The 3rd Universal Cup. Stage 39: Tokyo 题解

2025-06-18 02:26:55 **By** jiangly

A. Array Similarity

问题也就是要询问两个子数组的（非严格）前缀 $\max$ 位置是否相同。令 f_i 为 i 右边第一个 $\geq a_i$ 的数的位置，则只需要比较序列 $f_L - L, f_{f_L} - f_L, \ldots$（一直到最后一个 $\leq R$ 的位置）是否相同。

使用倍增 + 哈希即可，时间复杂度 $O((N+Q)\log N)$。

B. Bracket Character Frequency

考虑一个括号序列中左括号的位置，括号序列是合法的当且仅当左括号个数为 K，以及对每个 i（$1 \leq i \leq K$），前 $2i-1$ 个字符中至少有 i 个左括号。

因此要把这些左括号分给 N 个合法括号序列，只需要满足：

- $\sum_{i=1}^{2K} A_i = NK$。
- 对每个 i（$1 \leq i \leq K$），$A_1 + \cdots + A_{2i-1} \geq Ni$。

时间复杂度 $O(K)$。

C. Card Deck

给定一个集合，有一个简单的贪心算法可以判断该集合是否可以得到：

- 重复 M 次，每次取出牌堆顶 K 张牌，取走我们需要的牌，将剩余的牌放回牌堆顶。

设 $f(M,K)$ 为给定 M,K 的情况下，可以取到的不同集合数量，则有递推式：

$$f(0,K) = 1$$
$$f(M,K) = \sum_{i=0}^{K} \binom{K}{i} f(M-1,i) \quad (M \geq 1)$$

其中枚举的 i 就是牌堆顶的 K 张中属于集合的牌数，因为剩余的牌每次都需要放回，所以相当于接下来每次只能抽 i 张牌。

注意到这个式子的另一个组合意义是：$f(M,K)$ 表示长度为 K、每个元素是 0 到 M 的整数的数组个数。因此我们知道 $f(M,K) = (M+1)^K$。

但我们需要求的是每个集合大小的和，放在上面的转移中，集合大小就是每次转移选择的 i 之和。同样，我们发现另一个组合意义是长度为 K、每个元素是 0 到 M 的整数的数组的和的总和。因此答案为 $K \cdot \frac{M(M+1)}{2} \cdot (M+1)^{K-1}$。

时间复杂度 $O(\log K)$。

D. Digits of Prefix Product

有几种不同做法，但基本都依赖 $a_i = 999\ldots999$。

核心点在于令 $a_i = 10^d - 1$，其中 d 是 b_{i-1} 的位数，则 $b_i = b_{i-1} \cdot a_i = 10^d b_{i-1} - b_{i-1}$。如果 $b_{i-1} = \overline{x_1 x_2 \cdots x_d}$ 则

$$b_i = \overline{x_1 x_2 \cdots x_{d-1}(x_d - 1)(9-x_1)(9-x_2)\cdots(9-x_{d-1})(10-x_d)}$$

例如 $b_{i-1} = 18$，则 $b_i = 1800 - 18 = 1782$，$b_{i+1} = 17820000 - 1782 = 17818218$。末尾的两位会在两个数之间交替，因此只要初值选择合适，就能使得前后部分内部和交界处都不产生相邻相同的数对。

E. Edge Coloring Problem

每种颜色的边形成一个匹配，因此答案有下界 $\frac{N(N-3)}{2} / \lfloor N/2 \rfloor$，在 N 为偶数时为 $N-3$，N 为奇数时为 $N-2$，且都能取到。

当 N 为奇数时，因为 N 和 $N+1$ 答案一样，可以额外增加一个点 $N+1$，然后选择在原图中没有边的 (x,y)，添加边 (x,y) 和 $(z,N+1)$（$z \neq x, z \neq y$），归约到 N 是偶数的情况。

N 是偶数时，注意原图的补图是 2-正则图，即若干个环。先考虑所有环长都是偶数的情况，我们可以把所有点黑白染色，分成两个集合 L,R，满足 L,R 都是团，而 L 和 R 之间的边形成了 $(N/2-2)$-正则二分图。

对于正则二分图，我们知道一定能将所有的边分成若干个完美匹配，因为根据 Hall 定理，正则二分图一定存在完美匹配，只需不断求匹配即可。

对于团，又根据 $M = N/2$ 的奇偶性分成两种不同的情况：

- M 是奇数，则可以分成 M 组大小为 $\frac{M-1}{2}$ 的匹配，且每个匹配中剩下的点不同。构造方法是第 i 个匹配包含所有满足 $x + y \equiv 2i \pmod M$ 的边，而 i 是孤立点。
- M 是偶数时，可以分成 $M-1$ 组完美匹配。只需在 $M-1$ 的基础上，将点 M 和每次剩下的点连边即可。

