

Problem A. 皮卡丘与 Codeforces

签到题。

- $n = 1$ 时, 只有唯一一种打比赛的方案。
- $n > 1$ 时, 考虑当前 rating 最高的账号, 所有能涨分的比赛都用这个账号打, 其它比赛用其它账号打即可。

时间复杂度 $\mathcal{O}(n)$ 。

Problem B. Serval 的元素周期表

一个显然的做法是递推。为了表述方便, 设给出单词为 S 。

考虑 f_i 表示 $S[1 \dots i]$ 是否可以被拼出来。注意到元素符号长度不超过 2, 因此只需检查 S_i 和 $S[i-1 \dots i]$ 是否为元素符号, 结合 f_{i-2} 与 f_{i-1} 推出 f_i 即可。

边界条件 $f_0 = \text{true}$, 时间复杂度 $\mathcal{O}(|S|)$ 。

Problem C. Popcount Game

令 S_i 为所有二进制中有 i 个 1 的数的集合, 即 $S_i = \{x \mid \text{popcount}(x) = i\}$ 。对于给定的 n , 我们记:

- $f(i)$ 表示当前数属于 S_i 去除最大值的集合时, 先手的胜负状态 (先手必胜为 1, 必败为 0, 可以证明这个集合中所有数的胜负状态是相同的)。方便起见, 如果去除最大值的集合为空集, 则我们假定此时先手必胜;
- $g(i)$ 表示当前数为 S_i 中的最大值时, 先手的胜负状态 (先手必胜为 1, 必败为 0)。

$2^n - 1$ 是唯一的结束状态, 因此 $f(n) = 1, g(n) = 0$ 。当 $0 \leq i < n$ 时, 有:

- $g(i) = 1$, 当且仅当 $f(i+1) = 0$ 或 $g(i+1) = 0$;
- $f(i) = 1$, 当且仅当 $g(i) = 0$ 或 $f(i+1) = 0$ 或 $g(i+1) = 0$ 。

初始状态的先手胜负状态即为 $g(0)$ 。通过计算可以发现, 当 $n-i$ 为奇数时, 有 $f(i) = g(i) = 1$; 当 $n-i$ 为偶数时, 有 $f(i) = 1, g(i) = 0$ 。即是说, n 为奇数时先手必胜, n 为偶数时先手必败。

时间复杂度 $\mathcal{O}(1)$ 。

Problem D. 不动数组

对于 $n \geq 7$, 可以发现 $a_0 = n-4, a_1 = 2, a_2 = 1, a_{n-4} = 1$, 其他为 0 是一组可行解。 $n < 7$ 的情况可以暴力搜。结论是 $n \leq 3$ 或 $n = 6$ 时无解。

这一结论, 既可以通过打表观察得到, 也可以推理得出。容易证明 $0 \leq a_i \leq n-1$, 以及 $\sum_{i=0}^{n-1} a_i = \sum_{i=0}^{n-1} ia_i = n$ 。当 n 充分大时, 从 $\sum_{i=0}^{n-1} ia_i = n$ 可知非零的 a_i 只有 $\mathcal{O}(\sqrt{n})$ 级别, 因此 a_0 是 $n - \mathcal{O}(\sqrt{n})$ 级别。显然 a_{a_0} 只能等于 1, 因为不可能有第二种数有 a_0 那么多个了。因此 $a_1 \geq 1$, 但是 a_1 不能等于 1, 因为这会导致 $\text{cnt}_1 \geq 2$, 矛盾。不妨设 $a_1 = 2$ 及 $a_2 = 1$, 答案就出来了。

单组数据时间复杂度 $\mathcal{O}(n)$ 。

Problem E. 最后的轻语

简要题意：给两个串 s, t ，问 t 能否由若干个 s 的前缀拼接而成。

记 s, t 的长度分别为 n, m ，字符串下标从 0 开始。

将 s, t 连接起来，使用 KMP 算法求出其前缀函数，记为 π 。 t 能由若干个 s 的前缀拼接而成当且仅当对于 $\forall i \in [n, n+m)$ ，都有 $\pi_i > 0$ 。证明如下：

