神奇 DP 在哪里

nantf

2025 年 8 月 2 日

叠甲

- 接到的指标是“非常规 DP 相关技巧”。这是人类智慧的意思吧！
- 所以请不要问“这是怎么想到的”一类问题。我是真的不知道！
- 题目大致按难度排序。但我真的不会判断这些题的难度，太为难我了！
- 难度跨度（应该）很大，可以让大家都接近一下神明！

Gym103870Q Food Poisoning

- 维护一个 n 以内的质数集合 S，初始时包含最小的 m 个质数（保证均 $\leq n$）。
- 接下来 q 次操作，每次给定一个 n 以内的质数 x，若 S 包含 x 则将 x 删除，否则将 x 加入 S。
- 每次操作后，询问 1 到 n 中有多少个数被至少一个 S 中的质数整除。
- $1 \leq n \leq 10^6, 1 \leq q \leq 10^5$。

Gym103870Q Food Poisoning

- 定义 f_n 为 n 以内，有多少个数不能被任何 S 中的质数整除。
- 加入质数 p 时，f_n 需要减去能被 p 整除但不能被 S **中原有**质数整除的个数，即 $f_{\lfloor n/p \rfloor}$。更新顺序从大到小。
- 删除质数 p 时，f_n 需要加上能被 p 整除但不能被 S **中其它**质数整除的个数，即 $f_{\lfloor n/p \rfloor}$。更新顺序从小到大。
- 答案为 $n - f_n$。更新过程中只需要用到 $f_{\lfloor n/i \rfloor}$，这样的位置一共有 $O(\sqrt{n})$ 个。
- 时间复杂度 $O(n + q\sqrt{n})$。
- (虽然说是按难度排序，但这题我当时没做出来。足以证明我是真不会这类题！)

[CERC2017] Kitchen Knobs

- （题面经过微小转化。）
- 给定长度为 n 的序列 a_i，其中 $0 \le a_i \le 6$。
- 你可以进行如下操作若干次：选择一个区间以及 $0 \le x \le 6$，将区间内的所有 a_i 变为 $(a_i + x) \bmod 7$。
- 问将所有 a_i 变为 0 的最小操作次数。
- $1 \le n \le 501$。

[CERC2017] Kitchen Knobs

- 差分（$\bmod 7$ 意义下）得到 $d_1, \ldots, d_{n+1}$。操作即选择 $i < j$ 和 x，令 $d_i \leftarrow (d_i + x) \bmod 7, d_j \leftarrow (d_j - x) \bmod 7$。
- 进一步发现不需要 $i < j$，就是选一个加一个减。若对进行了操作的 i, j 连边，则一个连通块内的和（$\bmod 7$ 意义下）必然为 0，而又必然可以只用任一棵树。问题转换为将所有 d_i 分成尽可能多的组，每组和均为 0。
- 0 没必要再动，记录 1 到 6 各有多少个。先贪心凑 $16, 25, 34$ 的对子，这种对子分在不同组不优。
- 剩下三种数就可以 DP 了。令 $f_{i,j,k,l}$ 表示目前剩下的三种数各剩 i, j, k 个，正在凑的这组和为 l。转移选一个加进这组。
- 时间复杂度 $O(n^3)$。常数不大，已经可以通过。
- 事实上 l 这一维还能省，完全由 i, j, k 决定。其实也相当于把 i, j, k 按某个顺序轮流 -1 使得过程中和为 0 的次数最多。

CF1178F2 Long Colorful Strip

- 初始有一条长度为 m 的纸带，每个位置有颜色 0。
- 接下来 n 次操作。第 i 次操作选择纸带上颜色全部相同的一个区间，并将区间内的颜色均变为 i。
- 问有多少种选择区间的方案，使得最终纸带每个位置的颜色为 c_i。对大质数取模。
- $1 \leq n \leq 500, n \leq m \leq 10^6, c_i \neq 0$。
- 部分分：$n = m$ 且 c 是一个 1 到 n 的排列。

