2022 牛客

暑期多校训练营

EZEC and QCJJ fan club

Multi-University Round 11

写在前面的废话

- 花了不到 2 周时间准备的多校。
- 8.3 号接到出题任务，8.8 号选定题目，实际造题时间也就一周多一点。
- ~~时间比较仓促，金牌区题目难度以及难度上限可能未达到区域赛标准~~。
- 出难了。不过前期题还是比较有趣的（
- 命题组认为的难度升序：MHEJDGALKBFCI。
- 封榜前过题人数排序：MHEJKCGLBDAFI。

写在前面的废话

- 中间的一大堆题似乎都没定好位，令人震撼。
- C 是最被高估的题，看来 ACM 选手的字符串水平还是非常厉害的！
- D 是最被低估的题，可能和中途进行了题面修正，榜被带歪了有关。
- 答疑环节很有趣，滴叉获得了一它一日体验卡。

致歉

- B 有个不算原题的原题。
- C 有一点点卡常。
- D 的一处题面写挂了，原因是大家觉得现在改正后的版本不太好理解，结果换成了个错误的东西，还好 jp998244353 发现了，不然就没钱了（
- E 的题面充满了问题，已经严刑拷打出题人了，~~他的钱全归我~~。
- G 的 disjoint 定义的不是很清晰。
- J 的 mod 看起来不在下标里。
- 部分答疑可能过于及时，并且语气太不严肃，向大家道歉。

M - Maimai DX 2077

题目大意

- 有个音游叫 Maimai DX 2077。
- 这个音游有四种音符，每种音符有五个判定。
- 不同音符的不同判定会获得不同基础分数。
- 绝赞的判定单独计算分数。
- 给定一次游玩每种音符每种判定的数量，问达成度。

题解

- 将判定的表作为 4×5 数组输入程序。
- 读入时直接计算 A, A_0, B, B_0 即可。
- 希望无论是否玩过 Maimai DX 2077 的人都能切了这题。

H - Here is an Easy Problem of Zero-chan

题目大意

- 有一颗 n 个节点且以 1 为根的有根树。
- 第 i 个点的点权为 i。
- 多次查询编号为 x 的点，$\prod_{i=1}^{n} lca(i,x)$ 的末尾有多少个零。

题解

- 首先对于数 a 末尾有 x 个零等价于他能分解成 $a' \times 10^x$ 后 x 的最大值。
- 因为对于 x 进制而言，乘以 x 相当于将数字左移一位，然后末尾补 0。
- 问题便转变成了求 $\prod_{i=1}^{n} lca(i,x)$ 能分解出多少个 10 作为因子，而 10 的因子为 2和 5，所以等价于求上式能分解出的 2 与 5 的最小值。
- 随后我们可以将点分为两类，对于处于 x 的子树内的点，与 x 的 lca 均为 x，而不处于 x 的子树内的点，那么他们的 lca 均为 x 的祖先。

题解

- 所以对于查询 x 而言，我们可以枚举 x 到根节点的所有节点，对于每个点计算有多少个点与 x 不在同一个儿子子树内。
- 对于上述解法的时间复杂度上界为 n^2，此时我们可以发现，对于点 x，他的父亲节点 fa_x 对于祖先路径上的权值计算与 x 相同，所以我们可以采用树形 dp 来减少重复的计算。
- 则可以得到转移式 $dp_x = dp_{fa_x} + size_x \times (cnt_{x2|5} - cnt_{fa_x2|5})$，其中 $size_x$ 代表 x 的子树大小，$cnt_{x2|5}$ 代表节点 x 能分解出多少个 2 或 5。

E - Everyone is bot

题目大意

- 有 n 个人打算在群里复读。
- 一次复读的过程如下：
- 每一轮，n 个人按照编号从小到大依次执行以下操作。
- 如果这个人在前几轮已经进行过复读，他不会再次复读。也就是说，每个人最多只会复读一次。
- 否则他可以选择是否进行复读。
- 如果某一轮没有人进行复读，那么复读的过程结束。

