

Kruskal Cup 2026 Solution

中国科学技术大学新生程序设计竞赛

USTC 计算机编程俱乐部 & USTC 命题组

2026.9.13

University of Science and Technology of China

Outline

Summary	2	Solution K	34
Statistics	3	Solution L	39
Interesting Facts	8	Solution M	45
Solutions	9	Acknowledgments	49
Solution A	10	Thanks	50
Solution B	11		
Solution C	13		
Solution D	17		
Solution E	19		
Solution F	20		
Solution G & H	21		
Solution I	23		
Solution J	29		

Outline

Summary	2	Solution K	34
Statistics	3	Solution L	39
Interesting Facts	8	Solution M	45
Solutions	9	Acknowledgments	49
Solution A	10	Thanks	50
Solution B	11		
Solution C	13		
Solution D	17		
Solution E	19		
Solution F	20		
Solution G & H	21		
Solution I	23		
Solution J	29		

Statistics

本次比赛总通过率为 343/1697。

- A. Delete and Backspace. 通过率为 97/271，线下一血为 **黄子萌**。
- B. Vanishing Inversions. 通过率为 75/133，线下一血为 **黄普**。
- C. Range Minimization. 通过率为 29/363，线下一血为 **王凤池**。
- D. Corridor Tuning. 通过率为 22/332，线下一血为 **洪文杰**。
- E. Punctured Neighborhood. 通过率为 33/159，线下一血为 **华天和**。

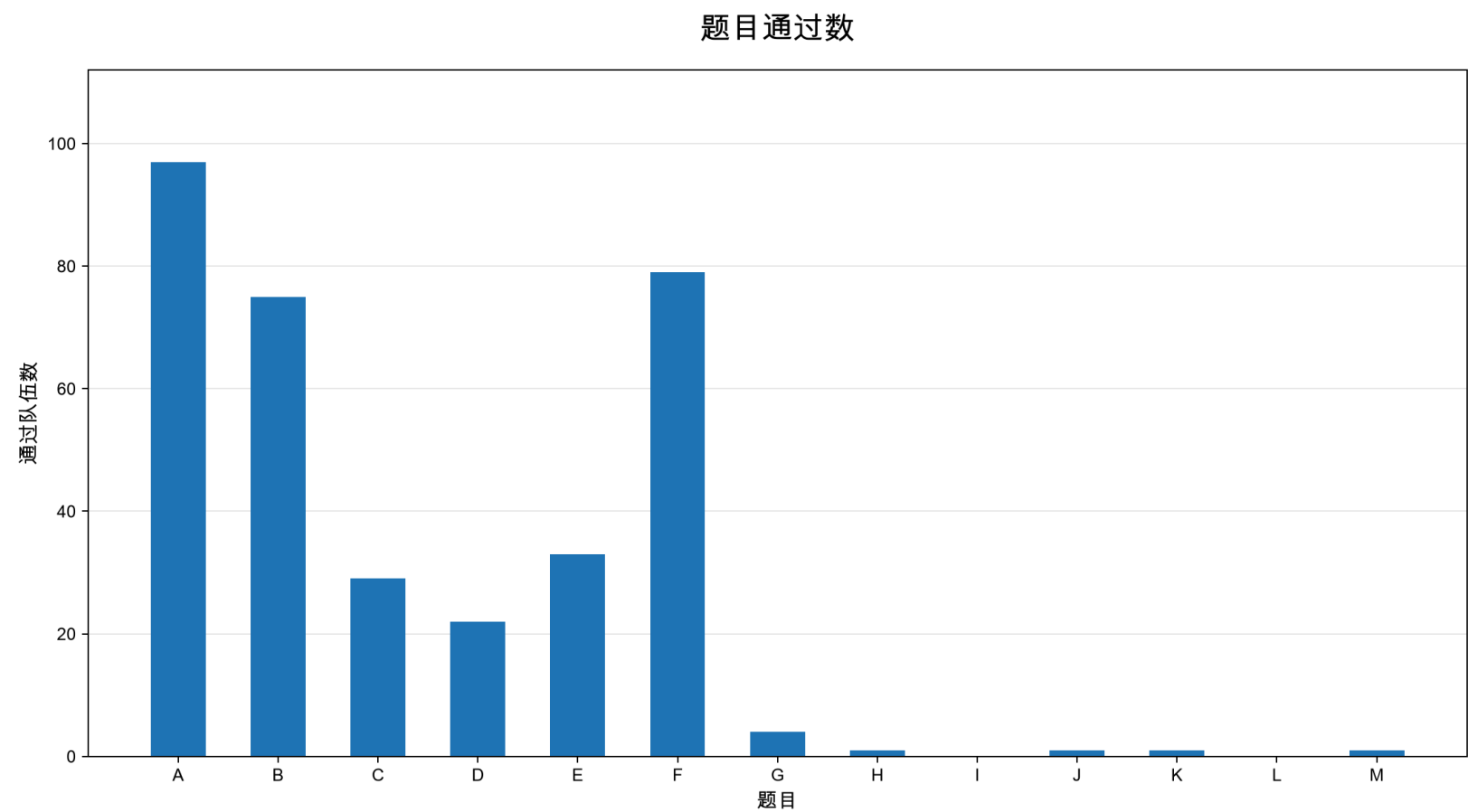
Statistics

- F. A Simple Construction Problem. 通过率为 79/276，线下一血为 **朱皓辰**。
- G. Nailoong vs. Bombloong 3.1. 通过率为 4/83，线下一血为 **洪文杰**。
- H. Nailoong vs. Bombloong 3.2. 通过率为 1/10，线下一血为 **洪文杰**。
- I. Same Frequency Echo. 通过率为 0/6，无人通过。
- J. Emordnilap. 通过率为 1/42. 线下一血为 **王凤池**。