CF1178F2 Long Colorful Strip

- 先看部分分。第 i 次操作选了一个区间染色后，所有操作再也不能跨过区间端点，于是区间内外独立了。这个区间必然包含排列中 i 所在的位置，而且这个位置以后不能再被染色，所以这两边也独立了。
- 于是这样分出了四个独立的区间，可以在这个子结构上 DP。
- 令 $f_{l,r}$ 表示现在 $[l,r]$ 内是同一种颜色，且每个位置都不是需要的颜色，且接下来的操作不能跨过端点的方案数。
- 转移看其中最小值的位置 $c_p = x$。第 x 次之前的操作和这个区间无关。枚举这次选择的区间 $l \le i \le x \le j \le r$，那么 $f_{l,r} = \sum f_{l,i-1} f_{i,x-1} f_{x+1,j} f_{j+1,r}$。
- 时间复杂度 $O(n^4)$。注意到转移式中 i,j 独立，优化为 $O(n^3)$。

CF1178F2 Long Colorful Strip

- 回到原题。现在 m 太大，不能直接区间 DP。找点性质：
  - 性质 1：如果两个相邻位置最终同色，那么它们一直同色，同时被某一操作覆盖或不覆盖。所以这两个位置可以缩成一个。
  - 性质 2：称相邻两个位置不同色的个数为转折数（包括虚拟的 0 和 $n+1$ 处颜色为 0），一次操作最多使转折数加 2。所以按性质 1 缩完后若长度大于 $2n-1$ 则无解。

CF1178F2 Long Colorful Strip

- 现在 $m \leq 2n-1$，几乎可以直接套用了。但是因为不是排列，有更多的一些细节。
- 比如 $[2,1,2]$ 是无解的，但是按照之前的会把两个 2 当成独立的。实际上就是要保证子结构与外界完全独立才行，所以如果 $[l,r]$ 中的最小值在外面也出现过，那么 $f_{l,r}=0$。
- 另外此时区间中的最小值不止一个，不过这个没有本质区别，这样拆出来的中间几个区间是固定的，只有两边的四个和原来一样。
- 时间复杂度 $O(n^3)$。序列长度 $2n$ 一眼常数很爆炸，但仔细分析一下其实很小。

[JLOI2016] 字符串覆盖

- 给定字符串 A 及其 n 个子串 B_i。
- 将每个 B_i 放到其在 A 中出现的某个位置，问总共被覆盖的字符数量的最大值与最小值。
- T 组数据，$1 \le T \le 10, 1 \le |A| \le 10^4, 1 \le n \le 4$。

[JLOI2016] 字符串覆盖

- 先看最小值。首先若存在子串是另一个子串的子串，其必然被完全覆盖，可以删去。
- 剩下的结构就很好看了。因为放一个子串不用担心跨过前面已有的子串还有影响，直接设 $f_{i,S}$ 表示前缀 i 中已经放了 S 中的串，有一个串恰好在 i 结束。

$$f_{i,S} = \min_{p \in S} \min_{0 \le j < i} (f_{j,S/\{p\}} + \min(len_p, i - j))$$

- 因为全都是 $\min$，维护一下前缀的 $f_{j,S}$ 和 $f_{j,S} - j$ 的最小值就行了。
- 时间复杂度 $O(Tn2^n|A|)$。

[JLOI2016] 字符串覆盖

- 再看最大值。$O(n!)$ 枚举每个子串左端点的大小关系，那么贪心的想法就是尽可能往前。这不完全对，因为如果这个串太前了与上一个串重叠，有可能不如不重叠。
- 继续 $O(2^n)$ 枚举每个子串和上个子串是否重叠。若不重叠则尽可能往前放，若重叠则尽可能往后放（注意左端点大小关系固定了）。
- 时间复杂度 $O(Tn!2^n n|A|)$。还可以精细实现到 $O(T(n|A| + n!2^n n\log|A|))$。

[JLOI2016] 字符串覆盖

- 那么有的观众就要问了：主播主播，我想不到最大值的贪心做法怎么办？其实也很简单，~~只需要想到 DP 做法就行了~~。
- 此时我们需要不同的第一步处理：不强制把所有子串都放完，不影响最大值。那么转移时也不考虑完全包含上一个串的情况：不如从前面第一个没被包含的串的状态转移过来。
- 不过这时转移式子是 $\max$ 里套 $\min$，需要用点 $\log$ 级数据结构才能优化转移。
- 时间复杂度 $O(Tn2^n|A|\log|A|)$。