E - Everyone is bot

题目大意

- 对于第 i 个人，如果他是所有人中第 j 个进行复读的，他会获得 $a_{i,j}$ 瓶冰红茶。
- 但是如果他是所有进行了复读的人当中倒数第 p 个进行复读的人，那么他不会获得任何冰红茶，并且需要交给咩噗特雷格博 bot 154 瓶冰红茶（即获得 -154 杯冰红茶）。
- 每个人都想最大化自己获得的冰红茶数量，求所有人一共会拿到多少冰红茶。

题解

- 总复读人数是 $n \bmod p$。
- 如果当前已经有 $n-p$ 个人复读了，那么后面不会有任何人复读。
- 一旦有人复读，剩下的人必然都会参与复读，那么这个人就会被禁言。所以他不会这么做。
- 同样，如果有 $n-2p$ 个人复读了，那么后面不会有任何人复读，因为一旦他复读了，接下来一定有 $p-1$ 个人加入复读，而这时候是 $n-p$ 个人复读的状态，剩下 p 个人一定不会复读，那么他就被禁言了。
- 同样可以推出，如果当前复读人数是 $n-kp$ 那么后面不会有人复读。

E - Everyone is bot

题解

- 前 $n \bmod p$ 个人必然一上来就复读。可以考虑如果有人不复读，后面本来没有机会的人必然会抓住机会，那么前面有人就会失去机会。

J - Jellyfish and its dream

题目大意

- 给一个序列，值域为 $0 \sim 2$。
- 如果 $(a_i + 1) \bmod 3 = a_{(i+1) \bmod n}$，就可以将 a_i 赋值为 $(a_i + 1) \bmod 3$。
- 问有限次操作后是否能使所有元素均相等。
- $1 \le \sum n \le 10^6$

题解

- 差分，后文中的变化全部指差分数组。
- 一次操作可以将相邻的 $(2,1)$ 变成 $(0,0)$，$(1,1)$ 变成 $(2,0)$，$(0,1)$ 变成 $(1,0)$。
- 不难发现 0 没有意义。可以通过 $(0,1)$ 变成 $(1,0)$这个操作来移动 1的位置。
- 只要 1 的数量不少于 2 的数量即可。
- $O(n)$

K - Killer Sajin's Matrix

题目大意

- 构造一个大小为 $n \cdot m$ 的二维网格，使其其中有 k 个 1，其余均为 0。
- 并且该网格的每一行和每一列的和均为奇数。

题解

- 对于每一个 1 来说，他能够给一行以及一列带来 1 的贡献，在一个点可以被重复放多次的情况下，每一个 1 对于行和列的贡献是分离的，所以我们考虑将行和列分开来分配需要给定 1 的数量。
- 按照题意，我们要抱着每行每列都最少拥有一个 1 ，在此基础上将剩余的 1 均匀分配给每一行或者每一列，因为每一列都是等价。所以分配便会出现以下情况 $2x+1,...,2x+1,2x-1,...,2x-1$。
- 现在考虑将行的分配和列的分配联合起来，即变成了对于一行分配的 a_x 个 1 ，通过反证法可以得需要选择 a_x 列与该行配对，且该 a_x 列应为未分配数量最多的列，应而推导出对于列的分配应该也从数量多到数量低进行枚举。

题解

- 同时对于上述 $2x+1,...,2x+1,2x-1,...,2x-1$ 的分配方案可证，分配数量 $2x+1,2x-1$ 一定优于 $2x+3,2x-3$，因为 $2x+1,2x-1$ 的分配方案中包含了 $2x+3,2x-3$ 的所有分配方案。
- 且对于特例当且仅当 $k=n\times m-2$ 并且 n，m 皆为奇数时无解。因为对于一个全 1 的矩阵来说，删除一个 1 之后，剩余一个无法使得该行和该列同时减少一个 1。
- 所以该方案无解，在此基础下其他能够分配的方案一定有解，所以可证该分配方案合理。

C - Cmostp

题目大意

- Cmostp (chain merging on suffix tree problem)
- 给定一母串 S，每次选择一个区间 $[l,r]$。
- 查询母串 S 中有多少本质不同的子串满足至少有一次在 $[l,r]$ 区间中结尾。
- 数据范围 $|S| \leq 5 \times 10^5$，$q \leq 5 \times 10^5$。

