2025 年广东工业大学 ACM 程序设计竞赛月赛题解

广东工业大学 ACM 集训队 24 级出题组

2025 年 10 月 19 日

题目难度分布

题目难度分布：

- 签到题：A、I、L
- Easy：E、H、J、M
- Easy-Medium：C、G、K
- Medium：D、F、N
- Medium-Hard：B
- 好像没有难题！

实际通过人数排序（封榜前）：

- A (153/263)、I (122/292)、L (60/119)、M (39/123)、J (32/89)、H (28/152)、E (24/250)、C (11/50)、K (8/71)、G (7/172)、F (1/5)、D (1/10)、B (1/23)、N (0/2)

A. %%% 自动机

简明题意

输出一个字符串，由 n 个字符‘%’组成。

签到题，按题意模拟即可，注意不要输出多余字符（如空格）。

时间复杂度 $O(n)$。

I. 运动世界校园

简明题意

每次操作有 p 概率成功，求三次操作至少一次成功的概率。

签到题，推公式并计算即可。

I. 运动世界校园

单次操作失败的概率为 $1-p$ 。

由于每次操作是独立的，所以三次操作全部失败的概率为 $(1-p)^3$。

故三次操作至少一次成功的概率为 $1-(1-p)^3$ 。

注意使用小数点表示法输出答案，并输出足够位数以满足输出精度要求。

时间复杂度 $O(1)$。

L. 硬币翻转

简明题意

有排成一排的 n 个硬币，每次操作可以翻转相邻两个硬币，判断是否可以经过若干次操作使全部硬币为正面。

签到题。

不妨记录反面硬币数量为 c，注意到每次翻转硬币后，c 的奇偶性总是不变的。

由于目标状态 $cnt = 0$ 为偶数，所以如果初始状态的 c 为奇数，则一定不能使所有硬币均变为正面；

否则，总可以通过贪心操作使得所有硬币为正。

L. 硬币翻转

我们可以按从第 1 枚硬币到第 $n-1$ 枚硬币的顺序执行以下操作：

如果第 i 枚硬币为反面，则翻转第 i 枚硬币和第 $i+1$ 枚硬币，否则跳过。

执行完毕后，从 1 到 $n-1$ 的每枚硬币均为正面。

现在只有第 n 枚硬币可能为反面，所以 c 只能为 0 或 1 。

但是已知 c 为偶数，即 $c=0$ ，第 n 枚硬币只能是正面。

时间复杂度 $O(n)$ 。

L. 硬币翻转

由于数据范围很小，我们还有一些更暴力的做法：

1. 枚举每对相邻位置是否取反，共 2^{n-1} 中可能；
2. 将整个字符串作为状态执行 DFS 或 BFS。

以上两种做法时间复杂度至少为 $O(n2^n)$，但是足以通过。

E. 新田忌赛马

简明题意

给定递减序列 a 和 b，匹配若干不重下标 (i, j)，在满足 $a_i > b_j$ 的情况下最大化 $\sum b_j$。

贪心题。我们总是可以考虑最大的 a_i：

- 只要当前存在 $a_i > b_j$，则 i 与最大的 b_j 对应的 j 匹配一定不劣；
- 否则 i 无法匹配任何 j，跳过这个 i。

然后变成同样的一个子问题，重复此贪心即可。

可以使用 set 等平衡树实现此贪心，时间复杂度 $O(n \log n)$。

也可以使用双指针实现此贪心，时间复杂度 $O(n)$。

H. 我不吃水果

简明题意

给定一个 $n \times n$ 的 $0/1$ 矩阵，求从左上角 $(1,1)$ 开始的最大全 0 子矩形的面积。

左上角的点已经固定是 $(1,1)$，可以枚举子矩形右下角的点 (i,j)，并快速判断该子矩形是否包含 1，考虑 DP。

令 $F[i][j]$ 为“从 $(1,1)$ 到 (i,j) 的子矩形是否包含 1”，包含记为 1，否则记为 0。

可以列出以下状态转移方程：

- $F[i][j] = F[i-1][j] \vee F[i][j-1] \vee a[i][j]$，其中 $\vee$ 代表逻辑或。

意思是 (i,j) 对应的子矩形包含 1，要么 $(i-1,j)$ 或 $(i,j-1)$ 对应的子矩形包含 1，要么 (i,j) 这个位置就是 1。

枚举所有 $F[i][j] = 0$ 的位置并计算 $i \times j$ 的最大值即可，时间复杂度 $O(n^2)$。

J. 逃出生天

简明题意

在 $(n+2) \times m$ 网格中，第 2 到 n 行每行有一个石像，石像会朝正前方发射激光。

小猪不能与石像同格或进入激光区域，判断从 $(1, m)$ 出发，是否可以到达 $(n+2, 1)$ 。

预处理一个 $(n+2) \times m$ 的地图，记录每个位置小猪是否可以进入，然后执行 DFS 或 BFS 即可。

时间复杂度 $O(n \times m)$。

J. 逃出生天

我们也可以考虑小猪如何到达下一行，只需要判断是否存在以下情况：

- 某个石像在第 1 列，朝向右。
- 某个石像在第 m 列，朝向左。
- 某一行的石像在第 x 列，朝向左，且下一行的石像列数小于等于 $x+1$，朝向右。
- 某一行的石像在第 x 列，朝向右，且下一行的石像列数大于等于 $x-1$，朝向左。

只要存在任一情况，则小猪不能到达终点，否则可以。

时间复杂度 $O(n)$。

M. 字符消消乐

简明题意

给定一个长度为 n 的字符串和最多 k 次消除操作，每次可以消除长度至少为 3 的相邻相同字符段，消除后左右部分拼接。游戏开始前可以任意重排字符串。问最终能缩短到的最小长度。