CF1149D Abandoning Roads

- 给定一个 n 个点 m 条边的无向连通图，只有 a, b 两种边权 $(1 \le a < b)$。
- 考虑这张图的所有最小生成树。对于每个点 i，求出 1 到 i 在所有最小生成树上的最短路的最小值。
- $2 \le n \le 70, 1 \le m \le 200$。

CF1149D Abandoning Roads

- 首先考虑一条简单路径有没有可能出现在最小生成树上，也就是试图把上面所有的边都放进生成树里。
- 对于边权为 a 的边，必然是都可以放进去的。对于边权为 b 的边，整张图在只保留 a 边的情况下会形成若干个连通块，若对这些连通块缩点，那么我们想强制加进去的边必须不成环。这是充要条件。
- 所以一条路径合法当且仅当不会离开一个连通块后再回来。但是连通块数仍然可以达到 n 个，看起来又变成旅行商问题了。

CF1149D Abandoning Roads

- 走出去再回来是个很劣的操作。事实上，如果特判掉连接同一个连通块的 b 边，那么走出去再回来至少需要 $2b$ 的代价。而在连通块大小 ≤ 3 时，一个连通块内用至多 $2a$ 的代价就可以互达了。
- 所以即使不考虑 ≤ 3 的连通块，也不会影响答案的正确性。而大小 ≥ 4 的连通块至多 $\frac{n}{4}$ 个，直接状压后跑分层图最短路就行了。
- 时间复杂度 $O(2^{n/4} m \log n)$。由于只有两种边权也可以做到 $O(2^{n/4} m)$。
- 其实理论复杂度可以做到更优，比如多记录一些走过的前几条 b 边的信息，要考虑的连通块个数可以更少。不过不重要了。

- 给定长度为 2^n 的 01 序列 $\{b_i\}$。保证 $b_0 = 1$。
- 我们称一个长度为 n 的序列 $\{a_i\}$ 是好的，当且仅当在每一个二进制位 k 下，所有 a_i 这一位的取值拼接成的 n 位二进制数 s 满足 $b_s = 1$。也即

$$\forall k \in \mathbb{N}, b_{\sum_{i=1}^{n} 2^{i-1} \lfloor \frac{a_i}{2^k} \rfloor} = 1$$

- 给定长度为 n 的序列 $\{r_i\}$。问有多少个满足 $a_i \le r_i$ 的序列 $\{a_i\}$ 是好的。对大质数取模。
- $1 \le n \le 18, 0 \le r_i < 2^{60}$。

[IOI2023 集训队互测 R3] 整数

- 直接使用数位 dp，从低位到高位，令 $f_{i,S}$ 表示填了最低的 i 位，目前 S 这些位置超过了 r 的限制。初始状态 $f_{-1,\varnothing}=1$，答案为 $f_{59,\varnothing}$。
- 转移就是看第 i 位填了什么（注意 b 的限制）：
  - 若 a_j 这一位填了 0，而 r_j 这一位是 1，那么就一定没有超过限制。
  - 若 a_j 这一位填了 1，而 r_j 这一位是 0，那么就一定超过了限制。
  - 否则是否超过限制与之前相同。
- 发现实际上是若 r_j 这一位是 1，则将 S 的第 j 位与 a_j 取与，若 r_j 这一位是 0 则为取或。所以可以直接看成 f_{i-1} 与 b 的某种卷积，使用每一维各自前缀和/后缀和的 FMT 就可以了。
- 时间复杂度 $O(2^n n\log r)$。

[IOI2023 集训队互测 R15] 翻修道路

- 给定一个 n 个点 m 条边的有向图，每条边有边权 a_i。你可以对每条边进行升级，第 i 条边升级后的边权会变为 b_i。保证 $1 \le b_i \le a_i$。
- 给定 k 个关键点的编号，对于每个 $x \in [0, m]$，问在升级至多 x 条边的情况下，从 1 号点到 k 个关键点的最短路的最大值最小是多少。
- $1 \le n, m \le 100, 1 \le k \le 8$。

