字符串 + 杂题分享

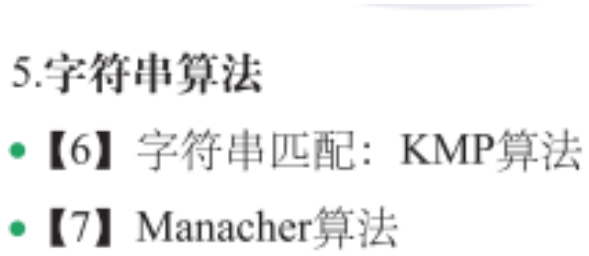
max0810

cdqz

2025 年 10 月 17 日

前言

本来是计划是讲字符串，但是：



所以可能很快就讲完了，所以我自己又准备了一些杂题，主要以 Ad-hoc 为主，然后还会有一些构造博弈类的题目。

希望大家听完后能有收获！

Contents

Contents

1

字符串

介绍

字符串是 OI 中一个比较重要的板块，但是因为 NOIP 范围内的内容太少，而且算法都比较简单，没有什么比较高深的应用。

基础的定义，比如子串，子序列，字典序等就不再过多赘述。

定义

给定一个长度为 n 的字符串 S，设 $f(S)$ 表示 S 的哈希值，一般有以下两种哈希方式：

- $f(S) = \sum_{i=1}^{n} B^{i-1} S_i \pmod{M}$
- $f(S) = \sum_{i=1}^{n} B^{n-i} S_i \pmod{M}$

其中 B, M 一般都是比较大的质数，比如 $2^{61} - 1$，或者也可以使用自然溢出，即 $M = 2^{64}$。有些时候为了提高哈希正确率，也可以用双哈希。

哈希冲突

设 $p(n,d)$ 表示 n 个字符串，取值范围为 d 的哈希冲突概率，则：

$$p(n,d)=1-\frac{d!}{d^n(d-n)!}\approx 1-\exp(-\frac{n(n-1)}{2d})$$

根据生日悖论，n 在 $\mathcal{O}(\sqrt{d})$ 时 $p(n,d)\approx\frac{1}{2}$，所以在哈希时需要保证模数远大于字符串数量的平方级别。
如果是双哈希，可以看成 d 是两个模数的 lcm。

哈希冲突

设 $p(n,d)$ 表示 n 个字符串，取值范围为 d 的哈希冲突概率，则：

$$p(n,d)=1-\frac{d!}{d^n(d-n)!}\approx 1-\exp(-\frac{n(n-1)}{2d})$$

根据生日悖论，n 在 $\mathcal{O}(\sqrt{d})$ 时 $p(n,d)\approx\frac{1}{2}$，所以在哈希时需要保证模数远大于字符串数量的平方级别。
如果是双哈希，可以看成 d 是两个模数的 lcm。

接下来介绍一下如何卡哈希。

卡哈希

如果模数很小的话，比如 10^9+7，直接构造大量字符串即可。

卡哈希

如果模数很小的话，比如 10^9+7，直接构造大量字符串即可。

如果是自然溢出的话，分 B 的奇偶性讨论。如果 B 是偶数，那么用一个映射出来是偶数的字符，放 64 个即可。否则，我们有如下构造方式：

- 设 $s_1=a$，设 $!s$ 为将 s 中的所有字符反转后的结果，即 a 变 b，b 变 a。
- $\forall i>1$，构造 $s_i=s_{i-1}+!s_{i-1}$，s_{12} 即为所求。

证明见：oi-wiki。

哈希应用

- 字符串匹配。

哈希应用

- 字符串匹配。
- 允许 k 次失配的字符串匹配，$n, m \leq 10^6, k \leq 5$。

哈希应用

- 字符串匹配。
- 允许 k 次失配的字符串匹配，$n, m \leq 10^6, k \leq 5$。
- 求最长回文子串，可以线性。

哈希应用

- 字符串匹配。
- 允许 k 次失配的字符串匹配，$n, m \leq 10^6, k \leq 5$。
- 求最长回文子串，可以线性。
- 最长公共子串（二分长度 + 匹配）。

哈希应用

- 字符串匹配。
- 允许 k 次失配的字符串匹配，$n, m \leq 10^6, k \leq 5$。
- 求最长回文子串，可以线性。
- 最长公共子串（二分长度 + 匹配）。
- 求字符串中不同子串数量，但是 $\mathcal{O}(n^2)$。

一些题目

- P3370 【模板】字符串哈希
- P3167 [CQOI2014] 通配符匹配
- P3809 【模板】后缀排序
- CF504E Misha and LCP on Tree
- P3501 [POI 2010] ANT-Antisymmetry
- CF580E Kefa and Watch
- 1476. sort
- P10634 BZOJ2372 music
- 1102. 串串串

算法

参考 oi-wiki。

一些应用：

- 字符串匹配。
- 求字符串最小周期（分是否能刚好拼完两种情况）。
- 求每个前缀出现次数。

例题

- P3375 【模板】KMP。
- P4391 [BalticOI 2009] Radio Transmission 无线传输
- UVA1328 Period
- P4824 [USACO15FEB] Censoring S
- P3435 [POI 2006] OKR-Periods of Words
- 1201. 后缀
- P2375 [NOI2014] 动物园

算法

求字符串的最长回文子串。

因为字符串只有 $\mathcal{O}(n)$ 个本质不同的回文串，所以可以线性求解。

算法流程不过多讲解，可以参考oi-wiki。

算法

求字符串的最长回文子串。

因为字符串只有 $\mathcal{O}(n)$ 个本质不同的回文串，所以可以线性求解。

算法流程不过多讲解，可以参考oi-wiki。

注意 Manacher 继承信息的时候可以暴力缩短，复杂度还是线性的。

例题

- P3805 【模板】manacher
- P3501 [POI 2010] ANT-Antisymmetry
- P4555 [国家集训队] 最长双回文串
- P4287 [SHOI2011] 双倍回文
- P11749 「TPOI-1C」Standard Problem.
- CF1827C Palindrome Partition
- AT_nikkei2019_2_final_h 逆にする関数（1539. 回文）
- P12977 泪雨 Namid[A]me

Contents

2

杂题选讲

介绍

主要讲贪心、构造、博弈，可以统称 Ad-hoc。

Ad-hoc，指没有确切的算法，主要是找各种性质的题目。一般遇到此类型的题目，都是手玩各种各样的数据，然后找规律，也可以用打表之类的方法。

具体的 trick 在每个板块内介绍

这里主要讲一些 ATCoder 上的题目。

介绍

对于贪心类的题目，一般是找最优解的性质，然后尝试去证明，一般可以使用调整法。

对于构造类题目，如果题目给的自由度太大，可以尝试自己增加一些条件，然后再去做。更常见的是直接打表找性质。

[AGC072A] Rhythm Game

给定 n 个按钮，第 i 个按钮会在 t_i 秒出现在坐标 x_i，每个按钮会在出现 $d+0.5$ 秒后消失。玩家从坐标 0 开始，以每秒至多 1 的速度在坐标轴上移动，可以以任意顺序按下所有按钮，每次按完按钮后需要回到坐标 0，求能否按完 n 个按钮。

