AtCoder Beginner Contest 452 题目讲解

云浅

2026-4-6

Outline

A. Gothec 2
B. Draw Frame 4
C. Fishbones 6
D. No-Subsequence Substring 10
E. You WILL Like Sigma Problem 13
F. Interval Inversion Count 16
G. 221 Substring 19

A. Gothec

A. Gothec

略

B. Draw Frame

B. Draw Frame

略

C. Fishbones

给定 N 个正整数数 A_i, B_i 以及 M 个字符串 $S_1, \cdots, S_M$.

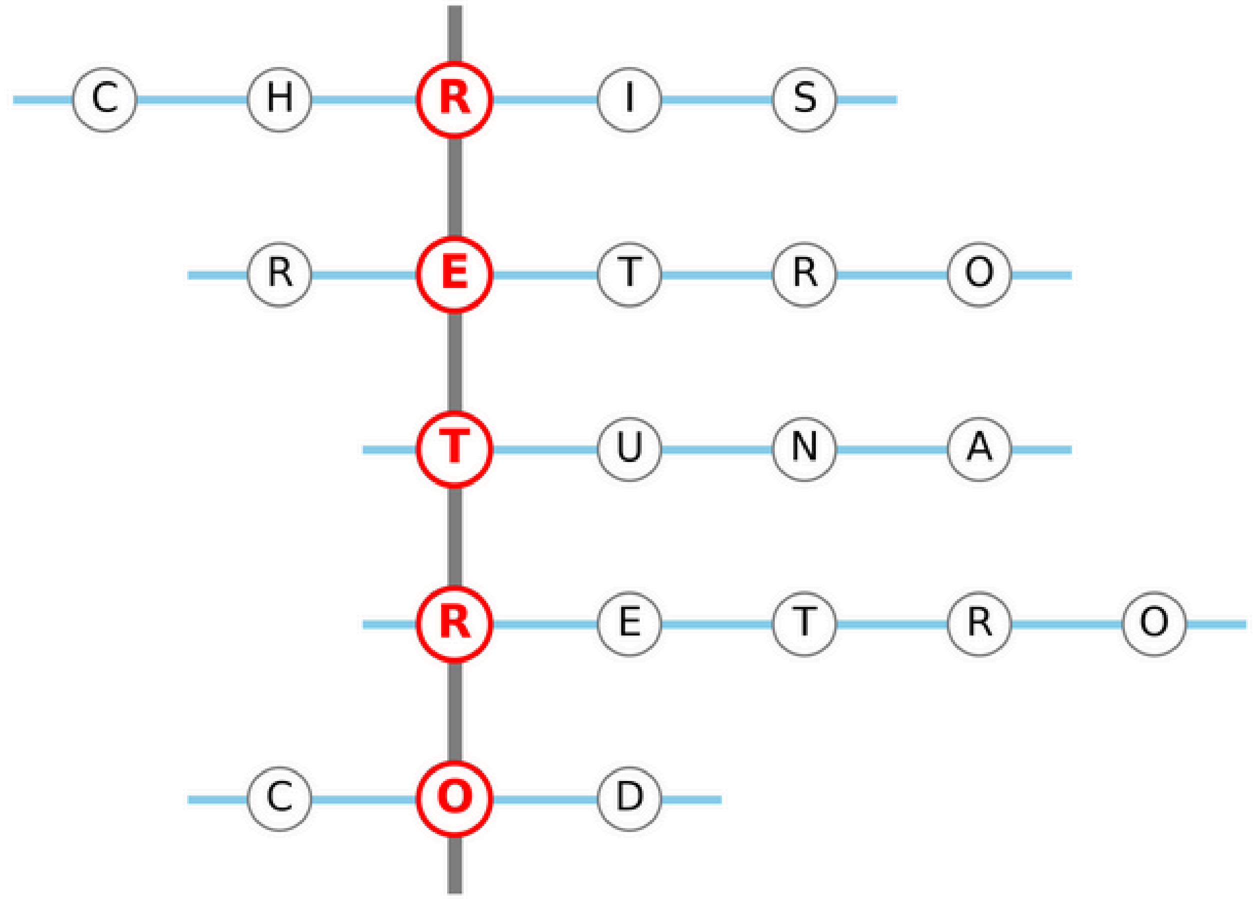
现在需要从这 M 个字符串中选出 $N+1$ 个字符串（可以重复选），假设选出的第一个字符串是 C，剩下的字符串是 $T_1, \cdots, T_N$，需要满足：

- $|C| = N$
- 对每个 $1 \le i \le N, |T_i| = A_i$，且 $T_i[B_i] = C_i$

对每个 $j = 1, 2, \cdots, M$，判断是否有一种选法使得 S_j 恰好作为第一个字符串 C.