[IOI2023 集训队互测 R15] 翻修道路

- 一个解下从 1 到 k 个关键点的最短路的并，形成了以 1 为根的外向树。在这个基础上可以考虑 dp。
- 令 $f_{u,i,S}$ 表示从 u 开始，升级至多 i 条边，走到 S 这个集合中最短路的最大值最小。对应了外向树上以 u 为根的子树，钦定这个子树里含有 S 这些点。这个 dp 形式上与斯坦纳树较为类似，但其实自然不少（
- 转移也与斯坦纳树类似，分为两种。第一种是枚举一条树边，分类讨论升级或不升级，从 (v, i, S) 转移到 $(u, i/i+1, S/S \cup \{u\})$。这一部分是 i, S 两维分层的分层图最短路，故复杂度为 $O(m2^k \times m \log n)$。

[IOI2023 集训队互测 R15] 翻修道路

- 第二种是合并一个点的两棵子树。从 $(u,i,S),(u,j,T)$ 转移到 $(u,i+j,S\cup T)$。朴素转移的时间复杂度为 $O(n\times m^2\times 3^k)$，不能通过。
- 注意转移方式是两边的 $\max$ 的最小值，所以可以在第二维用双指针优化，也即不妨只考虑 $f_{u,j,T}\le f_{u,i,S}$ 的转移，那么固定 u,S,T 时，对于每个 i 找到第一个满足的 j 就可以了，这个 j 随 i 递增不降。这一部分复杂度 $O(n\times m\times 3^k)$，可以通过。

- 对于一棵标准编号的无穷二叉树（也即根节点编号为 1，点 x 的左右儿子编号分别为 $2x$ 和 $2x+1$）。
- 给定正整数 s，问该二叉树上有多少条路径的点编号和为 s。对大质数取模。
- $1 \le s \le 10^{15}$。

- 首先考虑直上直下的路径。我们把这上面的点编号写成二进制后一行行写，最低位对齐，例如：

$$\begin{array}{r} 11010010 \\ 110100101 \\ 1101001011 \\ 11010010110 \end{array}$$

- 时刻牢记每条左下右上的对角线必须是一样的。

CF750G New Year and Binary Tree Paths

- 枚举点数 l。考虑右下角这个直角边长 $l-1$ 的三角形，这一部分贡献最大值在全部填 1 时取到，是 2^l-l-1。而设最浅点的编号是 x，贡献是 $x\times(2^l-1)$。所以实际上 x 的选择至多 1 种，就是 $\lfloor\frac{s}{2^l-1}\rfloor$。那么现在就是要右下角的三角形贡献等于 $s'=s \bmod (2^l-1)$。
- 这个从左到右一列一列求和，设 $f_{i,j,k}$ 表示目前填了 i 及以上的位，前面已经有了 j 个 1，还需要后面往前面进位 k 的方案数。转移就看新加的一位是 0 还是 1。因为后面往前面进位最多是 $O(l)$ 的，所以 k 这一维需记到 $O(l)$，加上外层枚举时间复杂度 $O(\log^4 s)$。

CF750G New Year and Binary Tree Paths

- 再考虑非直上直下的路径。同样枚举左右点数 l, r（包括最浅点 x），注意 $2x, 2x+1$ 都固定了要选，那么与 x 有关的部分贡献是 $x + 2x \times (2^{l-1} - 1) + (2x+1) \times (2^{r-1} - 1) = 2^{r-1} - 1 + (2^l + 2^r - 5)x$，无关的贡献是两个三角形，边长分别为 $l-2$ 和 $r-2$。
- 那么同样 x 的选择至多一种，现在要算两个三角形和为定值的方案数。发现我们不需要区分两个三角形已经填了的 1 的个数，所以状态仍然只有三维，转移就看两边分别填什么。
- 时间复杂度 $O(\log^5 s)$。