$n \le 5000, x_i, t_i, d \le 10^{12}$。

[AGC072A] Rhythm Game 题解

将一个按钮转化为在区间 $[t_i - x_i, t_i + x_i + d]$ 中选择一个长度为 $2x_i$ 的区间，要求所有选择的区间不交。于是可以将问题转化为有 n 个任务，每个任务是 (l_i, r_i, c_i)，表示任务在 l_i 时间派出，需要 c_i 时间完成，并且需要在 r_i 前完成。考虑如果所有 $l_i = 0$ 怎么做，显然可以贪心按 r_i 排序，然后依次完成每个任务，因为如果有两个任务的完成顺序与 r_i 大小关系不符，则交换这两个任务一定不劣。

[AGC072A] Rhythm Game 题解

将一个按钮转化为在区间 $[t_i - x_i, t_i + x_i + d]$ 中选择一个长度为 $2x_i$ 的区间，要求所有选择的区间不交。于是可以将问题转化为有 n 个任务，每个任务是 (l_i, r_i, c_i)，表示任务在 l_i 时间派出，需要 c_i 时间完成，并且需要在 r_i 前完成。考虑如果所有 $l_i = 0$ 怎么做，显然可以贪心按 r_i 排序，然后依次完成每个任务，因为如果有两个任务的完成顺序与 r_i 大小关系不符，则交换这两个任务一定不劣。

于是我们将所有任务按 r_i 排序。现在有了 l_i 的限制，也就可能想完成 r_i 最小的任务时还没派出，所以会先去完成其他任务。但是注意到，如果完成了第 j 个任务，那么所有 $i < j$ 的任务都已经派出，因为 $t_j + x_j \geq t_i + x_i \geq t_i - x_i$。

[AGC072A] Rhythm Game 题解

于是我们的策略一定是，设当前已经完成了前 i 个任务，然后完成一个任务 j，接着依次完成任务 $[i+1, j-1]$。

[AGC072A] Rhythm Game 题解

于是我们的策略一定是，设当前已经完成了前 i 个任务，然后完成一个任务 j，接着依次完成任务 $[i+1, j-1]$。

于是就可以 dp，设 f_i 表示完成前 i 个任务的最早时间，然后枚举 f_i 从 f_j 转移过来，设 $t = \max(f_j, l_i) + c_i$，首先要求 $t \le r_i$。然后依次枚举 $k \in [j+1, i-1]$，需要满足 $t + c_{j+1} + c_{j+2} + \ldots + c_k \le r_k$，记 c_i 前缀和为 s_i，就是 $t - s_j \le r_k - s_k$。那么从大到小枚举 j 的同时记录 $r_k - s_k$ 的最小值即可 $\mathcal{O}(1)$ 判断，总复杂度为 $\mathcal{O}(n^2)$。

[AGC063C] Add Mod Operations

给定两个长度为 n 的序列 a, b，判断能否用至多 n 次操作让 a 变成 b，操作如下：

- 选择两个整数满足 $0 \le x < y \le 10^{18}$，将所有 a_i 变为 $(a_i + x) \bmod y$

如果有解需要构造。$n \le 1000, 0 \le a_i, b_i \le 10^9$

[AGC063C] Add Mod Operations 题解

首先有解的一个必要条件为对于所有相同 a_i，它们对应的 b_i 也需要相同。判了这个之后我们将相同的 a_i 缩起来并按 a_i 排序。考虑如下操作，选择某个 x 和 $y = \max\{a_i\} + x$，将所有 a_i 画到数轴上，这相当于将 a_n 设为 0 并将其余 a_i 都向又平移 x。我们重复以上操作 $n-1$ 次，数轴上的顺序为 $a_2, a_3 \dots a_n, a_1$，满足 $a_2 = 0$ 并且任意两个 a_i 之间的距离都是我们可以决定的。

[AGC063C] Add Mod Operations 题解

首先有解的一个必要条件为对于所有相同 a_i，它们对应的 b_i 也需要相同。判了这个之后我们将相同的 a_i 缩起来并按 a_i 排序。考虑如下操作，选择某个 x 和 $y = \max\{a_i\} + x$，将所有 a_i 画到数轴上，这相当于将 a_n 设为 0 并将其余 a_i 都向又平移 x。我们重复以上操作 $n-1$ 次，数轴上的顺序为 $a_2, a_3 \dots a_n, a_1$，满足 $a_2 = 0$ 并且任意两个 a_i 之间的距离都是我们可以决定的。

现在 b_i 是乱序的不太好做，我们可以设一个 $inf = 3 \times 10^9$，然后最后一步模 inf，然后将每个 b_i 增加 inf 直到所有 b_i 按 $b_2, b_3 \dots, b_n, b_1$ 的顺序排序。那么最后一步的操作就是 $x = b_2, y = inf$，这样子 a_2 会由 0 变为 b_2。然后我们算出 $b_3 - b_2, \dots b_n - b_{n-1}, b_1 - b_n$，于是就可以推出我们每一步中的 x 是多少了，注意要有 $b_1 - b_n \geq a_1$，调大 b_1 即可。

[AGC068B] 01 Graph Construction

如果两个 01 串 S, T 的 0 和 1 的数量分别相等，那么设 S, T 生成的图为：

- 将 S 中的 0 和 T 中的 0 一一对应连边。
- 将 S 中的 1 和 T 中的 1 一一对应连边。

给定一个长度为 n 的序列 a，你需要构造两个 01 串 S, T，满足：

- S, T 的 0 和 1 的数量分别相等。
- $|S| \leq 10^5$。
- S, T 生成的图中 $\forall 1 \leq i, j \leq n$，$i, j$ 在同一连通块内当且仅当 $a_i = a_j$。

$n \leq 100$

[AGC068B] 01 Graph Construction 题解

初始时 S 全为 0，T 全为 1，有个两个排列 p, q 初始都为 $1 \sim n$。那么将 S, T 后面都添加一个 0 就相当于将 p 循环移位一次；将 S 后添加一个 1，T 后添加一个 0 就相当于将 p_1, q_1 连边，然后两个排列都删去第一位。

[AGC068B] 01 Graph Construction 题解

初始时 S 全为 0，T 全为 1，有个两个排列 p, q 初始都为 $1 \sim n$。那么将 S, T 后面都添加一个 0 就相当于将 p 循环移位一次；将 S 后添加一个 1，T 后添加一个 0 就相当于将 p_1, q_1 连边，然后两个排列都删去第一位。

于是我们可以在 $\mathcal{O}(n^2)$ 次数内让每个 i 都连向 p_i，其中 p_i 是某个排列。那么将每个连通块看成一个置换环即可。

[AGC059D] Distinct Elements on Subsegments

给定一个长度为 n 的序列 b 和整数 k，构造一个长为 $n+k-1$ 的正整数序列 a，满足 $\forall 1 \le i \le n$，b_i 等于 $a_i, a_{i+1}, \ldots, a_{i+k-1}$ 中不同的数的个数，或者报告无解。

