# 2022 中国大学生程序设计大赛（威海站）题解

## 2022 China Collegiate Programming Contest (Weihai Site) Tutorial

电子科技大学

UESTC

2022 年 11 月 6 日

# Problem A - Dunai

## 题目大意

- TI 比赛有 $n$ 只冠军队伍，每个队伍由五个选手组成，每个选手分别打 1-5 号位中的一个，一个选手永远只打一个位置，不会打其他位置。有 $m$ 名选手组队，问最多能组多少队，满足每个队内至少有一名冠军选手。
- 关键词：签到，Easy

# Problem A - Dunai

## 题目大意

- TI 比赛有 $n$ 只冠军队伍，每个队伍由五个选手组成，每个选手分别打 1-5 号位中的一个，一个选手永远只打一个位置，不会打其他位置。有 $m$ 名选手组队，问最多能组多少队，满足每个队内至少有一名冠军选手。
- 关键词：签到，Easy

- 不考虑每个队都有冠军这一限制条件，可以得到的最大队伍数为 $\min\{p_i\}$，其中 $p_i$ 指打 $i$ 号位的选手人数。

# Problem A - Dunai

## 题目大意

- TI 比赛有 $n$ 只冠军队伍，每个队伍由五个选手组成，每个选手分别打 1-5 号位中的一个，一个选手永远只打一个位置，不会打其他位置。有 $m$ 名选手组队，问最多能组多少队，满足每个队内至少有一名冠军选手。
- 关键词：签到，Easy

- 不考虑每个队都有冠军这一限制条件，可以得到的最大队伍数为 $\min\{p_i\}$，其中 $p_i$ 指打 $i$ 号位的选手人数。
- 先不考虑冠军这一角色，先将这些人组出最大队伍数支队伍，然后给每个人分配冠军角色。

# Problem A - Dunai

## 题目大意

- TI 比赛有 $n$ 只冠军队伍，每个队伍由五个选手组成，每个选手分别打 1-5 号位中的一个，一个选手永远只打一个位置，不会打其他位置。有 $m$ 名选手组队，问最多能组多少队，满足每个队内至少有一名冠军选手。
- 关键词：签到，Easy

- 不考虑每个队都有冠军这一限制条件，可以得到的最大队伍数为 $\min\{p_i\}$，其中 $p_i$ 指打 $i$ 号位的选手人数。
- 先不考虑冠军这一角色，先将这些人组出最大队伍数支队伍，然后给每个人分配冠军角色。
- 最多可以分配 $\sum c_i$ 个冠军角色，其中 $c_i$ 指的是打 $i$ 号位的冠军个数

# Problem A - Dunai

## 题目大意

- TI 比赛有 $n$ 只冠军队伍，每个队伍由五个选手组成，每个选手分别打 1-5 号位中的一个，一个选手永远只打一个位置，不会打其他位置。有 $m$ 名选手组队，问最多能组多少队，满足每个队内至少有一名冠军选手。
- 关键词：签到，Easy

- 不考虑每个队都有冠军这一限制条件，可以得到的最大队伍数为 $\min\{p_i\}$，其中 $p_i$ 指打 $i$ 号位的选手人数。
- 先不考虑冠军这一角色，先将这些人组出最大队伍数支队伍，然后给每个人分配冠军角色。
- 最多可以分配 $\sum c_i$ 个冠军角色，其中 $c_i$ 指的是打 $i$ 号位的冠军个数
- 得到的队伍数再与 $\sum c_i$ 取最小值即为最终答案。

# Problem B - Recruitment

## 题目大意

- 给定一个长度为 $n$ 的形如 $a_1 + a_2 + ... + a_n$ 的式子，每次修改一个加号为乘号，修改 $n-1$ 次，共有 $n$ 个式子。给定 $n$ 个式子的结果 $\{s_n\}$，要求构造数组 $\{a_n\}$ 和每次修改的加号位置，使得 $n$ 个式子均符合给定的结果。无解输出-1。$n \leq 10^5, s_i \leq 10^9$ 。
- 关键词：BFS，Medium

# Problem B - Recruitment

- 首先，我们可以把每个式子乘起来的部分视为一项。可以发现，如果某一次操作是将 $x+1$ 改为了 $x \times 1$ ，这样的操作一定会使得总和减小 $1$，我们可以将其特殊判断掉。在忽略这样的操作之后，由于 $n$ 个式子的给定结果不超过 $10^9$ ，剩余的操作一定不超过 30 次。

# Problem B - Recruitment

- 首先，我们可以把每个式子乘起来的部分视为一项。可以发现，如果某一次操作是将 $x+1$ 改为了 $x \times 1$，这样的操作一定会使得总和减小 $1$，我们可以将其特殊判断掉。在忽略这样的操作之后，由于 $n$ 个式子的给定结果不超过 $10^9$，剩余的操作一定不超过 30 次。
- 观察发现，第 $i$ 个式子相当于要找到一个大小为 $n-i+1$ 的无序集合，它们的和是 $s_i$ 而积是 $s_n$。由于 $s_i \leq 10^9$，因此对于第 $i$ 个式子，符合条件的集合非常有限。打表之后可以发现，对于任何固定的 $i$、$s_i$ 和 $s_n$，最多也只有 2000 个左右可能的集合。

# Problem B - Recruitment

- 首先，我们可以把每个式子乘起来的部分视为一项。可以发现，如果某一次操作是将 $x+1$ 改为了 $x \times 1$ ，这样的操作一定会使得总和减小 1，我们可以将其特殊判断掉。在忽略这样的操作之后，由于 $n$ 个式子的给定结果不超过 $10^9$ ，剩余的操作一定不超过 30 次。
- 观察发现，第 $i$ 个式子相当于要找到一个大小为 $n-i+1$ 的无序集合，它们的和是 $s_i$ 而积是 $s_n$。由于 $s_i \leq 10^9$，因此对于第 $i$ 个式子，符合条件的集合非常有限。打表之后可以发现，对于任何固定的 $i$ 、$s_i$ 和 $s_n$ ，最多也只有 2000 个左右可能的集合。
- 因为忽略掉 $+1$ 变为 $\times 1$ 的操作后最多剩余 30 次操作，所以可以考虑暴力搜索所有可能的状态，总共要搜索的状态数不超过 $6 \times 10^4$ 。

# Problem B - Recruitment

- 考虑从最后的结果 $s_n$ 开始向前倒推，将最终的式子视为只有一项，每次相当于要把上一个式子里的某一项拆成两项，这两项的积为原本的数，且这两项的和与原本的数之差为一个定值，这里对于每个式子的枚举次数不超过 $\sqrt{10^9}$，暴力搜索的时间开销可以接受。接下来需要比较小心地实现这个搜索过程，可以考虑使用 std::map 去重。

# Problem B - Recruitment

- 考虑从最后的结果 $s_n$ 开始向前倒推，将最终的式子视为只有一项，每次相当于要把上一个式子里的某一项拆成两项，这两项的积为原本的数，且这两项的和与原本的数之差为一个定值，这里对于每个式子的枚举次数不超过 $\sqrt{10^9}$，暴力搜索的时间开销可以接受。接下来需要比较小心地实现这个搜索过程，可以考虑使用 std::map 去重。
- 注意前述时间复杂度只是一个非常宽松的上界，实际在搜索过程中无论是拆分式子时的枚举次数还是总共需要搜索的状态数都远小于前述数量级，较好的实现可以在 100ms 以内通过本题。

# Problem C - Grass

## 题目大意

- 给定平面上 $n$ 个点，要求选出其中 $5$ 个不同的点，并选定 $1$ 个中心点 $A$ 向其他 $4$ 个点 $B, C, D, E$ 连出 $4$ 条线段，要求这四条线段任意两条的交点都仅有中心点 $A$。
- 关键词：几何，Easy-Medium

# Problem C - Grass

- 本题的关键结论：选出的 $5$ 个点能构造出可行解，当且仅当这 $5$ 个点不全共线。

# Problem C - Grass

- 本题的关键结论：选出的 $5$ 个点能构造出可行解，当且仅当这 $5$ 个点不全共线。
- 可以使用分类讨论的方式证明这个结论的正确性，但实际代码中可以利用这个结论避免分类讨论。

