牛客多校6

比赛时间	比赛名称	赛中过题	总计过题	题目总数	罚时	Dirt	校内排名
25.07.31	牛客多校6	4	10	13	552	2/6	12/19

赛时

C 00:32 +0

Ender_hz: 记 $f_{n, m}$ 为 n 元排列 p 满足 $f(p)=m$ 的个数$g_{n, k}=\sum\limits_{m=1}^{n}f_{n,m}m^k$打表注意到递推式 $f_{n, m}=(n-1)f_{n-1, m}+f_{n-1, m-1}$所以要求 $g_{n, 3}=\sum\limits_{m=1}^{n}f_{n, m}m^3$用递推式展开可得 $g_{n, 3}=(n-1)g_{n-1, 3}+\sum\limits_{m=1}^{n-1}f_{n-1, m}(m+1)^3=ng_{n-1,3}+3g_{n-1,2}+3g_{n-1,1}+3g_{n-1,0}$其他项也可以通过类似的式子得到，时间复杂度 $\mathcal O(n)$

题解的思路大致如下：在 n 给定的情况下，记随机变量 $X_i\in\{0,1\},1\le i\le n$ 为 i 是否出现在一 n 阶排列转化为单调栈后的序列中，则由古典概型和排列组合相关知识可知 $\mathbb E(X_i)=\mathbb P(X_i)=\dfrac{1}{i}$同理$\mathbb E(X_iX_j)=\dfrac{1}{ij}$$\mathbb E(X_iX_jX_k)=\dfrac{1}{ijk}$同时，所求结果

$$s_n=\sum\limits_{p\in\mathcal P_n} \left(\sum\limits_{i=1}^{n} X_i\right)^3=n! \mathbb E\left(\left(\sum\limits_{i=1}^{n}X_i\right)^3\right)$$

，其中

$$\left(\sum\limits_{i=1}^{n}X_i\right)^3=\sum\limits_{i=1}^{n} X_i^3 + 3\sum\limits_{1\le i, j\le n\\\,\,\,\, i\neq j} X_i^2X_j+3!\sum\limits_{1\le i<j<k\le n} X_iX_jX_k$$

，又因为 $X_i\in\{0,1\}$ 故 $X_i^k=X_i,k\in\mathbb N_+$那么就有

$$ \left(\sum\limits_{i=1}^{n}X_i\right)^3=\sum\limits_{i=1}^{n} X_i + 6\sum\limits_{1\le i< j\le n} X_iX_j+6\sum\limits_{1\le i<j<k\le n} X_iX_jX_k $$

，即

$$ s_n=n!\mathbb E\left(\sum\limits_{i=1}^{n} X_i + 6\sum\limits_{1\le i< j\le n} X_iX_j+6\sum\limits_{1\le i<j<k\le n} X_iX_jX_k\right)=n!\left(\sum\limits_{i=1}^{n}\dfrac{1}{i}+\sum\limits_{1\le i< j\le n}\dfrac{6}{ij}+\sum\limits_{1\le i<j<k\le n}\dfrac{6}{ijk}\right) $$

，显然右侧可以通过简单的递推得到。

L 00:43 +0

K 02:44 +2

Ender_hz: 贡献了一发乱搞，后面 _istina_ 想出正解做法之后我实现的过程中又罚了一发。

B 04:32 +0

Ender_hz: 倒推就行，就是每次考查 A_i 的值在 $[x, y]$ 内时对应底数的范围 $[l, r]$，然后作为 A_{i-1} 的值域，注意无穷大底数的单独处理和底数需要大于 $\max A_i$。

题解提供的做法是证明了合法底数的连续性和单调性然后用二分查找答案，实现起来应该略容易一些。

赛后

D（Ender_hz 补）

Ender_hz: 取 $\Delta_{i, j}=A_{i, j + 1}-A_{i, j}$，则由题给条件：

- $\sum\limits_{j=1}^{n-1} \Delta_{i, j}\le m$
- $\Delta_{i, j}\ge 0$
- $\Delta_{i, j}\ge \Delta_{i + 1, j}$