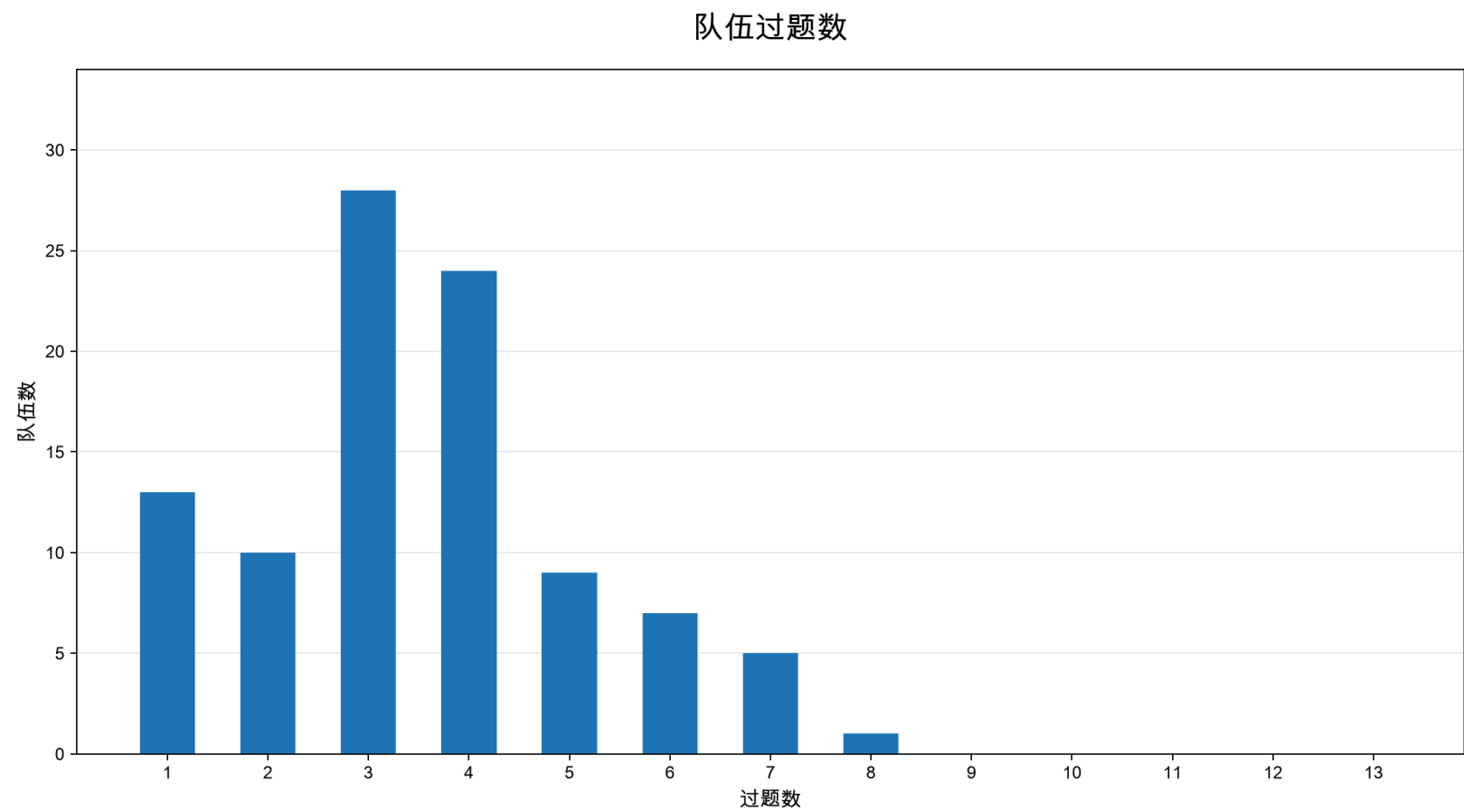
Statistics

- K. Card Packs. 通过率为 $1/6$ ，线下一血为 **林国盛**。
- L. AC Automaton. 通过率为 $0/3$ ，无人通过。
- M. Iterated Fixed Points. 通过率 $1/13$ ，线下一血为 **纪孝炎**。
-
- 顽强拼搏：最后一发有效的 AC 提交来自 **刘子诺**，于 17:29:26 递交了 G 题。

Statistics



Statistics



Interesting Facts

- 出题组认为 F 的通过率应该显著高于 B 题，但最后通过 F 的人数和通过 B 题的人数是接近的。
- 出题组认为 G 的通过率不会很低，但可能是因为过长的题面和通信题，最后只有 4 个人通过了这个题
- 选手 **蔡昀熹**，**陈溪** 是封榜后通过题目数量最多的选手，分别从 5 题上升到 7 题，从 3 题上升到 5 题。
- 选手 **吴一鸣** 是场上唯一一个 E 题做到 5 询的。

Outline

Summary	2	Solution K	34
Statistics	3	Solution L	39
Interesting Facts	8	Solution M	45
Solutions	9	Acknowledgments	49
Solution A	10	Thanks	50
Solution B	11		
Solution C	13		
Solution D	17		
Solution E	19		
Solution F	20		
Solution G & H	21		
Solution I	23		
Solution J	29		

Solution A

设第 k 次的答案为 b_k 。

考虑第 k 次操作，若我们想取到第 a_i ($i \leq k$)，则只需要先从后缀取 $k - i$ 个元素。若此时还没有取出 a_i ，则继续从前缀取 i 个元素即可。

因此，第 k 次操作能够取到的元素一定来自前 k 个元素或后 k 个元素，故

$$b_k = \max(a_1, a_2, \dots, a_k, a_{n-k+1}, \dots, a_n).$$

因此我们只需要一边输出答案，一边维护当前最大值即可。具体地，有

$$b_k = \max(b_{k-1}, a_k, a_{n+1-k}).$$

总时间复杂度为 $O(n)$ 。

Solution B

我们设排列 a 的逆序对个数为 I ，数组长度为 m ，起初 $m = n$ ，我们考虑 $P = I + \lfloor \frac{m}{2} \rfloor \bmod 2$ 。