# Problem C - Grass

- 本题的关键结论：选出的 $5$ 个点能构造出可行解，当且仅当这 $5$ 个点不全共线。
- 可以使用分类讨论的方式证明这个结论的正确性，但实际代码中可以利用这个结论避免分类讨论。
- 首先可以在 $O(n)$ 的时间复杂度下找到 $5$ 个不全共线的点（固定 $4$ 个点，枚举最后一个点，判断是否 $5$ 点共线），如果找不到，说明所有的点全部共线，此时本问题无可行解；找到 $5$ 个不全共线的点后，枚举这 $5$ 个点之一，判断该点作为中心点的解是否可行，由上述结论可知一定能找到一个可行解。

# Problem C - Grass

接下来是关键结论的证明：

# Problem C - Grass

接下来是关键结论的证明：

- 必要性：当 $5$ 个点全共线时，不难发现一定没有可行解，因此 $5$ 个点不全共线是有可行解的必要条件。

# Problem C - Grass

接下来是关键结论的证明：

- 必要性：当 $5$ 个点全共线时，不难发现一定没有可行解，因此 $5$ 个点不全共线是有可行解的必要条件。
- 充分性：$5$ 个点不全共线，可以分为以下几类：
  1. 任意 $3$ 点均不共线，此时选择任意一点作为中心均是可行解。
  2. 有且仅有 $4$ 点共线，此时选择不共线的一点作为中心即是可行解。
  3. 存在 $3$ 点共线，但没有 $4$ 点共线。将一组共线的 $3$ 点记为 $a, b, c$，其中 $b$ 在 $a, c$ 之间，另外 $2$ 点记为 $d, e$。此时又可以分为以下几类：

# Problem C - Grass

接下来是关键结论的证明：

- 必要性：当 $5$ 个点全共线时，不难发现一定没有可行解，因此 $5$ 个点不全共线是有可行解的必要条件。
- 充分性：$5$ 个点不全共线，可以分为以下几类：
  1. 任意 $3$ 点均不共线，此时选择任意一点作为中心均是可行解。
  2. 有且仅有 $4$ 点共线，此时选择不共线的一点作为中心即是可行解。
  3. 存在 $3$ 点共线，但没有 $4$ 点共线。将一组共线的 $3$ 点记为 $a, b, c$，其中 $b$ 在 $a, c$ 之间，另外 $2$ 点记为 $d, e$。此时又可以分为以下几类：
     1. 点 $e$ 不在射线 $da, db, dc$ 上，此时选择 $d$ 即是可行解。
     2. 点 $e$ 在线段 $da$ 或 $db$ 或 $dc$ 上，此时选择 $e$ 即是可行解。
     3. 点 $e$ 在 $da$ 或 $db$ 或 $dc$ 的延长线上，此时选择 $b$ 即是可行解。

# Problem C - Grass

- 本题原本的定位难度是 Easy，但验题时发现本题的过题情况并不理想，不少验题者在此题使用了分类讨论或者随机乱搞的方法，虽然这些方式也能通过，但正确率并不理想，因此本题的难度最终提升到了 Easy-Medium。

# Problem D - Sternhalma

## 题目大意

- 给定六边形棋盘每个格子的分数，询问若干初始的棋子摆放方式，问按照规则移除棋子最多得多少分。
- 移除棋子有两种方式，一种是直接移除一个棋子，不得分；另一种是用一个棋子跳过其相邻棋子，移除被跳过的棋子并且得分增加被移除棋子所在的格子的分数。
- 关键词：模拟，状压 DP，Easy

# Problem D - Sternhalma

- 使用 $19$ 位二进制数来存储棋盘状态（有棋子为 $1$，没有为 $0$），可以将所有状态映射为 $0$ 到 $2^{19}-1$ 的整数，然后根据规则可以找到状态间的转移关系。

# Problem D - Sternhalma

- 使用 $19$ 位二进制数来存储棋盘状态（有棋子为 $1$，没有为 $0$），可以将所有状态映射为 $0$ 到 $2^{19}-1$ 的整数，然后根据规则可以找到状态间的转移关系。
- 因为每进行一次操作一定会少一颗棋子，所以状态的转移一定是无后效性的，且每种状态的答案是固定的，因此可以使用动态规划求出每种状态的答案。

# Problem D - Sternhalma

- 使用 $19$ 位二进制数来存储棋盘状态（有棋子为 $1$，没有为 $0$），可以将所有状态映射为 $0$ 到 $2^{19}-1$ 的整数，然后根据规则可以找到状态间的转移关系。
- 因为每进行一次操作一定会少一颗棋子，所以状态的转移一定是无后效性的，且每种状态的答案是固定的，因此可以使用动态规划求出每种状态的答案。
- 可以先递推预处理出所有状态的答案，再 $O(1)$ 回答每个询问；也可以使用记忆化搜索的方式，两者时间复杂度相同。

# Problem D - Sternhalma

- 使用 $19$ 位二进制数来存储棋盘状态（有棋子为 $1$，没有为 $0$），可以将所有状态映射为 $0$ 到 $2^{19}-1$ 的整数，然后根据规则可以找到状态间的转移关系。
- 因为每进行一次操作一定会少一颗棋子，所以状态的转移一定是无后效性的，且每种状态的答案是固定的，因此可以使用动态规划求出每种状态的答案。
- 可以先递推预处理出所有状态的答案，再 $O(1)$ 回答每个询问；也可以使用记忆化搜索的方式，两者时间复杂度相同。
- 本题基本没有思维难度，主要考察六边形棋盘的代码处理，期望选手能使用尽可能简洁的代码快速通过此题。

# Problem E - Python Will be Faster than C++

## 题目大意

- 给出 $n$ 个 Python 版本的运行时间，依据最后两个版本的运行时间画出一次函数之后判断是否会在未来某个版本运行时间超过 C++。
- 关键词：签到，模拟，Easy

# Problem E - Python Will be Faster than C++

## 题目大意

- 给出 $n$ 个 Python 版本的运行时间，依据最后两个版本的运行时间画出一次函数之后判断是否会在未来某个版本运行时间超过 C++。
- 关键词：签到，模拟，Easy

- 注意到运行时间只有 $10^5$ 并且只有单组数据，所以可以直接 $10^5$ 模拟计算出各个版本的运行时间再来判断即可。
- 当 $a_{n-1} \leq a_n$ 时永远不会比 C++ 快。

# Problem F - Mooncake Delivery

## 题目大意

- 给出一张每个顶点 $v$ 带有颜色 $c_v$ 和点权 $w_v$ 的无向图。
- 顶点有激活和未激活两种状态，最开始每个顶点都是未激活的，激活一个顶点 $v$ 需要花费的代价为其点权 $w_v$。
- 你有一个棋子，棋子只能放在/移动到被激活的顶点上。
- 你可以随时选择一种颜色 $c$，回收这种颜色的所有已激活顶点的激活代价并将其全部设为未激活。
- 当然，$c$ 不能是你棋子当前所在顶点的颜色，因为棋子只能放在已激活顶点上。
- 现在对于图中的每一对顶点 $(s, t)$，询问若最开始图中所有点均未被激活，从 $s$ 出发走到 $t$ 所需的最小初始点权是多少。
- 关键词：图论，最短路，Medium

# Problem F - Mooncake Delivery

- 考虑答案形态。

# Problem F - Mooncake Delivery

- 考虑答案形态。
- 注意到每走到一个不同颜色的顶点上时你都可以回收其他颜色的所有顶点的点权。将答案（一条路径，视为顶点序列）按颜色分段，则其所需的最小初始点权为“每一段的点权和加上下一段的第一个顶点的点权”的最大值。

# Problem F - Mooncake Delivery

- 考虑答案形态。
- 注意到每走到一个不同颜色的顶点上时你都可以回收其他颜色的所有顶点的点权。将答案（一条路径，视为顶点序列）按颜色分段，则其所需的最小初始点权为“每一段的点权和加上下一段的第一个顶点的点权”的最大值。
- 于是我们可以的将图按颜色分为多个同色连通块，每块内部用 Floyd-Warshall 求两两之间最小点权路径。

# Problem F - Mooncake Delivery

- 然后通过枚举每段的起点，终点，以及下一段的起点建出一张带边权的有向图，边权即为连接起点与终点的最小点权路径的点权之和加上下一段起点的点权。

# Problem F - Mooncake Delivery

- 然后通过枚举每段的起点，终点，以及下一段的起点建出一张带边权的有向图，边权即为连接起点与终点的最小点权路径的点权之和加上下一段起点的点权。
- 最后在这张有向图上从每个顶点出发使用类似 Prim 的做法（每轮选择一个未连通的点权最小的出点）找出以每个点 $s$ 为根的有向瓶颈生成树。