$n, k \le 2 \times 10^5$

[AGC059D] Distinct Elements on Subsegments 题解

首先需要满足 $|b_i - b_{i+1}| \le 1$，我们考虑 b_i, b_{i+1} 之间的变化代表什么。设 l_i 表示 $a_{i-k+1\sim i-1}$ 中是否所有数都和 a_i 不同，r_i 表示 $a_{i+1\sim i+k-1}$ 中是否所有数都和 a_i 不同，那么有 $b_i + l_{i+k} - r_i = b_{i+1}$。所以如果 $b_i \neq b_{i+1}$，我们就可以确定出 r_i, l_{i+k} 的值，否则只能确定 $r_i = l_{i+k}$。并且还有 $\sum_{i=1}^{k} l_i = b_1$，$\sum_{i=n-k+1}^{n} r_i = b_n$。

[AGC059D] Distinct Elements on Subsegments 题解

首先需要满足 $|b_i - b_{i+1}| \le 1$，我们考虑 b_i, b_{i+1} 之间的变化代表什么。设 l_i 表示 $a_{i-k+1\sim i-1}$ 中是否所有数都和 a_i 不同，r_i 表示 $a_{i+1\sim i+k-1}$ 中是否所有数都和 a_i 不同，那么有 $b_i + l_{i+k} - r_i = b_{i+1}$。所以如果 $b_i \neq b_{i+1}$，我们就可以确定出 r_i, l_{i+k} 的值，否则只能确定 $r_i = l_{i+k}$。并且还有 $\sum_{i=1}^{k} l_i = b_1$，$\sum_{i=n-k+1}^{n} r_i = b_n$。

考虑什么样的序列 l, r 是合法的，即存在至少一个 a 对应 l, r。我们求出 nex_i, pre_i 表示 a_i 的前驱、后继，发现知道了前驱后继就可以确定 l, r，我们相当于知道了每个数前驱后继离自己的距离是否 $< k$。

[AGC059D] Distinct Elements on Subsegments 题解

而因为前驱后继是一一对应的，所以如果某个 $r_i = 0$，那么 $l_{nex_i} = 0$，反之同理。所以 l 中的每个 0 和 r 中的 0 是一一对应的，并且假设 l_i 对应了 r_j，则需要满足 $1 \le i - j < k$。所以我们一一对应的方式一定是 l 的第 i 个 0 匹配 r 的第 i 个 0，否则交换了一定更优。

[AGC059D] Distinct Elements on Subsegments 题解

而因为前驱后继是一一对应的，所以如果某个 $r_i = 0$，那么 $l_{nex_i} = 0$，反之同理。所以 l 中的每个 0 和 r 中的 0 是一一对应的，并且假设 l_i 对应了 r_j，则需要满足 $1 \le i - j < k$。所以我们一一对应的方式一定是 l 的第 i 个 0 匹配 r 的第 i 个 0，否则交换了一定更优。

设 pl_i, pr_i 表示 l, r 中第 i 个 0 的位置，则一组 l, r 合法的条件为 l, r 中 0 的个数相等，且 $1 \le pl_i - pr_i < k$。因为限制条件都是两个值相等，所以 l, r 中 0 的个数是否相等是已经确定好了的。接下来我们要设置还没确定的位置使得尽可能满足条件。

[AGC059D] Distinct Elements on Subsegments 题解

假设对于某个 i 有 $r_i = l_{i+k}$，如果 $b_i = b_{i+1} = k$，那么一定有 $r_i = l_{i+k} = 1$。否则，将 r_i, l_{i+k} 全都设置为 0 一定不劣，因为此时中间至少有两个数相同，即至少一个 l 和 r 为 0。我们开始先将所有 0 都匹配上，那么将 r_i, l_{i+k} 设为 0 就是调整中间的一些匹配，原来满足条件的现在依然是满足条件的，而如果原来中间有不能匹配的现在就可能可以匹配了，所以这样做一定不劣。

[AGC059D] Distinct Elements on Subsegments 题解

假设对于某个 i 有 $r_i = l_{i+k}$，如果 $b_i = b_{i+1} = k$，那么一定有 $r_i = l_{i+k} = 1$。否则，将 r_i, l_{i+k} 全都设置为 0 一定不劣，因为此时中间至少有两个数相同，即至少一个 l 和 r 为 0。我们开始先将所有 0 都匹配上，那么将 r_i, l_{i+k} 设为 0 就是调整中间的一些匹配，原来满足条件的现在依然是满足条件的，而如果原来中间有不能匹配的现在就可能可以匹配了，所以这样做一定不劣。

于是我们直接将所有这样的 l_{i+k}, r_i 设为 0，而对于 b_1，一定是将 $1 \sim b_1$ 设为 1，因为这些位置前面的数不足 k 个，所以 0 越靠近中间越好，b_n 同理。设置完之后判断一下即可，最后构造就是从前往后枚举，如果有匹配的前驱就设为前驱，否则就设为一个新的数。复杂度 $\mathcal{O}(n)$。

[AGC057C] Increment or Xor

给定 n 和一个 $0 \sim 2^n - 1$ 的排列 $a_0, \ldots, a_{2^n-1}$，你每次可以执行以下两种操作：

- 对所有 i，执行 $a_i \leftarrow (a_i + 1) \bmod 2^n$
- 选择一个 x，对所有 i，执行 $a_i \leftarrow a_i \oplus x$

判断能否通过若干次操作使得所有 $a_i = i$，如果可以，还需要构造一种操作数不超过 10^6 的方案。

$n \leq 18$

[AGC057C] Increment or Xor 题解

考虑又有 $+1$ 又有异或，于是我们在反 trie 上考虑。将所有数插到反 trie 中，就是一棵深度为 n 的满二叉树，记 tr_i 表示 i 二进制反过来的数，pos_i 表示 i 在 a 中的位置。于是我们的目标是将 pos_{tr_i} 变为 tr_i，$+1$ 操作就是对最右边一条链上的所有点依次做交换左右儿子，异或操作就是对一层的所有点做交换左右儿子，不妨设为从 p_i 变为 q_i。

[AGC057C] Increment or Xor 题解

考虑又有 $+1$ 又有异或，于是我们在反 trie 上考虑。将所有数插到反 trie 中，就是一棵深度为 n 的满二叉树，记 tr_i 表示 i 二进制反过来的数，pos_i 表示 i 在 a 中的位置。于是我们的目标是将 pos_{tr_i} 变为 tr_i，$+1$ 操作就是对最右边一条链上的所有点依次做交换左右儿子，异或操作就是对一层的所有点做交换左右儿子，不妨设为从 p_i 变为 q_i。

于是我们可以建出 p_i 和 q_i 的 trie 树，对于根节点，要么 p 的左子树与 q 的左子树集合相同，要么 p 的左子树与 q 的右子树集合相同，然后继续递归到每个节点判是否满足条件，并且也能求出每个节点是否发生交换。然而现在一次 $+1$ 操作只能给最右边的链的每个点执行交换，于是我们可以将每个 $+1$ 前面和后面都加上一个 $\oplus x$，这样子就是任选一条路径进行交换。