如果进行操作一，只有这一对元素的逆序关系发生改变，即 $I \rightarrow I - 1$ ，而排列长度不变，则 P 翻转一次。

如果进行操作二，考虑一个没有被删除的元素 y ，设被删去的元素是 x 和 $x + 1$ ，则只有可能 $x < x + 1 < y$ 或者 $y < x < x + 1$ ，则删去这两个元素之后逆序对的变化量一定是偶数，而 $m \rightarrow m - 2$ ，则 $\lfloor \frac{m}{2} \rfloor \rightarrow \lfloor \frac{m}{2} \rfloor - 1$ ， P 也翻转一次。

因此无论操作一还是操作二，都会使 P 发生翻转。考虑结束的状态，结束时一定没有逆序对，否则还能进行操作一。故排列只能是 $1, 2, \dots, m$ ，若 $m \geq 2$ ，则还能进行操作二。因此结束时排列只能为空或者为 1，这两种情况 $P = 0$ 。

Solution B

故初始时 $P = 1$ ，则 Alice 获胜。在本题的数据范围内，我们可以暴力求出逆序对，时间复杂度为 $O(n^2)$ ，最后直接计算即可。

事实上本题可以存在 $O(n \log n)$ 的做法（用树状数组统计逆序对）和 $O(n)$ 解法：

考虑把排列拆成 c 个环，则经过 $n - c$ 次交换后，排列可以恢复到恒等排列。

而排列中任意交换两个元素，都会使得逆序对奇偶性翻转，则 $I \equiv n - c \pmod{2}$ ，即 $P = n + \lfloor \frac{n}{2} \rfloor - c \pmod{2}$ ， c 可以 $O(n)$ 算出。

注：在审题中，我们发现了一个更简单的视角，我们只需要直接考虑顺序对即可，在 $\text{mod } 2$ 意义下，操作一会使得顺序对个数 $+1$ ，操作二会使得顺序对个数 -1 ，一样会进行翻转。

Solution C

对 n 进行分类讨论：

$n = 1$ 时，显然答案是 0。

$n = 2$ 时，我们考虑 $d = |a - b|$ ，则答案为 $\min(d, 4 - d)$ 。这也是显然的。

$n \geq 3$ 时，结论如下： $\sum a_i$ 为奇数且 n 为偶数时答案为 1，否则答案为 0。

为证明 $n \geq 3$ 的结论，我们引入以下引理：

引理 1：对任意长度 n ($n \geq 2$) 的数组，存在一组操作使得数组元素整体 +2。

引理 2：存在一组操作，使得 $n \geq 2$ 时能完成“等效”的“-3 +1”操作。

引理 3：存在一组操作，使得 $n \geq 2$ 时能完成“等效”的“-1 +1”操作。

Solution C

引理 1 的证明：将所有下标分成 2 个或 3 个一组，然后按下图方式操作即可。对任意 $n \geq 2$ 显然都可分组。



引理 2 的证明：引理 1 完成的数组元素整体 $+2$ 的操作不影响极差，那么在引理 1 操作的基础上“撤销”一个“ $+3 -1$ ”的操作即可将这组操作等效成一个“ $-3 +1$ ”操作。这个等效“ $-3 +1$ ”操作可以对任意 (i, j) 成功构造，因为总能把这对 (i, j) 放在引理 1 构造中的某个分组里。

Solution C

引理 3 的证明：由引理 2，对 (i, j) 操作 “+3 -1” 加上对 (i, k) 等效操作 “-3 +1” 即为对 (j, k) 等效操作 “-1 +1”。

原结论的证明：

题目原操作 “+3 -1” 会让 $\sum a_i$ 增大 2，所以 $\sum a_i$ 的奇偶性不变。

只需调整 $\sum a_i$ 为 n 的倍数，即可通过引理 3 的等效 “-1 +1” 操作使得所有元素相等，极差为 0。对于 n 奇 $\sum a_i$ 偶、 n 奇 $\sum a_i$ 奇、 n 偶 $\sum a_i$ 偶 这三种情况，都能做到。

那么唯独对于 n 偶 $\sum a_i$ 奇 这种情况无法做到极差为 0。

我们可以将其视为 n 偶 $\sum a_i + 1$ 偶 这种情况，所以我们总能做到极差为 1，证毕。

Solution C

$n \geq 3$ 的其他处理思路（更符合实践，来自久洵）

为了便于思考，我们可以在执行“+3 -1”操作时，把“+3”全部留给 $a_{\max}$ ，利用“-1”操作把除 $a_{\max}$ 外其他数全部弄相等。此时数组变成 1 个 $a_{\max}$ 和 $n-1$ 个 $a_{\min}$ 。

这时候可讨论 $c = a_{\max} - a_{\min}$ 的奇偶性。

若 c 为偶数，由引理 1 对 $n-1$ 个 $a_{\min}$ 不停 +2 即可做到极差为 0。

若 c 为奇数，则可尝试对 $n-1$ 个 $a_{\min}$ 做“-1”操作，对 $a_{\max}$ 做 $n-1$ 次“+3”操作。此时 c 会变化 $1 + 3 \times (n - 1) = 3 * n - 2$ 。当 n 为奇数时能让 c 变为偶数，此时可做到极差为 0。否则无法改变 c 的奇偶性，只能做到极差为 1。

Solution D

令 $c_i = b_i - a_i$ ，问题等价于：从全零序列出发，通过若干次前缀加和后缀加得到 c 。再对 c 做差分： $g_1 = c_1, g_i = c_i - c_{i-1} (2 \leq i \leq n)$ 。

考虑边界 i ，其中 $2 \leq i \leq n$ 。

- 给后缀 $[i, n]$ 加上 x ，只会令 g_i 增加 x ；
- 给前缀 $[1, i-1]$ 加上 x ，只会令 g_1 增加 x ，同时令 g_i 减少 x ；
- 给整个数组加上 x ，只会令 g_1 增加 x 。

因此，一次非整段操作只会影响 $g_2, g_3, \dots, g_n$ 中的一个位置。

令 $m = \#\{i \mid 2 \leq i \leq n, g_i \neq 0\}$ 。所以答案至少为 m 。

Solution D

恰好使用 m 次操作的充要条件是： $\exists S \subset \{2, 3, \dots, n\}, \sum_{i \in S} g_i = -g_1$.