# Problem F - Mooncake Delivery

- 然后通过枚举每段的起点，终点，以及下一段的起点建出一张带边权的有向图，边权即为连接起点与终点的最小点权路径的点权之和加上下一段起点的点权。
- 最后在这张有向图上从每个顶点出发使用类似 Prim 的做法（每轮选择一个未连通的点权最小的出点）找出以每个点 $s$ 为根的有向瓶颈生成树。
- 最终复杂度 $O(Tn^3)$。

# Problem G - Grade 2

## 题目大意

- 求

$$\sum_{k=l}^{r}[\gcd(kX \oplus X, X) = 1]$$

  多组询问。其中 $\oplus$ 为异或运算。
- 关键词：数论, Easy-Medium

# Problem G - Grade 2

## 题目大意

- 求

$$\sum_{k=l}^{r}[\gcd(kX \oplus X, X) = 1]$$

  多组询问。其中 $\oplus$ 为异或运算。
- 关键词：数论, Easy-Medium

- 本题有两种考虑方式。

# Problem G - Grade 2 - 解法一

- 此解法的动机为打表。

# Problem G - Grade 2 - 解法一

- 此解法的动机为打表。
- 容易发现 $\gcd(kX \oplus X, X)$ 的值有循环节，长度为 $2^{\lceil \log_2 X \rceil}$。

# Problem G - Grade 2 - 解法一

- 此解法的动机为打表。
- 容易发现 $\gcd(kX \oplus X, X)$ 的值有循环节，长度为 $2^{\lceil \log_2 X \rceil}$。
- 首先证明，对于 $\forall t \in \mathbb{N}, k < 2^{\lceil \log_2 X \rceil}$，有
  $\gcd(kX \oplus X, X) = \gcd((k + t \cdot 2^{\lceil \log_2 X \rceil})X \oplus X, X)$

# Problem G - Grade 2 - 解法一

## 证明

我们考虑有

$$a + b = (a \oplus b) + 2 \cdot (a \& b)$$

等式成立，其中 $\&$ 为按位与运算

# Problem G - Grade 2 - 解法一

## 证明

我们考虑有

$$a + b = (a \oplus b) + 2 \cdot (a \& b)$$

等式成立，其中 $\&$ 为按位与运算，那么

$$kX \oplus X = (k+1)X - 2 \cdot (kX \& X)$$

# Problem G - Grade 2 - 解法一

## 证明

我们考虑有

$$a+b=(a\oplus b)+2\cdot(a\&b)$$

等式成立，其中 $\&$ 为按位与运算，那么

$$kX\oplus X=(k+1)X-2\cdot(kX\&X)$$

同时

$$\begin{aligned}&(k+t\cdot 2^{\lceil\log_2 X\rceil})X\oplus X\\=&(k+1+t\cdot 2^{\lceil\log_2 X\rceil})X-2\cdot((k+t\cdot 2^{\lceil\log_2 X\rceil})X\&X)\end{aligned}$$

# Problem G - Grade 2 - 解法一

## 证明

使用更相减损术的初级版本 $\gcd(a,b)=\gcd(a,a-b)$ 我们总可以将要证的等式右侧

$$(k+t\cdot 2^{\lceil \log_2 X\rceil})X\oplus X$$

化为

$$(k+1)X-2\cdot((k+t\cdot 2^{\lceil \log_2 X\rceil})X\&X)$$

# Problem G - Grade 2 - 解法一

## 证明

使用更相减损术的初级版本 $\gcd(a,b)=\gcd(a,a-b)$ 我们总可以将要证的等式右侧

$$(k+t\cdot 2^{\lceil \log_2 X\rceil})X\oplus X$$

化为

$$(k+1)X-2\cdot((k+t\cdot 2^{\lceil \log_2 X\rceil})X\&X)$$

下面我们只需说明下式即可。

$$kX\&X=(k+t\cdot 2^{\lceil \log_2 X\rceil})X\&X$$

# Problem G - Grade 2 - 解法一

## 证明

我们用二进制形式来分别考虑等式的左右两端，等式左侧 $kX\&X$ 的二进制形式长度一定不会超过 $X$ 的二进制长度 $\lceil \log_2 X \rceil$。

# Problem G - Grade 2 - 解法一

## 证明

我们用二进制形式来分别考虑等式的左右两端，等式左侧 $kX\&X$ 的二进制形式长度一定不会超过 $X$ 的二进制长度 $\lceil \log_2 X \rceil$。我们展开等式右侧的括号

$$(tX \cdot 2^{\lceil \log_2 X \rceil} + kX)\&X$$

可以理解为 $tX$ 先左移 $\lceil \log_2 X \rceil$ 位后，再加上 $kX$，那么这个值的最后 $\lceil \log_2 X \rceil$ 位全部由 $kX$ 确定，再与上 $X$，结果一定等于 $kX\&X$。

# Problem G - Grade 2 - 解法一

## 证明

我们用二进制形式来分别考虑等式的左右两端，等式左侧 $kX\&X$ 的二进制形式长度一定不会超过 $X$ 的二进制长度 $\lceil \log_2 X \rceil$。我们展开等式右侧的括号

$$(tX \cdot 2^{\lceil \log_2 X \rceil} + kX)\&X$$

可以理解为 $tX$ 先左移 $\lceil \log_2 X \rceil$ 位后，再加上 $kX$，那么这个值的最后 $\lceil \log_2 X \rceil$ 位全部由 $kX$ 确定，再与上 $X$，结果一定等于 $kX\&X$。因此上述结论成立。

# Problem G - Grade 2 - 解法一

- 上述结论说明 $\gcd(kX \oplus X, X)$ 的结果是循环的，并以 $C = 2^{\lceil \log_2 X \rceil}$ 长度为循环节长度。

# Problem G - Grade 2 - 解法一

- 上述结论说明 $\gcd(kX \oplus X, X)$ 的结果是循环的，并以 $C = 2^{\lceil \log_2 X \rceil}$ 长度为循环节长度。
- 我们只需找出 $[0, C-1]$ 范围内的答案，之后根据循环节分块计算即可。

# Problem G - Grade 2 - 解法一

- 上述结论说明 $\gcd(kX \oplus X, X)$ 的结果是循环的，并以 $C = 2^{\lceil \log_2 X \rceil}$ 长度为循环节长度。
- 我们只需找出 $[0, C-1]$ 范围内的答案，之后根据循环节分块计算即可。
- 时间复杂度：$\mathcal{O}(X \log X + Q)$，空间复杂度：$\mathcal{O}(X)$。时间复杂度中的 $\log X$ 来自求 $\gcd$。

# Problem G - Grade 2 - 解法二

- 下面从 Möbius 反演角度考虑。

# Problem G - Grade 2 - 解法二

- 下面从 Möbius 反演角度考虑。
- 不妨转化为求

$$f(k)=\sum_{i=1}^{k}[\gcd(iX\oplus X,X)=1]$$

则对于每个询问，答案为 $f(r)-f(l-1)$。

# Problem G - Grade 2 - 解法二

考虑 Möbius 反演

# Problem G - Grade 2 - 解法二

## 考虑 Möbius 反演

$$f(k) = \sum_{i=1}^{k} [\gcd(iX \oplus X, X) = 1]$$

# Problem G - Grade 2 - 解法二

## 考虑 Möbius 反演

$$f(k) = \sum_{i=1}^{k} [\gcd(iX \oplus X, X) = 1]$$

$$= \sum_{i=1}^{k} \sum_{d \mid \gcd(iX \oplus X, X)} \mu(d)$$

# Problem G - Grade 2 - 解法二

## 考虑 Möbius 反演

$$
\begin{aligned}
f(k) &= \sum_{i=1}^{k} [\gcd(iX \oplus X, X) = 1] \\
&= \sum_{i=1}^{k} \sum_{d \mid \gcd(iX \oplus X, X)} \mu(d) \\
&= \sum_{d=1}^{X} \mu(d) \sum_{i=1}^{k} [d \mid \gcd(iX \oplus X, X)]
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

## 考虑 Möbius 反演

$$
\begin{aligned}
f(k) &= \sum_{i=1}^{k} [\gcd(iX \oplus X, X) = 1] \\
&= \sum_{i=1}^{k} \sum_{d \mid \gcd(iX \oplus X, X)} \mu(d) \\
&= \sum_{d=1}^{X} \mu(d) \sum_{i=1}^{k} [d \mid \gcd(iX \oplus X, X)] \\
&= \sum_{d \mid X} \mu(d) \sum_{i=1}^{k} [d \mid (iX \oplus X)]
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