题解

- 平凡题，偏套路了。
- 建出 SAM，我们可以考虑在后缀树上一条从根节点到表示字符串前缀 $s[1,i]$ 的节点的链路径，因为后缀自动机自动帮你去重，你需要数的就是编号为 $l\sim r$ 的链并节点和。
- 下文中 $s[1,i]$ 都是指代后缀树上表示 $s[1,i]$ 的节点。
- 题目的全称为 chain merging on suffix tree problem，但事实上链并问题个人认为是个比较简单的问题，这里给出三种做法：

第一种做法

- $O((n+m)\sqrt{n})$做法。
- 记 v_i 表示根节点到 $s[1,i]$ 节点的路径权值和，我们可以将链并换成表示前缀 $s[1,l] \sim s[1,r]$ 共 $r-l+1$ 个节点按照 dfs 序排序后的 $\sum_{i=l}^{r} v_{p_i} - \sum_{i=l}^{r-1} v_{lca(p_i,p_{i+1})}$。
- 其中 p 是一个对 $[l,r]$ 节点按照 dfs 排序后的重标号序列。

第一种做法

- 显然原问题变成了个区间有序序列相关，且贡献只和一个位置的前后项有关的问题。
- 可以通过不添加莫队维护链表做到时间复杂度 $O((n+m)\sqrt{n})$，具体地设右指针为单调指针，我们每次进行 l 落在第 i 块的询问时，对落在第 i 块以及以后块中的节点拉出来排好序。
- 用链表先存储下来，对于右端点暴力删，对于左端点开栈删，回撤时用栈还原信息即可。
- 不过遗憾的是，这道题将这个做法卡掉了，因为在不久之前的 WC2022 T2 中就有一道相似的题，并且存在时间复杂度更优秀的做法，我们还是卡掉了这个做法。

第二种做法

- $O(n\log^2 n)$ 做法之 LCT，考虑扫描线，我们维护树上每个节点在当前加入第 i 条链时最后的可以属于哪条链。开一个维护值的数组 w，假设一个节点最后属于第 p 条链就加到 w_p 上面，最后的答案即 $\sum_{i=l}^{r} w_i$。
- 不难发现链的性质可以很好的 access，上动态树。
- 初始时所有点都是相互独立的，即有树大小条链没条链记录的标号都是 0。
- 加入第 i 条链时把对应的一条链上的所有节点的标号都置为 i，同时回撤更改的其它实链上之前的贡献。

第二种做法

- 例如我们 access 的时候找到之前一条有过对其它位置贡献的实链 d，d 上我们取下来的一部分节点设其之前贡献给的是第 o 条链，则从 w_o 减去这些节点的贡献。
- 最后对整条拉出来的实链全部节点执行一次对 w_i 的贡献。
- 由于动态树本身优秀的性质，我们知道整个问题中遍历到的实链条数，即对于 w 的修改次数数量级为 $O(n\log n)$，用树状数组
- 维护一下单点加区间查即可做到 $O(n\log^2 n)$。
- 一个更优秀的理论上界是通过调整分治结构的叉数 $O(\frac{n\log^2 n}{\log\log n})$，但是实现不精细常数更大没有任何必要。
- 同时存在用树剖替代 LCT access 的一种维护方法也是同时间复杂度的。

第三种做法

- $O(n\log^2 n)$ 之颜色段均摊，考虑还是一样的维护方法。
- 通过树链剖分，我们等价于对 $\log n$ 个区间执行颜色段覆盖，由于颜色段的数量肯定是 $n+q\log n$ 级别的，所以直接用一个 set 暴力维护一下颜色段的插入/删除即可。
- 这个做法的常数极大，不知是否能通过，我本质上是没卡的。
- 同时存在一种区间虚树的做法，那个东西如果有谁会的话可以赛后分享一下，我实在是不想再讲了。
- 顺带一提，用 SA 建立笛卡尔树当后缀树用的做法是等价的，std 用时是 1s，为了防止常数过大的选手被卡掉我开了 2s。

G - Good red-string

题目大意

- 给定一个长度为 $3n$ 的字符串，字符串中仅包含'r','e','d','?' 四种字符。
- 你可以将‘?’变换成其他 3 个字符中的任意一种。
- 你需要将 $3n$ 个字符切分成 n 个不相交的子序列，且每个子序列的长度均为 3，并且每个子序列均为“red”。
- 不相交指不存在任意一个字符属于两个子序列。