我们总是可以把相同的字符重排到相邻位置，所以只要出现次数大于 3 的字符都可以在一次操作内消除。

记录 26 种不同的字符的出现次数，排序后删除出现次数前 k 大的字符即可。

时间复杂度 $O(n)$ 。

C. 序列重构（简单版）

简明题意

有一个隐藏的长度为 n 的排列 p。

每次可以查询一个长度为 n 的 **序列**中有多少个位置正确。

在 n^2 次查询内确定隐藏的排列 p 。

简单交互题。考虑逐个确定某个位置 i 的值：

我们可以将其余位置设置为 $n+1$，然后在该位置枚举 1 到 n 每个整数填入并查询，返回 1 则代表位置 i 当前填入的数正确。

以上方法可以在 n 次查询内确定一个位置，总查询次数不超过 n^2 。

G. 俄罗斯方块的博弈

简明题意

在 4×4 的网格上轮流放置 T 形方块或 L 形方块，无法放置者失败，求先手必胜或必败。

此题分类讨论题，需要耐心且细心地分类。

不难发现，先手必胜只有以下情况：

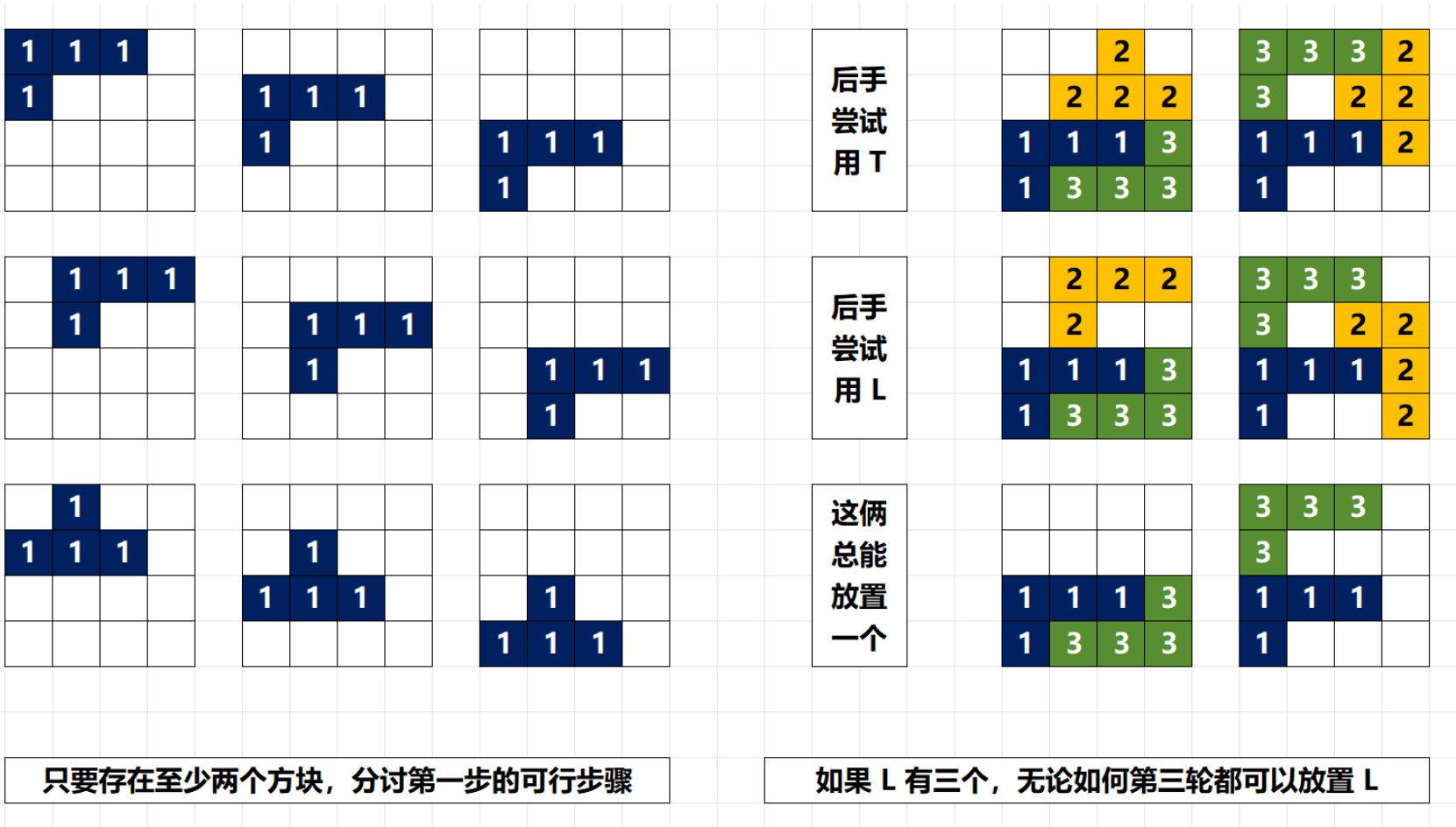
- 两种方块个数之和为 1 。
- 有 3 个 L 形方块。

时间复杂度 $O(1)$ 。

下文将提供一个完整的分讨流程，仅供参考。

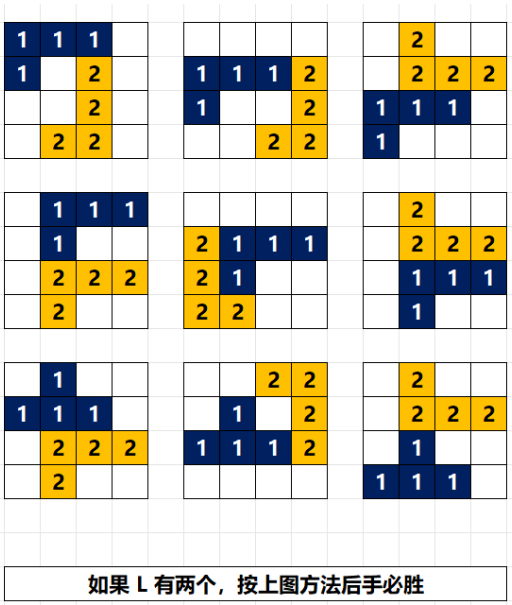
G. 俄罗斯方块的博弈