若不存在这样的子集， m 次操作不可能完成；但 $m + 1$ 次操作一定足够。因此答案只可能是 m 或 $m + 1$ 。

题目保证 $\sum_{i=2}^n |g_i| \leq 20000$ 直接背包可过。还可使用 bitset 来优化维护 0-1 背包。设 bitset 宽度为 $W = 40001$ ，本测试用例中非零的 g_i 数量为 $m \leq 20000$ 。

- 构造差分： $O(n)$ ；
- 每次 bitset 转移约为 $O(\frac{W}{64})$ ；
- 总时间复杂度为 $O(n + mW/64)$ ；
- 空间复杂度为 $O(W)$ 。

可以尝试思考 $O\left(n + \frac{V^{3/2}}{w}\right)$ 的做法。

Solution E

$n = 4$ 的情况很简单，将 $m = 2, r = 1$ 和 $m = 3, r = 1$ 两次询问结果相加即可。

考虑 $n = 5$ 的情况，依次记为 a, b, c, d, e ，手玩一下不难发现将 $a + c, b + d, c + e, a + b + d + e$ 四个询问的结果相加恰好是 $2(a + b + c + d + e)$ ，因此这 5 个比特的和可以确定。进一步地，结合 $a + b + d + e$ 的结果，可以确定 c 的值，而结合 $a + c$ 以及 $c + e$ 的结果， a 和 e 的值也可以确定。这意味着：对于任意连续 5 个比特，可以用 4 次查询，得到首、尾、中心 3 个比特的值。

由此，当 $n \geq 6$ 时可以采用如下的询问方式：

- 若 n 为奇数，先询问整个 B ，然后利用上述手段用 4 次询问把中间的比特补上，一共 5 次询问。
- 若 n 为偶数，先使用 n 为奇数时的方法，用 5 次询问把前 $n - 1$ 个比特之和确定（因为 $n - 1$ 是奇数），然后用 4 次询问把最后的比特补上，一共 9 次询问。

因此，最多只需要 9 次询问。存在至多 5 次询问的做法，感兴趣可以自行思考。

Solution F

我们用 $\forall a \in \mathbb{N}, a^2 \equiv 0 \text{ 或 } 1 \pmod{4}$ 。

法一（风之混氏新子：以 4 为循环，构造 $a_k = [4k + 1, 4k + 2, 4k + 4, 4k + 3]$, $b_k = [4k + 3, 4k + 4, 4k + 2, 4k + 1]$ ，将 $a_1, b_2, a_3, b_4, \dots$ 相接即可，简单处理首数/尾数即可。

法二（插座也会求偏导）：设 $1 \sim n$ 中偶数从大到小排列为 a ，将除了 5 以外的模 4 余 1 的数从小到大排列为 b ，模 4 余 3 的数从小到大排列为 c ，则 $a, b, 5, c$ 相接为一个解。 $n \leq 4$ 输出 $1 \sim n$ 即为正确答案。

时间复杂度均为 $O(n)$ 。

Fun Fact：验题时每个 AI 均给出了不同的构造。

Solution G & H

注意到发送的是染色后的任意一个欧拉序。

先考虑染 n 种颜色怎么做，只需要把欧拉序相邻两个点连起来就好了。

考虑 3 种颜色。我们钦定一个根，考虑按深度 $d(u) \bmod 3$ 染色， d_u 表示节点 u 的深度。如果欧拉序中模 3 意义下后一个数比前一个数加一，则向下走，否则向上走，这样很好的还原。

考虑 2 种颜色。如果相邻染色不同会丢失所有信息。我们事实上想知道的是欧拉序在哪里从“向下”变成“向上”或者从“向上”变成“向下”一个朴素的想法是利用一个性质：欧拉序中，从一个节点到叶子再返回，叶子两边的点深度相同；同理，从 v_1 到 u 再到 v_2 ，其中 v_1, v_2 为 u 的儿子， v_1 与 v_2 深度也相同。因此，我们尝试让第 i 层和第 $i + 2$ 层的颜色不同。

Solution G & H

我们钦定一个根 r ，然后染色 $c_u = \lfloor \frac{d_u \bmod 4}{2} \rfloor$ 。还原时我们枚举当前深度模 4，然后找到深度最小的位置作为起点按编码的逆操作还原即可。

考虑根的选取会不会导致还原出来两棵树。选择错误当且仅当把向上和向下全部反过来。考虑树的深度(以边数记)为 E ，若原树每个节点的 u 深度为 d_u ，新树中则为 $E - d_u$ 。要求编码相同，则必须有 $\lfloor \frac{d_u \bmod 4}{2} \rfloor = \lfloor \frac{(E - d_u) \bmod 4}{2} \rfloor$ 对所有的 u 都成立。这表明 $E \equiv 1 \pmod 4$ 。我们只需要在选根的时候选择深度模 4 不为 1 的根即可。

考虑树的直径，直接选择直径端点，如果深度不满足要求，则偏移一个节点一定满足要求。

时间复杂度和空间复杂度均为 $O(n)$ 。评测时瓶颈在判断两棵树是否同构上。

Solution I

核心转化

固定频段 c ，记同频节点集合为 S_c ，令 $d_u = \text{dist}(1, u)$ 。若 $z = \text{lca}(u, v)$ ，则 $D(u, v) = |a_u - a_v| + d_u + d_v - 2d_z$ 。定义 $g_{c(z)}$ 为所有满足 $u, v \in S_c$ 且 $\text{lca}(u, v) = z$ 的点对中的最小 $D(u, v)$ ；不存在则视为 $+\infty$ 。

只需在 S_c 的虚树上计算 g_c 。任何两个同频点的 LCA 都在虚树中；若该频段出现 k 次，虚树大小为 $O(k)$ ，所以所有频段的虚树总大小为 $O(n)$ 。