题解

- 由于我们需要将整个字符串切分成 n 份，且每个子序列均为"red" 所以我们要保证对于每一个前缀，均有 $cnt_d \le cnt_e \le cnt_r$，且对于每一个后缀也有 $cnt_r \le cnt_e \le cnt_d$。
- 将条件拆分来看，前缀有 $cnt_e \le cnt_r$，$cnt_d \le cnt_e$，而后缀有 $cnt_r \le cnt_e$ ，$cnt_e \le cnt_d$。
- 对于前缀，我们先维护 $cnt_d \le cnt_e$ 。
- 对于后缀，我们维护 $cnt_r \le cnt_e$。

题解

- 由于整个数组有 $cnt_r = cnt_e = cnt_d$。
- 对于维护后缀的 $cnt_r \le cnt_e$ 等价于前缀满足 $cnt_e \le cnt_r$。
- 对于维护前缀的 $cnt_d \le cnt_e$ 等价于后缀满足 $cnt_e \le cnt_d$。
- 所以仅维护前缀满足 $cnt_d \le cnt_e$，后缀满足 $cnt_r \le cnt_e$，等价于维护所有条件。
- 对于维护前缀我们仅需记录所有'?' 的位置，如果出现'd' 的数目多余'e' 的数目，那么我们需要将前缀中靠后的'?' 转变成'e', 维护后缀同上。
- 当上述维护完成之后，字符串中可能存在'?'，我们从前往后将剩余的'r','e','d' 进行填充。
- 最后判断该字符串是否合法即可。

L - Lndjy and the mex

题目大意

- 给定多重集 S，满足元素是 $[0, n]$ 内的整数，且 $|S| = n$。
- 一个序列的权值定义为所有区间的 mex 之和。
- 计算所有长度为 n 且元素与 S 完全相同的序列的权值之和。
- 取模 998244353。
- $1 \leq n \leq 10^5$。

L - Lndjy and the mex

题解

- 对于集合 S，考虑

$$\begin{aligned}\operatorname{mex}(S) &= \max\{k \mid \forall j \in \mathbb{Z} \cap [0, k), j \in S\} \\ &= \sum_{k \geq 0} [\forall j \in \mathbb{Z} \cap [0, k], j \in S]\end{aligned}$$

题解

- 同时注意到，若枚举数列 $b_0, b_1, \ldots, b_n\ (b_i \le a_i)$ 为区间内可重集的构成，其贡献应为

$$\binom{\sum_{i=0}^{n} b_i}{b_0, \ldots, b_n}\binom{n-\sum_{i=0}^{n} b_i}{a_0-b_0, \ldots, a_n-b_n}\left(n-\sum_{i=0}^{n} b_i+1\right)$$

- 即分别枚举区间内与区间外的方案数，再将区间插入。

L - Lndjy and the mex

题解

- 结合开头提到的和式，可知答案为

$$\sum_{k\geq 0}\sum_{\forall i,[i\leq k]\leq b_i\leq a_i}\binom{\sum_{i=0}^{n} b_i}{b_0,\ldots,b_n}\binom{n-\sum_{i=0}^{n} b_i}{a_0-b_0,\ldots,a_n-b_n}\left(n-\sum_{i=0}^{n} b_i+1\right)$$

- 其中艾佛森括号 $[P]$ 在 P 成立时值为 1，否则为 0。
- 考虑枚举 $\sum_{i=0}^{n} b_i$ 统计答案。

题解

- 构造 $f_i(x)=\sum_{j=0}^{a_i}\frac{x^j}{j!(a_i-j)!}$ 可知原式等于

$$\sum_{s\geq 0} s!(n-s+1)![x^s]\sum_{k\geq 0}\left(f_0(x)-\frac{1}{a_0!}\right)\cdots\left(f_k(x)-\frac{1}{a_k!}\right)f_{k+1}(x)\cdots f_n(x)$$