[AGC057C] Increment or Xor 题解

那么现在两种操作就变成了任选一条路径进行交换，或者选一层所有点进行交换，那可以发现这两种操作与原问题是等价的，因为对于每一种方案，我们都可以让 $\oplus x, +1$ 变为 $(\oplus x, +1, \oplus x), \oplus x$。那么我们设 tp_i 表示节点 i 是否需要进行交换操作，每次就是路径异或，或者同一层异或。于是我们依次枚举最后一层需要交换的点，将根到这个点的路径进行异或，最后再看每一层的点状态是否一致即可。

[AGC057C] Increment or Xor 题解

那么现在两种操作就变成了任选一条路径进行交换，或者选一层所有点进行交换，那可以发现这两种操作与原问题是等价的，因为对于每一种方案，我们都可以让 $\oplus x, +1$ 变为 $(\oplus x, +1, \oplus x), \oplus x$。那么我们设 tp_i 表示节点 i 是否需要进行交换操作，每次就是路径异或，或者同一层异或。于是我们依次枚举最后一层需要交换的点，将根到这个点的路径进行异或，最后再看每一层的点状态是否一致即可。

操作次数最多为 $3 \times 2^{n-1} + 1$，显然不超过 10^6。

trick

对于判断类的题目，一般是找充要条件，可以尝试不断找必要条件，然后证明这些条件是充分的。

而对于找必要条件，如果找不到限制比较严的，去找那些非常显然的条件也可以。

[AGC071C] Orientable as Desired

给定一个 n 个点 m 条边的无向简单连通图，判断是否存在一个长度为 n 的非负整数序列 $(x_1, x_2, \ldots, x_n)$，满足：

- $x_i \leq deg_i$，deg_i 表示点 i 的度数。
- 不存在一种给原图的每条边定向的方案，使得 $\forall 1 \leq i \leq n$，满足点 i 的出度或入度等于 x_i。

$n, m \leq 2 \times 10^5$

[AGC071C] Orientable as Desired 题解

先考虑一些特殊情况，比如 x 全为 0，那么一定是将 n 个点划分为两个集合，一个集合全部连向另一个集合，即要求原图是一个二分图，所以如果原图不是二分图就可以直接输出 YES。

[AGC071C] Orientable as Desired 题解

先考虑一些特殊情况，比如 x 全为 0，那么一定是将 n 个点划分为两个集合，一个集合全部连向另一个集合，即要求原图是一个二分图，所以如果原图不是二分图就可以直接输出 YES。

接下来考虑 x 中某一位有值，其余为全是 0。不妨设 $x = (y, 0, 0, \ldots)$，我们尝试找到一种定向方案满足这个 x。将原图去掉 1，会形成一些连通块，显然每个连通块都是二分图，所以只有两种定向方案。

[AGC071C] Orientable as Desired 题解

设 1 在每个连通块中的度数分别为 $a_1, a_2 \ldots, a_k$，那么我们需要能找出一些 a_i 使得和为 y，显然 y 可以取遍 $0 \sim deg_1$，所以我们要求 $a_1, a_2, \ldots, a_k$ 能凑出所有 $0 \sim deg_1$。于是如果有一个 u，满足 u 的 $a_1 \ldots a_k$ 不能凑出 $0 \sim deg_u$ 中的所有数，我们就找到了一种 x。

[AGC071C] Orientable as Desired 题解

设 1 在每个连通块中的度数分别为 $a_1, a_2 \ldots, a_k$，那么我们需要能找出一些 a_i 使得和为 y，显然 y 可以取遍 $0 \sim deg_1$，所以我们要求 $a_1, a_2, \ldots, a_k$ 能凑出所有 $0 \sim deg_1$。于是如果有一个 u，满足 u 的 $a_1 \ldots a_k$ 不能凑出 $0 \sim deg_u$ 中的所有数，我们就找到了一种 x。

接下来大胆猜测如果不满足上述条件，则一定不存在 x，相当于对于所有 x 都存在边的定向方案。

[AGC071C] Orientable as Desired 题解

设 1 在每个连通块中的度数分别为 $a_1, a_2 \ldots, a_k$，那么我们需要能找出一些 a_i 使得和为 y，显然 y 可以取遍 $0 \sim deg_1$，所以我们要求 $a_1, a_2, \ldots, a_k$ 能凑出所有 $0 \sim deg_1$。于是如果有一个 u，满足 u 的 $a_1 \ldots a_k$ 不能凑出 $0 \sim deg_u$ 中的所有数，我们就找到了一种 x。

接下来大胆猜测如果不满足上述条件，则一定不存在 x，相当于对于所有 x 都存在边的定向方案。

考虑证明，我们建出原图的圆方树，那每个方点就有两种定向方案。然后从根节点往下 dfs，每次到一个圆点 u 时，只有其父亲的方点被定向了，则我们要定向 u 的所有儿子方点。

[AGC071C] Orientable as Desired 题解

设 u 在每个方点的度数为 $a_1 \dots a_k$，其中父亲方点为 a_1，有 $a_1 \dots a_k$ 能凑出 $0 \sim deg_u$ 的所有数。则我们需要在 $a_2 \dots a_k$ 中选出一些 a_i 满足和为 x_u 或 $x_u - a_1$，这一定是存在的，因为如果不存在的话，再来一个 a_1 也凑不出 x_u，与条件矛盾。

[AGC071C] Orientable as Desired 题解

设 u 在每个方点的度数为 $a_1 \dots a_k$，其中父亲方点为 a_1，有 $a_1 \dots a_k$ 能凑出 $0 \sim deg_u$ 的所有数。则我们需要在 $a_2 \dots a_k$ 中选出一些 a_i 满足和为 x_u 或 $x_u - a_1$，这一定是存在的，因为如果不存在的话，再来一个 a_1 也凑不出 x_u，与条件矛盾。

所以无解的充要条件为对于所有点 u，满足 $a_1 \dots a_k$ 能凑出来 $0 \sim deg_u$ 的所有数，这个判断是容易的，将 a_i 排一遍序，那么充要条件为 $\forall 1 \le i \le k, \sum_{j=1}^{i-1} a_j \ge a_i - 1$。求 a_i 用 tarjan 即可，复杂度为 $\mathcal{O}(n + m \log m)$。

[AGC068C] Ball Redistribution

有 n 个编号为 $1 \sim n$ 的球和 n 个编号为 $1 \sim n$ 的盒子，初始时你可以选择每个球放入哪个盒子，然后依次对 $1 \sim n$ 执行以下操作：

- 如果盒子 i 中没有球，则跳过。
- 否则，你可以按任意顺序排列盒子 i 中的球，设排好后的顺序依次为 $x_1, x_2, \ldots, x_k$，则对于每个 $1 \leq j \leq k$，将球 x_j 放入盒子 $x_{j \bmod k+1}$ 中。

给定最终状态中每个球在哪个盒子中，你需要确定是否存在一种初始的状态和操作顺序能得到最终状态。

$n \leq 2.5 \times 10^5$

[AGC068C] Ball Redistribution 题解

考虑倒着做，然后 i 向所在的盒子编号 p_i 连边，会得到一个内向基环树森林。从 $n \to 1$ 依次考虑，每次相当于可以将一个环拆掉让其全部连向 i。我们发现到一个 i 时，如果有一个不在环上的点连向了 i，那么就无法操作了。否则如果有环上的点连向 i，则必须操作这个环，如果没有点连向 i，那么我们就可以任选一个环进行操作或者不操作。