- 若存在 $\pi_i = 0$ ，由前缀函数的定义可以得知，串 $t[0 \dots i-n]$ 的所有后缀均不是串 s 的前缀，因此无法用 s 的前缀拼接出 $t[0 \dots i-n]$ ，也就无法拼接出 t 。
- 若对于 $\forall i \in [n, n+m)$ ，都有 $\pi_i > 0$ 。从后往前考虑，设当前考虑到了 t 中的第 j 位（从第 $m-1$ 位开始），对应的前缀函数为 π_{j+n} ，那么串 $t[j-\pi_{j+n}+1 \dots j]$ 是 s 的前缀。接下来只需要考虑 $j := j - \pi_{j+n}$ 。因为前缀函数均为正数， j 是单调递减的， t 中的每一位都会出现在 s 的前缀中。由此我们得到了一种将 s 的前缀拼接成 t 的方法。

时间复杂度 $\mathcal{O}(n+m)$ 。

Problem F. 魔法少女莫卡的诞生

简单起见，我们记给出的 $2n$ 阶排列为 $A_1, \dots, A_{2n}$ ，记交换后的排列为 $A_{p_1}, \dots, A_{p_{2n}}$ 。对于某个排列 p ，最少交换次数为 p 的逆序数 $\tau(p)$ 。

我们有如下引理：

引理 存在一种交换次数最小的方案，使得对于任意 $1 \leq i \leq 2n$ ，有 $|p_i - i| \leq 1$ 。

证 首先在 $2i-1$ 和 $2i$ 连边，假设某次交换了 $2i$ 和 $2i+1$ ，那么我们将 $2i$ 和 $2i+1$ 连边，显然不同连通块之间不会相互影响，不难发现每个连通块的点数一定是偶数，那么对于一个连通块，如果它有 $2x$ 个点，那么至少需要进行 $x-1$ 次操作，而对于这个连通块，我们如果直接交换其中每对元素，那么需要 x 次，即一个连通块至多操作 x 次，否则我们可以直接交换每一对元素。如果整个连通块共进行了 x 次操作，那么对于当前连通块，我们就可以将它的方案转换为交换每对元素，此时所有交换没有交集。如果整个连通块共进行了 $x-1$ 次操作，那么每次操作一定是交换每对相邻元素相邻的两个点，该操作也没有交集。那么不难存在一种最小交换次数的方案，使得数列中的任意一个位置至多会被交换一次。假设有位置交换了两次，那么两次操作必然存在交集，与上文矛盾。因此 p_i 只有可能是 $i, i-1, i+1$ ，引理得证。

引理说明，存在一种交换次数最小的方案，只包含不动连续段和相邻交换连续段。我们需要求出在这种情况下交换次数最小是多少。

方便起见，我们称 A_{2k-1} 与 A_{2k} 为一对元素对。

考虑以元素对为单位动态规划。对于元素对 A_{2k-1}, A_{2k} 以及与之相关的交换，只要交换方案确定，元素对就对应地给出了限制条件，共有五种情况：

- 元素对未交换。

这要求 $A_{2k-1} < A_{2k}$ 。

- 交换 A_{2k-1}, A_{2k} 。

这是元素对内的交换，要求 $A_{2k} < A_{2k-1}$ 。

- 交换 A_{2k-2}, A_{2k-1} 。

这是左相邻元素对之间的交换，要求 $A_{2k-2} < A_{2k}$ 。

- 交换 A_{2k}, A_{2k+1} 。

这是右相邻元素对之间的交换，要求 $A_{2k-1} < A_{2k+1}$ 。

- 交换 A_{2k-2}, A_{2k-1} 以及 A_{2k}, A_{2k+1} 。

这是左右相邻元素对之间的交换，要求 $A_{2k-2} < A_{2k+1}$ 。

前两种情况中，元素对之间是独立的；后三种情况中，相邻元素对之间是相关的。设 $f_{i,j}$ 表示考虑完前 i 对元素对，第 i 与 $i+1$ 对元素对之间独立 ($j=0$) 或是相关 ($j=1$) 时，前 i 对元素对内的最小交换次数。分类讨论可得状态转移方程：

$$f_{k,0} = \min \begin{cases} f_{k-1,0}, & A_{2k-1} < A_{2k} \\ f_{k-1,0} + 1, & A_{2k} < A_{2k-1} \\ f_{k-1,1} + 1, & A_{2k-2} < A_{2k} \end{cases}$$

$$f_{k,1} = \min \begin{cases} f_{k-1,0}, & A_{2k-1} < A_{2k+1} \\ f_{k-1,1} + 1, & A_{2k-2} < A_{2k+1} \end{cases}$$

