GDKOI 2025 Day 1 解题报告

2025-12-31

填数问题

题目描述

给定一个长度为 n 的正整数序列 $e_1, e_2, ..., e_n$，以及模数 m。需要回答 q 次询问，每次询问给出上限值 M，需要回答如下问题：存在多少个长度为 n 的正整数序列 $a_1, a_2, ..., a_n$，使得

$$\prod_{i=1}^{n} a_i^{e_i} \leq M$$

输出答案对 m 取模后的结果。

约束条件：$n, q \leq 10^5, M \leq 10^7$。

子任务 1

子任务 1 占 6 分，额外的约束条件为：$n \leq 10$；对于所有 $1 \leq i \leq q$，均有 $M_i \leq 10$。

对于这个子任务，只需要暴力搜索序列 a 即可。在这一数据范围下，搜索的量级较小，可以通过。

子任务 2

子任务 2 占 11 分，额外的约束条件为：$n \leq 1000$；对于所有 $1 \leq i \leq q$，均有 $M_i \leq 1000$。

可以使用动态规划解决这一个子问题。令 $f_{i,j}$ 表示在考虑了乘积的前 i 项时，满足结果为 j 的方案数，在层间转移的方式如下：

$$f_{i,j} = \sum_{\substack{k=1 \\ k^{e_i} | j}}^{+\infty} f_{i-1,j/k^{e_i}}$$

从 f_{i-1} 转移到 f_i 的复杂度不会超过 $\mathcal{O}(M \log M)$（最劣的情况为 $e_i = 1$ 的情况，此时需要枚举 $[1, M]$ 内每个数的倍数），因此总体的时间复杂度就是 $\mathcal{O}(nM \log M)$。

子任务 3

子任务 3 占 13 分，额外的约束条件为：对于所有 $1 \leq i \leq n$，均有 $e_i \geq 12$。

对于这个子任务，需要注意到 $2^{24} > 10^7$，因此不等于 1 的项只会有不超过一个。枚举这一个项所在的位置，即可得到所有可能的数列乘积。合并这些乘积并排序之后，对每次询问二分就能得到答案。

子任务 4

子任务 4 占 19 分，额外的约束条件为：对于所有 $1 \le i \le n$，均有 $e_i \ge 9$。

对于这个子任务，除了使用前一个子任务的提示，枚举两个不为 1 的位置并进行适当优化之外，还存在另一种可行的做法。注意到 $7^9 > 10^7$，可以设立如下状态：$f_{i,j,k,l}$ 表示考虑了前 i 项，乘积为 $2^j 3^k 5^l$ 的情况数，转移和子任务 1 的转移类似。这一转移实际上效率较高，可以通过。

子任务 5

子任务 5 占 24 分，额外的约束条件为：对于所有 $1 \le i \le q$，均有 $M_i \le 10^6$。

考虑计算满足如下要求的序列个数：

$$\prod_{i=1}^{n} a_i^{e_i} = x$$

将结果记为 $f(x)$，对 $f(x)$ 求前缀和即可回答询问。因此问题在于如何计算 $f(x)$。

为了计算 $f(x)$，不妨考虑将 x 进行质因数分解，假设有 $x = p_1^{\alpha_1} p_1^{\alpha_2} ... p_k^{\alpha_k}$。不难发现，每个质数的决策存在独立性，例如我们可以先决定 α_1 个 p_1 因子放在序列的哪些位置上，然后决定 α_2 个 p_2 因子放在序列的哪些位置上，以此类推。

为了计算每个质数对应的子问题，可以发现答案等价于对序列 e_i 计算多重背包得到的方案，因此可以预先在 $[0, \log M]$ 值域上进行一次多重背包，随后将“凑出 α_i 的方案数”相乘即可得到答案。

此时的时间复杂度为 $\mathcal{O}((n + M) \log M)$，可以通过。

子任务 6

子任务 6 占 27 分，没有额外的约束条件。

从前面的推导可以发现，$f(x)$ 实际上是一个积性函数，因此可以使用线性筛求算。此时的时间复杂度为 $\mathcal{O}(M + n \log M)$，可以通过。

实际上，时间复杂度为 $\mathcal{O}(M \log \log M)$ 的埃氏筛在常数较小的前提下也可以通过，不过复杂度更劣的筛法或者质因数分解后计算的方法理论上都无法得到这一档分数。

树上跳跃

题目描述

给定一棵包含 n 个结点的树，结点 1 是树的根结点。若当前在结点 x，则有三种跳跃方式可以选择：

- 跳跃到结点 x 的父亲结点，代价为结点 x 的第一个参数 a_x；
- 跳跃到结点 x 的某个子结点，记为结点 y ，代价为结点 y 的第二个参数 b_y；
- 跳跃到某个和结点 x 深度相同的点，记为结点 y ，且满足 $x \neq y$。假设结点 x 和结点 y 的最近公共祖先为结点 z，则代价为 c_d，其中 d 表示结点 x 到结点 z 的路径上包含的边数。

求出从每个结点出发，到达结点 1 所需的最小代价。

约束条件：$1 \leq \sum n \leq 2 \times 10^5$，$0 \leq a_i, b_i, c_i \leq 10^9$。

子任务 1 & 子任务 2 & 子任务 3

子任务 1 占 6 分，额外的约束条件为：$\sum n \leq 5000$；对于所有 $1 \leq i < n$，均有 $x_i = i$，$y_i = i + 1$。

子任务 2 占 16 分，额外的约束条件为：$\sum n \leq 5000$。

子任务 3 占 9 分，额外的约束条件为：对于所有 $1 \leq i < n$，均有 $x_i = i$，$y_i = i + 1$。

这一个子任务存在两种解决方案：

- 可以发现链上的答案是容易计算的，可以直接通过求 a_i 的前缀和得到。因此可以在 $\mathcal{O}(n)$ 的时间复杂度下解决这个问题。这个算法同时也可以解决子任务 3；
- 根据跳跃方式信息，暴力建图，然后使用最短路算法在反向图上计算结点 1 到达所有点的最短路，根据实现可以做到 $\mathcal{O}(\sum n^2 \log n)$ 或 $\mathcal{O}(\sum n^2)$（例如使用无堆优化的 Dijkstra 算法）。这个算法在常数小的前提下也可以解决子任务 2。

在赛时，考虑到评分方式为计算每个子任务的历史最大得分并相加，那么可以通过对上述两种做法分别实现一份代码并提交，得到这三个子任务的分数。

子任务 4

子任务 4 占 19 分，额外的约束条件为：对于所有 $1 \leq i < n$，均有 $c_i \leq c_{i+1}$。

依然考虑建图之后使用最短路算法求解。前两种跳跃处理起来较为容易，而最后一种跳跃则需要进行一些特殊处理。

本档子任务是为虚树做法设立的。考虑到最后一种跳跃只会在深度相同的点之间进行，故考虑对同层的点建立虚树，虚树上的边权使用 c_i 的差分得到。在这个子任务的约束条件下，虚树上的边权非负，因此可以直接将虚树视为辅助点和辅助边，放在原图上跑最短路即可。时间复杂度为 $\mathcal{O}(\sum n \log n)$，可以通过。

子任务 5 & 子任务 6

子任务 5 占 29 分，额外的约束条件为：$\sum n \leq 10^5$。

子任务 6 占 21 分，没有额外的约束条件。

实际上，本题的优化建图方式非常多样，标准算法采用的方案是使用长链剖分。长链剖分本质上可以在 $\mathcal{O}(n)$ 的时间复杂度内维护一棵子树内部每一层的点信息，而这里，我们需要将子树内同一层的点缩在一起。

使用经典的长链剖分方法维护缩点的结果（包含所有点能到达的出点，以及能到达所有点的入点），在合并轻链的时候，需要对每一层将来自轻链的若干个结点和当前重链结果合并，将所有出点向入点连边，对应的边权为 c_i 序列上的某个固定值。为了避免连边过多，可以使用前后缀的思想进一步优化这部分建图，从而保证最终的边数在 $\mathcal{O}(n)$ 量级内。最后使用 Dijkstra 即可通过此题，时间复杂度为 $\mathcal{O}(n \log n)$。

一些不太优秀的建图方案可能无法得到子任务 6 的分数。

嵌套区间

题目描述

对三个闭区间 $[l_1, r_1], [l_2, r_2], [l_3, r_3]$，如果满足 $l_1 < l_2 < l_3 < r_3 < r_2 < r_1$，那么这三个区间被称为一组“嵌套区间”。

一个无序集合 $S = \{[l_1, r_1], [l_2, r_2], ..., [l_n, r_n]\}$ 是“嵌套集合”，当且仅当：

- 对于所有 $1 \le i \le n$，均有 $1 \le l_i < r_i \le 2n$；
- 对于所有 $1 \le x \le 2n$，均有 x 在 $2n$ 个端点里出现了恰好一次；
- 存在三个闭区间 $[l_i, r_i], [l_j, r_j], [l_k, r_k] \in S$，满足 $l_i < l_j < l_k < r_k < r_j < r_i$。

求不同的“嵌套集合”个数模质数 p 的结果。

约束条件：$1 \le n \le 10^6$，$9 \times 10^8 \le p \le 10^9 + 7$。

子任务 1 & 子任务 2

子任务 1 占 6 分，额外的约束条件为：$n \leq 8$。

子任务 2 占 7 分，额外的约束条件为：$n \leq 10$。

使用搜索即可解决这个问题。具体而言，每次从未选出的数字中选出最小值，并决定最小值和剩余数字中的哪个数字进行匹配。在确认下来后暴力判断是否存在一组嵌套区间即可。

使用子任务 1 的程序打表即可通过子任务 2。

子任务 3

子任务 3 占 12 分，额外的约束条件为：$n \leq 50$。

注意到题目等价于数区间嵌套数不小于 3 的方案数，反过来就只需要求嵌套数小于 2 的方案数。

考虑从前往后进行动态规划，在枚举到一个位置的时候，当前有一些左端点还没有匹配到右端点。可以把这些左端点分为两类：如果在此刻提供右端点，那么一类左端点已经有了包含的区间，而另一类还没有任何包含的区间。设前面的一类有 t_1 个，后面的一类有 t_2 个。注意到第一类的 t_1 个左端点一定在第二类的前面。

子任务 3

设 f_{i,t_1,t_2} 表示考虑前 i 个位置，第一类左端点有 t_1 个，第二类左端点有 t_2 个的方案数。转移如下：

- $i+1$ 作为左端点，此时 $t_2 \leftarrow t_2+1$。
- $i+1$ 作为 t_1 里的一个右端点，此时 $i+1$ 只能匹配这 t_1 个左端点里最左边的点，否则一定存在嵌套 ≥ 3 的区间。于是 $t_1 \leftarrow t_1-1$。
- $i+1$ 作为 t_2 里的一个右端点。假设其匹配的是从左往右第 j 个左端点，那么 $1 \sim j-1$ 的左端点对应区间都包含该区间，而 $j+1 \sim t_2$ 则无影响。于是枚举 j，有转移 $t_1 \leftarrow t_1+j-1$，$t_2 \leftarrow t_2-j$。

上述转移总体就是 $\mathcal{O}(n^4)$ 量级的，可以通过。

子任务 4

子任务 4 占 18 分，额外的约束条件为：$n \leq 300$。

注意到上面的动态规划算法瓶颈只在于处理最后一类转移，而不难发现最后一类转移可以使用前缀和优化，这样就能够将上述算法的复杂度降低至 $\mathcal{O}(n^3)$，可以通过。