当 M 是偶数时，原图中的边可以分成二分图中的 $(N/2-2)$ 组匹配和 $(N/2-1)$ 组（L 的完美匹配 + R 的完美匹配）。

当 M 是奇数时，取出二分图中的任意一组匹配，对于匹配中的每条边 (u,v)（$u \in L, v \in R$），L 中没有用到 u 的匹配、R 中没有用到 v 的匹配和 (u,v) 形成了原图的一组完美匹配，共 $N/2$ 组。再加上二分图中剩余的 $(N/2-3)$ 组完美匹配。

此时已经解决了补图所有环都是偶环的情况，考虑有奇环的情况。

假设有 $2k$ 个奇环，我们仍然将每个奇环尽量黑白染色，但是会存在两个相邻（在原图中没有边）的点被分在同一侧。我们将其中 k 个环的这两个点分给 L，另外 k 个分给 R，保证 L 和 R 的大小都还是 $M = N/2$。设 L 中的这些特殊点对为 $(u_1, u_2), \ldots, (u_{2k-1}, u_{2k})$，$R$ 中的为 $(v_1, v_2), \ldots, (v_{2k-1}, v_{2k})$。

我们先对原图进行一些修改，最后再还原。对每个 $i = 1, \ldots, k$，我们在原图中添加边 $(u_{2i-1}, u_{2i}), (v_{2i-1}, v_{2i})$，删除边 $(u_{2i-1}, v_{2i-1}), (u_{2i}, v_{2i})$。然后按照上述只有偶环的情况进行构造。

这里的区别在于，我们需要通过改变完全图分解成匹配时的编号，将 L 中的新添加的所有边分到同一个匹配，R 中的同理。如果 M 是奇数，还需要满足 L 的这个匹配中没有用到的点和 R 的这个匹配中没有用到的点之间是有边的。这样我们就能将所有新添加的边分到同一个完美匹配中。最后只需将这些新添加的边替换为被我们删除的边即可。

复杂度瓶颈在于求正则二分图的完美匹配，因为要求 $O(N)$ 次，所以时间复杂度 $O(N^{3.5})$（当然也存在更优的算法）。

F. Fourier Coefficients

因为 $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$，令 $g(x) = \sum_{k=0}^{N-1} A_k \frac{x^k+x^{-k}}{2}$，则 $f(x) = g(e^{ix})$。

假设我们知道了 $g(x)$ 的 $2N-1$ 个点值（因为指数是 $-N+1$ 到 $N-1$），则可以使用多项式插值求出 $g(x)$ 的系数。

我们希望对于一个复数求出 $g(z^0), g(z^1), \ldots, g(z^{N-1})$， 因为 $g(x) = g(x^{-1})$，我们事实上还知道 $g(z^{-1}), \ldots, g(z^{-N+1})$，因此总共是 $2N-1$ 个点值，且这些点值是一个等比数列，可以在 $O(N \log N)$ 时间内进行插值（见 https://noshi91.github.io/algorithm-encyclopedia/polynomial-interpolation-geometric#noredirect）。

设 $z = a + bi$，若 $e^{ix} = z^k$，则 $\cos x + i \sin x = z^k, \cos x - i \sin x = z^{-k}$，即 $\cos x = \frac{z^k+z^{-k}}{2}$，因此询问 $f(x)$ 也就是 $f\left(\arccos\left(\frac{z^k+z^{-k}}{2}\right)\right)$。在 $\bmod 998244353$ 意义下选择一个 z 使得 $z^{-N+1}, \ldots, z^{N-1}$ 互不相同即可。

时间复杂度 $O(N \log N)$。

G. Guarding Plan

原来在右上凸包上的点一定是关键点。我们可以在凸包的边上再加一些点，使得剩余的关键点尽可能少。

考虑可能被凸包的每条边覆盖的点，这些点形成了一个横坐标递增、纵坐标递减的序列。我们需要把这些点分成若干段，每一段要么本身是一个关键点，要么被一个新添加的关键点覆盖。

设 $\mathrm{dp}(i) = (a, b)$ 表示前 i 个点至少要分成 a 段，在分成 a 段的前提下至少要添加 b 个新点。

第一种转移就是 $\mathrm{dp}(i) = (a, b) \to \mathrm{dp}(i+1) = (a+1, b)$。

第二种转移是 $\mathrm{dp}(l-1) = (a, b) \to \mathrm{dp}(r) = (a+1, b+1)$，且要求 $[l, r]$ 内的点要能被凸包上的一个点覆盖，即点 (x_r, y_l) 在凸包内。对每个 r，可以转移的最小的 l 是单调的，因此可以用单调队列优化转移。

