2025 年中国大学生程序设计竞赛全国邀请赛（郑州）暨第七届 CCPC 河南省大学生程序设计竞赛 题目解析

BUAA 命题组

2025 年 6 月 2 日

预期难度

- Easy: DJ
- Easy-Medium: FHM
- Medium: EGK
- Medium-Hard: BC
- Hard: IL
- Very-Hard: A

Toxel 与独一无二的序列

注意到操作时可以替换成 1 到 n 的各种数。每次操作时一定会将区间中的数替换成一些当前不存在的数，使得该区间以后都可以继续操作。如果有一个操作和之前的操作区间重合，那么可以将它扩展到之前操作区间的边界，这不影响答案和操作结果。因此，所有的操作可构成一棵树，这样就可以自然地想到区间 dp。

Toxel 与独一无二的序列

设 $f_{l,r}$ 表示将 l 到 r 中所有数变成互不相同的最小操作次数，答案是 $f_{1,n}$。考虑如何转移到 $f_{l,r}$，只需要枚举最外层操作了哪些区间，这些区间一定是互不相交的，且它们的 f 值需要加上，因为只有互不相同了才能操作。这些区间没有覆盖到的剩余位置不用操作。使 $[l,r]$ 变成互不相同的充要条件是这些不用操作的位置互不相同（操作了的地方可以变成任意的数）。因此，可以考虑一个内层的状态压缩 dp，令 $g_{l,r,S}$ 表示 l 到 r 中有 S 这个集合的数没被操作时的最小操作次数。转移时枚举当前位置是不操作的，或者枚举操作区间的长度。因此转移时有一个 $\mathcal{O}(n)$ 的复杂度。

Toxel 与独一无二的序列

转移方程大意是（省略了初值或边界）：

$$f_{l,r} = \min\{g_{l,r,S}|S\}$$

$$\mathrm{g}_{l,r,S} = \min\left(\{g_{l,i-1,S} + f_{i,r} + 1 | l < i \le r\} \cup \{g_{l,r-1,S-\{a_r\}}\}\right)$$

考虑到总状态数为 $\mathcal{O}(2^n n^2)$，转移的复杂度为 $\mathcal{O}(n)$，总复杂度为 $\mathcal{O}(2^n n^3)$，不能通过本题。

Toxel 与独一无二的序列

注意到很多状态是无用的，我们并不需要记录所有 n 个数是否都在 S 中。考虑以下两个性质：

1. 若一个数没有在 $a_l, \ldots, a_r$ 中出现，则不需要在 $g_{l,r,S}$ 中记录它；
2. 若一个数没有在 $a_{r+1}, \ldots, a_n$ 中出现，则也不需要在 $g_{l,r,S}$ 中记录它，因为之后不论怎么转移这个数都不会再有影响了。

利用上面这两个性质，我们会发现 $g_{l,r,S}$ 中的 S 至多只需要记录 $2^{\min(r-l,n-r)}$ 个状态。可以证明，总状态数已经降低到了 $2^{n/2}$。读者可以自行简单计算。

这样一来，乘上转移的复杂度仅为 $\mathcal{O}(2^{n/2}n)$。

随机栈 II

设 $dp_{i,j,k}$ 表示目前处理了前 i 个操作，最后一次取出的数为 j，且 j 最后连续出现了 k 个，取出的序列有序的概率。那么我们有转移方程（仅对于取出操作，cnt_u 表示到当次取出前 u 出现的次数，p 表示到当次取出前集合的大小）：

$$dp_{i,j,k} \leftarrow \sum_{u=1}^{j-1} \sum_{v=1}^{\mathsf{cnt}_u} dp_{i-1,u,v} \cdot \frac{\mathsf{cnt}_j}{p}$$

$$dp_{i,j,k} \leftarrow dp_{i-1,j,k-1} \cdot \frac{\mathsf{cnt}_j - (k-1)}{p}, \quad \text{if } 1 < k \leq cnt_j$$

随机栈 II

对于第一种转移，我们可以对 u 维护一个前缀和来实现 $O(1)$ 转移。第二种转移则显然是 $O(1)$ 的。而我们又注意到对于某一个 i 来说，不同的 j, k 数量至多就是当前的插入集合操作次数。因此有用的总状态数是 $O(n^2)$ 的。从而本题的总复杂度即为 $O(n^2)$。

在实现本题时，可能需要注意空间复杂度。如果使用滚动数组，并且将 j, k 两维拍平成一维，可以做到 $O(n)$ 的空间复杂度。

另外，由于出题人比较友善，除法如果使用了快速幂求逆元也可以通过（复杂度为 $\mathcal{O}(n^2 \log n)$）。预处理出 $1 \sim n$ 的逆元即可规避这一问题。