如此，$A_{n\times n}$ 与 $\Delta_{n\times (n-1)}$ 一一对应，统计 A 的个数即统计 Δ 的个数。

不难发现 Δ 的每一列之间相互独立，且每一列均单调不增，不难证明每一列在 $\Delta_{1, j}\ge \Delta_{2, j}\ge \cdots\ge \Delta_{n, j}\ge 0$ 约束下的方案数为 $\dbinom{\Delta_{1, j}}{n - 1 + \Delta_{1, j}}$，因此目标转化为 $\sum\limits_{\Delta_{1,1}+\Delta_{1, 2}+\cdots+\Delta_{1, n}\le m} \prod\limits_{1\le j\le n-1} \dbinom{\Delta_{1, j}}{n - 1 + \Delta_{1, j}}$。

设 $F(x)=\sum\limits_{i\ge 0} \dbinom{i}{n-1+i}x^i$，目标转化为 $\sum\limits_{i=0}^{m}[x^i]F^{n-1}$。

由广义二项式定理

$$ (1-x)^{-n}=\sum\limits_{k=0}^{+\infty}\binom{-n}{k}(-x)^k=\sum\limits_{k=0}^{+\infty} \binom{n+k-1}{k}x^k $$

可得 $F(x)=(1-x)^{-n}, F^{n-1}(x)=(1-x)^{-n(n-1)}$，故所求即为 $\sum\limits_{i=0}^{m}\dbinom{n(n-1)-1+i}{i}=\dbinom{n(n-1)+m}{m}$。

由于 $n(n-1) +m$ 较大，计算时使用 Lucas 定理后做 $2m$ 次运算即可，时间复杂度 $\mathcal O(m\log P)$。

E（_istina_ 补）

考虑朴素的分块。记块长为 L，a 的第 i 个块为 A_i，a_p 的第 i 个块是 B_i。

再记 A_i 整块加一后 B_j 的增量为 $f_{i, j}$，并对 f_i 做前缀和 $S_{i, j}$。（可以在预处理时 $\mathcal O(n + (\frac{n}{L})^2)$ 求得）

修改涉及整块时，对 A_i 维护一个懒标记 tag_i，则其对 $B_{l..r}$ 的影响之和为 $tag_i \sum_{j = l}^{r}f_{i,j}$，即 $tag_i (S_{i, r} - S_{i, l - 1})$。至于不成整块的部分，直接暴力修改。单次修改时间复杂度为 $\mathcal O(\frac nL + L)$。

查询时类似。整块用懒标记统计增量，不成整块的部分直接暴力统计。单次修改时间复杂度也为 $\mathcal O(\frac n L + L)$ 。

取 $L = \sqrt n$ 则总时间复杂度为 $\mathcal O(n + q\sqrt n)$ 可以通过本题。

F（Ender_hz 补）

Ender_hz: 给定一个类线段树的数据结构，根节点对应区间 $[1, n]$ 对于区间为 $[l, r]$ 的节点：

- 若 $2|(r-l+1)$ 则两个子节点对应区间分别为 $\left[l, \dfrac{l+r-1}{2}\right], \left[\dfrac{l+r+1}{2}, r\right]$
- 若 $2\not\mid (r-l+1)$ 则两个子节点有 $\dfrac{1}{2}$ 的概率对应 $\left[l, \dfrac{l+r}{2}-1\right], \left[\dfrac{l+r}{2}, r\right]$ 有 $\dfrac{1}{2}$ 的概率对应 $\left[l, \dfrac{l+r}{2}\right], \left[\dfrac{l+r}{2}+1, r\right]$

在树上查询区间 $[x, y]$ 时，从根节点开始。在某节点时：

- 若该节点对应区间是 $[x, y]$ 的子集，则结束；
- 否则，若该节点左（右）子节点对应区间与 $[x, y]$ 的交集不为空，则访问左（右）节点；
- 最终返回访问的节点数。