## 考虑 Möbius 反演

$$
\begin{aligned}
f(k) &= \sum_{i=1}^{k} [\gcd(iX \oplus X, X) = 1] \\
&= \sum_{i=1}^{k} \sum_{d \mid \gcd(iX \oplus X, X)} \mu(d) \\
&= \sum_{d=1}^{X} \mu(d) \sum_{i=1}^{k} [d \mid \gcd(iX \oplus X, X)] \\
&= \sum_{d \mid X} \mu(d) \sum_{i=1}^{k} [d \mid (iX \oplus X)] \\
&= \sum_{d \mid X} \mu(d) \sum_{i=1}^{k} [(iX \oplus X) \bmod d = 0]
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

考虑 $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，且 $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$

# Problem G - Grade 2 - 解法二

**考虑** $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，**且** $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$ ，**则**

$$(iX \oplus X) \bmod d = \left[(2^{\lceil \log_2 X \rceil} \cdot t + c)X \oplus X\right] \bmod d$$

# Problem G - Grade 2 - 解法二

**考虑** $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，**且** $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$ ，**则**

$$
\begin{aligned}
(iX \oplus X) \bmod d &= \left[(2^{\lceil \log_2 X \rceil} \cdot t + c)X \oplus X\right] \bmod d \\
&= \left[(2^{\lceil \log_2 X \rceil} \cdot tX + cX) \oplus X\right] \bmod d
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

**考虑** $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，**且** $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$ ，**则**

$$
\begin{aligned}
(iX \oplus X) \bmod d &= \left[(2^{\lceil \log_2 X \rceil} \cdot t + c)X \oplus X\right] \bmod d \\
&= \left[(2^{\lceil \log_2 X \rceil} \cdot tX + cX) \oplus X\right] \bmod d \\
&= \left[\left(tX + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor\right) \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

**考虑** $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，**且** $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$ ，**则**

$$
\begin{aligned}
(iX \oplus X) \bmod d &= \left[(2^{\lceil \log_2 X \rceil} \cdot t + c)X \oplus X\right] \bmod d \\
&= \left[(2^{\lceil \log_2 X \rceil} \cdot tX + cX) \oplus X\right] \bmod d \\
&= \left[\left(tX + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor\right) \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d \\
&= \left[tX \cdot 2^{\lceil \log_2 X \rceil} + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

**考虑** $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，**且** $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$ ，**则**

$$
\begin{aligned}
(iX \oplus X) \bmod d &= \left[(2^{\lceil \log_2 X \rceil} \cdot t + c)X \oplus X\right] \bmod d \\
&= \left[(2^{\lceil \log_2 X \rceil} \cdot tX + cX) \oplus X\right] \bmod d \\
&= \left[\left(tX + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor\right) \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d \\
&= \left[tX \cdot 2^{\lceil \log_2 X \rceil} + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d \\
&= \left[tX \cdot 2^{\lceil \log_2 X \rceil} + (cX \oplus X)\right] \bmod d
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

**考虑** $i = 2^{\lceil \log_2 X \rceil} \cdot t + c$，**且** $t \in \mathbb{N}, c < 2^{\lceil \log_2 X \rceil}$ ，**则**

$$
\begin{aligned}
(iX \oplus X) \bmod d &= \left[(2^{\lceil \log_2 X \rceil} \cdot t + c)X \oplus X\right] \bmod d \\
&= \left[(2^{\lceil \log_2 X \rceil} \cdot tX + cX) \oplus X\right] \bmod d \\
&= \left[\left(tX + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor\right) \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d \\
&= \left[tX \cdot 2^{\lceil \log_2 X \rceil} + \left\lfloor \frac{cX}{2^{\lceil \log_2 X \rceil}} \right\rfloor \cdot 2^{\lceil \log_2 X \rceil} + \left[(cX \bmod 2^{\lceil \log_2 X \rceil}) \oplus X\right]\right] \bmod d \\
&= \left[tX \cdot 2^{\lceil \log_2 X \rceil} + (cX \oplus X)\right] \bmod d \\
&= (cX \oplus X) \bmod d
\end{aligned}
$$

# Problem G - Grade 2 - 解法二

以下解释中，记 $2^{\lceil \log_2 X \rceil} = L$。

- 上述推导中第二行到第三行，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此可以发现 $2^L \cdot tX + cX$ 的二进制由三部分组成：
  - 低 $L$ 位为 $cX$ 的低 $L$ 位

# Problem G - Grade 2 - 解法二

以下解释中，记 $2^{\lceil \log_2 X \rceil} = L$。

- 上述推导中第二行到第三行，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此可以发现 $2^L \cdot tX + cX$ 的二进制由三部分组成：
  - 低 $L$ 位为 $cX$ 的低 $L$ 位
  - 中间部分为 $tX$ 的低位加 $cX$ 的高位

# Problem G - Grade 2 - 解法二

以下解释中，记 $2^{\lceil \log_2 X \rceil} = L$。

- 上述推导中第二行到第三行，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此可以发现 $2^L \cdot tX + cX$ 的二进制由三部分组成：
  - 低 $L$ 位为 $cX$ 的低 $L$ 位
  - 中间部分为 $tX$ 的低位加 $cX$ 的高位
  - 高位部分为 $tX$ 的高位

# Problem G - Grade 2 - 解法二

以下解释中，记 $2^{\lceil \log_2 X \rceil} = L$。

- 上述推导中第二行到第三行，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此可以发现 $2^L \cdot tX + cX$ 的二进制由三部分组成：
  - 低 $L$ 位为 $cX$ 的低 $L$ 位
  - 中间部分为 $tX$ 的低位加 $cX$ 的高位
  - 高位部分为 $tX$ 的高位
- 因此考虑把 $cX$ 在二进制低 $L$ 位处分解，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此加 $cX$ 后低 $L$ 位不会产生任何进位，所以 $tX$ 的低位和 $cX$ 的高位可以直接相加。

# Problem G - Grade 2 - 解法二

以下解释中，记 $2^{\lceil \log_2 X \rceil} = L$。

- 上述推导中第二行到第三行，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此可以发现 $2^L \cdot tX + cX$ 的二进制由三部分组成：
  - 低 $L$ 位为 $cX$ 的低 $L$ 位
  - 中间部分为 $tX$ 的低位加 $cX$ 的高位
  - 高位部分为 $tX$ 的高位
- 因此考虑把 $cX$ 在二进制低 $L$ 位处分解，由于 $2^L \cdot tX$ 的低 $L$ 位均为 $0$，因此加 $cX$ 后低 $L$ 位不会产生任何进位，所以 $tX$ 的低位和 $cX$ 的高位可以直接相加。
- 由于 $X$ 只有 $L$ 位，因此异或仅对 $cX$ 的低 $L$ 位有贡献。

# Problem G - Grade 2 - 解法二

- 第四行到第五行，和式中后两项代表 $cX$ 的高位和 $cX$ 的低 $L$ 位异或 $X$。由于 $X$ 只有 $L$ 位，并不会对 $cX$ 的高位部分产生影响，因此可以直接合并。

# Problem G - Grade 2 - 解法二

- 第四行到第五行，和式中后两项代表 $cX$ 的高位和 $cX$ 的低 $L$ 位异或 $X$。由于 $X$ 只有 $L$ 位，并不会对 $cX$ 的高位部分产生影响，因此可以直接合并。
- 第五行到第六行，由于 $d \mid X$，因此 $X \bmod d = 0$。

# Problem G - Grade 2 - 解法二

- 第四行到第五行，和式中后两项代表 $cX$ 的高位和 $cX$ 的低 $L$ 位异或 $X$。由于 $X$ 只有 $L$ 位，并不会对 $cX$ 的高位部分产生影响，因此可以直接合并。
- 第五行到第六行，由于 $d \mid X$，因此 $X \bmod d = 0$。
- 由此可以证明 $(iX \oplus X) \bmod d$ 的结果以 $2^{\lceil \log_2 X \rceil}$ 长度为循环节。

# Problem G - Grade 2 - 解法二

- 第四行到第五行，和式中后两项代表 $cX$ 的高位和 $cX$ 的低 $L$ 位异或 $X$。由于 $X$ 只有 $L$ 位，并不会对 $cX$ 的高位部分产生影响，因此可以直接合并。
- 第五行到第六行，由于 $d \mid X$，因此 $X \bmod d = 0$。
- 由此可以证明 $(iX \oplus X) \bmod d$ 的结果以 $2^{\lceil \log_2 X \rceil}$ 长度为循环节。
- 我们只需计算一个循环节内部有多少 $c$ 满足 $(cX \oplus X) \bmod d = 0$ 即可。