Toxel 与宝可梦图鉴

我们考虑用珂朵莉树维护所有连续的区间。每次修改操作时，使用 `set.lower_bound()` 查找出第一个被修改的区间，并依次连续处理后续被修改的区间。需要注意开头和结尾的两个区间有可能有一部分需要放回 `set`。随后把修改的区间也放入 `set` 中。

接下来，用一棵权值线段树维护每个值出现的次数。每次修改区间时，只需要支持区间加操作即可。最终的答案只需要线段树求区间最小值。

注意到每次操作至多产生 3 个新区间，所以总复杂度均摊为 $\mathcal{O}(n \log n)$。

2025

签到题，只需要按照题意实现即可。需要注意 `sqrt` 的结果为 double，应该转换为 `int` 类型再做后续判断。

双生魔咒

考虑任意一个字符串 S，设其在所有题目给定的 $2n$ 个串中作为前缀出现了 m 次，那么，由基本不等式得知，其最大贡献次数将会是 $\left\lfloor \frac{m}{2} \right\rfloor \times \left\lfloor \frac{m+1}{2} \right\rfloor$ 次。

又对于任意的 S，其不包括自身的 $|S| - 1$ 个前缀也一定会包含这些贡献次数，所以我们不妨记其每次对总答案的贡献为 1。

接下来，我们要证明可以取到这样的最大贡献。

双生魔咒

考虑将题目给定的 $2n$ 个串，将其按字典序排序后，按奇偶分到「引」与「根」两类去。可以发现，对于任意字符串 S，其作为前缀出现的字符串在排序之后一定是连续排列的。所以，恰好会有 $\left\lfloor \frac{m}{2} \right\rfloor$ 和 $\left\lfloor \frac{m+1}{2} \right\rfloor$ 个串分别分到「引」与「根」(或是反之)，这就证明了最大贡献的成立。

双生魔咒

那么，只需要考虑会有贡献的 S 即可。可以发现，每个有贡献的 S 都会对应到 trie 树上的一个点，因此将字符串加到 trie 树上，遍历全树，将每个点（对经过每个点的字符串数记作 m ）的贡献 $\left\lfloor \frac{m}{2} \right\rfloor \times \left\lfloor \frac{m+1}{2} \right\rfloor$ 加到答案里即可。时间复杂度 $\mathcal{O}(\sum |S|)$ 。

P.S. 也可以考虑排序后使用 SA 的 height 数组求解答案。

幻形之路

首先，可以通过 BFS/DFS 分别求出起点/终点在不经过障碍的前提下，可以到达哪些点，称这些点为起点/终点可达点。然后，对每个点分别求出到最近的起点/终点可达点的距离（不考虑障碍，所有格子都可以走），这一步可以通过多源最短路实现。最后，所有点之中，到最近的起点可达点距离 $+$ 到最近的终点可达点距离 -1 的最小值即为答案。注意若起点可以不经过障碍直接到达终点，那么路径上的点既是起点可达点也是终点可达点，因此求出来是 -1，还需和 0 取 $\max$。

时间复杂度取决于多源最短路的复杂度，由于边权只有 1，因此可以用 BFS 做到 $\mathcal{O}(nm)$，当然 Dijkstra 也可以通过。

直径与最大独立集

$n \leq 4$ 的情况是平凡的。

当 $n > 4$ 时，一定有解。构造的方法有很多，可以通过打表观察规律或者手玩出来。下面给出一种可行的构造。大体思路是，构造一条链加上一个菊花图的形状：

令 $t = \lfloor \frac{n+2}{3} \rfloor + 2$。

当 $n \bmod 3 \neq 1$ 时：

$\forall i \in [2, t]$，与 1 连边；$\forall i \in [t, n)$，与 $i+1$ 连边。

当 $n \bmod 3 = 1$ 时：

$\forall i \in [2, t]$，与 1 连边；$\forall i \in [t, n-1)$，与 $i+1$ 连边；3 与 n 连边。

树论函数

注意到，有如下等式：

$$n(n+1) = \frac{(n^2+2n)(n^2+2n+1)}{(n+1)(n+2)}$$

即节点 n 始终与 $n+1$ 联通，故所有结点均联通。

直接输出 $r-l+1$ 即可。

大多数队伍会通过「打表大法」获得该结论。

有的兄弟，有的

为了方便描述，我们将题意转换为“染色问题”。

首先就是判断出无解的情况，不难发现就两种情况：

- 出现了原本不存在的颜色；
- 每个颜色在 a 中仅出现一次，但是 b 不等于 a；

随后只需构造有解的情况。

有的兄弟，有的

先回想一下，借助“中间变量”tmp 交换变量的值的思路：int tmp = a; a = b; b = tmp;