记 $query(x, y)$ 为在树上查询区间 $[x, y]$ 的结果，求满足 $query(x, y)=i, \forall 1\le i\le 2n$ 的区间期望数量。

思路如下：不难证明在对应区间 $[l, r]$ 的节点查询子区间 $[x, y]$ 的结果的分布仅与区间长度 $r-l+1$ 有关。

对于区间长度 len 我们维护 $dp_{len,k}$ 为查询结果恰为 k 的子区间期望数量，同时维护 $pre_{len, k}$ 为查询结果恰为 k 的前缀区间期望数量 $suf_{len, k}$ 为查询结果恰为 k 的后缀区间期望数量，那么我们有:

边界条件：

- $dp_{1,1}=pre_{1,1}=suf_{1,1}=1$

转移方程：

- 对于 dp:
  - 若访问的子区间恰为自身，则 $dp_{len, 1}=1$
  - 若访问的子区间恰在某个儿子内，则对 $dp_{len, k},k\ge 2$ 的贡献为 $dp_{len^L, k-1}+dp_{len^R, k-1}$
  - 若访问的子区间跨越两个儿子且不为自身，则对 $dp_{len, k},k\ge 3$ 的贡献为 $\sum\limits_{\quad i,j\ge 1,i+j=k-1} suf_{len^L, i}\cdot pre_{len^R, j}$
- 对于 pre suf 同理）：
  - 若访问的子区间恰为自身，则 $pre_{len, 1}=1$
  - 若访问的子区间恰在左儿子内，则对 $pre_{len, k}, k\ge 2$ 的贡献为 $pre_{len^L, k - 1}$
  - 若访问的子区间跨越两个儿子且不为自身，则对 $pre_{len, k}, k\ge 3$ 的贡献为 $pre_{len^R, k - 2}$

特别地，当 $2\not\mid len$ 时需要将两种 (len^L,len^R) 的情况取平均。

G□Ender_hz 补）

Ender_hz: 题意可以简化为：给定一个字符串 $|s|=n$□带修□□$s_i\in\{\texttt{L},\texttt{R}\}$□在每一时刻，将 s 中所有的子串 $\mathtt{RL}$ 替换为 $\mathtt{LR}$□问多少时间后 s 到达终态。

首先考虑 $\mathtt{L}$ 的行为（注意到 $\mathtt{L}$ 只会向左移动）：

- 左侧全部都是 $\mathtt{L}$□已经到达终态；
- 左侧有紧邻的 $\mathtt{R}$□与 $\mathtt{R}$ 交换位置；
- 左侧有 $\mathtt{R}$ 但不紧邻□$\iff$ 左侧紧邻的是 $\mathtt{L}$□□待左侧的 $\mathtt{L}$ 在某一时刻与 $\mathtt{R}$ 交换位置后变为上一个状态。

由此可以得到：一个 $\mathtt{L}$ 到达终态前一定会与左边的每一个 $\mathtt{R}$ 交换，需要的时间至少为其个数；同时它有可能在交换过程中被左侧的 $\mathtt{L}$ 卡住。综上：到达终态需要的时间等于其左侧最近的 $\mathtt{L}$ 到达终态的时间 $+1$ 和左侧 $\mathtt{R}$ 的个数两者取最大值（开始时就在最左侧的 $\mathtt{L}$ 需要的时间为 0）。

因此，对每一个位置维护 (dp, num) 二元组，分别代表该位置起（含）左侧第一个 $\mathtt{L}$ 到达终态的时间和左侧 $\mathtt{R}$ 的个数，有转移方程

$$ (dp_i,num_i)=\begin{cases} (\max\{dp_{i-1}+1, num_{i-1}\}, num_{i-1}), &s_i=\mathtt{L}\\ (dp_{i-1}, num_{i-1}+1), &s_i=\mathtt{R} \end{cases} $$