[AGC068C] Ball Redistribution 题解

考虑倒着做，然后 i 向所在的盒子编号 p_i 连边，会得到一个内向基环树森林。从 $n \to 1$ 依次考虑，每次相当于可以将一个环拆掉让其全部连向 i。我们发现到一个 i 时，如果有一个不在环上的点连向了 i，那么就无法操作了。否则如果有环上的点连向 i，则必须操作这个环，如果没有点连向 i，那么我们就可以任选一个环进行操作或者不操作。

这个时候就非常考验注意力了，任选一个环进行操作自由度太大了，我们可以弱化为只能拆掉所在的内向基环树中的环。

[AGC068C] Ball Redistribution 题解

然后模拟一下这种情况下的条件是什么，可以发现对于一个点 u，如果 u 的所有儿子 v 的子树最大值都大于 u，那么等操作到 u 时 u 一定处于环或叶子上。

[AGC068C] Ball Redistribution 题解

然后模拟一下这种情况下的条件是什么，可以发现对于一个点 u，如果 u 的所有儿子 v 的子树最大值都大于 u，那么等操作到 u 时 u 一定处于环或叶子上。

进一步发现，这个条件也是必要条件，因为如果有一条边 $u \to v$ 满足 u 不是环上节点并且 u 子树内最大值 $< v$，那么操作到 v 时 u 子树内一定没有动过，所以一定不合法。即这是一个充要条件，$\mathcal{O}(n)$ 判断即可。

[AGC067A] Big Clique Everywhere

给定一个 n 个点 m 条边的简单无向图，判断此图是否满足以下条件：

- 对于任意一个 $\{1, 2, \ldots, n\}$ 的子集 X，都存在一个 X 的子集 Y 满足 $|Y| \geq \frac{|X|}{2}$ 且 Y 中的点构成了一个团。

$n \leq 10^5, m \leq 10^6$

[AGC067A] Big Clique Everywhere 题解

在补图上考虑，相当于对于任意一个子集 X，都存在大小 $\geq \frac{|X|}{2}$ 的独立集。如果补图中有奇环，那么这个奇环一定不合法；否则，补图是个二分图，对于任意一个集合 X，X 中左部点集合和右部点集合一定有一个 $\geq \frac{|X|}{2}$，所以合法的充要条件就是补图是一个二分图。

[AGC067A] Big Clique Everywhere 题解

在补图上考虑，相当于对于任意一个子集 X，都存在大小 $\geq \frac{|X|}{2}$ 的独立集。如果补图中有奇环，那么这个奇环一定不合法；否则，补图是个二分图，对于任意一个集合 X，X 中左部点集合和右部点集合一定有一个 $\geq \frac{|X|}{2}$，所以合法的充要条件就是补图是一个二分图。

补图边数可能很多，但是我们发现原图中需要至少有两个并集为 $\{1, 2, \ldots, n\}$ 的团，所以 $m \geq 2\binom{\lfloor \frac{n}{2} \rfloor}{2}$，即 m 是 $\mathcal{O}(n^2)$ 级别的。所以我们可以先判断是否满足边数限制，然后再判断补图是否为二分图。

[AGC065C] Avoid Half Sum

给定一个长度为 n 的非负整数序列 a，保证 $S = \sum a_i$ 为偶数。请判断是否存在满足以下条件的长度为 n 的非负整数序列对 b 和 c。

- $\forall 1 \le i \le n, a_i = b_i + c_i$
- 对于任意长度为 n 的整数序列 x，$\forall 1 \le i \le n, x_i = b_i \vee x_i = c_i$，则 $\sum x_i \neq \frac{S}{2}$。

$n \le 2 \times 10^5, a_i \le 10^9$

[AGC065C] Avoid Half Sum 题解

x_i 为 b_i, c_i 的其中一个不太好做，要让 $\sum x_i = \frac{S}{2}$，相当于把 b, c 分成两个和相等的序列，设 $d_i = |b_i - c_i|$，则每次可以选择 $-d_i$ 或 d_i 使得总和为 0。

[AGC065C] Avoid Half Sum 题解

x_i 为 b_i, c_i 的其中一个不太好做，要让 $\sum x_i = \frac{S}{2}$，相当于把 b, c 分成两个和相等的序列，设 $d_i = |b_i - c_i|$，则每次可以选择 $-d_i$ 或 d_i 使得总和为 0。

也可以换一种理解，我们要在 d_i 中选出一些数使得这些数的和为 $\frac{\sum d_i}{2}$。每个 d_i 可以选 $[0, a_i]$ 中任意一个与 a_i 奇偶性相同的数。

[AGC065C] Avoid Half Sum 题解

x_i 为 b_i, c_i 的其中一个不太好做，要让 $\sum x_i = \frac{S}{2}$，相当于把 b, c 分成两个和相等的序列，设 $d_i = |b_i - c_i|$，则每次可以选择 $-d_i$ 或 d_i 使得总和为 0。

也可以换一种理解，我们要在 d_i 中选出一些数使得这些数的和为 $\frac{\sum d_i}{2}$。每个 d_i 可以选 $[0, a_i]$ 中任意一个与 a_i 奇偶性相同的数。

我们先考虑必要条件，相当于求在不满足什么条件时一定可以选出来这些 d_i 使得这些数和为总和的一半。不妨弱化一下条件，我们要求一定可以选出 $[0, \sum d_i]$ 中的任意一个数。

[AGC065C] Avoid Half Sum 题解

那么考虑归纳法，设前 $k-1$ 个数已经可以凑出 $[0, \sum_{i=1}^{k-1} d_i]$ 中的任意一个数，要求前 k 个数能凑出 $[0, \sum_{i=1}^{k} d_i]$ 中的所有数，即 $\sum_{i=1}^{k-1} d_i \geq d_k - 1$。而 $\sum_{i=1}^{k-1} d_i$ 最小为 $< a_i$ 的数中的奇数个数，$d_k - 1$ 最大为 $a_k - 1$，那么可以得出条件为对于每个 i 都满足小于 a_i 的奇数个数都 $\geq a_i - 1$。

[AGC065C] Avoid Half Sum 题解

那么考虑归纳法，设前 $k-1$ 个数已经可以凑出 $[0, \sum_{i=1}^{k-1} d_i]$ 中的任意一个数，要求前 k 个数能凑出 $[0, \sum_{i=1}^{k} d_i]$ 中的所有数，即 $\sum_{i=1}^{k-1} d_i \geq d_k - 1$。而 $\sum_{i=1}^{k-1} d_i$ 最小为 $< a_i$ 的数中的奇数个数，$d_k - 1$ 最大为 $a_k - 1$，那么可以得出条件为对于每个 i 都满足小于 a_i 的奇数个数都 $\geq a_i - 1$。