于是我们可以花 $\mathcal{O}(1)$ 步找到一个“中转站”，让它持续当做这个 tmp 变量。

但是仅靠上述的一级结论，这道题依旧很难写，需要讨论维护的东西很多，容易出现想到一级结论就上机冲，导致疯狂 WA 的情况。

有的兄弟，有的

所以需要进一步挖掘二级结论，来简化代码的书写。让 b 中每种颜色都保留至少一个“种源”，做法很多，但编写时也有不少坑。一个相对简单的构造方法是：

- 第一阶段：让 b 中每一种颜色都至少一个位置正确
- 第二阶段：让“种源”扩散到 b 中对应的位置

现在问题变成第一阶段中，选择哪个位置当做“中转站”，考虑颜色至少一个“种源”，至多五六步就可以弄成一个始终为中转站的位置。

有的兄弟，有的

本题中还有一个辅助编码的技巧：因为本题的 SPJ 检验器非常好写。于是选手可以输入的时候随机点数据，然后最后模拟看 a 是否等于 b，即可避免很多无效的 WA。

可能会有选手觉得 $3n + \mathcal{O}(1)$ 的操作次数还是很多，还有操作次数更少的算法吗？

出题人的回答当然是：有的兄弟，有的。

有的兄弟，有的

本题最优的算法是 $1.5n+1$ 次操作的基环树做法，但考虑到现场参赛选手的平均水平，最终出成 $3n+\mathcal{O}(1)$ 次的。

在每个颜色中选一个代表点，然后让位置 i，指向「颜色 b_i 的代表点位置」，就构造出一个基环树森林。

每次找一个度为 0 的点当中间变量，先记录一个环上第一个点的颜色，然后整个环就可以转一圈。之后环的外围树形部分，就可以从内部向外扩散。

总操作次数 $=$ 点数 $+$ 环数 $+$ 最后把中间点改回去次数，全为二元环的时候最坏，为 $1.5n+1$ 次操作。

Ring Trick

签到题，依照题意模拟即可。

Ring Trick II

朴素做法枚举 $k \in [0, m-1]$，计算每个移位后序列的洞数和，复杂度过高。我们可以优化：设 $\mathrm{freq}[x]$ 为 x 在原序列的出现次数，预处理 $\mathrm{holes}[x]$ 为数字 x 的洞数。移位 k 后的总洞数可以表示为：

$$\mathrm{TotalHoles}(k) = \sum_{v=0}^{m-1} \mathrm{freq}[(v - k + m) \pmod m] \cdot \mathrm{holes}[v]$$

这个求和式本质上是一个卷积。稍作变换或许更容易看出这一点：将 freq 反转，并把 holes 自身重复延长一倍。$\mathrm{TotalHoles}(k)$ 就可以利用这两者的卷积得到。

Ring Trick II

循环卷积可以使用 FFT 或者 NTT 在 $\mathcal{O}(N \log N)$ 时间内高效计算。总时间复杂度优化为 $\mathcal{O}(n + m \log m)$。

如果不会卷积科技，这道题验题人还给出了拆位 + 树状数组的做法，复杂度是 $\mathcal{O}(m \log^2 m)$，由于常数小实际上速度跟卷积差不多，也可以通过此题。

Astral Decay

关键观察：最优的 B 和 C 都在凸包上。让我们用调整法证明：假定固定 A 和 B，那么 C 是在 $\overrightarrow{AB}$ 投影方向上最靠近负侧的点，因此 C 应该在所有点构成的凸包上；同理，B 也应该在凸包上。

因此，我们考虑枚举 A，此后在凸包上按顺序枚举 B，那么最优的 C 显然也是在凸包上顺次移动的，可以用双指针来维护，总复杂度是 $O(n^2)$。

需要注意，当 A 也在凸包上时的一些细节。一种处理方法是：如果 A 在凸包上，枚举 B 时就不要枚举 A 了，从 A 的下一个点开始枚举 B，转回来到 A 的上一个点结束枚举。

川陀航空学院

一个 n 个点的树有且仅有 $n-1$ 条边，假设原图中联通块个数为 t，那么首先需要将每个连通块变为树，这里需要 $m-(n-t)$ 次，再加边将连通块联通，这里需要 $t-1$ 次，总次数即为 $m-n+2t-1$ 次。

计算连通块个数可以使用并查集或 DFS 等多种方法，时间复杂度 $\mathcal{O}(n\alpha(n))$ 或 $\mathcal{O}(n)$，均可通过。