[USACO19DEC Platinum] Tree Depth

- 给定 n, k 以及大质数模数 M，对于每个 $1 \le i \le n$，解决以下问题：
  - 对于每个有恰好 k 个逆序对的 1 到 n 的排列，以最小值为子树根建立笛卡尔树，求出点 i 的深度（注意是第 i 个位置而非 i 所在的位置）之和，其中根节点深度为 1。对 M 取模。
- $1 \le n \le 300$。

[USACO19DEC Platinum] Tree Depth

- 把深度拆成每个祖先各贡献 1。也即枚举 j，计算 j 为 i 的祖先的方案数。以下 $j \neq i$（$j = i$ 的方案数就是合法排列数）。
- 这等价于排列 p 满足 $p_j < p_i$ 且 i 到 j 之间不存在 $< p_j$ 的。
- 套用求合法排列数的方法，还是一个一个确定相对大小。先确定 i, j 之间的（不含），再确定 i, j，最后确定外面的。发现除了 j 只能选在当时 $j - i + 1$ 个中的最小值，剩下的完全没有额外限制。
- 先按照正常的方法求出 $f_{n,k}$ 表示长度为 n 的排列 k 个逆序对的方案数。枚举到 i, j 时（像除掉生成函数一样）撤销掉 j 处就可以了。
- 时间复杂度 $O(n^3)$。

CF1158F Density of Subarrays

- 如果一个正整数数组 $a_1, a_2, \ldots, a_n$ 满足对所有 i 都有 $1 \le a_i \le c$，则称其为 c-数组。
- 定义 c-数组 $a_1, a_2, \ldots, a_n$ 的密度为最大的非负整数 p，使得任意长度为 p 的 c-数组都是 $a_1, a_2, \ldots, a_n$ 的一个子序列。
- 现给定一个整数 c 和一个 c-数组 $a_1, a_2, \ldots, a_n$。对于所有 $0 \le p \le n$，求出其密度为 p 的非空子序列的个数。对大质数取模。
- $1 \le n, c \le 3000$。

CF1158F Density of Subarrays

- 先考虑如何计算一个数组的密度。按照贪心匹配子序列的思路，如果想尽可能让特定序列不为子序列，先让第一个数是第一次出现最晚的那个数，第二个数是那之后第一次出现最晚的那个数，以此类推直到存在数没有出现过。
- 可以设 $f_{i,j}$ 表示有多少个 a 的子序列，使第 j 个找到了第 i 个位置。转移的话枚举下一个位置 k，要求 $[i+1, k-1]$ 中与其相同的都不选，与其不同的都至少选一个。
- 这个东西可以对每个 (i, k) 预处理，时间复杂度 $O(n^3)$。
- 注意每次转移时要求 $[i+1, k]$ 中 c 以内的数均出现过，所以 $k - i \geq c$，第二维只需要记到 n/c，时间复杂度 $O(n^3/c)$。

CF1158F Density of Subarrays

- 那么 c 小的时候怎么做呢？这次不枚举区间，一个一个位置转移。设 $f_{i,j,S}$ 表示有多少个前缀 i 的子序列，使得现在正在找第 $j+1$ 个，且从第 j 个到现在已经选了 S 这些数。时间复杂度 $O(n^2 2^c/c)$。
- 平衡一下，$c > \log n$ 时选第一个，否则选第二个。时间复杂度 $O(n^3/\log n)$。注意常数。

CF1605F PalindORme

- 称一个长度为 n 的序列 a 是 PalindORme 的，当且仅当对于任意 $1 \le i \le n$，满足 $a_1|a_2|\dots|a_i = a_n|a_{n-1}|\dots|a_{n-i+1}$，其中 '$|$' 表示按位或运算。
- 称一个长度为 n 的序列 b 是 good 的，当且仅当它可以重排成一个 PalindORme 的序列。
- 给你 n, k, m，求长度为 n，每个元素值域为 $[0, 2^k)$ 的序列中有多少个是 good 的，对 m 取模。
- $1 \le n, k \le 80$。$10^8 < m < 10^9$ 且 m 为质数。