- 设 $g_i(x)=f_i(x)-\frac{1}{a_i!}$。
- 考虑分治 $[l,r]$，维护 $\sum_{k=l}^{r} g_l\cdots g_k f_{k+1}\cdots f_r$ 和 f,g 的区间积，用 NTT 合并。可以直接取 l,r 中点，也可以按照多项式的度数取带权中点，复杂度均为 $O(n\log^2 n)$。

B - Bustling City

题目大意

- 给定一个内向基环树森林，对于每个 i 求出最小的 ans_i 满足至少 k 个点满足从这点出发走 ans_i 条边恰好到达 i。

题解

- 对于树上的点和环上的点分别计算答案。
- 树上问题即对于每个点求出以它为跟的子树里最小满足条件的深度。
- 处理深度的问题可以使用长剖。
- 记录每个点的长儿子和这个点子树内深度为 d 的点的数量。
- 计算长儿子的答案，其余儿子的答案直接按照深度暴力合并到父亲上，同时更新答案。
- 这一部分的时间复杂度是 $O(\sum len) = O(n)$。

题解

- 环上问题处理方法比较多。
- 一种方法是记录当前连通块中每个不在环上的点什么时候走在环上。
- 但是走到环上的时候这个点上原来的人可能已经走掉了，换成了新的人。
- 显然和某个人 i 在同一个点上的人数不会减少。
- 因此不妨计算出和对于每个刚开始在环上的人 i 在同一点的人数什么时候满足条件。
- 这是容易计算的，只要按照时间依次合并到环的对应位置即可。
- 最后要满足 u 和 a_u 的答案差不超过 1，扫一遍调整即可。

D - Directions

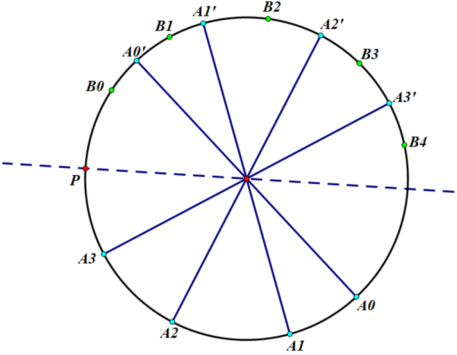
题目大意

- 赤道上有逆时针排列的 n 座岛屿。
- 给定其中一些岛屿的方位关系（东，西）。
- 问满足这些条件的基础上，n 座岛屿之间的关系矩阵有多少种。
- 答案对 $998,244,353$ 取模。
- $1 \leq \sum n \leq 500$

题解

- 我们考虑枚举所有点和 1 号点的方位关系来将点分为两类。
- 将 1 西部的点记为 B 类点，1 东部的点记为 A 类点：
- 将 A 类点全部关于圆心中心对称，如图；

图



题解

- 我们发现，第 x 个 A 类点在第 y 个 B 类点的西/东可以等价于第 x 个 A' 类点在第 y 个 B 类点的前面/后面。
- 直接单次 $O(n^2)$ 做一次 dp 即可通过此题。
- $O(n^3)$，小常数。
- Bonus：在 $O(n^2\text{polylog}(n))$ 的时间复杂度内解决问题。

A - Alternating 2.0

题目大意

- 对于一个 01 串 x，你可以将其任意一个非空子串反转（即 $0 \to 1, 1 \to 0$），称为一次操作。
- 定义 $f(x)$ 为使得 x 满足 01 交替的最少操作次数。
- 现给定一长为 n 的 01 串 s，你需要回答 q 个询问。
- 对于每个询问，求出 $\sum_{k \in \text{ non-empty subsequences of } s_{[l:r]}} f(k)$。
- $1 \le n, q \le 5 \cdot 10^6$

题解

- 首先考虑如何计算 $f(x)$。
- 令 $w=\sum\limits_{i=1}^{k-1}[x_i=x_{i+1}]$，则我们的目标就是将 w 减少至 0。
- 而显然，每次操作中，我们至多可以改变两对相邻字符的相等/不等关系，即至多将 w 减小 2。
- 故字符串 $f(x)=\left\lceil\frac{w}{2}\right\rceil$。
- 发现这个上取整运算较难直接处理，故考虑将其化为 $\frac{1}{2}(w+[2\nmid w])$，这样我们仅需求出 $s_{[l:r]}$ 每个子序列的 w 之和及 w 为奇数的子序列个数即可。