。如此，单次查询复杂度为 $\mathcal O(n)$□

为了降低时间复杂度，我们定义三元组 (A, B, C)□它与二元组复合时具体如下：

$$ (dp, num)\oplus(A, B, C)=(\max\{dp + A, num + B\}, num + C) $$

，由此转移方程可以改写为：

$$ (dp_i,num_i)=\begin{cases} (dp_{i-1}, num_{i-1})\oplus(1, 0, 0), &s_i=\mathtt{L}\\(dp_{i-1}, num_{i-1})\oplus(0, -\infty, 1), &s_i=\mathtt{R} \end{cases} $$

。同时，定义三元组之间的复合如下：

$$ (A, B, C)\oplus(A',B',C')=(A + A', \max\{B + A', C + B'\}, C + C') $$

，这样就使得结合律被满足：

$$\begin{align*} &((dp, num)\oplus(A, B, C))\oplus(A', B', C')&\\=&(\max\{dp + A, num + B\}, num + C)\oplus(A', B', C')&\\=&(\max\{\max\{dp + A, num + B\} + A', num + C + B'\}, num + C + C')&\\=&(\max\{dp + A + A', num + \max\{B + A', C + B'\}\}, num + C + C')&\\=&(dp, num)\oplus(A + A', \max\{B + A', C + B'\}, C + C')&\\=&(dp, num)\oplus((A, B, C)\oplus(A', B', C'))& \end{align*} $$

，因此可以用线段树维护三元组，在单次查询前通过线段树上二分找到第一个 $\mathtt{R}$ 出现的位置，再在线段树上查询从第一个 $\mathtt{R}$ 到字符串尾的每一个位置的三元组的复合，最后与二元组 $(0, 0)$ 复合得到答案，单次查询时间复杂度 $\mathcal O(\log n)$□

M□Ender_hz 补）

Ender_hz: 题意为 m 进制下，给定 n 个互不相同的数码 $0\le a_i < m,1\le i\le n$□恰组成两个数 x,y□求 $\min |x-y|$□

不妨设 $x<y$□a 递增。记 $x=\overline{x_{k-1}\cdots x_1x_0}, y=\overline{y_{k-1}\cdots y_1y_0}$□高位可补 0），此时 $y-x=\sum\limits_{i=0}^{k-1}(y_{i}-x_{i})m^i$□

1. $2|n$

显然要使 $y-x$ 尽量小，就有 $k=\dfrac{n}{2}$□$y_{k-1}>x_{k-1}$ 但 $y_{k-1}-x_{k-1}$ 尽量小□$\iff$ 两者在 a 中相邻），且 $y_{k-2}<y_{k-3}<\cdots<y_1<y_0<x_0<x_1<\cdots<x_{k-3}<x_{k-2}$□容易证明，当且仅当 $y_{k-2}<\cdots<y_0<x_{k-1}<y_{k-1}<x_0<\cdots<x_{k-2}$ 时，$\sum\limits_{i=0}^{k-2}(y_{i}-x_{i})m^i$ 最小，且 x_{k-1}, y_{k-1} 单向移动时，前式一定单调增，因此只需找到两侧距离 a_k,a_{k+1} 最近的 a_p,a_{p+1} 满足 $a_{p+1}-a_p=\min\limits_{1\le i<n} a_{i+1}-a_{i}$ 作为 x_{k-1},y_{k-1} 后选择两者中的最小值即为所求。

2. $2\not\mid n$

显然要使 $y-x$ 尽量小，就有 $k=\dfrac{n+1}{2}$□此时有 $x_{k-1}=0$□□

(1) $a_1 = 0$

若 $y_{k-1}=a_1=0$□则可转化为剩余 $a_{2..n}$ 的 $n-1$ 个数的问题，通过第 1 种情况解决；

若 $y_{k-1}\neq a_1$□不难证明要使 $y-x$ 最小，一定有 $y_{k-1}=a_2$□剩余部分与确定了高位的第 1 种情况同理。

(2) $a_1 \neq 0$

不难证明要使 $y-x$ 最小，一定有 $y_{k-1}=a_1$□剩余部分与确定了高位的第 1 种情况同理。

总结

Ender_hz:

istina:

MeowScore: