GDKOI 2025 ACM 赛事活动暨 2025 年中山大学程序设计邀请赛 解题报告

2025-12-31

目录

前言

预期难度：

$$C \approx H \ll L < G < D < F \ll A < E \approx J < M < K \ll B < I$$

C. 回文

解法

注意到 $f_i \le i$，那么让 i 往 f_i 连边，最后会形成一个森林，一次操作会将一个点向父亲走，把 x, y 变成相同数的最小操作次数实际上就是森林上两点间路径长度，如果 x, y 不在一棵树里就不合法。

实际实现可以把 $f_i = i$ 的点连向 0 号点减少代码复杂度，时间复杂度为 $O(n \log n + q \log n)$。

H. 基础概率练习题

解法一

分组总方案数是 $(2n+1)!!$，先选择一个 x，然后剩下每次找到未选出来的最大值，从剩下的 $2k-1$ 个球里面找一个和他放一组。

合法的总方案数是 $(2n)!!$，每次找到最大值，当前最大值一定是需要找到一个配对的，从剩下的 $2k$ 个球里找到一个和他放到一组。这样最后剩下的一个球自然就是 x。

因此答案就是 $\frac{(2n)!!}{(2n+1)!!} = \frac{1}{2n+1}$。

解法二

考虑容斥，钦定 i 组都 $< x$，方案是我们从 n 组中选出 i 组，然后 x 是这 $2i+1$ 个球中最大的，概率是 $\frac{1}{2i+1}$。

因此答案是 $\sum_{i=0}^{n} \frac{\binom{n}{i}(-1)^i}{2i+1}$。

可以证明两个答案是相等的（经典恒等式）。

L. 旅行

解法

考虑如果能通过一次操作在某个贮藏点放一瓶水并回到起点，那么一定可以通过有限次操作将其放满。

那么最终一定可以放满一个前缀的贮藏点。最终答案是从头开始往右走一遍，到某个位置再掉头走回来。

考虑如何维护，我们考虑记录一个 bag 表示当前的背包里有多少水，$back$ 表示从当前位置走回起点至少需要背包里留多少水。每次走一个贮藏点需要消耗 $dis = p_i - p_{i-1}$，回去也需要消耗 dis。能在这个贮藏点放水的条件是 $bag \geq 2dis + back + 1$。若到尽头或者无法抵达下一个贮藏点，就可以计算最后一波的时候能走到哪里。

实际上只需要维护 $bag - back$ 即可，单次扫描 $O(n)$。

G. 串符字炒翻

解法

观察到一个长度大于 1 的区间，如果有除了自己以外的 border，那么可以写成 sts（t 可能为空），$\mathrm{rev}(sts) = \mathrm{rev}(s)\ \mathrm{rev}(t)\ \mathrm{rev}(s)$，会计算重复。

因此合理猜测：只考虑划分成一些无除自己以外的 border 的区间能刚好计算所有情况（不翻转的考虑划分成一堆长度为 1 的区间）。

所有情况都会被统计到是显然的，接下来证明不会有两种方案得到的串相同。

假设两种方案：$s = s_1 s_2 ... s_n = t_1 t_2 ... t_m$，每个串都无 border，我们有 $\mathrm{rev}(s_1)\ \mathrm{rev}(s_2) ...\ \mathrm{rev}(s_n) = \mathrm{rev}(t_1)\ \mathrm{rev}(t_2) ...\ \mathrm{rev}(t_m)$。

找到第一个 $|s_i| \neq |t_i|$，不妨设 $|s_i| < |t_i|$，根据 $\mathrm{LCP}(s_i, t_i) = \mathrm{LCP}(\mathrm{rev}(s_i), \mathrm{rev}(t_i)) = |s_i|$，我们有 s_i 是 t_i 的 border，矛盾了。

因此这样数数是对的，接下来选择一个喜欢的能判断一个区间有没有 border 的算法做个 $O(n^2)$ 的 dp 即可。

D. 城市

解法一

选择一种你喜欢的哈希方式维护。这里提供一种冷门（?）的。

假设我们找到矩阵内随机两个数 w_a, w_b，我们可以通过对矩阵内函数 $f(x) = (w_x - w_a)(w_x - w_b) = w_x^2 - (w_a + w_b)w_x + w_a w_b$ 求和。前缀和维护即可，接下来问题是如何找到两个数 w_a, w_b。

一种方法是每个点维护向下/向右第一个不同的位置，先找第一列有没有两种颜色，如果没有，再找这一段区间向右能找到最近的位置，可以用 st 表做到 $O(n^2 \log n + q)$。

解法二

考虑扫描线，从左往右扫，对每个区间维护区间最近出现的三种颜色及其位置，用 st 表更新维护可以做到 $O(n^2 \log n + q)$，用线段树/树状数组之类的数据结构也可以做到 $O(n^2 \log n + q \log n)$。

没想到最后有一些卡常，不过没写快读的 std 跑了 1.2s，最后开了 3s，大部分合理做法也过了。出题人滑跪。

F. 【模板】幂函数

解法

实际上是构造一些置换环，最终循环节是这些置换环长度的最小公倍数，记这个值为 d。分两种情况讨论：

- 如果 $m_1 \neq 0$，那么需要满足一些类似 $d \mid k$ 的条件，取所有 m_1 个限制的 k 的 gcd，满足这些限制的充要条件是 $d \mid \gcd$，枚举它的约数，然后即可在 $O(m_2)$ 的时间复杂度判断 m_2 个限制是否满足。如果一个 d 满足条件，n 最小的构造方法是对 d 分解质因数，取每个质因子对应的 p^k 拼起来。
- 如果 $m_1 = 0$，注意到 $8 \times 3 \times 5 \times 7 \times 11 \times 13 = 120120$，因此答案 n 不超过 47，因此我们可以枚举 $n \leq 47$ 的有效情况。注意到仅保留有效状态不超过 1000 个，因此可以爆搜出所有情况，每次 $O(m_2)$ 判断即可。

复杂度 $O(m \cdot d(n))$ 或者 $O(m)$，其中 $d(n)$ 表示 n 的因子个数。

A. 贵校是构造王国吗 V

解法一

直接随机一些解，实测直接随能随 490 个点左右，加一些随机化或者手动随即构造一些解即可通过。

解法二

找一类解，最终方案数如果不是像 $a \times b \times c \times d$ 之类的，会具有一定的随机性。只要解密度够大，大部分解都能存在一种构造。

我们构造了一种解，大概长这样：

```
1      |
2    _ _
3   |   |
4 -       -
5   |   |
6    _ _
7      |
```

分成 12 块，每一块放一些点，实测可以在 $n \leq 15$ 内构造出所有 $k \leq 1000$ 的解。后来考虑到后面题目偏难，把 k 调到了 500。

可能有点随机区分了，出题人滑跪！

E. 尖尖的

解法

注意到 p 的字典序实际上是在 q 上从小到大填数，然后满足一个 W/N/M 状的波浪形拓扑序关系。因此我们的做法大概是枚举 lcp 和 lcp 后一位然后数确定一些位置的方案。

注意到一个关键性质：确定了一些位置之后，剩下的位置会被切成一些段，每个段之间是独立的。因此，我们每次枚举 lcp 和 lcp 下一位后，只需要根据每一段的大小求出内部方案，然后用组合数把每一段拼起来即可。

求内部方案考虑 dp：设 f_i 表示长度为 i 的 W 形排列的数量。转移枚举 1 号点填在哪里，然后把方案分成两段。

$$f_n = \sum_{i=1}^{n} f_{i-1} f_{n-i} \binom{n-1}{i-1} [i \bmod 2 = 0]$$

这部分复杂度是 $O(n^2)$。总复杂度 $O(n^3)$。

瓶颈是枚举 lcp 和 lcp 下一位后还要重新统计每一段的长度。这个很垃圾，我们可以考虑把重新统计变成“单点修改”，每次分裂一个段或者合并两个段，总复杂度就变成 $O(n^2)$。

解法

接下来考虑把“枚举 lcp 下一位“这个操作干掉。

注意到实际上是在一个已经挖掉一些位置的东西上，对一个前缀统计挖掉每个点的方案数的和。那么会是前面有一些段完全在前缀内，有一个段有一部分在这个前缀内。

对于完全在前缀内的段，由于每个段答案独立，可以对每个段统计内部挖掉一个点的方案，这个也是卷积，可以做 NTT 预处理，然后用一个数据结构维护。然后对于一部分被选中的段，注意到假设这个段长度是 k，被覆盖的部分是 j，那么这一次统计完后，就会分裂成两个段 j 和 $k-j$，因此我们可以采用启发式分裂的办法，用 $O(\min(j,k-j))$ 的复杂度，选择直接统计，或者用总数减去剩下的两款暴力中的一个即可，这部分复杂度是 $O(n\log n)$。

接下来考虑优化 dp 数组的预处理。

$$f_n=\sum_{i=1}^{n}f_{i-1}f_{n-i}\binom{n-1}{i-1}[i \bmod 2=0]$$

可以采用 LOJ575 不等关系 的容斥 + 分治 NTT 做法做到 $O(n\log^2 n)$；也可以直接使用半在线卷积优化上面的 dp 式子做到 $O(n\log^2 n)$，足以通过此题。

但是由于我们求的东西很特殊，因此我们还有更快的办法求出 f_n。

解法

法一：我们考虑沿用 LOJ575 的做法，对下降的限制进行容斥。如果 n 是偶数，那么会变成一些排列长度为偶数的上升段拼起来，这个的 EGF 是：

$$G = \frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!} - \ldots = 1 - \cos x$$

拼接之后得到：

$$F_0 = \frac{1}{1-G} = \sec x$$

n 是奇数时要卷上最后一段长度为奇数的段，那么：

$$F_1 = \sin x \times \sec x = \tan x$$

因此 f_n 的 EGF 就是

$$F = F_0 + F_1 = \frac{1 + \sin x}{\cos x}$$

采用多项式求逆即可做到 $O(n \log n)$。

法二：直接把式子写出微分方程：

$$\frac{2\mathrm{d}F}{\mathrm{d}x} = F^2 + 1$$

解出来：

$$2\arctan(F) = x + C$$

带入 $F(0) = 1$ 得 $C = \frac{\pi}{2}$，因此我们有：

$$F = \tan\left(\frac{x}{2} + \frac{\pi}{4}\right) = \frac{1 + \sin x}{\cos x}$$

最后总复杂度是 $O(n \log n)$。

不过本题实际上不需要会任何和生成函数有关的部分，仅需要半在线卷积即可通过。

J. 幸福指数

解法

考虑 Product Trick，对于每个点，选定一个指向的点作为贡献。

令 C_i 代表 i 号点的所有可能的取值，而 S_i 为向 i 号点连边的点集。对于每个点 i 计算出有 j 个点选定它作为贡献时，权值的期望，也就是 $\frac{\sum_{v\in C_i} v^j}{|C_i|}$。

考虑集合幂级数，设 f_i 为点 i 给一个集合贡献对应的集合幂级数。我们要求的就是 $(\prod f_i)_U$。

暴力 FWT 即可做到 $O(2^n n^3)$。

考虑优化，我们将原式写成 $\exp(\sum \ln f_i)$，注意到 $\ln f_{i,s}$ 只和 i 和 $|s\cap S_i|$ 有关，只有 $\operatorname{poly} n$ 种方案，对于每一种 $O(n^2)$ 暴力 $\ln$ / $\exp$ 即可。

复杂度 $O(2^n n^2)$。

M. 无限背包

问题其实就是在求解

$$[x^k]\prod_{i=1}^{n}\frac{1}{(1-w_ix)}$$

考虑对这个式子进行有理分解。例如，如果所有 w_i 只出现一次，能够说明存在一个长度为 n 的序列 A，满足：

$$\prod_{i=1}^{n}\frac{1}{(1-w_ix)}=\sum_{i=1}^{n}\frac{A_{i,1}}{1-w_ix}$$

如果某个 w_i 出现多次，例如 2 次，则把对应项换成：

$$\frac{A_{i,1}}{1-w_ix}+\frac{A_{i,2}}{(1-w_ix)^2}$$

解法

可以发现，在通过高斯消元预处理系数 A 之后，在每次询问中就可以分别对每个分式计算 x^k 的系数，相加即可得到答案。为了计算这个答案，可以观察到

$$[x^k](1-w_jx)^{-r}=\binom{k+r-1}{r-1}w_j^k$$

因此直接计算对应组合数即可。使用光速幂计算 w_j^k，即可做到 $O(n\sqrt{V}+nq+n^3)$ 的时间复杂度。

考虑怎么加快 $O(n^3)$ 高斯消元这个过程。考虑 w_j 出现了 k 次，对应的有理分解结果为

$$\frac{A_{j,1}}{(1-w_jx)^1}+\frac{A_{j,2}}{(1-w_jx)^2}+\cdots+\frac{A_{j,k}}{(1-w_jx)^k}$$

定义 $Q(x)=\prod_{w_i\neq w_j}(1-w_ix)$，那么有

$$\prod_{i=1}^{n}\frac{1}{(1-w_ix)}=\frac{P(x)}{Q(x)}+\frac{A_{j,1}}{(1-w_jx)^1}+\frac{A_{j,2}}{(1-w_jx)^2}+\cdots+\frac{A_{j,k}}{(1-w_jx)^k}$$

两边同时乘以 $(1-w_jx)^k$：

$$\frac{1}{Q(x)}=(1-w_jx)^k\frac{P(x)}{Q(x)}+A_{j,1}(1-w_jx)^{k-1}+A_{j,2}(1-w_jx)^{k-2}+\cdots+A_{j,k}$$

令 $t = 1 - w_j x$，那么 $x = \frac{1-t}{w_j}$。带入可得

$$\frac{1}{Q(\frac{1-t}{w_j})} = t^k \frac{P(\frac{1-t}{w_j})}{Q(\frac{1-t}{w_j})} + A_{j,1} t^{k-1} + A_{j,2} t^{k-2} + \cdots + A_{j,k}$$

那么只需要计算出 $\frac{1}{Q(\frac{1-t}{w_j})}$，并**只取结果的低 k 位**，就能够得到所需的参数了。随后可以发现：

$$Q^{-1}\left(\frac{1-t}{w_j}\right) = \prod_{w_i \neq w_j} (1 - w_i \times \frac{1-t}{w_j})^{-1} = w_j^{n-k} \prod_{w_i \neq w_j} (w_j - w_i(1-t))^{-1}$$

一次函数的逆直接使用组合数计算即可，而乘法也可以暴力计算。这部分的时间复杂度就是 $O(n^2)$，过程中需要时刻注意函数的截断。

因此总时间复杂度为 $O(n^2 + nq + n\sqrt{V})$。

K. 线性的

解法

为了避免歧义，本文中排列表示大小为 m 的元素，序列表示“线性的”的定义中操作过程的每个排列按照被表示出来的先后顺序构成的序列。

先考虑对于一个排列的集合 S，如何判断其是否是“线性的”。

定义关于排列的二元函数 $dis(p,q)$，表示以 p 为开始，每次交换相邻两个元素，至少要几次变成 q。这个东西相当于“相对的逆序对”，单次可以用树状数组做到 $O(m\log m)$。

接下来考虑如下事实：根据上述方案可以构造出从 p_1 到 p_2 的最优交换序列，以及 p_2 到 p_3 的最优交换序列，以此类推。根据“线性”对应的要求，交换过程中不会出现两个值 x,y 被交换至少两次（这说明 x 和 y 的相对位置关系在排列序列中切换了至少两次），因此将整个交换序列拼接后形成的就应该是 p_1 到 p_n 的最优交换序列。

可以得到如下充要条件：集合 S 是“线性的”，当且仅当存在一个排序方式，满足 $dis(p_1,p_n)=\sum_{i=1}^{n-1}dis(p_i,p_{i+1})$。

接下来考虑如何给这些排列排序，注意到 $dis(p_1,p_i)$ 应当递增，所以我们把排列按照 $dis(p_1,p_i)$ 排序然后检测一遍即可。

接下来考虑如何确定 p_1，注意到对于一个 p_i，$\max_j dis(p_j,p_i)=\max(dis(p_1,p_i),dis(p_n,p_i))$，类似直径的性质，随便找一个排列 p_i，找到距离它最远的排列，就确定了这个序列的一端。这样我们成功在 $O(nm(\log n+\log m))$ 即 $O(nm\log nm)$ 的时间内完成了一次询问。其中 $O(\log m)$ 对应求 dis 的复杂度，$O(\log n)$ 对应排序的复杂度。只有插入的数据可以二分 check 变成离线问题，但是多个 log。

接下来考虑只有插入的时候能否少个 log。考虑持续维护这个序列，每次插入一个排列就把他往这个序列里插，用个支持插入的数据结构维护一下，如果插不了就 No 了。我们往 p_k,p_{k+1} 中间插入了一个新的 p_{new}，那么有 $dis(p_j,p_{new})$ 是单谷函数，并且有 $\min_j dis(p_j,p_{new})=\min(dis(p_k,p_{new}),dis(p_{k+1},p_{new}))$，因此我们可以三分找到谷底并检查相邻两个位置是否满足要求即可。

解法

接下来考虑删除操作。注意到还有一个偏序性质：从“线性的”集合中删除一些排列，依然是“线性的”；向非“线性的”集合中加入一些排列，依然是非“线性的”。考虑“滞后操作”，每次插入，如果能插入，就插入，否则就标记 No，然后后面的插入操作直接罢工，使用一个待处理序列把这些插入存起来，我们把这个存起来的东西叫做“罢工队列”。如果后面删除了一个在罢工队列的排列，就直接踢掉，否则需要重新启用罢工队列，重新插入，如果罢工队列清空了，那么就是 Yes 了，否则到某个时刻接着罢工。这样每个元素只会被成功插入一次，不成功插入只会在每次删除时触发最多一次，这样的事件复杂度时 $O(nm \log n \log nm)$，这里是 $O(\log n \log nm)$ 的原因是三分的时候每次要在数据结构上拿个排列出来并且求 dis 是嵌套的。

因为性质比较好，直接在数据结构上维护一些东西也能做到这个复杂度，或者 $O(nm \log n \log m)$。

解法

那么就有人问了：主播主播有什么办法把这两个 log 变成一个 log 吗？有的兄弟有的。

接下来考虑假设有一个排列 p_ϵ 永远也不会删除。我们可以把所有排列 p 按照在序列中是在 p_ϵ 的左侧还是右侧分成两波，两波都相当于有了个 p_1，然后它们的 $dis(p_1,p_i)$ 就应该递增。我们称这个 p_ϵ 为基准点。因此有一个做法，是以 p_ϵ 为“基准点”，每次插入一个排列的时候往两波都分别插入一下，如果能插入就插入，不能插入就相当于 No。然后我们判断一下会不会有“人”的情况出现即可（即 $dis(p_l,p_\epsilon)+dis(p_\epsilon,p_r)=dis(p_l,p_r)$，其中 p_l,p_r 分别为两波中的任意一个排列）。这个基准点强势在我们把顺序拍成了一个数 $dis(p_\epsilon,p_i)$，然后拿着这个丢到数据结构里定位，然后 check。这样就做到了 $O(n\log n+nm\log m)$。

但是如果上面的“基准点”p_ϵ 被删了就倒闭了。对此我们有两个解决方案：

解法

1. 考虑每次删掉基准点时直接暴力重构，每次选择当前留下来的排列中最晚被删的作为“基准点”。这个复杂度是对的，因为每个排列最多参与一次重构。
2. 如果强制在线，考虑每次删掉基准点时直接暴力重构，每次随机选择一个排列作为“基准点”。这个复杂度是对的。假设每个点有个删除时间 t_i，如果不删除就设为 $+\infty$。每次选择一个“基准点”p_ϵ，会把所有 $t_i < t_\epsilon$ 的 i 干掉，在下一次重构时删掉，每次期望干掉一半的点。也就是说，一个序列不断重构的期望复杂度是 $O(n)$，因此总复杂度仍为 $O(n \log n + nm \log m)$。
3. 离线线分治，这样删掉“基准点”时已经删空了，复杂度多个 $\log$。

注意到就算没有插入删除值判断一次也要这个复杂度，因为需要排序和求逆序对，因此这已经是基于逆序对判断做法的最优复杂度了。

这个题还有一些神秘做法，比如 $O(n^2 m)$ 和 $O(nm^2)$ 拼在一起的根号做法，比如加入一些剪枝的线分治做法，比如一些随机做法，可能可以通过。

B. 嘟南玻行山

解法

容易把 u, v 的贡献放在 LCA 上。

考虑树剖，维护 LCA 在每条重链上的答案。对于重链的每个区间，维护三个凸包 T_1, T_2, T_3，分别表示这段区间中：当前重链上所有一次函数构成的凸包，当前重链所有轻儿子子树的一次函数，答案的凸包。那么我们需要线段树合并：

$$T_1 = T_1' + T_1''\ T_2' = T_2' + T_2''\ T_3 = T_3' + T_3'' + T_2' \times (T_1'' + T_2'')$$

其中 $+, \times$ 分别表示凸包合并和闵可夫斯基和。

直接维护是 $O(n \log^3 n)$，采用全局平衡二叉树维护即可做到 $O(n \log^2 n)$。

注意到这个东西叫做 Top Tree，可以封装实现 R 和 C 两种操作减少代码复杂度。

将凸包序列处理为斜率的拐点序列之后，使用分散层叠可以做到 $O(n \log n)$。

I. 基础卷积练习题

解法

考虑将 f 和 g 序列分块，每块包含 B 个元素，共有 $K = \lceil \frac{n+1}{B} \rceil$ 个部分。两个序列进行卷积可以据此拆分为 K^2 个块。考虑两个长度为 B 的序列进行卷积，为了求某一个指数下的系数，可以对 $2B$ 个值进行插值，并且将结果根据固定系数进行线性组合。这样就能通过维护插值结果实现删除和增加一个块的贡献。

考虑没有修改操作对应的答案。在分块后，对所有 K^2 个块进行插值，并得到每个指数下的系数。对于每个参数 $0 \le k \le 2(K-1)$，考虑对所有 $0 \le i \le k$，将 f 的第 i 块和 g 的第 $k-i$ 块的卷积结果相加，就能得到卷积后一条斜线上的块的卷积和，称为第 k 组卷积和。在计算答案的时候，只需要根据当前的变量 d 找到相邻两组卷积和，即可计算出系数。

解法

接下来考虑修改。为所有 K^2 个块上添加懒标记，在某一个位置发生变化时，将所有受这一位置影响的块打上标记，并从对应的卷积和组中减去其贡献。在查询时，则需要枚举卷积和组内的所有块，将所有被标记的块重算，并加入到结果中。

设 $B = \sqrt{n}$，可以证明上述算法中需要重算的次数不超过 $O(n)$。简单来说，即使变量 d 可以通过不断自增自减不断对相邻三个斜线进行查询，但是除了第一次查询可能需要重算 $O(\sqrt{n})$ 个块之外，后续的重算次数是均摊 $O(1)$ 的。另外，为了前往下一组斜线需要 $\sqrt{n}$ 步，因此可以得到重算次数不超过 $O(n)$ 次。在使用连续点值进行插值，适当预处理插值相关的系数之后，即可在 $O(n\sqrt{n})$ 的时间复杂度解决这个问题。