# Problem G - Grade 2 - 解法二

- 第四行到第五行，和式中后两项代表 $cX$ 的高位和 $cX$ 的低 $L$ 位异或 $X$。由于 $X$ 只有 $L$ 位，并不会对 $cX$ 的高位部分产生影响，因此可以直接合并。
- 第五行到第六行，由于 $d \mid X$，因此 $X \bmod d = 0$。
- 由此可以证明 $(iX \oplus X) \bmod d$ 的结果以 $2^{\lceil \log_2 X \rceil}$ 长度为循环节。
- 我们只需计算一个循环节内部有多少 $c$ 满足 $(cX \oplus X) \bmod d = 0$ 即可。
- 对于多次询问，由于 $X$ 是固定的，因此可以对于每个 $d \mid X$ 且 $\mu(d) \neq 0$ 分别处理每一个询问。

# Problem G - Grade 2 - 解法二

- 第四行到第五行，和式中后两项代表 $cX$ 的高位和 $cX$ 的低 $L$ 位异或 $X$。由于 $X$ 只有 $L$ 位，并不会对 $cX$ 的高位部分产生影响，因此可以直接合并。
- 第五行到第六行，由于 $d \mid X$，因此 $X \bmod d = 0$。
- 由此可以证明 $(iX \oplus X) \bmod d$ 的结果以 $2^{\lceil \log_2 X \rceil}$ 长度为循环节。
- 我们只需计算一个循环节内部有多少 $c$ 满足 $(cX \oplus X) \bmod d = 0$ 即可。
- 对于多次询问，由于 $X$ 是固定的，因此可以对于每个 $d \mid X$ 且 $\mu(d) \neq 0$ 分别处理每一个询问。
- 时间复杂度：$\mathcal{O}(2^{\omega(X)}(X+Q))$，空间复杂度：$\mathcal{O}(X+Q)$。

# Problem H - Party Animals

## 题目大意

- 有一行人在玩石头剪刀布。每个人均有一个最喜欢的手势(石头剪刀布之一)，每次游戏每个人均会使用其最喜欢的手势。两个人玩完一轮之后输的那个人的最喜欢的手势会变成赢的那个人的手势。
- 现在给出这行人每个人的最喜欢的手势，你需要进行两种操作：
  1. 指定一个区间，该区间内每一对相邻的人从左往右依次进行游戏；
  2. 查询某个人最喜欢的手势。
- 关键词：数据结构，平衡树，Medium-Hard

# Problem H - Party Animals

- 需要进行以下几点观察：

# Problem H - Party Animals

- 需要进行以下几点观察：
  1. 平局则两方手势均无变化；

# Problem H - Party Animals

- 需要进行以下几点观察：
  1. 平局则两方手势均无变化；
  2. 若石头后面跟了一串剪刀，则那串剪刀都会变成石头；

# Problem H - Party Animals

- 需要进行以下几点观察：
  1. 平局则两方手势均无变化；
  2. 若石头后面跟了一串剪刀，则那串剪刀都会变成石头；
  3. 若一个剪刀后面跟了一个石头，则那个剪刀最后的那个人的手势会变成石头。

# Problem H - Party Animals

- 需要进行以下几点观察：
  1. 平局则两方手势均无变化；
  2. 若石头后面跟了一串剪刀，则那串剪刀都会变成石头；
  3. 若一个剪刀后面跟了一个石头，则那个剪刀最后的那个人的手势会变成石头。
- 如果左边的人赢了，则分割点要一直右移直到碰到区间边界或者其他手势的人。

# Problem H - Party Animals

- 需要进行以下几点观察：
  1. 平局则两方手势均无变化；
  2. 若石头后面跟了一串剪刀，则那串剪刀都会变成石头；
  3. 若一个剪刀后面跟了一个石头，则那个剪刀最后的那个人的手势会变成石头。
- 如果左边的人赢了，则分割点要一直右移直到碰到区间边界或者其他手势的人。
- 如果右边的人赢了，则分割点需要往左移一格。因此我们只需要使用一些数据结构来维护分割点的位置和类型。

# Problem H - Party Animals

- 对于操作 1，在碰到第一对左边赢的分割点之前，所有分割点左移一格。最左边的分割点移出范围时需考虑是否需要和前面的分割点合并。

# Problem H - Party Animals

- 对于操作 1，在碰到第一对左边赢的分割点之前，所有分割点左移一格。最左边的分割点移出范围时需考虑是否需要和前面的分割点合并。
- 在碰到第一对左边会赢的人之后，若其后第一个分割点在范围内，则将其删除并循环，否则将其移至范围尽头。

# Problem H - Party Animals

- 对于操作 1，在碰到第一对左边赢的分割点之前，所有分割点左移一格。最左边的分割点移出范围时需考虑是否需要和前面的分割点合并。
- 在碰到第一对左边会赢的人之后，若其后第一个分割点在范围内，则将其删除并循环，否则将其移至范围尽头。
- 因为分割点只会越来越少，且每执行一次第二步要么结束循环要么少一个分割点，所以可以使用平衡树结合懒标记来维护分割点来达成 $O((N+Q)\log N)$ 的总复杂度。

# Problem I - Dragon Bloodline

## 题目大意

- 合成一个龙蛋需要 $n$ 种精华，其中第 $i$ 种精华需要 $a_i$ 个单位。
- 有 $k$ 种工具龙，第 $j$ 种工具龙总共有 $b_j$ 条，且每单位时间能产出 $2^{j-1}$ 单位的任意一种精华。
- 你需要将每条工具龙分配去生产某种精华，并最大化每单位时间内能产生的龙蛋数量。
- 关键词：贪心，二分，Easy-Medium

# Problem I - Dragon Bloodline

- 考虑二分答案 $M$，问题转化为能否将每条工具龙分配去生产某种精华，使得负责生产第 $i$ 种精华的工具龙的单位时间产出之和达到 $M \times a_i$。

# Problem I - Dragon Bloodline

- 考虑二分答案 $M$，问题转化为能否将每条工具龙分配去生产某种精华，使得负责生产第 $i$ 种精华的工具龙的单位时间产出之和达到 $M \times a_i$。
- 观察到如果工作效率最高的工具龙的工作效率能够被完全利用，那么我们可以直接将它分配去生产对应的精华，理由如下：

# Problem I - Dragon Bloodline

- 考虑二分答案 $M$，问题转化为能否将每条工具龙分配去生产某种精华，使得负责生产第 $i$ 种精华的工具龙的单位时间产出之和达到 $M \times a_i$。
- 观察到如果工作效率最高的工具龙的工作效率能够被完全利用，那么我们可以直接将它分配去生产对应的精华，理由如下：
  - 设那条工具龙的工作效率为 $2^{j-1}$，且在这种分配方式中它被分配去生产第 $i$ 种精华。对于任何一个合法解，因为工具龙的工作效率都是 $2$ 的幂，且 $M \times a_i \geq 2^{j-1}$，所以我们总是能够在被分配去生产第 $i$ 种精华的工具龙中找出工作效率之和为 $2^{j-1}$ 的一组工具龙与其互换，因此总是存在一个最优解，那条工具龙被分配去生产了第 $i$ 种精华。

# Problem I - Dragon Bloodline

- 初始化 $c_i = M \times a_i$。我们按工作效率从高到低考虑每一种工具龙。对于工作效率为 $2^{j-1}$ 的工具龙，我们先将它们分配给所有 $c_i$ 大于等于 $2^{j-1}$ 的精华 $i$。如果龙的数量不够，则继续考虑下一种工具龙。否则我们设剩下 $b'_j > 0$ 条工具龙，并将这些工具龙全部分配去生产那些 $c_i$ 前 $b'_j$ 大的精华。

# Problem I - Dragon Bloodline

- 初始化 $c_i = M \times a_i$。我们按工作效率从高到低考虑每一种工具龙。对于工作效率为 $2^{j-1}$ 的工具龙，我们先将它们分配给所有 $c_i$ 大于等于 $2^{j-1}$ 的精华 $i$。如果龙的数量不够，则继续考虑下一种工具龙。否则我们设剩下 $b'_j > 0$ 条工具龙，并将这些工具龙全部分配去生产那些 $c_i$ 前 $b'_j$ 大的精华。
- 每一次二分可通过 std::nth_element 在 $O(kn)$ 内完成，因此为总复杂度 $O(Tnk \log C)$，其中 $C = \sum b_j 2^{j-1} < 2 \times 10^{15}$。