子任务 5 & 子任务 6

子任务 5 占 31 分，额外的约束条件为：$n \leq 2000$。

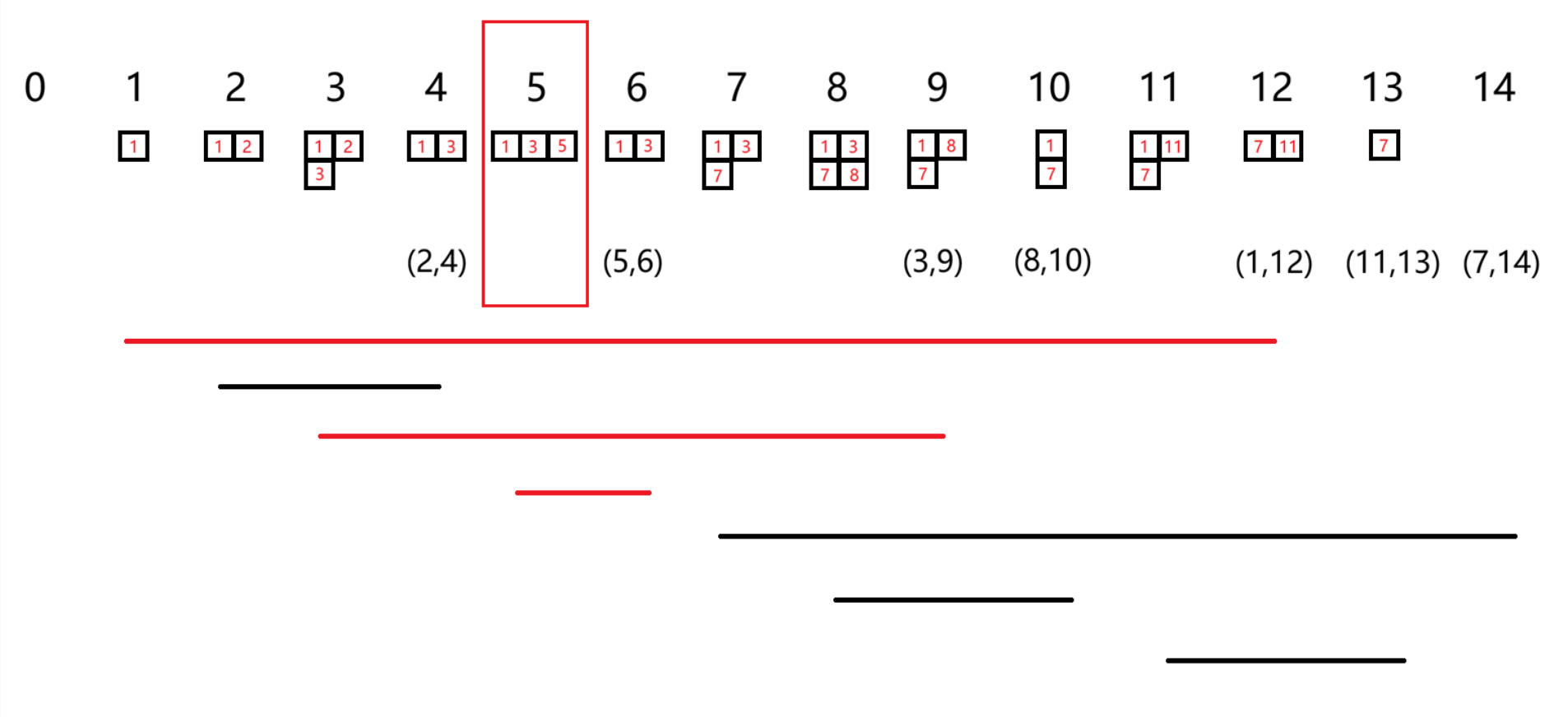
子任务 6 占 24 分，没有额外的约束条件。

实际上，合法的方案可以与大小为 $2n$ 且列数不超过 2 的振荡杨表形成双射。从区间集合到样表序列的双射如下：

1. 若 $|\lambda_{i-1}| < |\lambda_i|$，则将 i 填入 $\lambda_i \setminus \lambda_{i-1}$ 的格子中；
2. 若 $|\lambda_{i-1}| > |\lambda_i|$，则找到唯一一个 $< i$ 的数 x 使得 $(x \to \lambda_i) = \lambda_{i-1}$，将 x 与 i 配对。

双射根据构造不难证明，且当列数超过 2 时，一定会出现三个相互包含的区间。于是只需要对列数不超过 2 的振荡杨表序列进行计数。

下面是一个不符合要求的例子，可以发现这里出现了三列的杨表：



设 a 和 b 分别代表杨表中第一列和第二列的大小，那么等价于从 $(a,b)=(0,0)$ 开始，每次可以从 $(0,1)$，$(0,-1)$，$(1,0)$ 和 $(-1,0)$ 中选择一个加上，最终回到 $(0,0)$，并且需要保证 $a\geq b\geq 0$。

上述问题等价于二维平面的路径计数。为了让两个坐标轴的移动独立，不妨令 $p=a+b,q=a-b$，可以发现四个向量变成了 $(1,1)$，$(1,-1)$，$(-1,1)$ 和 $(-1,-1)$，并且限制变成了 $p\geq q\geq 0$。

记录第 i 个时刻的横坐标 p_i 和纵坐标 q_i，可以刻画两条路径 (i,p_i+2) 和 (i,q_i)，满足二者点不相交，并且不会到达第四象限。处理不相交的问题可以使用 LGV 引理，而为了处理象限问题，可以使用格路计数或者卡特兰数。

最终可以得到答案为 $C_nC_{n+2}-C_{n+1}^2$，用总方案数 $(2n-1)!!$ 减去即为答案。时间复杂度为 $\mathcal{O}(n)$。