不满足这个性质显然是一个必要条件，下面我们说明这是一个充分条件。

[AGC065C] Avoid Half Sum 题解

我们任意找一个满足 $< a_i$ 的奇数个数 $< a_i - 1$ 的 a_i：

- 如果 a_i 是奇数，对于所有偶数取 $d_j = 0$，对于所有 $< a_i$ 的奇数取 1，所有 $\geq a_i$ 奇数取 $d_j = a_i$，此时如果 $d_j = a_i$ 的出现次数为偶数，就将一个 $= a_i$ 的 d_j 设为 1，考虑第一种意义（选正负号加起来为 0），因为有奇数个 a_i，所以这些数凑数来绝对值至少为 a_i，剩下最多 $a_i - 1$ 个 1 也凑不出 0。
- 如果 a_i 是偶数，那么第一个 $> a_i$ 的奇数也一定符合条件，如果存在这样的 a_i 就选，否则就是所有 $\geq a_i$ 的都是偶数，同上述构造。

[AGC065C] Avoid Half Sum 题解

我们任意找一个满足 $< a_i$ 的奇数个数 $< a_i - 1$ 的 a_i：

- 如果 a_i 是奇数，对于所有偶数取 $d_j = 0$，对于所有 $< a_i$ 的奇数取 1，所有 $\geq a_i$ 奇数取 $d_j = a_i$，此时如果 $d_j = a_i$ 的出现次数为偶数，就将一个 $= a_i$ 的 d_j 设为 1，考虑第一种意义（选正负号加起来为 0），因为有奇数个 a_i，所以这些数凑数来绝对值至少为 a_i，剩下最多 $a_i - 1$ 个 1 也凑不出 0。
- 如果 a_i 是偶数，那么第一个 $> a_i$ 的奇数也一定符合条件，如果存在这样的 a_i 就选，否则就是所有 $\geq a_i$ 的都是偶数，同上述构造。

于是所有 $< a_i$ 的奇数个数 $\geq a_i - 1$ 就是一个充要条件，排序后判断即可。

[AGC059E] Grid 3-coloring

有一个 $n \times n$ 的矩阵，共有三种颜色，给定了最外面一层的颜色，求是否存在一种染色方案，使得相邻格子颜色不同。

$n \leq 2 \times 10^5$

[AGC059E] Grid 3-coloring 题解

矩阵存在一种染色方案，设 $c_{i,j}$ 表示颜色，则我们一定能给每个格子赋一个整数权值 $d_{i,j}$，满足 $c_{i,j} \equiv d_{i,j} \pmod 3$，并且相邻两个格子权值相差 1。考虑证明，我们发现对于一种染色方案，只要确定了某个格子的权值，就能确定整个矩阵的权值。并且对于每个 2×2 的子矩阵，根据左上角的权值算出来的另外三个权值一定都满足限制，即如果左下角颜色等于右上角颜色，则右下角权值一定同时符合条件；如果左下角不等于右上角，则右下角权值一定等于左上角权值，且也符合条件。

[AGC059E] Grid 3-coloring 题解

矩阵存在一种染色方案，设 $c_{i,j}$ 表示颜色，则我们一定能给每个格子赋一个整数权值 $d_{i,j}$，满足 $c_{i,j} \equiv d_{i,j} \pmod 3$，并且相邻两个格子权值相差 1。考虑证明，我们发现对于一种染色方案，只要确定了某个格子的权值，就能确定整个矩阵的权值。并且对于每个 2×2 的子矩阵，根据左上角的权值算出来的另外三个权值一定都满足限制，即如果左下角颜色等于右上角颜色，则右下角权值一定同时符合条件；如果左下角不等于右上角，则右下角权值一定等于左上角权值，且也符合条件。

于是问题就变成了是否存在一个合法的赋权值方案满足条件，我们首先设 $d_{1,1} = c_{1,1}$，然后就可以算出整个外层的权值，先判断这个环的头尾的权值是否满足条件。

[AGC059E] Grid 3-coloring 题解

接下来我们继续找必要条件，发现对于任意两个点 $(i,j),(i',j')$，这两个点的权值之差一定不超过它们的曼哈顿距离，即 $|d_{i,j}-d_{i',j'}|\le|i-i'|+|j-j'|$。而如果有这样的两个两个点不满足条件，一定是正方形的两条对边上，假设是左边和右边，我们将右边的点的横坐标设为左边点的横坐标，此时这个两个点依然不合法，所以只需要判断所有相对的两个点即可。

[AGC059E] Grid 3-coloring 题解

接下来我们继续找必要条件，发现对于任意两个点 $(i,j),(i',j')$，这两个点的权值之差一定不超过它们的曼哈顿距离，即 $|d_{i,j}-d_{i',j'}|\le|i-i'|+|j-j'|$。而如果有这样的两个两个点不满足条件，一定是正方形的两条对边上，假设是左边和右边，我们将右边的点的横坐标设为左边点的横坐标，此时这个两个点依然不合法，所以只需要判断所有相对的两个点即可。

如果所有点都满足条件，那么一定有解，我们可以构造 $d_{i,j}=\max\{d_{i,1}-(j-1),d_{i,n}-(n-j),d_{1,j}-(i-1),d_{n,j}-(n-i)\}$。这样子最外层的点一定都满足条件，并且相邻两个点差不超过 1，而且每个值的奇偶性都和 $i+j$ 相同，所以一定有相邻两个格子权值差为 1。

[AGC060B] Unique XOR Path

给定一个 $n \times m$ 的网格和 k，和一条从 $(1,1)$ 到 (n,m) 的由向下和向右构成的路径，判断能否给每个格子填一个 $[0, 2^k - 1]$ 的数，满足在所有从 $(1,1)$ 到 (n,m) 的由向下和向右的路径中，给定的路径是唯一一条异或为 0 的路径。

$T \leq 100, n, m, k \leq 30$

[AGC060B] Unique XOR Path 题解

一个初步的想法是，给 $(1,1)$ 填上 2^k-1，然后给每个拐点处填上某个 2^p，于是条件就是拐点数 $\le k$。但是这不是必要的，我们考虑，一条路径在每个拐点处都可以换一种拐弯方式，形成一个只和原路径只差一格的路径。设有 x 个拐点，如果拐点间是独立的，那么相当于是 x 个数的线性基能否凑出 0，需要有 $x\le k$。

[AGC060B] Unique XOR Path 题解

一个初步的想法是，给 $(1,1)$ 填上 2^k-1，然后给每个拐点处填上某个 2^p，于是条件就是拐点数 $\le k$。但是这不是必要的，我们考虑，一条路径在每个拐点处都可以换一种拐弯方式，形成一个只和原路径只差一格的路径。设有 x 个拐点，如果拐点间是独立的，那么相当于是 x 个数的线性基能否凑出 0，需要有 $x\le k$。

但是如果有连续两个拐弯，那么这两个拐弯不是独立的，可以发现连续 x 个拐弯只需要 $\lceil\frac{x}{2}\rceil$ 个数即可，所以我们统计出 $\sum\lceil\frac{x}{2}\rceil$ 即可。最后构造的话，我们将这条路径上的数放 0，拐点处放某个 2^p，然后一直向左下或右上延伸到边缘。

[AGC058C] Planar Tree

有一个 n 个顶点的圆环，每个点上有一个 $1 \sim 4$ 的权值 a_i，保证 $1 \sim 4$ 的每个数都至少出现了一次。现在要在 n 个点间添加 $n-1$ 条边使其构成一棵树，并满足以下要求：