# Problem J - Eat, Sleep, Repeat

## 题目大意

- 给定 $n$ 个数 $a_1, a_2..., a_n$，以及 $k$ 条限制，每条形如 $limit[x_i] = y_i$。两个人进行游戏，每次选一个大于 0 的数并减去 1，如果出现某个数 $t$ 的出现次数 $cnt[t]$ 大于上限 $limit[t]$ 或者无法操作就输了，问谁赢。$1 \leq n, k \leq 10^5$，$0 \leq k \leq 10^5$ , $0 \leq x_i \leq 10^9$，$0 \leq y_i \leq n$。
- 关键词：博弈，贪心，Easy-Medium

# Problem J - Eat, Sleep, Repeat

- 可以将 $limit[-1]$ 视为 0，且任何不属于 $\{x_i\}$ 的数 $z$ 都有 $limit[z] = \infty$。可以发现，每个数都可以减小若干次，但是无法越过形如 $limit[x_i] = 0$ 的限制。

# Problem J - Eat, Sleep, Repeat

- 可以将 $limit[-1]$ 视为 0，且任何不属于 $\{x_i\}$ 的数 $z$ 都有 $limit[z] = \infty$。可以发现，每个数都可以减小若干次，但是无法越过形如 $limit[x_i] = 0$ 的限制。
- 按照这些个数上限为 $0$ 的限制进行分段，对于每一段内的数，令其尽可能减小，只要不超过个数限制即可。

# Problem J - Eat, Sleep, Repeat

- 可以将 $limit[-1]$ 视为 0，且任何不属于 $\{x_i\}$ 的数 $z$ 都有 $limit[z] = \infty$。可以发现，每个数都可以减小若干次，但是无法越过形如 $limit[x_i] = 0$ 的限制。
- 按照这些个数上限为 $0$ 的限制进行分段，对于每一段内的数，令其尽可能减小，只要不超过个数限制即可。
- 可以用 std::map 等数据结构来记录限制，由此可以计算出总共可行的减小次数，并判断其奇偶性即可知道获胜者。可以证明，总共可行的操作次数与双方的策略无关。

# Problem K - I Wanna Maker

## 题目大意

- 有一个集合包含 $[l, r]$ 内所有整数,$l, r > 0$, 现在你知道 $n$ 个条件:
  1. $1\ k\ x$: 这个集合存在 $k$ 个元素，并且这 $k$ 个元素和为 $x$;
  2. $2\ k\ x$: 这个集合不存在 $k$ 个元素或者没有 $k$ 个元素和为 $x$。
- 问你有多少个合法的集合?
- 关键词：思维，Medium

# Problem K - I Wanna Maker

- 在区间 $[l, r]$ 中选取 $k$ 个不同的整数求和，其最小值为 $S_{min} = l + (l+1) + \cdots + (l+k-1)$，最大值为 $S_{max} = (r-k+1) + (r-k+2) + \cdots + r$。并且，对于任意的整数 $S \in [S_{min}, S_{max}]$，一定能在区间 $[l, r]$ 中找到 $k$ 个不同的整数使其和为 $S$。

# Problem K - I Wanna Maker

- 在区间 $[l, r]$ 中选取 $k$ 个不同的整数求和，其最小值为 $S_{min} = l + (l+1) + \cdots + (l+k-1)$，最大值为 $S_{max} = (r-k+1) + (r-k+2) + \cdots + r$。并且，对于任意的整数 $S \in [S_{min}, S_{max}]$，一定能在区间 $[l, r]$ 中找到 $k$ 个不同的整数使其和为 $S$。
- 于是，我们可以将题目中的条件转化为：

# Problem K - I Wanna Maker

- 在区间 $[l, r]$ 中选取 $k$ 个不同的整数求和，其最小值为 $S_{min} = l + (l+1) + \cdots + (l+k-1)$，最大值为 $S_{max} = (r-k+1) + (r-k+2) + \cdots + r$。并且，对于任意的整数 $S \in [S_{min}, S_{max}]$，一定能在区间 $[l, r]$ 中找到 $k$ 个不同的整数使其和为 $S$。
- 于是，我们可以将题目中的条件转化为：
  1. $S_{min} \le x_i$ 且 $S_{max} \ge x_i$
  2. $S_{min} > x_i$ 或 $S_{max} < x_i$

# Problem K - I Wanna Maker

- 在区间 $[l, r]$ 中选取 $k$ 个不同的整数求和，其最小值为 $S_{min} = l + (l+1) + \cdots + (l+k-1)$，最大值为 $S_{max} = (r-k+1) + (r-k+2) + \cdots + r$。并且，对于任意的整数 $S \in [S_{min}, S_{max}]$，一定能在区间 $[l, r]$ 中找到 $k$ 个不同的整数使其和为 $S$。
- 于是，我们可以将题目中的条件转化为：
  1. $S_{min} \le x_i$ 且 $S_{max} \ge x_i$
  2. $S_{min} > x_i$ 或 $S_{max} < x_i$
- 将 $S_{min}, S_{max}$ 带入后化简，并考虑到 $0 < l \le r$，得：

# Problem K - I Wanna Maker

- 在区间 $[l, r]$ 中选取 $k$ 个不同的整数求和，其最小值为 $S_{min} = l + (l+1) + \cdots + (l+k-1)$，最大值为 $S_{max} = (r-k+1) + (r-k+2) + \cdots + r$。并且，对于任意的整数 $S \in [S_{min}, S_{max}]$，一定能在区间 $[l, r]$ 中找到 $k$ 个不同的整数使其和为 $S$。
- 于是，我们可以将题目中的条件转化为：
  1. $S_{min} \le x_i$ 且 $S_{max} \ge x_i$
  2. $S_{min} > x_i$ 或 $S_{max} < x_i$
- 将 $S_{min}, S_{max}$ 带入后化简，并考虑到 $0 < l \le r$，得：
  1. $0 < l \le \left\lfloor \dfrac{2x_i - (k-1)k}{2k} \right\rfloor$ 且 $r \ge \left\lceil \dfrac{2x_i + (k-1)k}{2k} \right\rceil$
  2. $l > \left\lfloor \dfrac{2x_i - (k-1)k}{2k} \right\rfloor$ 或 $r < \left\lceil \dfrac{2x_i + (k-1)k}{2k} \right\rceil$，且 $0 < l \le r$

# Problem K - I Wanna Maker

- 此时可以发现条件 1 在 $lOr$ 平面上的图像是某个无限高矩形，但条件 2 的图像不太规则。考虑到条件 1 中已隐含 $r-l+1 \geq k$，可将条件 2 改写为

# Problem K - I Wanna Maker

- 此时可以发现条件 1 在 $lOr$ 平面上的图像是某个无限高矩形，但条件 2 的图像不太规则。考虑到条件 1 中已隐含 $r-l+1 \geq k$，可将条件 2 改写为
  - 满足 $0 < l \leq \left\lfloor \frac{2x_i-(k-1)k}{2k} \right\rfloor$ 且 $r \geq \left\lceil \frac{2x_i+(k-1)k}{2k} \right\rceil$ 的区间不合法。

# Problem K - I Wanna Maker

- 此时可以发现条件 1 在 $lOr$ 平面上的图像是某个无限高矩形，但条件 2 的图像不太规则。考虑到条件 1 中已隐含 $r-l+1 \geq k$，可将条件 2 改写为
  - 满足 $0 < l \leq \left\lfloor \frac{2x_i-(k-1)k}{2k} \right\rfloor$ 且 $r \geq \left\lceil \frac{2x_i+(k-1)k}{2k} \right\rceil$ 的区间不合法。
- 即条件 2 表示某个无限高矩形不合法。

# Problem K - I Wanna Maker

- 此时可以发现条件 1 在 $lOr$ 平面上的图像是某个无限高矩形，但条件 2 的图像不太规则。考虑到条件 1 中已隐含 $r-l+1 \geq k$，可将条件 2 改写为
  - 满足 $0 < l \leq \left\lfloor \frac{2x_i-(k-1)k}{2k} \right\rfloor$ 且 $r \geq \left\lceil \frac{2x_i+(k-1)k}{2k} \right\rceil$ 的区间不合法。
- 即条件 2 表示某个无限高矩形不合法。
- 于是将条件 1 代表的区域求交，并去掉其与条件 2 代表的区域的交即可。做法有很多，如线段树、排序后单调栈等。另需注意只有一种条件的情况。