从上至下依次对应前述的五种情况。边界条件为 $f_{0,0} = 0, f_{0,1} = +\infty, f_{n,0}$ 即为所求的最小交换次数。

时间复杂度 $\mathcal{O}(n)$ 。

Problem G. Serval 的数学课堂

先求出不除去区间最大值与最小值的情况下每个数对答案的贡献系数，再减去每个数作为最大值或最小值的贡献系数。

每个数作为最大值或最小值的区间是一个区间，可以用单调栈求出。对于位置 p , $[\text{lmax}_p, \text{rmax}_p]$ 表示其所有包含 p 的子区间均以 A_p 为最大值， $[\text{lmin}_p, \text{rmin}_p]$ 同理。

因此答案为：

$$\sum_{p=1}^n (f(1, n, p) - f(\text{lmax}_p, \text{rmax}_p, p) - f(\text{lmin}_p, \text{rmin}_p, p)) A_p$$

其中 $f(l, r, p)$ 的含义是：

$$f(l, r, p) = \sum_{i=l}^p \sum_{j=p}^r \frac{[j-i \geq 2]}{j-i-1}$$

答案括号中 3 项分别代表了不除去区间最大值与最小值的情况下 A_p 对答案的贡献系数、 A_p 作为最大值的贡献系数、 A_p 作为最小值的贡献系数。

显然有 $\text{lmax}_p \leq p \leq \text{rmax}_p$ 且 $\text{lmin}_p \leq p \leq \text{rmin}_p$, 因此我们限制 $f(l, r, p)$ 的参数满足 $l \leq p \leq r$ 。

在正式计算 $f(l, r, p)$ 之前, 先定义辅助函数 $g(m)$:

$$g(m) = \sum_{i=1}^m \frac{m-i+1}{i}$$

注意到 $g(m)$ 可改写为下式:

$$g(m-2) = \sum_{i=1}^m \sum_{j=i}^m \frac{[j-i \geq 2]}{j-i-1} = \sum_{i=l}^{l+m-1} \sum_{j=i}^{l+m-1} \frac{[j-i \geq 2]}{j-i-1}$$

也即有:

$$g(r-l-1) = \sum_{i=l}^r \sum_{j=i}^r \frac{[j-i \geq 2]}{j-i-1}$$

考虑 $f(l, r, p)$ 如何计算, 依容斥原理可得:

$$\begin{aligned} f(l, r, p) &= \sum_{i=l}^p \sum_{j=p}^r \frac{[j-i \geq 2]}{j-i-1} \\ &= \left(\sum_{i=l}^r \sum_{j=l}^r - \sum_{i=l}^r \sum_{j=l}^{p-1} - \sum_{i=p+1}^r \sum_{j=l}^r + \sum_{i=p+1}^r \sum_{j=l}^{p-1} \right) \frac{[j-i \geq 2]}{j-i-1} \end{aligned}$$

注意到只有当条件 $j-i \geq 2$ 成立时, 对应项才对 $f(l, r, p)$ 有贡献。利用 $l \leq p \leq r$, 简单限制求和的上下限可得:

$$\begin{aligned} f(l, r, p) &= \left(\sum_{i=l}^r \sum_{j=l}^r - \sum_{i=l}^r \sum_{j=l}^{p-1} - \sum_{i=p+1}^r \sum_{j=l}^r + \sum_{i=p+1}^r \sum_{j=l}^{p-1} \right) \frac{[j-i \geq 2]}{j-i-1} \\ &= \left(\sum_{i=l}^r \sum_{j=l}^r - \sum_{i=l}^{p-1} \sum_{j=l}^{p-1} - \sum_{i=p+1}^r \sum_{j=p+1}^r \right) \frac{[j-i \geq 2]}{j-i-1} \\ &= \left(\sum_{i=l}^r \sum_{j=i}^r - \sum_{i=l}^{p-1} \sum_{j=i}^{p-1} - \sum_{i=p+1}^r \sum_{j=i}^r \right) \frac{[j-i \geq 2]}{j-i-1} \end{aligned}$$

将上式与 $g(r-l-1)$ 对应的和式比较, 显见上式即为:

$$f(l, r, p) = g(r-l-1) - g(p-l-2) - g(r-p-2)$$

考虑 $g(m)$ 的计算:

$$\begin{aligned} g(m) &= \sum_{i=1}^m \frac{m-i+1}{i} \\ &= \sum_{i=1}^m \frac{m+1}{i} - 1 \\ &= -m + (m+1) \sum_{i=1}^m \frac{1}{i} \end{aligned}$$