CF1605F PalindORme

- 首先如何判断一个给定序列是不是好的呢？
- 重排后第一个位置和最后一个位置一定是一样的，所以一定是选一个出现次数 ≥ 2 的放到首尾。设当前已经在首尾的数或为 s，那么定义两个数等价当且仅当在 s 为 0 的位下全部相同。于是就是选两个等价的数放到首尾。
- 若 n 是偶数那么当且仅当能选完，n 是奇数就当且仅当能选到最后只剩一个数。
- 那么每一步选哪个呢？发现选哪个都行。因为现在能放的以后也可以。所以贪心判断的策略就是每次任取一对能放到首尾的。

CF1605F PalindORme

- 那么怎么计数呢。思考很久发现怎么数好像都会数重，于是正难则反，计算不好的序列个数。
- 不好的序列，就是在进行上面的过程中，到某一步所有数都不等价了（且不止一个）。
- 我们发现，如果这时剩下的所有数都和 0 不等价，那么无论每一步怎么选，最后分到两边的数集是唯一的，因为剩下的数永远没法配对。而如果存在和 0 等价的数，也显然只有恰好一个。
- 换句话说，我们定义**合法**好序列：
  - 对于长度为偶数的序列，定义与好序列相同；
  - 对于长度为奇数的序列，还要求重排后中间的数等价于 0。
- 那么每个不好的序列，都有唯一的最长合法好子序列。而且设这个子序列的或为 s，则这个子序列以外的数在 s 的等价意义下，互不等价且都不等价于 0。

CF1605F PalindORme

- 那么就可以开始数数了。为了在状态中体现所有数的或，记 $f_{i,j}$ 表示长度为 i 的 $[0,2^k)$ 的序列，或的 popcount 恰好为 j，合法好序列的个数。
- 直接数还是有困难。容斥一下，再记 $g_{i,j}$，意义为枚举 popcount 为 j 的一个数，钦定或是这个数的子集，合法好序列的个数之和。
- 转移就统计不好的序列以及不合法的好序列个数。前者直接枚举最长合法好子序列。后者一定是 i 为奇数时，从 $i-1$ 的好序列任意添加一个不等价于 0 的数，因为不等价于 0 所以一定不会数重。
- 时间复杂度 $O(n^2k^2)$。

CTT2022 D4T3

- 给定正整数 n，对于每个 $0 \le i, j \le n-1$，计算 1 到 n 的所有排列中，有恰好 i 个超过和 j 个下降的个数。对给定模数取模。
- (对于排列 p，位置 i 是超过当且仅当 $p_i > i$，位置 i 是下降当且仅当 $p_i > p_{i+1}$。)
- $1 \le n \le 60$。

- 听 EI 说有 0 次容斥，1 次容斥，2 次容斥的做法。来讲一个 0 次容斥的。
- 这种题看起来就比较像往排列里面一个一个填数的。然后在 1 到 n 的排列里插入一个 $n+1$ 听起来就不太靠谱。
- 因为超过数就是在 $y=x$ 上方的点数。所以考虑从左往右填，并且将所有数分成 $\leq i$ 和 $> i$ 两种情况。
- 设 $f_{i,j,k,a,b}$ 表示前 i 个位置中，有 j 个 $\leq i$ 的，**位置 i 填了一个 $\leq i$ 的数**且还没填的数中有 k 个小于它的，a 个超过 b 个下降。
- $g_{i,j,k,a,b}$ 表示**位置 i 填了一个 $> i$ 的数**且是已填的数中的第 k 小。

- k 这一维看起来有点莫名其妙。实际上是，在前缀 i 中 $> i$ 的数的具体取值暂时不重要，会在以后的过程中确定取值（也即每次 i 到 $i+1$ 时，先讨论已经填了的 $> i$ 中的最小值是不是 $i+1$）。于是 $> i$ 的数只用管相对大小，这也就是 g 的由来。
- f 相对来说更好理解了，就是用来算新加的能否组成一个下降。也就是说，f, g 中 $\leq i$ 的数是确定取值的，而 $> i$ 的数只确定了相对大小，求的是这个条件下的方案数。
- 时间复杂度 $O(n^6)$。可以对 k 这维做前缀和一类的东西做到 $O(n^5)$，常数很优秀。

Thanks!