令 $s_{c(v)}$ 为虚树子树 v 中所有 g_c 的最小值。若虚树边为 $p \rightarrow v$ ，则原树路径 $(p, v]$ 上任意节点 x 都有 $\min_{z \in \text{subtree}(x)} g_{c(z)} = s_{c(v)}$ ；虚树根 r 对应路径 $[1, r]$ ，阈值为 $s_{c(r)}$ 。因此每个频段只产生 $O(k)$ 个事件：当 $K \geq s_{c(v)}$ 时，给路径 $(p, v]$ 加上 w_c 。

Solution I

祖先链加无需树剖。对 Euler 序做差分：事件 $(p, v]$ 激活时令 $B[\text{dfn}_v] += w_c$ 、 $B[\text{dfn}_p] -= w_c$ ($p = 0$ 时忽略第二项)，则节点 x 当前收到的贡献恰好是 B 在 $\text{subtree}(x)$ 对应 DFS 区间上的和。把事件按阈值排序、询问按 K 排序，用 Fenwick 扫描即可，公共部分复杂度为 $O((n + q) \log n)$ ，空间为 $O(n + q)$ 。

下面只需解决：对一个频段在 $O(k \log k)$ 时间、 $O(k)$ 空间内求出虚树上所有 g_c 。

Solution I

做法一：虚树 + 可合并 Treap

在虚树节点 z 处，只需考虑来自 z 自身和不同儿子子树之间的点对。记 $H(u, v) = d_u + d_v + |a_u - a_v|$ ，则 $g_c(z) = \min H(u, v) - 2d_z$ 。

对一个点集按 a_u 建随机 Treap。每棵 Treap 子树维护 $M_- = \min(d_u - a_u)$ 和 $M_+ = \min(d_u + a_u)$ ；相同的 a 只保留最小的 d 。合并两个集合 A, B 时用标准 Treap Union：若当前取 A 的根 x ，按 a_x 把 B 分成键值小于、等于、大于 a_x 的三部分，再递归合并左右同侧部分。跨过 a_x 的候选只需比较两项： A 中不大于 a_x 的最小 $d - a$ 加上 B 中不小于 a_x 的最小 $d + a$ ，以及交换 A, B 后的同类候选；它们都能由 M_-, M_+ 在 $O(1)$ 时间取得。

在虚树上后序处理。节点 z 先把自身（若颜色为 c ）作为一个集合，再依次 Union 各儿子的 Treap。每次合并得到的跨集合最优值都来自不同虚树分支，所以这些点对的 LCA 恰好为 z ，用该值减去 $2d_z$ 更新 $g_c(z)$ ；同时维护 $s_{c(z)} = \min(g_c(z), s_{c(\text{children})})$ ，即可直接生成事件。

Treap Union 合并大小为 $x \leq y$ 的两棵随机 Treap，期望代价为 $O(x \log(\frac{x+y}{x}))$ 。用势能 $F(t) = t \log t$ 计费，整棵虚树从若干单点集合合并为大小 k 的集合，总代价为 $O(k \log k)$ ，而不是 $O(k \log^2 k)$ 。因此该做法单个频段期望 $O(k \log k)$ ，总时间期望 $O((n + q) \log(n + q))$ ，总空间 $O(n + m + q)$ 。

Solution I

做法二：按 a 分治 + 虚树压缩

将同频节点按 (a_u, u) 排序为 $p_0, p_1, \dots, p_{k-1}$ ，定义 $L_i = d_{p_i} - a_{p_i}$ 、 $R_i = d_{p_i} + a_{p_i}$ 。对 $i < j$ 有 $a_{p_i} \leq a_{p_j}$ ，因此 $D(p_i, p_j) = L_i + R_j - 2d_{\text{lca}(p_i, p_j)}$ 。

对下标区间 $[l, r)$ 分治，令 $m = \frac{l+r}{2}$ 。当前层只处理 $i \in [l, m)$ 、 $j \in [m, r)$ 的点对；任意一对 $i < j$ 恰好在它们第一次被分到左右两侧的那一层出现一次。

当前区间已有一棵虚树。在虚树上后序 DP：对每个节点 u ，维护其子树内左半部分的最小 L_i 和右半部分的最小 R_j 。扫描 u 的自身和各儿子组，记此前各组的两个最小值为 P, Q ，新组的两个最小值为 X, Y ，则用 $P + Y - 2d_u$ 和 $Q + X - 2d_u$ 更新 $g_{c(u)}$ ，再令 $P = \min(P, X)$ 、 $Q = \min(Q, Y)$ 。因为只组合不同组，得到的点对 LCA 恰好是 u 。

关键是递归时不能重新求 LCA 建虚树，否则会多一个 $\log$ 。左右子问题的虚树直接从当前虚树线性压缩得到：对某一侧，若节点 u 本身属于该侧，或至少有两个儿子子树在该侧

非空，则保留 u ；若 u 不属于该侧且只有一个非空儿子，则把 u 压掉，直接向上返回该儿子。一次后序即可同时生成左右两棵子虚树，时间为 $O(|VT|)$ 。

分治树同一层的各区间互不相交，真实节点数总和为 k ，对应虚树大小总和也是 $O(k)$ ，所以每层 $O(k)$ 、共 $O(\log k)$ 层，总时间 $O(k \log k)$ 。深度优先处理分治时，沿递归链暂存的兄弟虚树规模为 $\frac{k}{2} + \frac{k}{4} + \dots < k$ ，因此辅助空间仍为 $O(k)$ 。全部频段总计确定性 $O((n + q) \log(n + q))$ 时间、 $O(n + m + q)$ 空间。

实现说明

原树只需维护 fa 、 dep 、 siz 、 son 、 top 、 dfn 和 d ，用 HLD 求 LCA，空间仍是线性的。同频节点可按全局 DFS 序扫描后直接装入各颜色的连续数组，因此无需再按 dfn 排序；虚树也可以对已按 dfn 排序的节点在线用栈构造，不必把所有 LCA 加入后再排序。