通过预处理 $\frac{1}{i}$ 的前缀和即可快速计算 $g(m)$ 。

需要注意有多个数相等时的处理。时间复杂度 $\mathcal{O}(n)$ 。

Problem H. 莫卡与阿拉德大陆

将圆之间的包含关系定义为一个偏序，显然这个偏序关系构成了一个森林，即每个圆被其祖先包含且不被其它点包含。按半径排序将所有圆排序后，即可 $\mathcal{O}(n^2)$ 建出森林，需要注意完全重合的圆。

注意到莫卡对每个圆最多释放一次反转魔法，因为更多次释放只会相互抵消影响。考虑使用树上背包解决这一问题。设 $dp_{u,i,0/1}$ 表示点 u 及其子树选择 i 个节点反转时的最大受诅咒面积/未受诅咒面积。采用树上背包的方式更新 dp 并计算答案。假设当前 dp_u 已经计算了 u 的一些子树，考虑加入一棵新子树 v ，转移方式比较简单，如下：

$$\begin{aligned} dp'_{u,i+j,0} &\leftarrow \max(dp'_{u,i+j,0}, dp_{u,i,0} + dp_{v,j,0}) \\ dp'_{u,i+j,1} &\leftarrow \max(dp'_{u,i+j,1}, dp_{u,i,1} + dp_{v,j,1}) \end{aligned}$$

还需要考虑 u 是否释放反转魔法，分别考虑这两种情况，并记 A_u 表示圆 u 去除它包含的其它圆的剩余面积，有：

$$\begin{aligned} dp'_{u,i,0} &\leftarrow A_u + dp_{u,i,0} \\ dp'_{u,i+1,0} &\leftarrow \max(dp'_{u,i+1,0}, dp_{u,i,1}) \\ dp'_{u,i,1} &\leftarrow dp_{u,i,1} \\ dp'_{u,i+1,1} &\leftarrow \max(dp'_{u,i+1,1}, A_u + dp_{u,i,0}) \end{aligned}$$

注意到该森林可能由多棵树组成。此时添加一个虚拟树根，将所有真实树根作为儿子即可。易知 $dp_{\text{root},i,1}$ 表示释放 i 次魔法时最大的未受诅咒面积，等价于圆的总面积减去最小的受诅咒面积，与 k 比较即可得出答案。

背包时 i, j 显然分别小于等于 sz_u, sz_v ，可以证明加入这样的上界优化后，树上背包的时间复杂度为 $\mathcal{O}(n^2)$ 。

Problem I. 皮卡丘与 PCPC 集训队

由于 $x_i = (ax_{i-1} + b) \bmod p$ ，因此存在一个整数 k_i ，使得 $ax_{i-1} + b - x_i = k_i p$ 。换句话说， $p \mid (ax_{i-1} + b - x_i)$ 。那么， $p \mid \gcd_{i=2}^n (ax_{i-1} + b - x_i)$ 。除此之外，还需要满足 $ax_{i-1} + b - x_i \geq 0$ ，以及 $p > a, b, x_i$ ，否则不满足取模的定义。即， p 必须满足 $\max(a, b, x_1, \dots, x_n) < p \leq c$ 。记这个区间为 $[l, r]$ 。

若 $\gcd$ 不为 0，即 $ax_{i-1} + b - x_i$ 不全为 0，要求出它的所有因数。显然 $\gcd$ 最大可达 10^{12} 级别，若枚举所有小于根号的数试除，复杂度过高。因此需要枚举根号内的所有质数，将 $\gcd$ 质因数分解，再使用 dfs 生成所有约数。需要注意检查约数是否在 $[l, r]$ 间。

若 $\gcd$ 为 0，即所有 $ax_{i-1} + b - x_i$ 全为 0，相当于要求区间 $[l, r]$ 内所有整数的异或和。这里需要注意特判 $l > r$ 的情况。它相当于 $[0, r]$ 和 $[0, l-1]$ 内所有整数的异或和的异或和。有

$$\bigoplus_{i=0}^r i = \begin{cases} r & (r \equiv 0 \pmod{4}) \\ 1 & (r \equiv 1 \pmod{4}) \\ r+1 & (r \equiv 2 \pmod{4}) \\ 0 & (r \equiv 3 \pmod{4}) \end{cases}$$

单组数据时间复杂度 $\mathcal{O}(\frac{\text{gcd}}{\log \text{gcd}} + d(\text{gcd}))$ 。其中 d 是约数个数函数。查表可知 10^{12} 内的约数个数至多为 6720，可以接受。