题解

- 考虑如何求出每个子序列的 w 之和。
- 对于一对满足 $l \le i < j \le r$ 且 $s_i = s_j$ 的字符，显然只有当它们相邻且同时出现在子序列中才会对答案有 1 的贡献，而情况数总共有 $2^{i-l} \cdot 2^{r-j} = 2^{r-l+i-j}$ 种，故总贡献为

$$\sum_{l \le i < j \le r, s_i = s_j} 2^{r-l+i-j} = 2^{r-l} \sum_{l \le i < j \le r, s_i = s_j} 2^{i-j}$$

$$= 2^{r-l} \left(\sum_{1 \le i < j \le r, s_i = s_j} 2^{i-j} - \sum_{1 \le i < l, l \le j \le r, s_i = s_j} 2^{i-j} - \sum_{1 \le i < j < l, s_i = s_j} 2^{i-j} \right)$$

- 将 $s_i = s_j = 0$ 及 $s_i = s_j = 1$ 的情况分别预处理即可 $O(1)$ 计算。

题解

- 考虑如何求出 w 为奇数的子序列个数。
- 容易证明一个子序列的 w 的奇偶性等于子序列长度与 $[s_{st}=s_{ed}]$ 之和的奇偶性，其中 st,ed 分别为子序列在 s 中的首末位置。
- 对于所有 $ed-st\le 1$ 的子序列，特判即可；否则若 st,ed 固定，那么长度为奇数的子序列个数与长度为偶数的子序列个数相等，均为 $2^{ed-st-2}$。
- 故总贡献为

$$\sum_{l\le i<r}[s_i=s_{i+1}]+\sum_{l\le i<j-1<r}2^{j-i-2}=\sum_{l\le i<r}[s_i=s_{i+1}]+\sum_{2\le i\le r-l}(r-l-i+1)\cdot 2^{i-2}。$$

- 预处理 $[s_i=s_{i+1}]$，2^i 及 $i\cdot 2^i$ 的前缀和即可 $O(1)$ 计算。
- 总时间复杂度为 $O(n+q)$。

题解

- 当然本题还可以通过线段树维护矩阵支持修改操作，时间复杂度为 $O(V^3(n+q)\log n)$，其中 V 为矩阵大小。

F - Flame blast magician master qcjj

题目大意

- 在一个二维平面上有 N 只怪物。
- 可以使用技能攻击怪物 x 坐标在某个区间 $[l, r]$ 或者 y 坐标在某个区间 $[l, r]$ 的全部怪物。
- 若击杀则计算得分。
- 过程中可能产生新的怪物
- 每次使用技能攻击的操作强制在线。

F - Flame blast magician master qcjj

题解

- 先想一个本题的退化版本，假设 $Q=1$，从二维矩形区间退化成序列上维护若干个怪物的血量。
- 显然这个问题是可以使用势能线段树的方式解决，只需要维护线段树上区间 $[l,r]$ 中怪物血量的最小值。

F - Flame blast magician master qcjj

题解

- 当某次操作后发现线段树上区间的最小值 $=0$，就暴力到叶子结点将该怪物删除，同时加上相应的 $score$，叶子节点可以套一个 std::priority_queue 维护一下当前叶子中所有怪物血量的最小值。
- 因为怪物至多 $n+m$ 个，所以暴力的复杂度就是 $O(\log(n+m))$，同时优先队列是套在叶子结点上的，所以优先队列也不会额外增加时间复杂度，总复杂度仍然是 $\log$ 级别的。

F - Flame blast magician master qcjj

题解

- 如果是二维版本是不是需要一个二维数据结构维护比如写个树套树什么的呢，其实并不用。
- 这里用到一个 trick，想这么一个问题，假设怪物的血量是 H，设在 x 轴累计的伤害为 dx，在 y 轴累计的伤害为 dy。
- 若存在不等式 $\begin{cases} dx < \frac{H}{2} \\ dy < \frac{H}{2} \end{cases}$，根据不等式加法，则有 $dx + dy < H$。
- 意思就是说，如果我们分开维护怪物的半血血量的话，只要 x 轴或者 y 轴中累计的伤害没达到怪物的半血线，就不可能杀死怪物。