两种做法的公共事件数都是 $O(n)$ 。Treap 做法代码更直接但为期望复杂度；按 a 分治的做法完全确定性，且主要是顺序数组扫描，通常常数更稳定。

Solution J

验题时得到了多种做法，下面介绍最具有启发性的四种。

这些做法均涉及到两个子串的字典序的比较，二分哈希、后缀数组结合 RMQ 等技术都可以通过 $O(n \log n)$ 或 $O(n)$ 的预处理实现单次 $O(\log n)$ 或 $O(1)$ 询问。本题时限确保 $O(n \log n)$ 时间复杂度且常数合理的做法可以通过。

借助 $O(n)$ 后缀数组以及 $O(n) - O(1)$ RMQ，做法一和做法四均可以导出 $O(n)$ 时间复杂度的确定性算法。

以下记以 i 为中心，半径为 r 的回文串为 $P(i, j)$ ，中心 i 的最长回文半径为 f_i 。

Solution J

做法一

对每个 i 和 $j \leq f_i$ ，维护 $p_{i,j}$ ，表示以 i 为中心、半径不超过 j 的所有回文串中，字典序最小者的半径。

在 Manacher 的过程中复用镜像点的信息并扩展回文串的长度，本质上归结为两种操作：

- 取某个已有版本的前缀作为新版本；
- 在某个已有版本的尾部添加新值作为新版本。

因此可以离线地把版本按照依赖关系建成一棵树。

在版本树上 DFS，整个过程中复用同一个数组维护当前版本的 p 。在尾部扩展时，用新值替换数组中对应位置的元素，递归结束时撤销，使对应位置的元素恢复为父版本的值。

瓶颈是扩展过程中产生的 $O(n)$ 次子串字典序比较。

Solution J

做法二

考虑把每个本质不同的回文子串用字符串哈希表示，并以哈希值作为 key 存入哈希表。

对于每个本质不同的回文串 X ，维护它所有内部同心回文串（包括自身）中字典序最小的一个，记为 best_X 。

在 Manacher 扩展中心 i 时，由镜像点可以先得到一段确定能够复用的回文半径。

设这段回文串对应 X 。利用字符串哈希可以 $O(1)$ 求出 X 的哈希值，于是可以直接从哈希表中得到 best_X ，不需要重新处理这部分内部结构。

之后只需要从这段已经复用的回文串继续向外扩展，并在出现新的本质不同回文串时更新哈希表。复杂度瓶颈是 $O(n)$ 次子串字典序比较。

Solution J

做法三

固定一个中心 i 。对于每个合法半径 r ，回文串 $P(i, r)$ 的左端点为 $i - r$ ，因此它对应于后缀 $s_{i-r}s_{i-r+1}\dots$ 的一个前缀。

考虑所有这些左端点： $i - f_i, i - f_i + 1, \dots, i$ 。利用后缀数组，可以找到这些位置中字典序最小的后缀。设其对应半径为 r^* ，注意到以下性质：

中心 i 的答案是使得 $P(i, r)$ 的字典序小于 $P(i, r^*)$ 的最小的 r 。

因此可以从小到大枚举半径，找到第一个满足 $P(i, r) < P(i, r^*)$ 的 r ；如果不存在，则答案就是 r^* 。

总比较次数为 $O(\sum \text{ans})$ ，可以证明（过程较为复杂，详见这个链接）： $\sum \text{ans} = O(n \log n)$ ，因此复杂度瓶颈是 $O(n \log n)$ 次子串字典序比较。事实上，这个界是紧的。

Solution J

做法四

建立回文自动机 (PAM)。每个结点对应一个本质不同的回文子串。对于结点 X ，维护其所有内部同心回文串（包括自身）中字典序最小的一个，记为 best_X 。在转移边形成的奇回文串树上做 DP，即可计算每个结点的 best_X 。

对于每个中心 i ，最长回文串 $P(i, f_i)$ 的右端点为 $r = i + f_i$ ，预先把查询 (i, f_i) 挂在位置 r 上。从左到右建立回文自动机时，处理完位置 r 后，当前结点 last 表示以 r 结尾的最长回文后缀。而所有以 r 结尾的回文串都位于 last 的 fail 链上。根据目标长度 $2f_i + 1$ 即可在 fail 链上找到 $P(i, f_i)$ 对应的结点。得到这个结点 X 后，中心 i 的答案就是 best_X 对应的半径。

复杂度瓶颈在于 $O(n)$ 次在 fail 树上找指定长度的祖先以及 $O(n)$ 次子串字典序比较，前者只需要在 fail 树 DFS 过程中维护根到当前结点的链上，长度与结点的映射数组即可做到线性。

Solution K

由于 $\gcd(p, q) = 1$ ，根据中国剩余定理，每个卡片编号 $x \in [0, pq)$ 与二元组 $(x \bmod p, x \bmod q)$ 一一对应。

对于一次询问，定义 $C_{u,v}$ 为满足 $x \equiv u \bmod p$ 且 $x \equiv v \bmod q$ 的卡片数量，其中 $0 \leq u < p$ ， $0 \leq v < q$ 。于是可以将 C 看作一个 $p \times q$ 的矩阵。

第一类编号为 u 的标准卡包恰好使第 u 行每个位置增加 1；第二类编号为 v 的标准卡包恰好使第 v 列每个位置增加 1。若两类卡包分别使用 R_u, S_v 次，则位置 (u, v) 中共有 $R_u + S_v$ 张卡片。因此区间合法，当且仅当存在非负整数 R_u, S_v ，使得

$$C_{u,v} = R_u + S_v.$$

特别地，若 $p = 1$ 或 $q = 1$ ，则存在只含一张卡片的标准卡包，因此任意询问均合法。下面只考虑 $p, q \geq 2$ 。

Solution K

上述条件等价于：对任意 u, u', v, v' ，均有

$$C_{u,v} - C_{u',v} - C_{u,v'} + C_{u',v'} = 0.$$

必要性直接代入 $C_{u,v} = R_u + S_v$ 即可得到。

下面证明充分性。固定 $u' = 0$ ，由上式可知，对固定的 u ， $C_{u,v} - C_{0,v}$ 与 v 无关，记这个差为 d_u 。于是 $C_{u,v} = C_{0,v} + d_u$ 。