数据范围：$1 \le N, A_i, |S_i| \le 10, 1 \le M \le 200000$.

C. Fishbones



C. Fishbones

一个字符串 S_j 合法，当且仅当 $|S_j| = N$，且对每个 $1 \le i \le N$，都存在某个字符串 S_k 满足 $|S_k| = A_i, S_k[B_i] = S_j[i]$.

考虑怎么判断后面那个条件，我们可以预处理一个数组 $\text{exist}[\text{len}][p][c]$ 表示：是否存在一个字符串 S_k，满足 $|S_k| = \text{len}, S_k[p] = c$.这个可以扫一遍字符串预处理出来.

最后判断一个 S_j 是否合法只要对每个 i 都看一下 $\text{exist}[A_i][B_i]\left[S_j[i]\right]$ 是否都是 true 就可以了。时间复杂度是 $O(N^2|\Sigma| + NM)$.

D. No-Subsequence Substring

D. No-Subsequence Substring

给定由小写英文字母组成的字符串 S 和 T。

在 S 的所有**非空子串** s 中，统计那些**不**将 T 作为（不一定连续的）子序列包含的子串数量。

注意：即使两个子串作为字符串相等，只要它们在 S 中的位置不同，就被视为不同的子串。

数据范围：

- $1 \leq |S| \leq 2 \times 10^5$
- $1 \leq |T| \leq 50$

考虑对每个 i 算出来第一个 $j \geq i$，使得 $S[i,j]$ 不是 T 的子序列。那么 j 后面的肯定也不是 T 的子序列了。

怎么判断一个字符串 A 是不是另一个字符串 B 的子序列？可以从前往后扫每个 A_i，贪心地找 B 中下一个和 A_i 相同的字符。形式化地，可以维护一个 p 表示 $B[1,p]$ 是最短的一段前缀满足 $A[1,i]$ 是 $B[1,p]$ 的子序列，每次找最小的 $p' \geq p$，使得 $A[i+1] = B[p']$。

可以发现这个过程支持我们一个一个加字符，那么我们枚举 i，然后不断尝试把 j 往后移动，这样一直找下一个字符即可。

由于 $|T| \leq 50$，至多找 50 个字符就会结束，时间复杂度是 $O(|S| \times |T|)$.

E. You WILL Like Sigma Problem

E. You WILL Like Sigma Problem

给定两个序列 A, B，求

$$\sum_{i=1}^{N}\sum_{j=1}^{M} A_i \times B_j \times (i \bmod j)$$

答案对 998244353 取模

数据范围：

- $1 \leq N, M \leq 5 \times 10^5$

E. You WILL Like Sigma Problem

看到 $i \bmod j$ 这样的形式，经验告诉我们可以写成 $i - \lfloor i/j \rfloor \times j$，然后在 $\lfloor i/j \rfloor$ 上面做一些手段，可以利用调和级数 or 整除分块。

那么这个题可以拆一下式子变成

$$\sum_{i=1}^{N}\sum_{j=1}^{M} A_i B_j (i - \lfloor i/j \rfloor \times j)$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} i \times A_i \times B_j - \sum_{i=1}^{N}\sum_{j=1}^{M} A_i \times \lfloor i/j \rfloor \times B_j \times j$$

前面那部分就是 $\left(\sum_{i=1}^{N} A_i \times i\right) \times \left(\sum_{j=1}^{M} B_j\right)$