# Problem K - I Wanna Maker

- 此时可以发现条件 1 在 $lOr$ 平面上的图像是某个无限高矩形，但条件 2 的图像不太规则。考虑到条件 1 中已隐含 $r-l+1\geq k$，可将条件 2 改写为
  - 满足 $0<l\leq\left\lfloor\frac{2x_i-(k-1)k}{2k}\right\rfloor$ 且 $r\geq\left\lceil\frac{2x_i+(k-1)k}{2k}\right\rceil$ 的区间不合法。
- 即条件 2 表示某个无限高矩形不合法。
- 于是将条件 1 代表的区域求交，并去掉其与条件 2 代表的区域的交即可。做法有很多，如线段树、排序后单调栈等。另需注意只有一种条件的情况。
- 时间复杂度 $O(n\log n)$。

# Problem L - Novice Magician

## 题目大意

- 设 $m = 2^n$，初始数组为长度为 $m$ 的全 0 数组。使用 $k$ $(k \leq m)$ 次魔法，每次魔法会对数组中的 $m/2$ 个不同位置加上一个数。
- 设第 $i$ 次魔法的值为 $x_i$，对应的顺序为 $p_i = (p_{i,0}, p_{i,1}, \ldots, p_{i,m/2-1})$，那么数组中 $p_{i,j}$ 位置将会加上 $x_i + 2j$，即 $a_{p_{i,j}} \leftarrow a_{p_{i,j}} + x_i + 2j$。
- 构造一个使用魔法的方案，即构造每个魔法的值 $x_i$ 和对应使用顺序 $p_i$，使得最后这个数组为整数数组 $A$。要求 $x_i$ 为整数。
- 关键词：构造，Medium

# Problem L - Novice Magician

- 注意：以下题解中规定，矩阵所有下标，使用魔法次数等计次变量均从 $0$ 开始。

# Problem L - Novice Magician

- 注意：以下题解中规定，矩阵所有下标，使用魔法次数等计次变量均从 $0$ 开始。
- 若使用 $k$ 次魔法，则应满足如下等式

# Problem L - Novice Magician

- 注意：以下题解中规定，矩阵所有下标，使用魔法次数等计次变量均从 $0$ 开始。
- 若使用 $k$ 次魔法，则应满足如下等式

$$\sum_{i=0}^{k-1}\left(\frac{m}{2}x_i+\frac{m(m-2)}{4}\right)=\sum_{i=0}^{m-1}a_i$$

# Problem L - Novice Magician

- 注意：以下题解中规定，矩阵所有下标，使用魔法次数等计次变量均从 $0$ 开始。
- 若使用 $k$ 次魔法，则应满足如下等式

$$\sum_{i=0}^{k-1}\left(\frac{m}{2}x_i+\frac{m(m-2)}{4}\right)=\sum_{i=0}^{m-1}a_i$$

- 提出公因子

# Problem L - Novice Magician

- 注意：以下题解中规定，矩阵所有下标，使用魔法次数等计次变量均从 $0$ 开始。
- 若使用 $k$ 次魔法，则应满足如下等式

$$\sum_{i=0}^{k-1}\left(\frac{m}{2}x_i+\frac{m(m-2)}{4}\right)=\sum_{i=0}^{m-1}a_i$$

- 提出公因子

$$\frac{m}{2}\sum_{i=0}^{k-1}\left(x_i+\frac{m-2}{2}\right)=\sum_{i=0}^{m-1}a_i$$

# Problem L - Novice Magician

- 注意：以下题解中规定，矩阵所有下标，使用魔法次数等计次变量均从 $0$ 开始。
- 若使用 $k$ 次魔法，则应满足如下等式

$$\sum_{i=0}^{k-1}\left(\frac{m}{2}x_i+\frac{m(m-2)}{4}\right)=\sum_{i=0}^{m-1}a_i$$

- 提出公因子

$$\frac{m}{2}\sum_{i=0}^{k-1}\left(x_i+\frac{m-2}{2}\right)=\sum_{i=0}^{m-1}a_i$$

- 上式中 $m-2=2^n-2=2(2^{n-1}-1)$ 为偶数，则 $\frac{m-2}{2}$ 为整数。同时数组 $A$ 为整数数组，等式右侧也为整数。如果要满足 $x_i$ 为整数，则必须满足 $\frac{m}{2}\mid\sum_{i=0}^{m-1}a_i$，否则不能构造出全为整数的 $x_i$。

# Problem L - Novice Magician

- 对于有解情况有许多构造方法，下文介绍一种比较简单的方法。一共进行 $k$ 次操作，固定了每次魔法指定的对象和顺序后，数组中的每个元素加的数值中 $2j$ 部分是固定的，就可以把问题简化成每次操作只是选择一个数加上固定选择的对象上去，这可以方便地用矩阵形式来列出方程。

# Problem L - Novice Magician

- 对于有解情况有许多构造方法，下文介绍一种比较简单的方法。一共进行 $k$ 次操作，固定了每次魔法指定的对象和顺序后，数组中的每个元素加的数值中 $2j$ 部分是固定的，就可以把问题简化成每次操作只是选择一个数加上固定选择的对象上去，这可以方便地用矩阵形式来列出方程。
- 用矩阵 $F = f_{i,j}$ 表示魔法的策略，$f_{i,j}$ 表示第 $i$ 次魔法中数组第 $j$ 个元素将要加的值，那么 $a_j = \sum_{i=0}^{k-1} f_{i,j}$。实际上这一求和可以看成两个矩阵相乘，即全 $1$ 的 $k$ 维行向量右乘 $F$。

# Problem L - Novice Magician

$$
\begin{aligned}
A_{1\times m} &= \begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}_{1\times k} \begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,m-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{k-1,0} & f_{k-1,1} & \cdots & f_{k-1,m-1} \end{bmatrix}_{k\times m}
\end{aligned}
$$

# Problem L - Novice Magician

$$
\begin{aligned}
A_{1\times m} &= \begin{bmatrix} a_0 & a_1 & \dots & a_{m-1} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}_{1\times k} \begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,m-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{k-1,0} & f_{k-1,1} & \cdots & f_{k-1,m-1} \end{bmatrix}_{k\times m}
\end{aligned}
$$

- 设 $X = \begin{bmatrix} x_0 & x_1 & \dots & x_{k-1} \end{bmatrix}_{1\times k}$，并把 $F$ 中 $x_i$ 和与 $j$ 相关的项分离，考虑构造 $W, C$ 使得

$$XW + C = A$$

# Problem L - Novice Magician

$$A_{1\times m} = \begin{bmatrix} a_0 & a_1 & \dots & a_{m-1} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}_{1\times k} \begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,m-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{k-1,0} & f_{k-1,1} & \cdots & f_{k-1,m-1} \end{bmatrix}_{k\times m}$$

- 设 $X = \begin{bmatrix} x_0 & x_1 & \dots & x_{k-1} \end{bmatrix}_{1\times k}$，并把 $F$ 中 $x_i$ 和与 $j$ 相关的项分离，考虑构造 $W, C$ 使得

$$XW + C = A$$

- 其中 $W$ 第 $i$ 行将会有 $\frac{m}{2}$ 个 $1$，被使用魔法的位置为 $1$，其他位置为 $0$；$C$ 中第 $j$ 个元素为使用魔法时与 $j$ 有关的项的和。

# Problem L - Novice Magician

- 若构造好了策略 $F$，那么这道题就是求解方程 $XW + C = A$，即 $XW = A - C$，其中 $X$ 为 $1 \times k$ 行向量，$W$ 为 $k \times m$ 矩阵，$A - C$ 为 $1 \times m$ 行向量。

# Problem L - Novice Magician

- 若构造好了策略 $F$，那么这道题就是求解方程 $XW + C = A$，即 $XW = A - C$，其中 $X$ 为 $1 \times k$ 行向量，$W$ 为 $k \times m$ 矩阵，$A - C$ 为 $1 \times m$ 行向量。
- 考虑使用 $m$ 次魔法，对以每一个位置 $i$ $(1 \le i < m)$ 为起点长度为 $\frac{m}{2}$ 的区间做一次魔法，超出数组的部分从位置 $1$ 开始，最后再对区间 $[0, \frac{m}{2} - 1]$ 做一次魔法。可以发现这样构造的 $W$ 是可逆的，可以保证 $X$ 有唯一解。