令 $m = \min_v C_{0,v}$ ，取 $R_u = \min_v C_{u,v} = m + d_u$ ， $S_v = C_{0,v} - m$ 。显然二者均为非负整数，并且 $R_u + S_v = C_{u,v}$ ，故充分性成立。

直接检查所有四元组显然不可行，因此使用随机赋权压缩这些条件。

取质数 $M > 2n$ ，所有运算均在模 M 意义下进行。随机生成 $r_0, r_1, \dots, r_{p-2}$ ，并令 $r_{p-1} = -\sum_{u=0}^{p-2} r_u$ ，从而 $\sum_u r_u = 0$ ；类似地生成 s_v ，使得 $\sum_v s_v = 0$ 。

Solution K

定义位置 (u, v) 的随机权值为 $r_u s_v$ ，即编号 x 的卡片权值为 $w(x) = r_{x \bmod p} s_{x \bmod q}$ 。询问区间的总权值为

$$H = \sum_{u,v} C_{u,v} r_u s_v.$$

若区间合法，则将 $C_{u,v} = R_u + S_v$ 代入后，两部分分别含有因子 $\sum_v s_v$ 与 $\sum_u r_u$ ，因此均为 0，故 $H = 0$ 。

下面估计非法区间被误判的概率。

若区间非法，则一定存在两列 v, v' ，使 $D_u = C_{u,v} - C_{u,v'}$ 不关于 u 为常数。定义 $T_v = \sum_u C_{u,v} r_u$ 。利用 $r_{p-1} = -\sum_{u=0}^{p-2} r_u$ ，可将 $T_v - T_{v'}$ 写成 $\sum_{u=0}^{p-2} (D_u - D_{p-1}) r_u$ 。

因为 D_u 不全相同，所以其中至少有一个系数非零。又有 $-2n \leq D_u - D_{p-1} \leq 2n$ ，且 $M > 2n$ ，因此这个非零系数在模 M 下仍然非零。

Solution K

于是 $T_v - T_{v'}$ 是关于随机变量 r_u 的非零线性式。固定其余随机变量后，只有唯一一个 r_u 会使它等于 0，故第一层失败的概率为 $\frac{1}{M}$ 。

当第一层成功时，所有 T_v 不完全相同。再利用 $s_{q-1} = -\sum_{v=0}^{q-2} s_v$ ，可将 H 写成 $\sum_{v=0}^{q-2} (T_v - T_{q-1})s_v$ 。其中至少有一个系数非零，因此同理，在第一层成功的条件下，第二层仍误判的概率为 $\frac{1}{M}$ 。

所以一次随机化的误判概率满足

$$\Pr(\text{误判}) \leq \frac{1}{M} + \frac{1 - \frac{1}{M}}{M} < \frac{2}{M}.$$

实际实现中使用三组相互独立的随机赋权。若三个区间权值和均为 0，输出 YES，否则输出 NO。因此非法区间三组同时误判的概率小于 $(\frac{2}{M})^3$ 。

Solution K

取 $M = 10000000007$ 时, 单次询问的误判概率小于 8×10^{-27} ; 即使有 3×10^5 次询问, 总误判概率也可以忽略。

最后, 分别使用三棵树状数组维护三个权值 $w_1(a_i), w_2(a_i), w_3(a_i)$ 的区间和。修改时删去旧权值并加入新权值, 询问时检查三个区间和是否均为 0。

生成随机数组需要 $O(p + q)$ 时间与空间, 初始化树状数组需要 $O(n \log n)$ 时间, 每次修改和询问均为 $O(\log n)$ 。因此总时间复杂度为 $O((n + Q) \log n + p + q)$, 总空间复杂度为 $O(n + p + q)$ 。

Solution L

无效点删除

先删去所有不可能出现在答案中的状态。具体地，从所有接受状态出发在反图上做一次DFS/BFS，能被访问到的状态称为“有用状态”；其余状态以及连向它们的边都可以直接忽略。若状态 1 不是有用状态，则不存在任何接受串，输出 WA。

Solution L

计算 f

记 f_u 为从 u 出发、最终停在接受状态的路径的最大长度；若长度可以任意大，则令 $f_u = \infty$ 。

只考虑有用状态组成的图，令 $\deg_u$ 为 u 指向有用状态的出边数，然后不断剥去出度为 0 的点。初始令所有 $f_u = 0$ ，每次删除 v 时，对每条反向边 $u \rightarrow v$ 更新 $f_u = \max(f_u, f_v + 1)$ ，并将 $\deg_u$ 减一；若其变成 0，就把 u 加入队列。

最后被剥掉的点恰好是 f 有限的点，未被剥掉的点全部满足 $f_u = \infty$ 。这是因为未被剥掉的点可以沿有用边一直走下去，有限图中必然进入环，而所有有用点又都能到达接受状态，所以可以在环上绕任意多圈后再走到接受状态。另一方面，被剥掉的部分是 DAG，因此有限的 f_u 一定不超过 $n - 1$ 。

若 f_1 有限且 $f_1 < k$ ，显然不存在长度至少为 k 的接受串，输出 WA。

Solution L

贪心

设当前已经得到长度为 l 的前缀，并位于状态 u 。若 $l \geq k$ 且 u 是接受状态，应当立即结束，因为一个字符串总比它的任何真扩展字典序更小。

否则考虑一条字符为 c 、终点为 v 的边。它能够被选择，当且仅当 v 是有用状态，并且 $f_v = \infty$ ，或者 $l + 1 + f_v \geq k$ 。这个条件正好表示走完这条边以后，仍然存在某个接受串能使最终长度达到 k 。