时间复杂度 $O(N \log N)$，瓶颈在于排序。

H. Hidden Sequence Rotation

考虑类似后缀排序，每次已知长度为 l 的最小子串，求出所有长度为 $2l$ 的最小子串。

直接做显然会导致总长度超过 N。我们有如下观察：如果左端点为 i 和 $i+d$ 的长度为 l 的子串都是字典序最小的，且 $d \le l$，但左端点为 $i+2d$ 的不是，则长度为 $2l$ 时，左端点为 $i+d$ 的子串字典序一定大于左端点为 i 的子串。

因此只有两种情况：

- 整个串有长度为 $d \le l$ 的循环节，此时所有长度为 l 的最小子串就是所有最小循环移位。
- 否则，如果两个长度为 l 的最小子串相交，靠后者一定不可能是最小循环移位。剩余的所有子串两两不相交，因此可以询问它们之后的长度 l 的子串，总长度不会超过 N。

I. Insert AB or BA

不妨设 $X \le Y$。

假设确定了 S 的每个字符对应的 T 中的位置，则只需要计算这些位置之间每一段的代价之和。

区间 $[l, r]$ 的代价计算方式如下：

- 令 s_i 表示 T 中前 i 个字符中 `A` 的个数减去 `B` 的个数。如果 `A` 的个数不等于 `B` 的个数，即 $s_r \ne s_{l-1}$，则代价为 $+\infty$。否则因为 $X \le Y$，我们希望尽量少插入 `BA`，而这个数量等于 $s_r - \min_{i=l-1}^{r} s_i$。

设 $\mathrm{dp}(i, j)$ 表示 S_i 匹配 T_j 的最小代价，则转移形如 $\mathrm{dp}(i, l-1) + \mathrm{cost}(l, r) \to \mathrm{dp}(i+1, r+1)$。

暴力做的复杂度是 $O(|S||T|^2)$。要优化每一轮的转移，首先把 s_i 相同的放在一起处理，注意到代价的形式，可以建立笛卡尔树后在笛卡尔树上 DFS 转移，或是用单调栈维护，每一轮转移是线性的，总时间复杂度 $O(|S||T|)$。

J. Journey through the Fractal

只需观察一下合法路径的形式即可。L 级三角形可以在两侧经过一个 1 级三角形到达 $L-1$ 级三角形，这个经过的 1 级三角形可以被替换为 $1, L-2, 1$ 的序列，以此类推。

举例来说，当 $L = 4$ 时，一个极长的路径形如 $1, 1, 1, 2, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1, 1$。

最优路径可以看作由如下的方法生成：

- 初始时 $L-1$。
- 添加至多两组 $(1, L-2)$。
- 添加至多四组 $(1, L-3)$。
- ......
- 添加 1 直到总长度到达 2^L-1。

时间复杂度 $O(\log L)$。

K. K-rep Array

数组可以是 K-rep 当且仅当不存在 i, j 满足 $A_i \neq -1, A_j \neq -1, A_i \neq A_j$ 且 $(i-j) \bmod K = 0$。

我们先计算有哪些 d 使得存在满足上述条件的 i, j 满足 $i-j=d$，然后扩展到每个 d 的约数。

上述过程可以看作求 $f_d = \sum_{i-j=d}(A_i+1)(A_j+1)(A_i-A_j)^2$，则存在满足条件的 i, j，$i-j=d$ 当且仅当 $f_d \neq 0$。

上式可以用 FFT 计算，时间复杂度 $O(N \log N)$。

L. LIS Triangle

如果 $K=1$ 一定无解，因为所有数互不相同，1 不可能跟任意两个数组成三角形。

剩余情况都有解，一种构造方法是构造一个单峰序列，先递增再递减，除最大值外，两侧的长度分别是 $L-1$ 和 $N-L$。将 K 和剩余的较大的数放在较小的一侧，剩余数放在较多的一侧即可。原理是让较大的一侧靠近最大值的数尽可能大一些。

M. Minimum Distance Tree

显然在有解的情况下，答案是原图唯一的最小生成树。只需要判断非树边的边长都不小于树上路径长度即可。

时间复杂度 $O(N \log N)$。

N. Nice Bouquets

一组花可以被分成若干组当且仅数量是 3 的倍数，且当将 `R`、`G`、`B` 分别看作 $0, 1, 2$ 后，总和是 3 的倍数。

因此对每一棵树，我们额外加上一列 1，表示花的数量，则满足条件当且仅当每一列的总和是 3 的倍数。

把每一行看作 $\mathbb{F}_3$ 上的向量，则操作一棵树可以看作将对应的行乘以 0 或 2。因此令所有行向量的总和是 $\mathbf{s}$，问题被转化为求最小的 r，使得 $\mathbf{s}$ 在 $\mathbf{v}_1, \ldots, \mathbf{v}_r$ 张成的空间内。