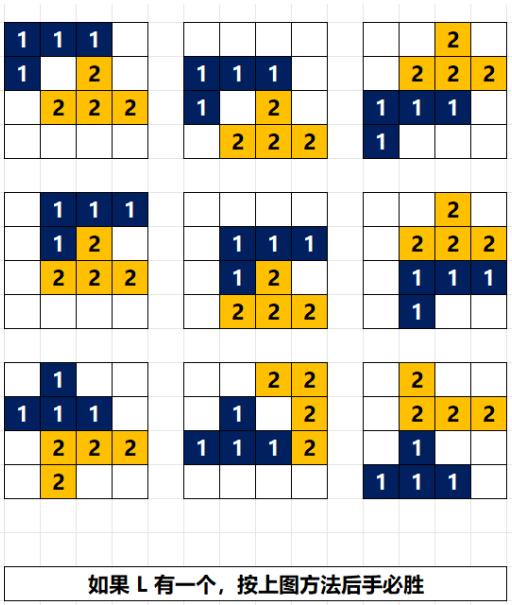
首先特判掉总方块个数不超过 1 的平凡情况，然后考虑样例 $x = 3,\ \ y = 3$ 先手如何获胜。可以枚举所有本质不同的第一步下法共九种，然后发现先手有且只有一种必胜下法：

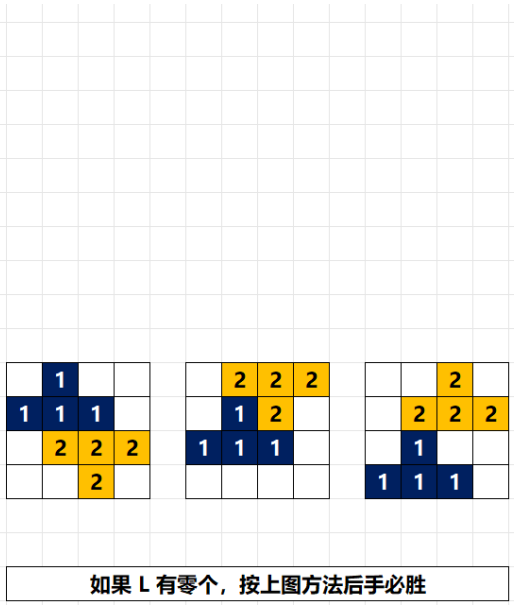


G. 俄罗斯方块的博弈

启发我们可以先枚举了 L 的个数，其次枚举所有本质不同第一步，然后我们只需要从后手角度考虑，尽可能让先手失败即可：







如图分讨，然后发现如果 L 的个数不为 3 ，无论先手如何下，后手都有必胜策略。

注意有些情况下网格仍然可以放置 L 方块，但是因为 L 方块的个数不足，先手无法放置。

G. 俄罗斯方块的博弈

分类讨论的情况较为复杂，我们还可以考虑状压 DP 或搜索。

将网格压缩成 16 位二进制串表示，并将所有的可能的 T 方块放置方法和 L 方块放置方法也用 16 位二进制串表示，然后就可以进行经典的博弈论 DP 。

至多 2^{16} 种状态，转移很少，并且难以跑满，足以通过。

此方法的代码实现颇为复杂，同样需要耐心且细心地实现代码。

K. 最不上升也不下降序列

简明题意

构造一个长度为 n 的排列，其 LIS 长度与 LDS 长度之和最小。

构造题，观察样例即可得到构造方法，以下给出证明。

K. 最不上升也不下降序列