因此每一步先尝试 A，若 A 不可行再选择 C。只要贪心过程最终停下，得到的就一定是合法串中字典序最小者。

Solution L

跳过很长的前缀

直接模拟仍然可能需要 $O(k)$ 步。关键是有限的 f 至多为 $n - 1$ 。

令 $L = \max(0, k - n)$ 。当当前长度 $l < L$ 时，有 $k - l > n$ 。若下一步走到一个 f_v 有限的状态，则 $l + 1 + f_v < k$ ，所以这条边一定不可行。也就是说，前 L 步只能在 $f = \infty$ 的状态之间移动。

于是对每个无限状态 u 定义固定转移：若 A 边的终点也是无限状态就走 A ，否则走 C 。无限状态至少有一条边仍指向无限状态，因此该转移总是存在。前 L 步完全按照这个固定转移进行，而这是一个函数图；从状态 1 出发记录首次出现位置，一旦进入环，就可以用环长取模， $O(n)$ 求出走恰好 L 步后的状态，而无需真正模拟 L 次。

Solution L

模拟剩余部分

从长度 L 开始，再按照前面的完整贪心规则逐步模拟。到长度 k 最多还剩 n 步，因此这一段只有 $O(n)$ 步。

当 $l \geq k$ 后，所有指向有用状态的边都自动满足长度限制，此后的规则也重新变成固定的：若当前状态已经接受就结束，否则优先走通向有用状态的 A，再考虑 C。因此再经过至多 n 个不同状态，要么到达接受状态，要么重复访问某个非接受状态。

若发生后者，贪心路径将永远循环，此时应输出 WA。因为贪心路径的每个有限前缀都仍然可以扩展为某个合法答案，而任意合法答案若在某处第一次偏离这条无限贪心路径，只可能选择更大的字符；继续沿贪心路径多走一些再扩展，就能得到一个严格更小的合法答案，所以根本不存在字典序最小的字符串。

Solution L

输出与复杂度

显式模拟从 L 开始至结束至多经过 $O(n)$ 步，过程中只需维护最近 n 个字符。若最终答案长度不超过 n ，此时 $L = 0$ ，保存的就是完整答案；若答案长度超过 n ，直接输出维护的最后 n 个字符即可。

时间复杂度为 $O(n)$ 。

Solution M

任意函数都可以看成一张函数图：每个点出度恰为 1，因此每个弱连通块由一个有向环和若干棵指向环的树组成。

若 x 不在环上，则不断迭代 f 后不可能回到自身；若 x 位于长度为 d 的环上，则 $f^{k(x)} = x \Leftrightarrow d \mid k$ 。因此， k 阶迭代不动点恰为所有长度整除 k 的环上的点。

设这些点的集合为 P ， $|P| = p$ 。先从 n 个点中选出 P 。由于 P 中所有点都在合法环上， f 在 P 上的限制是一个所有循环长度均整除 k 的置换。

定义

$$A(x) = \exp \left(\sum_{d \mid k} \frac{x^d}{d} \right) = \sum_{i \geq 0} a_i x^i.$$

则 $i!a_i$ 等于 i 个有标号元素上，所有循环长度均整除 k 的置换数。

Solution M

对 $A(x)$ 求导：

$$A'(x) = A(x) \sum_{d \mid k} x^{d-1}.$$

比较 x^{i-1} 的系数，得到

$$a_0 = 1, \quad a_i = \frac{1}{i} \sum_{d \mid k, d \leq i} a_{i-d}.$$

令 $m = n - p$ 。固定 P 后，剩余点的像仍可任取 $1, 2, \dots, n$ ；唯一禁止的是这些点内部出现长度整除 k 的环，否则会产生新的 k 阶迭代不动点。

对这些坏环做容斥。若钦定若干互不相交的坏环，共占用 r 个点，则其余 $m - r$ 个点仍各有 n 种像。长度为 d 的有向环在指数生成函数中的贡献为 $\frac{x^d}{d}$ 。

Solution M

故

$$B(x) = \exp(nx) \exp\left(-\sum_{d \mid k} \frac{x^d}{d}\right) = \exp\left(nx - \sum_{d \mid k} \frac{x^d}{d}\right) = \sum_{i \geq 0} b_i x^i.$$

于是 $i!b_i$ 即为指定的 i 个剩余点中，不出现长度整除 k 的内部环的方案数。

对 $B(x)$ 求导：

$$B'(x) = B(x) \left(n - \sum_{d \mid k} x^{d-1} \right).$$

比较 x^{i-1} 的系数，得到

Solution M

$$b_0 = 1, \quad b_i = \frac{1}{i} \left(nb_{i-1} - \sum_{d \mid i, d \leq i} b_{i-d} \right).$$

因此答案为

$$\binom{n}{p} p! a_p (n-p)! b_{n-p} = n! a_p b_{n-p}.$$

设 $\tau(k)$ 为 k 的约数个数。枚举 k 的所有约数后，两组递推总时间复杂度为

$$O(n\tau(k)),$$

空间复杂度为 $O(n)$ 。

Outline

Summary	2	Solution K	34
Statistics	3	Solution L	39
Interesting Facts	8	Solution M	45
Solutions	9	Acknowledgments	49
Solution A	10	Thanks	50
Solution B	11		
Solution C	13		
Solution D	17		
Solution E	19		
Solution F	20		
Solution G & H	21		
Solution I	23		
Solution J	29		

Thanks

组织者：Huasushis。

出题人：矧可射思(EJL)、久洵(ABKM)、风之混氏新子(D)、插座也会求偏导(FGH)、David24(I)、1xx55(C)。

验题人：JerrySyr, caidx233, pjz_pjz, Sayu_zzz, a_little_cute, AllFiction, syk070531, GUAIKATTO, dremlory, yixiuge777, chen_zida, xuanxuan001, Dinshey, gty314159, Serval, WannaWalnut, Illyasv1el, retrosidus, XiaoXia, bookcat, o_yingxing_o, Kown。

AI 验题人：Kimi k3, Qwen 3.8-max, ChatGPT5.6 sol extra-high, Gemini 3.1 Pro

Thanks

特别鸣谢：浅梦，XeF2。

感谢老师们的鼎力相助！

也感谢同学们的积极参与！

Kruskal2026 完结撒花！