F - Flame blast magician master qcjj

题解

- 所以只要分开维护 x, y 轴的伤害，直接退化成一维版本。当 x 轴或者 y 轴其中之一到达某只怪物的半血血线时暴力检查是否杀死该怪物，否则更新新的半血血线。
- 每次暴力时，怪物在 x, y 轴上累计的伤害总量一定大于 $\frac{H}{2}$，所以至多暴力 $\log H$ 次。
- 总时间复杂度为 $O((P+Q)\log(n+m)\log H)$。

I - Innocent Longing

题目大意

- 给定 n, k，求出至少存在一个长度在 $[k+1, n]$ 内的连续段的 n 阶排列个数。
- 取模 998244353。
- $1 \le k < n \le 10^5$。

I - Innocent Longing

题目大意

- 给定 n, k，**求出所有连续段长度都在** $[1, k] \cup \{n\}$ **内的连续段的** n **阶排列个数。**
- **取模** 998244353。
- $1 \le k < n \le 10^5$。

I - Innocent Longing

题目大意

- 给定 n, k，求出所有连续段长度都在 $[1, k] \cup \{n\}$ 内的连续段的 n 阶排列个数。
- 取模 998244353。
- $1 \le k < n \le 10^5$。
- 取补集即可。下文将以此题意为基础展开讨论。

连续段结构：析合树

- 对于排列 π 和一个连续段 $[l, r]$，若不存在与之相交但不包含的连续段，则称其为本原连续段。
- 显见，本原连续段之间构成一棵树。这就是**析合树**。
  且原排列中一个连续段必然由若干连续且互不相交的本原连续段组合而成。
- 对于析合树上一个结点，若其儿子的连续段值域上呈升序或降序，则称其为**合点**。
  否则，称其为**析点**。

连续段结构：析合树

- 对于一个合点，显然，其儿子的任意连续子序列都是一个连续段。
- 对于一个析点，有结论，其儿子的连续子序列是连续段，当且仅当其恰好包含 1 个或所有儿子。
- 否则，可以取出新的本原连续段。

连续段结构：析合树

- 对于一个合点，显然，其儿子的任意连续子序列都是一个连续段。
- 对于一个析点，有结论，其儿子的连续子序列是连续段，当且仅当其恰好包含 1 个或所有儿子。
- 否则，可以取出新的本原连续段。
- 于是，析合树反映了排列所有连续段的信息。

子任务：单排列

- 即 $k=1$ 的情况。
- 这是不那么容易计数的。但是我们观察到这类似于排列一个析点的所有儿子的方案数，所以我们不妨先来考虑所有排列的析合树结构。
- 先考虑根结点为合点。不妨仅讨论升序的情况，降序的情况显然是对称的。

  设 $P(x)=\sum_{i\geq 1} i!x^i$，$G(x)$ 为这样的合点的一个儿子的 OGF。
  显然，G 中的排列都是无法划分为升序的本原段的。因此就有

$$P=\sum_{i\geq 1} G^i=\frac{G}{1-G} \iff G=\frac{P}{1+P}$$

子任务：单排列

- 而所有根结点为合点的排列，即

$$2(P-G)=\frac{2P^2}{1+P}$$

- 再考虑析点，设单排列的 OGF 为 $F(x)$。
  注意其包含 $[1],[1,2],[2,1]$，因此用以排列儿子顺序时应当为 $F-x-2x^2$。
  而析点的儿子显然可以是任意排列。因此根结点为析点的排列数即

$$(F-x-2x^2)\circ P=F(P)-P-2P^2$$

子任务：单排列

- 立即列出方程

$$F(P)-P-2P^2=P-\frac{2P^2}{1+P}-x$$

- 两侧代入复合逆 $P^{\langle -1\rangle}$，有

$$F(x)-x-2x^2=x-\frac{2x^2}{1+x}-P^{\langle -1\rangle}$$

- 至于怎么计算，目前还不重要。

完整思路与计算

- 对于 $k>1$，考虑根结点是析点还是合点。
- 若是合点，则意味着所有儿子的子串的长度都不得超过 k。
  显然，最长的两个子串是最大的真前缀与真后缀。故条件即第一个与最后一个儿子的大小都不小于 $n-k$。即
  $$[x^n]2(P+1)\left(\sum_{i\ge n-k} g_i x^i\right)^2$$
- 若是析点，则意味着每个儿子的大小都不超过 k。
  设 $H(x)=\sum_{i=1}^{k} i!x^i$，我们希望计算复合 $(F-x-x^2)\circ H$ 的第 n 项系数。
- 那么接下来有两条路可走。