定理

对于任意长度为 $mn+1$ 的排列 p，必然存在一个长度为 $m+1$ 的递增子序列或一个长度为 $n+1$ 的递减子序列。

证明

定义 $f(i)$ 为以 p_i 结尾的 LIS 长度，如果存在某个 $f(i) \geq m+1$，则定理成立。

否则，所有 $f(i) \leq m$，即 $f(i)$ 只能取 1 到 m 之间的整数。

由于序列有 $mn+1$ 个元素，根据鸽巢原理，至少有一个值 $1 \leq k \leq m$ 被 $f(i)$ 取到了至少 $n+1$ 次，也就是存在至少 $n+1$ 个位置 $i_1 < i_2 < \cdots < i_{n+1}$ 使得 $f(i_j) = k$ 。

考虑这些位置对应的元素 $p_{i_1}, p_{i_2}, \ldots, p_{i_{n+1}}$，这些元素一定构成一个递减序列。

综上，如果不存在长度为 $m+1$ 的递增子序列，则必然存在长度为 $n+1$ 的递减子序列。定理得证。

K. 最不上升也不下降序列

推论

对于任意长度为 n 的排列，有 $\text{LIS} \times \text{LDS} \geq n$。

根据基本不等式，有 $\text{LIS} + \text{LDS} \geq 2\sqrt{\text{LIS} \times \text{LDS}} \geq 2\sqrt{n}$ 。

