置插入了 $[l_j, r_j]$，然后在 d_j 处删去，那么我们就相当于每次查询现在的区间里，有哪些 $[l_j, r_j], [l_k, r_k]$ 使得 $[l_i, r_i], [l_j, r_j], [l_k, r_k]$ 两两有交。那么再次容斥，就变成了选两个区间和 $[l_i, r_i]$ 有交的方案减去选两个区间和 $[l_i, r_i]$ 有交且这两个区间无交的方案，前者可以直接算，后者考虑相当于满足 $l_i < r_j < l_k < r_i$，那么用线段树维护每个区间里有多少个区间的左端点和有多少个线段的右端点即可。

复杂度 $O(n \log n)$。

XX opencup GP of Poland E

如果我们可以求出每条边的断裂时间，那么就可以用 `kruskal` 重构树在 $O((n+m)\log n + q)$ 的时间内回答询问。那么接下来只考虑求出边的断裂时间。

考虑如果初始图是树怎么做。

每次修改 (x, h)，那么先检查 x 和父亲的边是否断裂，然后考虑如何快速地检查 x 和儿子的边。考虑儿子 v 假如 z_v 固定，那么满足 (x, v) 这条吧不断裂的 z_x 形成一段区间，把这段区间记录在 x 节点上。那么修改 x 时，只需要检查有没有区间不包含修改后的 z_x，用 `set` 维护，复杂度 $O(m \log m + k \log m)$（注意，还需要修改 x 放在父亲处的区间）。

考虑转化为一般图的做法，相当于我们给每条边定向，每次我们暴力检查一个点的出边，然后用 `set` 维护入边，复杂度就是 $O(m\log m+dk\log m)$，其中 d 是最大出度。所以接下来我们需要让 d 在一个可以接受的量级里。

一个比较直觉的做法是，我们每次拿出度数最小的点，把和它相连的边都定向为出边，然后删掉这个点继续做。

假如是一般图，那么 d 是 $O(\sqrt{m})$ 级别的。但是题目给出的图是平面图，可以知道平面图最小的度数肯定是小于等于 5 的。

那么总复杂度就是 $O(m\log m+k\log m+q)$。

THUPC 2024 A

待定阈值 B，记 c_x 为 $[1,x]$ 这个前缀被操作了几次，把 $[1,n]$ 划分为若干个区间 $[l_1,r_1],[l_2,r_2],\ldots,[l_k,r_k]$，其中 $l_1=1,r_k=n,\forall i,r_i=l_{i+1}-1$。使得每个区间满足 $\sum_{j=l_i}^{r_i}c_j\le B\vee(l_i=r_i\wedge c_{l_i}>B)$，那么有 $k=O(n/B)$。

对于每个区间尝试对所有询问求出答案，对于完整覆盖了区间的操作，可以使用前缀和直接记掉。考虑此时零散的操作只有 $O(B)$ 个，故本质不同的询问只有 $O(B^2)$ 个，分治求出即可，复杂度是 $T(B)=2T(B/2)+O(B^2)=O(B^2)$。

总复杂度为 $O(n/B(n+m)+nB)=O((n+m)\sqrt{n})$。