I. 拉格朗日反演

- 施拉格朗日反演，可得

$$[x^n](F-x-x^2)\circ H=\frac{1}{n}[x^{n-1}](F-x-x^2)'\left(\frac{x}{H^{\langle -1\rangle}}\right)^n$$

- 或另类拉格朗日反演，可得

$$[x^n](F-x-x^2)\circ H=[x^n](F-x-x^2)(H^{\langle -1\rangle})'\left(\frac{x}{H^{\langle -1\rangle}}\right)^{n+1}$$

- 而根据前文，

$$F(x)-x-2x^2=x-\frac{2x^2}{1+x}-P^{\langle -1\rangle}$$

I. 拉格朗日反演

- 那么，我们现在要分别计算 $P^{\langle -1\rangle}$ 与 $(\sum_{i=1}^{k} i!x^i)^{\langle -1\rangle}$。
- 令 $A = P^{\langle -1\rangle}$。由 $n! = (n-1)(n-1)! + (n-1)!$ 可得 ODE

$$P = x(1 + P + xP')$$

- 代入 $x = A$，得

$$\begin{aligned} x &= A(1 + x + AP'(A)) \\ &= A\left(1 + x + \frac{A}{A'}\right) \\ xA' &= (1+x)AA' + A^2 \end{aligned}$$

I. 拉格朗日反演

- 令 $a_n = [x^n]A(x)$，下文会用到类似的记号。
- 两边提取 $[x^n]$ 得在线卷积式

$$a_n = -\sum_{k=1}^{n-1}(k+1)(a_k + a_{k+1})a_{n-k} \qquad (n \geq 2)$$

- 可以 CDQ 分治 NTT。但在分治区间形如 $[0, r]$ 的时候可能涉及没算出的位置。
  一个做法是套一层倍增。也等价于特殊处理此类区间，将将来可能算不出的贡献提前计算。

I. 拉格朗日反演

- 类似地，令 $Q = H^{\langle -1 \rangle}$。将上面的 ODE 略作改动可得

$$H = x(1 + H + xH') - (k+1)!x^{k+1}$$

$$x = Q\left(1 + x + \frac{Q}{Q'}\right) - (k+1)!Q^{k+1}$$

$$xQ' = (1+x)QQ' + Q^2 - (k+1)!Q'Q^{k+1}$$

- 令 $R = Q^{k+1}$，众所周知有 $R'Q = (k+1)RQ'$，从而可以并行维护。

II. 直接计算复合

- 但我们已经得到了 A 的 ODE，根据经验，我们应该能直接得到 $A(H)$ 的 ODE。
- 设 $B = A(H)$，回顾前文，有

$$
\begin{aligned}
xA' &= (1+x)AA' + A^2 \\
\frac{H}{H'}B &= \frac{1+H}{H'}BB' + B^2
\end{aligned}
$$

- 将其展开为在线卷积会遇到诸多细节。出题人采用的方法是并行维护 $\left(\frac{1+H}{H'} - 1\right)B$，其中减一是为了保持计算顺序。需提前计算出 $\frac{1}{H'}$ 和 $\frac{H}{H'}$。

尾声

- 标程采用的是计算复合的方法，而验题人实现了计算复合逆的方法。
- 同时，标程采用的是 $\Theta\left(\frac{n\log^2 n}{\log\log n}\right)$ 的多叉 CDQ 分治 NTT。
  不过这样实现在线卷积是很麻烦的。所以标程也运用倍增将问题归约到了半在线卷积。
  非常快，在牛客上只跑了不到 600ms。
- 另外，注意到我们只求一项。能不能拓展到前 n 项呢？
  析点的情况只需使用复合的方法即可，但合点的情况难以对所有 n 算出。
  如果有大师会的话请务必教育出题人。

THANKS!

AC.NOWCODER.COM