由于 LIS 和 LDS 均为正整数，因此 $\text{LIS} + \text{LDS} \geq \lceil 2\sqrt{n} \rceil$ 。

我们可以构造一个排列，使得上述和达到这个下界。

K. 最不上升也不下降序列

令 $s = \lceil \sqrt{n} \rceil$，　$t = \left\lceil \frac{n}{s} \right\rceil = \lfloor \sqrt{n} \rfloor$ 。

我们可以将将数字 $1, 2, \ldots, n$ 分成 s 组，每组 t 个元素（最后一组可能不够 t 个）：

- 第 1 组：$1, 2, \ldots, t$
- 第 2 组：$t+1, t+2, \ldots, 2t$
- ...
- 第 s 组：$(s-1)t+1, \ldots, n$

将数字按从第 s 组到第 1 组这样的顺序连接起来，组内顺序不变，得到排列 p 。

这样构造的排列 p 中，$\mathrm{LDS} = s$，　$\mathrm{LIS} = t$ ，满足 $\mathrm{LIS} + \mathrm{LDS} = \lceil 2\sqrt{n} \rceil$ 最小。

D. 序列重构（困难版）

简明题意

有一个隐藏的长度为 n 的排列 p 。

每次可以查询一个长度为 n 的 **排列**中有多少个位置正确。

在 n^2 次查询内确定隐藏的排列 p 。

本题有很多做法，以下介绍一种简单的方法：

由于只能查询排列，我们考虑交换某个数对，通过两次查询之间返回值的差来得到信息。

从一个初始的排列开始，仍然考虑逐个确定某个位置 i 的值：

我们只需要将这个位置和所有其余位置交换，只要返回值变大就保留，否则不变。

由于我们考虑了 i 与所有位置的交换，如果当前 i 位置填入的数不正确，则当其与正确的数的实际位置执行交换后，位置 i 一定是正确的。

以上方法可以在 n 次查询内确定一个位置，总查询次数不超过 n^2 。

F. 雪莉的预言

简明题意

维护一个可重集，有添加元素和删除元素两种操作，每次操作后输出出现次数第 k 大的所有数之和。

本题使用随机数生成器生成数据，要求单次操作为时间复杂度 $O(1)$，单次操作时间复杂度为 $O(logm)$ 的做法（如树状数组上二分）预期无法通过本题。

F. 雪莉的预言

我们只需要维护以下几个数组：

每个数的出现次数 *cnt*，每个出现次数对应的数字个数 *freq* ，每个出现次数对应的数字之和 *fsum* ，以上信息均可以 $O(1)$ 修改。

然后记录当前第 k 大的出现次数 *curk* 以及出现次数严格大于 *curk* 的数字个数 *gr*，不难发现 *curk* 即满足对应 $gr < k$ 的最小值。

满足注意到每次操作后，*curk* 的变化不会超过 1 ，使用 *while* 循环移动 *curk* 即可。

时间复杂度 $O(n+q)$ 。

N. 缩花

简明题意

给定一棵树，每次操作可以选择一个根和一个节点，并将该节点的所有儿子重新连接到父亲上，计算如何通过最少的操作将其变成菊花图。

称“叶子节点”为度数恰好为 1 的节点，则“菊花图”实际上是恰好只有 1 个非叶子节点的图。

我们先计算给定的树中的叶子节点个数 k，则树中有 $n-k$ 个非叶子节点。

N. 缩花

容易看出，每次操作至多将树中一个非叶子节点变成叶子节点，故至少需要 $n-k-1$ 次操作。如果每次操作都能将树中叶子节点个数增加 1 ，则总操作次数取到该下界。