- 如果一条边连接了两个点 x, y，则必须满足 $|a_x - a_y| = 1$。
- 如果将所有边看成线段，那么任意两条线段不会在除端点外的地方相交。

判断是否有满足条件的方案。

$n \leq 3 \times 10^5$

[AGC058C] Planar Tree 题解

首先我们可以将条件改为 $|a_x - a_y| \leq 1$ 就能连边，那么最后考虑某个权值相同的连通块，一定有一个点连向了一个权值不同的点，那么所有连通块内的点都连向这个点即可。于是我们就可以将所有相邻的权值相同的点合并起来。因为 $|a_x - a_y| \leq 1$ 就能连边，那么 1 能连向的权值 2 一定都可以连，所以如果有两个 $1, 2$ 相邻就可以将 1 删去，$3, 4$ 同理。

[AGC058C] Planar Tree 题解

首先我们可以将条件改为 $|a_x - a_y| \leq 1$ 就能连边，那么最后考虑某个权值相同的连通块，一定有一个点连向了一个权值不同的点，那么所有连通块内的点都连向这个点即可。于是我们就可以将所有相邻的权值相同的点合并起来。因为 $|a_x - a_y| \leq 1$ 就能连边，那么 1 能连向的权值 2 一定都可以连，所以如果有两个 $1, 2$ 相邻就可以将 1 删去，$3, 4$ 同理。

那么剩下的环一定是 ≤ 2 和 ≥ 3 的点间隔排列。接下来考虑如下的性质：一棵合法的树一定存在一条原来环上相邻的两个点的边，满足其中一个点是叶子。证明可以手玩以下，那么答案一定可以通过每次连接相邻两个点并删去其中一个点的方式生成。因为剩下的点中连边一定是 $(2, 3)$ 之间连，接下来考虑删去其中一个点之后会有什么影响。

[AGC058C] Planar Tree 题解

如果删去了 3，3 另一边无论是 1 还是 2 都可以和 2 合并，然后变成原来点数减 2 的问题。删去 2 同理，一定会恰好合并一个点。可以发现如果剩下的数全是 $2,3$，那么一定可以合并完，所以我们每次要尽量删去 $1,4$。每次删去一个 2 最多会删去一个 4，删去一个 3 最多会删去一个 2，并且如果剩下 $1,2,3,4$ 各一个是无解的，所以记 c_1,c_2,c_3,c_4 表示出现次数，一个必要条件是 $c_2>c_4,c_3>c_1$。可以发现这也是充要条件，因为每次都一定能找到一个符合条件的合并方法。
于是直接判断即可，复杂度为 $\mathcal{O}(n)$。

[AGC058E] Nearer Permutation

对于两个 $1 \sim n$ 的排列 p, q，定义 $d(p, q)$ 表示通过交换 p 的相邻两个元素变为 q 的最小操作次数。设 I 表示 $I_i = i$ 的排列，定义 $f(p)$ 为所有满足 $d(p, q) \leq d(q, I)$ 的排列 q 中，字典序最小的 q。现在给定一个排列 q，求是否存在一个排列 p 满足 $f(p) = q$。

$n \leq 3 \times 10^5$

[AGC058E] Nearer Permutation 题解

先考虑给定一个 p，怎么求 $f(p)$。因为要字典序最小，所以我们依次枚举 i 从 $1 \sim n$，每次考虑 q_i 最小填多少。假设 $q_i = x$，我们要尽量让后面的 $d(p, q) \le d(q, I)$，最优方式一定是直接将 p 中的 x 移到最开头，而每次移动都会让 $d(p, q)$ 加一，$d(q, I)$ 减一。于是设 $s = \frac{inv(p)}{2}$，假设排列下标从 0 开始，那么每次操作为：

- 找到 $p_{0,\ldots,\min(\lfloor s \rfloor, n-1)}$ 中最小的 p_i，将 p_i 从 p 中删除并加到 q 末尾，然后将 s 减去 i。

[AGC058E] Nearer Permutation 题解

先考虑给定一个 p，怎么求 $f(p)$。因为要字典序最小，所以我们依次枚举 i 从 $1 \sim n$，每次考虑 q_i 最小填多少。假设 $q_i = x$，我们要尽量让后面的 $d(p,q) \le d(q,I)$，最优方式一定是直接将 p 中的 x 移到最开头，而每次移动都会让 $d(p,q)$ 加一，$d(q,I)$ 减一。于是设 $s = \frac{inv(p)}{2}$，假设排列下标从 0 开始，那么每次操作为：

- 找到 $p_{0,\dots,\min(\lfloor s \rfloor, n-1)}$ 中最小的 p_i，将 p_i 从 p 中删除并加到 q 末尾，然后将 s 减去 i。

接下来注意到，每次 s 都是不增的，那么 $[0, \min(s, n-1)]$ 的范围就会在减少，并且每次都移除了最小值，那么 q_i 应该都是递增的。于是我们找到第一个 k 满足 $q_k > q_{k+1}$（如果没找到那么 $q = I$，当 $p = I$ 时有 $f(p) = q$），设在做了 k 次操作后的序列为 a，一定有：

[AGC058E] Nearer Permutation 题解

- $a_0 = q_k$，并且 a_0 是 $a_{0,\dots,s}$ 中的最小值，且 $a_{s+1} = q_{k+1}$。

[AGC058E] Nearer Permutation 题解

- $a_0 = q_k$，并且 a_0 是 $a_{0,\dots,s}$ 中的最小值，且 $a_{s+1} = q_{k+1}$。

那么此时删掉 q_k 后一定会删掉 q_{k+1}，并且这是唯一一种可能出现 $q_k > q_{k+1}$ 的情况。可以发现，a 中剩下的数每次都会删掉最开头的数，相当于按原顺序排列。于是做了 k 次后的序列一定长成：$q_k, q_{k+2}, q_{k+3}, \dots, q_x, q_{k+1}, q_{x+1}, \dots, q_n$。

[AGC058E] Nearer Permutation 题解

- $a_0 = q_k$，并且 a_0 是 $a_{0,\dots,s}$ 中的最小值，且 $a_{s+1} = q_{k+1}$。

那么此时删掉 q_k 后一定会删掉 q_{k+1}，并且这是唯一一种可能出现 $q_k > q_{k+1}$ 的情况。可以发现，a 中剩下的数每次都会删掉最开头的数，相当于按原顺序排列。于是做了 k 次后的序列一定长成：$q_k, q_{k+2}, q_{k+3}, \dots, q_x, q_{k+1}, q_{x+1}, \dots, q_n$。

那么我们考虑倒着做操作，即令 $p = q$，然后依次枚举 $q_{k+1}, q_{k-1}, q_{k-2}, \dots, q_1$，每次将当前的 q_i 向后移动 c_i 个位置。现在就是要求一组 c_i，考虑怎么判断 c_i 是否合法，设在执行 $k+1$ 次操作后，Δ 为 q_{k+1} 的位置与 s 的差，那么我们需要保证 $\Delta \in \{0, \frac{1}{2}\}$。可以发现，$c_i$ 每增加 1，都会让 $inv(p)$ 增加 1，如果 $i \neq k+1$，就会让最后的 s 减一，否则就会让 q_{k+1} 加一，即都会让 Δ 减去 $\frac{1}{2}$。