后面那部分可以枚举 j，然后再枚举 $\lfloor i/j \rfloor$ 的值，那么满足 $\lfloor i/j \rfloor = k$ 的 i 是一段区间，即 $[j \times k, (j+1) \times k - 1]$，只需要算出来这段区间的 A_i 之和，再乘上 $B_j \times j$ 即可。

根据调和级数，总的复杂度是 $\sum \frac{N}{j} = O(N \log N)$ 的。

F. Interval Inversion Count

F. Interval Inversion Count

给定一个排列和正整数 K。

求有多少个区间 $[l, r]$ 满足 $(P_l, \cdots, P_r)$ 的逆序对数恰好为 K。

数据范围：

- $1 \leq N \leq 5 \times 10^5, 0 \leq K \leq \frac{N(N-1)}{2}$

F. Interval Inversion Count

考虑到固定左端点 l 的时候，随着 r 增大，逆序对数只会变大。

那么只需要算出：

- f_i：最小的 $j \geq i$ 使得 $[i,j]$ 的逆序对数 $\geq K$，如果不存在则为 $N+1$
- g_i：最大的 $i \leq j \leq N$ 使得 $[i,j]$ 的逆序对数 $\leq K$

答案就是 $\sum_{i=1}^{N}(g_i - f_i + 1)$。

算 f, g 可以用双指针，例如算 f，我们倒着扫一遍，从 f_{i+1} 尝试算出 f_i，每次尝试把右端点往左移动一步；同时维护一个值域上的树状数组存储当前区间内的这些数，就可以支持两端插入一个数，以及维护总的逆序对个数。

时间复杂度是 $O(N \log N)$。

G. 221 Substring

G. 221 Substring

对于一个由正整数组成的序列 $X = (X_1, ..., X_n)$，如果把它的每个连续段缩起来之后中，每个（长度，值）对中的长度和值相等，则称 X 为 **221 序列**。

例如，$(2, 2, 3, 3, 3, 1, 2, 2)$ 是一个 221 序列（2 个 2，3 个 3，1 个 1），但 $(1, 1)$ 和 $(4, 4, 1, 4, 4)$ 不是 221 序列。

给定一个长度为 N 的正整数序列 $A = (A_1, ..., A_N)$。求 A 中作为非空连续子串出现的不同 221 序列的数量。

注意：即使两个子串取自不同位置，只要它们作为序列相等，就计为一个。

数据范围：

- $1 \leq N \leq 500000$
- $1 \leq A_i \leq 9$

G. 221 Substring

先分析一下 221 substring 出现的条件，它可能会跨过 A 中的若干个连续段，那么要求这些连续段的长度和值都相等；对于两侧的连续段，可能只选一部分，那么这就要求这个连续段的长度比值更大。

那么我们可以做一些转化，对于一个 A 中的连续段 $(\mathrm{val}, \mathrm{cnt})$，我们有三种情况：

- $\mathrm{val} > \mathrm{cnt}$：这个连续段不可能和任何 221 substring 相交。
- $\mathrm{val} = \mathrm{cnt}$：这个连续段可以作为某个 221 substring 的一部分，并且要么不包含他，要么一定是完整包含这一段。
- $\mathrm{val} < \mathrm{cnt}$：这个连续段只能作为某个 221 substring 的两侧，并且如果包含了这个连续段内的数，那么包含的长度是确定的。

于是我们可以把这三种连续段分别改成 $(0), (\mathrm{val}), (\mathrm{val}, 0, \mathrm{val})$，然后问题就变成了：计算有多少个本质不同的子串，满足子串中不包含 0。

G. 221 Substring

这个问题怎么做呢。。。我觉得绕不开 SA 或者 SAM，下面讲一下使用 SA 的做法

考虑 SA 是怎么求本质不同子串个数的，相当于每次计算新增的那个后缀里面，哪些子串是之前没有出现过的，出现过的一定是一个前缀，也就是它和之前所有后缀的 LCP 的最大值。

那么不包含 0 的限制就相当于限制了新增的这些子串的长度也不能太长，于是合法的子串的右端点就是一段区间，直接把这段区间的长度加到答案上就好了。

时间复杂度就是求 SA 的复杂度，常见的做法是 $O(N \log N)$ 或者 $O(N|\Sigma|)$ 的。