我们只需要任取一个非叶子节点作为根节点 r ，再取另一个非叶子节点作为 u 并执行操作，则节点 u 在操作后一定变成叶子节点，且其余节点“是否为叶子节点”不变。

时间复杂度 $O(n)$ 。

B. k-冲突数对

简明题意

在数组 a 中找最长连续子数组，使得其中不存在（可相同的）两个数满足 $x + y - gcd(x, y) = k$ 。

先分析冲突数对的性质：

平凡地，当 $x > k$ 则不与任何 y 冲突，于是只需要考虑 $x \leq k$。

不妨令 $d = \gcd(x, y)$

则 $x + y - \gcd(x, y) = k$

变为 $x - td - d = k$ ，其中 t 为整数。

整理得 $(t - 1)d = k - x$

所以 $d \mid (k - x)$ 且 $d \mid x$ 。

由整除的可加性得到 $d \mid k$ ，综上 $d \mid \gcd(x, k)$ 。

B. k-冲突数对

对于求最长连续子数组，我们可以使用双指针，左指针为 l，右指针为 r：

每次右指针向右移动，新增一个值 $x = a_r$ 时，枚举 $d \mid \gcd(x, k)$，通过 x 和 d 可以确定唯一的 y 并验证是否确实为冲突数对。

使用计数数组记录双指针维护的区间 $[l, r]$ 中存在的所有数，对于与 x 冲突的所有 y，不断右移 l 指针直到 $[l, r]$ 内不存在 y。

对于如何枚举 $d \mid \gcd(x, k)$，注意到 $\gcd(x, k)$ 一定是 k 的因数，因此预处理 k 的所有因数的因数数组即可，这一步的时间复杂度为 $O(d(k)^2) = O(k)$。

B. k-冲突数对

定理

$\sum_{i=1}^{n} d(\gcd(i, n)) = O(n \log \log n)$

推论

对值域 V 以内所有的 i 枚举 $d \mid \gcd(x, i)$ 的总枚举次数是 $O(V \log \log V)$ 级别的。

故枚举 $d \mid \gcd(a_r, k)$ 的双指针做法总时间复杂度为 $O(n \log k \log \log n)$ ，其中 $\log k$ 项由 gcd 运算带来，常数很小，足以通过本题。

如果只枚举 $d \mid k$ ，在 $k = 1081080$ 时，时间复杂度将变为 $O(n \log n \times 255)$ ，预期无法通过。

如果只枚举 $d \mid a_r$ ，需要预处理所有 $x \leq k$ 的因数，需要 $O(n \log n)$ 的空间和 $O(n \log n \log k)$ 时间且常数较大，预期无法通过。

B. k-冲突数对

由于 gcd 运算贡献了时间复杂度中的 $\log k$ 项，我们可以进一步优化 gcd 的时间复杂度。

本题需要用到的 gcd 值域为 k，理论上可以使用“洛谷 P5435 基于值域预处理的快速 GCD”将 gcd 运算优化为 $O(1)$，但该方法实现复杂且常数巨大，对通过本题没有帮助。

注意到前文所述的枚举中只需要计算形如 $\gcd(i,k)$ 形式的 gcd，可以按以下方法预处理：

- 枚举 k 的每个因数 d，将满足 $d \mid i$ 的所有 $\gcd(i,k)$ 对 d 取最小值。

这样就能处理出了枚举部分所需要的 gcd，时间复杂度同埃氏筛的 $O(k \log\log k)$。

枚举 d 并计算出 y 后还需要检查形如 $\gcd(\frac{k}{d}+1, \frac{a_r}{d}) \neq 1$ 的式子，注意到需要判断的 $\frac{a_r}{d} \leq \frac{k}{d}$，因此需要对每个不同的 $\frac{k}{d}$ 预处理 $[1, \frac{k}{d}]$ 哪些数与 $\frac{k}{d}+1$ 互质。

其中 $\frac{k}{d}$ 为 k 的因子，因此 $\sum_d \frac{k}{d} = O(k \log\log k)$，使用筛法求出每个数的最小质因数后可以 $O(1)$ DP $[1, \frac{k}{d}]$ 哪些数与 $\frac{k}{d}+1$ 互质，这一步的时间复杂度也是 $O(k \log\log k)$。

B. k-冲突数对

将 n 和 k 视为同阶，利用前文提到的预处理 gcd 后，做法的总时间复杂度变为 $O(n \log \log n)$，空间复杂度变为 $O(n \log \log n)$，以很小的常数达到目前已知最优复杂度，运行很快。

结语

Thank you!