# Problem L - Novice Magician

例如当 $n = 2$, 即 $m = 4$ 时,

# Problem L - Novice Magician

例如当 $n = 2$, 即 $m = 4$ 时,

$$W = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 4 & 2 & 2 \end{bmatrix}$$

# Problem L - Novice Magician

例如当 $n = 2$, 即 $m = 4$ 时,

$$W = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 4 & 2 & 2 \end{bmatrix}$$

对应的逆矩阵为

# Problem L - Novice Magician

例如当 $n = 2$, 即 $m = 4$ 时,

$$W = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 4 & 2 & 2 \end{bmatrix}$$

对应的逆矩阵为

$$W^{-1} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 1 \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

# Problem L - Novice Magician

例如当 $n = 3$, 即 $m = 8$ 时,

# Problem L - Novice Magician

例如当 $n = 3$, 即 $m = 8$ 时,

$$W = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 14 & 16 & 18 & 12 & 12 & 12 & 12 \end{bmatrix}$$

# Problem L - Novice Magician

对应的逆矩阵为

# Problem L - Novice Magician

对应的逆矩阵为

$$W^{-1} = \begin{bmatrix} -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 1 \\ \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & 0 \\ \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & 0 \end{bmatrix}$$

# Problem L - Novice Magician

再由

$$X = (A - C)\, W^{-1}$$

# Problem L - Novice Magician

再由

$$X = (A - C)\, W^{-1}$$

可得

$$x_i = \sum_{j=0}^{m-1} (a_j - c_j)(\, W^{-1})_{ji}$$

# Problem L - Novice Magician

再由

$$X = (A - C)\, W^{-1}$$

可得

$$x_i = \sum_{j=0}^{m-1} (a_j - c_j)(\, W^{-1})_{ji}$$

总结归纳一下可得

# Problem L - Novice Magician

再由

$$X = (A - C)\, W^{-1}$$

可得

$$x_i = \sum_{j=0}^{m-1} (a_j - c_j)(\, W^{-1})_{ji}$$

总结归纳一下可得

$$x_i = \begin{cases} d_0 & i = m-1 \\ -\frac{2}{m} \sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} + d_0 & i = \frac{m}{2} - 1 \\ -\frac{2}{m} \sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} & \text{else} \end{cases}$$

其中

$$d_i = a_i - c_i$$

# Problem L - Novice Magician

$$x_i = \begin{cases} d_0 & i = m-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} + d_0 & i = \frac{m}{2}-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} & \text{else} \end{cases}$$

# Problem L - Novice Magician

$$x_i = \begin{cases} d_0 & i = m-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} + d_0 & i = \frac{m}{2}-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} & \text{else} \end{cases}$$

- 其中 $st, ed$ 分别为做第 $i$ 次魔法时序列的最左端和最右端。进一步考察，由于 $\frac{m}{2} \mid \sum_{i=0}^{m-1} a_i$，且 $\sum_{i=0}^{m-1} c_i = \frac{m^2(m-2)}{4}$，因此可以保证所有的 $x_i$ 均为整数。直接考虑构造每次均对选中的序列顺序释放魔法，可以对任意有解的情况求出一组解。实际上，对选中的序列以随机顺序释放魔法也是可行的，只不过需要统计 $c_i$

# Problem L - Novice Magician

$$x_i = \begin{cases} d_0 & i = m-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} + d_0 & i = \frac{m}{2}-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} & \text{else} \end{cases}$$

- 其中 $st, ed$ 分别为做第 $i$ 次魔法时序列的最左端和最右端。进一步考察，由于 $\frac{m}{2} \mid \sum_{i=0}^{m-1} a_i$，且 $\sum_{i=0}^{m-1} c_i = \frac{m^2(m-2)}{4}$，因此可以保证所有的 $x_i$ 均为整数。直接考虑构造每次均对选中的序列顺序释放魔法，可以对任意有解的情况求出一组解。实际上，对选中的序列以随机顺序释放魔法也是可行的，只不过需要统计 $c_i$
- 关于逆矩阵的求解，由于矩阵十分特殊，可以通过高斯消元求逆矩阵然后总结归纳逆矩阵的通用表达形式，也可以直接尝试写优化后的高斯消元直接求逆矩阵。

# Problem L - Novice Magician

$$x_i = \begin{cases} d_0 & i = m-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} + d_0 & i = \frac{m}{2}-1 \\ -\frac{2}{m}\sum_{j=0}^{m-1} d_j + d_{st} + d_{ed} & \text{else} \end{cases}$$

- 其中 $st, ed$ 分别为做第 $i$ 次魔法时序列的最左端和最右端。进一步考察，由于 $\frac{m}{2} \mid \sum_{i=0}^{m-1} a_i$，且 $\sum_{i=0}^{m-1} c_i = \frac{m^2(m-2)}{4}$，因此可以保证所有的 $x_i$ 均为整数。直接考虑构造每次均对选中的序列顺序释放魔法，可以对任意有解的情况求出一组解。实际上，对选中的序列以随机顺序释放魔法也是可行的，只不过需要统计 $c_i$
- 关于逆矩阵的求解，由于矩阵十分特殊，可以通过高斯消元求逆矩阵然后总结归纳逆矩阵的通用表达形式，也可以直接尝试写优化后的高斯消元直接求逆矩阵。
- 此外，还有许多其他的构造方法，如分治等也可以通过此题。

# Problem M - String Master

## 题目大意

- 有一个无限长的字符串 S，由 $0, 1, 2, 3, \ldots$ 的二进制构成 (即 011011100101....), 问字符串 $S[l, r]$ 子串内，长度为 $n$ 的子串中字典序最大的串。
- 关键词：杂题，Medium-Hard

# Problem M - String Master

- 较为显然但是并不容易证明的是，任意两个子串的 LCP 的长度是 $O(\log n)$。

# Problem M - String Master

- 较为显然但是并不容易证明的是，任意两个子串的 LCP 的长度是 $O(\log n)$。
- 首先，除去边界情况，最优解第一个字符一定是 1，前一个字符一定是 0。边界情况暴力枚举即可。

# Problem M - String Master

- 较为显然但是并不容易证明的是，任意两个子串的 LCP 的长度是 $O(\log n)$。
- 首先，除去边界情况，最优解第一个字符一定是 1，前一个字符一定是 0。边界情况暴力枚举即可。
- 考虑所有可能是答案的起点，首先枚举起点所在的二进制长度，然后枚举起点是在二进制的第几位开始，总共有 $O(\log^2 n)$ 个。

# Problem M - String Master

- 较为显然但是并不容易证明的是，任意两个子串的 LCP 的长度是 $O(\log n)$。
- 首先，除去边界情况，最优解第一个字符一定是 1，前一个字符一定是 0。边界情况暴力枚举即可。
- 考虑所有可能是答案的起点，首先枚举起点所在的二进制长度，然后枚举起点是在二进制的第几位开始，总共有 $O(\log^2 n)$ 个。
- 按位贪心，枚举下一个数取 1 是否在原串内能出现，具体的做法是询问一个区间内是否存在二进制数，有一些位置确定了 0 和 1，另一些位置任意，问是否存在。能则放 1，不能则放 0，只要枚举 $O(\log n)$ 就只剩下了一个起点，所以这部分的时间复杂度是 $O(\log^3 n)$。

# Problem M - String Master

- 较为显然但是并不容易证明的是，任意两个子串的 LCP 的长度是 $O(\log n)$。
- 首先，除去边界情况，最优解第一个字符一定是 1，前一个字符一定是 0。边界情况暴力枚举即可。
- 考虑所有可能是答案的起点，首先枚举起点所在的二进制长度，然后枚举起点是在二进制的第几位开始，总共有 $O(\log^2 n)$ 个。
- 按位贪心，枚举下一个数取 1 是否在原串内能出现，具体的做法是询问一个区间内是否存在二进制数，有一些位置确定了 0 和 1，另一些位置任意，问是否存在。能则放 1，不能则放 0，只要枚举 $O(\log n)$ 就只剩下了一个起点，所以这部分的时间复杂度是 $O(\log^3 n)$。
- 最后，将全部可能是答案的起点暴力比较，一共 $O(log^2 n)$ 个可能的答案起点，每次比较最大复杂度是 LCP 长度 $O(logn)$，所以最后总复杂度为 $O(\log^3 n)$。