用线性基求解，时间复杂度 $O(NK \min\{N, K\})$。

O. One Different Inequality

如果 $S_i \neq S_{i+1}$，则对应的两对相邻的数，至少有一对的差不等于 1。例如对于所有的 <>，不可能形如 $x-1, x, x-1$。因此我们要选择尽量少的 S_i，使得上述的每一对 (S_i, S_{i+1}) 至少有一个被选择。而一个选择的方案一定存在一个合法的解，因为剩下的每一段 S_i 都是相等的，因此排列被分成了若干段连续上升或下降的数，这些段之间要满足被选择的 S_i 限制的大小关系，而这是一定有解的。

要计算方案数，只需要计算对于每一种选择 S_i 的方案，满足其代表的大小关系的排列的个数之和。例如，$S=$ <<><>><<>，选择了 S_2, S_4, S_6, S_9，即 <<>> ，我们需要计算有多少个大小为 5 的排列 P，满足 $P_1 < P_2, P_2 < P_3, P_3 > P_4, P_4 > P_5$。

固定了选择的子序列的情况下，计算这个方案数可以使用容斥，我们将 > 容斥成没有限制减去 <，容斥后的方案数等于

$$\frac{K!}{\prod L_i}$$

其中 K 是排列的长度，L_i 是受到 < 限制的每一段的长度，也是相邻两个被容斥成无限制的 > 之间的距离（如果认为开头和结尾各有一个）。因此我们设 dp_i 表示把 i 位置上的 > 容斥成无限制，前缀的贡献之和。转移是

$$\mathrm{dp}_i = [S_i = >] \sum_{j=0}^{i-1} \mathrm{dp}_j \cdot \frac{1}{(i-j)!} \cdot (-1)^{\sum_{k=j+1}^{i-1}[S_k =>]}$$

可以使用分治 FFT（半在线卷积）优化。

现在的问题是选择的 S_i 不确定，但是其长度是确定的，且可以分成若干段（每一段对应原串中交替、形如 <><><> 的一段），根据每一段长度的奇偶性，有两种情况：

- 长度是奇数，如 <><><：则方案唯一，都是 < 或都是 >。
- 长度是偶数，如 <><><>：则有 $\frac{\text{len}}{2}+1$ 种方案，每种方案是 i 个 > 后 $\frac{\text{len}}{2}-i$ 个 <（或相反）。

对于第二类情况，我们可以在必要的时候再决定其选择的方案。例如是 > 在 < 之前的情况，则可选择的方案只和我们最后一个选择的 > 有关，因此我们在做从这一段到之后的段的转移时乘上这部分的系数。同理，对于 < 在 > 之前的情况，我们在做从之前的段到这一段的转移时乘上这部分的系数。段内的转移可以看作全是 > 来处理。

时间复杂度 $O(N \log^2 N)$。

P. Perfect Suika Game on a Tree

考虑判断是否合法的一个贪心算法：

- 自底向上进行合并，每次将一个点 x 不断和它的孩子进行合并。

可以发现，在合并过程中，如果存在 x 孩子的值小于 x 的值，或是 x 有两棵子树有相同的值，则一定无解。否则 x 的子树合并完后一定是父亲的值小于儿子的值，我们只需要直到子树中出现的数的集合即可。

换句话说，令 S_x 为 x 子树内的 2^{A_y} 之和，则如果合法，x 合并后的子树中包含的数恰好就是 S_x 所有为 1 的二进制位。而合法的条件如下：

- 设 1 为根，则 S_1 是 2 的幂。
- 在 $T_x = \sum_{y\in \mathrm{ch}(x)} S_y$ 这个求和中没有进位，即 $\mathrm{popcount}(T_x) = \sum_{y\in \mathrm{ch}(x)} \mathrm{popcount}(S_y)$。
- $2^{A_x} \le \mathrm{lowbit}(T_x)$。

我们可以使用可持久化线段树合并来求出所有的 S_x 和 T_x（过程中需要处理进位）。

修改操作只会交换相邻的两个点，设它们是 x 和 p_x，则受到影响的数只有 $A_x, A_{p_x}, T_{p_x}, S_x$ 这些。只要能够快速进行给一个数加上或减去 2^k 的操作，就能够快速维护上述的条件。

时间复杂度 $O(N\log N)$。

Q. Quadratic Pieces

一个序列是平方的，当且仅当对每个 $1 \le i \le N-3$ 有 $A_{i+3} - 3A_{i+2} + 3A_{i+1} - A_i = 0$，这也就是序列的三阶差分。每次贪心地取尽量长的段即可。

时间复杂度 $O(N)$。

Universal Cup ucup 题解 [+19]

Comments

No comments yet.

Post a comment

You can refer to mike by using "@mike", and "mike" will be highlighted. If you want to type the character "@", please use "@@" instead.

You can enter "/kel" to use the emoticon "kel".

Content

Submit