一些中间结果可能可以导向 $\mathcal{O}(n^2)$ 的算法。

三元组

题目描述

给定 n 个三元组 (a_i, b_i, c_i)，需要对每个整数 $k \in [1, n]$ 回答如下问题的答案：

- 从 n 个元组中选出 k 个，令 A, B, C 分别表示所选元组的 a，b，c 三个分量的和，求 $A + \frac{B^2}{C}$ 的最大值。

约束条件：$1 \le n \le 3000$，$|a_i|, |b_i| \le 10^5$，$1 \le c_i \le 10^5$。

子任务 1 & 子任务 2

子任务 1 占 6 分，额外的约束条件为：$n \leq 10$。

子任务 2 占 19 分，额外的约束条件为：$n \leq 40$。

可以暴力检查 2^{2n} 个子集，计算每个子集的 A, B, C 并更新答案。时间复杂度为 $\mathcal{O}(n2^n)$，可以通过子任务 1。

结合一定的乱搞可能可以通过子任务 2。

子任务 3 & 子任务 4 & 子任务 5

子任务 3 占 8 分，额外的约束条件为：$n \leq 300$；对于所有 $1 \leq i \leq n$，均有 $c_i \leq 10$，$b_i = 1$。

子任务 4 占 13 分，额外的约束条件为：$n \leq 300$；对于所有 $1 \leq i \leq n$，均有 $a_i = 0$。

子任务 5 占 16 分，额外的约束条件为：$n \leq 90$；对于所有 $1 \leq i \leq n$，均有 $c_i = 1$。

可以看到，在这些问题下，表达式的某一个维度被固定了下来，因此可以使用一系列特殊做法完成。

对于子任务 3，可知 $B = k$，考虑进行动态规划：

$f_{i,j,k}$ 表示在前 i 个三元组中取了 j 个三元组，在满足 $C = k$ 时的最大 A 值。这个数组的转移复杂度为 $\mathcal{O}(10n^3)$，可以通过。

子任务 4 和子任务 5 都与子任务 6 的做法密切相关，只不过对应的函数具有一定的特殊性，因此不再赘述。

子任务 6

子任务 6 占 16 分，额外的约束条件为：$n \leq 300$。

需要注意到 $A + \frac{B^2}{C}$ 实际上是一元二次方程 $f(x) = -Cx^2 + 2Bx + A$ 的最大值点，并且这一个一元二次方程实际上可以通过每个三元组自身对应的函数 $f_i(x) = -c_i x^2 + 2b_i x + a_i$ 相加得到，因此可以考虑从这 n 个函数出发。

考虑令 $x = x_0$ 时的函数值 $f(x_0)$，为了让它尽可能大，显然是需要选择最大的 k 个 $f_i(x_0)$ 进行相加。需要注意到，在大多数情况下，一个邻域 $[x_0 - \varepsilon, x_0 + \varepsilon]$ 内最大的 k 个函数值来自同样的 k 个函数，因此可以直接将它们的系数求前缀和，使用得到的系数构造出一个新的函数 $F(x)$，然后计算 $F(x)$ 的最大值即可（这里不需要计算这一区域内的最大值，而是计算在全体实数上的最大值，因为额外算的部分不会影响答案）。

子任务 6

在什么时候会使得最大的 k 个函数来源会发生变化呢？显然，只有在二次函数相交前后，相交的两个二次函数的相对顺序会发生变化，此时会使得最大的 k 个函数来源发生变化。因此，可以枚举所有二次函数对，计算它们的交点，并将这些交点按照从小到大的顺序排序。随后从左到右扫描这些交点，在两个交点的中点处计算点值并从大到小排序，取前 k 个函数进行相加并计算最大值。

考虑到有 $\mathcal{O}(n^2)$ 个交点，每次处理一个交点需要 $\mathcal{O}(n \log n)$ 的排序，因此总体复杂度为 $\mathcal{O}(n^3 \log n)$，可以通过。

子任务 7

子任务 7 占 22 分，没有额外的约束条件。

这个子任务只需要注意到一个简单的优化：在处理交点的时候，不需要每次都对所有函数进行排序，因此在两个函数相交的时候，实际上只有这两个函数的相对顺序发生了变化。考虑到我们计算的是前缀和，那么前缀和本身实际上只有一个位置发生了变化（比如第 p 大的二次函数和第 $p+1$ 大的二次函数相交，那么在交换二者之后，只有 $p+1$ 个二次函数的和发生了变化），此时只需要更新对应位置的答案。

使用任意一种你喜欢的数据结构维护前 k 大的二次函数和即可。时间复杂度为 $\mathcal{O}(n^2 \log n)$，可以通过。本题还需要注意一些细节，例如交点重合、数据精度等问题。