[AGC058E] Nearer Permutation 题解

于是，c_i 每增加 1 都会让 Δ 减少 $\frac{1}{2}$，当 c_i 都为 0 时有 $\Delta = \frac{inv(q)}{2}$，而 Δ 应该为 0 或 $\frac{1}{2}$，于是 $\sum c_i$ 应该为 $inv(q)$ 或 $inv(q) - 1$，我们可以分别判断这两种情况。因为知道了 $\sum c_i$，于是就知道 $inv(p) = inv(q) + \sum c_i$，那么此时的最优策略一定是依次枚举 $q_1, q_2, \ldots, q_{k-1}, q_{k+1}$，求出它们 c_i 最大能取到多少。

[AGC058E] Nearer Permutation 题解

于是，c_i 每增加 1 都会让 Δ 减少 $\frac{1}{2}$，当 c_i 都为 0 时有 $\Delta = \frac{inv(q)}{2}$，而 Δ 应该为 0 或 $\frac{1}{2}$，于是 $\sum c_i$ 应该为 $inv(q)$ 或 $inv(q) - 1$，我们可以分别判断这两种情况。因为知道了 $\sum c_i$，于是就知道 $inv(p) = inv(q) + \sum c_i$，那么此时的最优策略一定是依次枚举 $q_1, q_2, \ldots, q_{k-1}, q_{k+1}$，求出它们 c_i 最大能取到多少。

因为我们知道了 $inv(p)$，相当于知道了 $s = \frac{inv(p)}{2}$，就可以模拟每次操作了。设 res 表示剩下的 $\sum c_i$，则一定有 $res \le s$，那么每次操作要么是移到序列末尾，要么是将 res 减为 0，所以剩下的数的相对位置不会发生变化，只需记录 pos_i 表示 q 中 i 的位置即可 $\mathcal{O}(n)$ 判断。总复杂度为 $\mathcal{O}(n \log n)$，瓶颈在于求 $inv(q)$。具体细节可以参考代码。

介绍

注意这类博弈论题目一般不会用到比较高深的 SG 函数，或者其他博弈模型，仍然也只是找性质。

一般可以从小数据开始模拟，或者通过打表，来找出一些规律。

[AGC069B] Pair Guessing

给定一个 $n \times n$ 的 01 矩阵，甲和乙在上面玩游戏，流程如下：

- 甲选择一个满足 $a_{i,j} = 1$ 的 (i, j)。
- 乙有 n 次机会进行提问，每次可以询问一个 (i', j')，甲会回答是否有 $i = i' \vee j = j'$。
- 乙需要回答 (i, j) 是什么。

你需要判断，无论甲选择哪个 (i, j)，乙是否都有合适的策略回答出 (i, j) 是什么。

$n \leq 500$

[AGC069B] Pair Guessing 题解

不妨设乙第一次选择了 $(1,1)$，如果回答 0，那么就可以去掉矩阵的第一行第一列，一直询问直到甲回答 1。此时如果第一行第一列中全是 1，那么乙就一定猜不出来（除非只剩一个 1），否则，乙可以选择含 0 的那列或行，然后剩下的一个一个猜一定能猜出来。

[AGC069B] Pair Guessing 题解

不妨设乙第一次选择了 $(1,1)$，如果回答 0，那么就可以去掉矩阵的第一行第一列，一直询问直到甲回答 1。此时如果第一行第一列中全是 1，那么乙就一定猜不出来（除非只剩一个 1），否则，乙可以选择含 0 的那列或行，然后剩下的一个一个猜一定能猜出来。

于是问题变成了每次选择矩阵中的一行一列，如果这一行一列中包含 0，就可以删去这一行一列，求能否将矩阵删到只剩一个数。那么对于所有 $a_{i,j}=0$ 的 (i,j)，将第 i 行向第 j 列连边，对于每个连通块，找到一棵生成树，从叶子依次往上删，一共可以删 $siz-1$ 次。所以我们统计所有连通块的 $siz-1$ 之和 sum，然后判断是否有 $sum\geq n-1$ 即可。

[AGC069B] Pair Guessing 题解

但是还有个 cornercase（极为隐蔽的 corner），比如下面这个矩阵：

$$\begin{matrix}011\\111\\111\end{matrix}$$

[AGC069B] Pair Guessing 题解

但是还有个 cornercase（极为隐蔽的 corner），比如下面这个矩阵：

$$\begin{matrix}011\\111\\111\end{matrix}$$

如果问 $(1,1)$ 后回答了否，我们可以通过问 $(1,2),(2,1)$ 来唯一确定点的位置。也就是说如果 $sum = n-2$ 且 $n>2$ 那么也是合法的。

[AGC048D] Pocky Game

有 n 堆石子排成一排，第 i 堆有 a_i 个，有两个人在上面玩游戏。每次先手在最左端的石子中拿至少一个（至多拿完），然后后手在最右端的石子中拿至少一个（至多拿完）。无法操作的人失败，求先手必胜还是后手必胜。

$T, n \leq 100, a_i \leq 10^9$

[AGC048D] Pocky Game 题解

对于一堆石子，假设除了第一堆外的石子数量都是固定的，那么对于先手，第一堆的石子数量一定是越多越好，因为少的石子第一步能达到的状态多的石子肯定也能达到。所以一定有一个分界点，满足第一堆石子数量如果少于就必败，否则就必胜。

[AGC048D] Pocky Game 题解

对于一堆石子，假设除了第一堆外的石子数量都是固定的，那么对于先手，第一堆的石子数量一定是越多越好，因为少的石子第一步能达到的状态多的石子肯定也能达到。所以一定有一个分界点，满足第一堆石子数量如果少于就必败，否则就必胜。

于是我们可以用 $f_{l,r}$ 表示在区间 $[l,r]$ 中，固定了 $[l+1,r]$ 的数量，a_l 最小是多少时，先手必胜。$g_{l,r}$ 表示固定了 $[l,r-1]$ 的石子数量，a_r 最小是多少，后手必胜（假设这里后手先操作）。初始化有 $f_{i,i}=g_{i,i}=1$。

[AGC048D] Pocky Game 题解

对于一堆石子，假设除了第一堆外的石子数量都是固定的，那么对于先手，第一堆的石子数量一定是越多越好，因为少的石子第一步能达到的状态多的石子肯定也能达到。所以一定有一个分界点，满足第一堆石子数量如果少于就必败，否则就必胜。

于是我们可以用 $f_{l,r}$ 表示在区间 $[l,r]$ 中，固定了 $[l+1,r]$ 的数量，a_l 最小是多少时，先手必胜。$g_{l,r}$ 表示固定了 $[l,r-1]$ 的石子数量，a_r 最小是多少，后手必胜（假设这里后手先操作）。初始化有 $f_{i,i}=g_{i,i}=1$。

考虑求 $f_{l,r}$，首先如果有 $a_r<g_{l+1,r}$，那么先手可以直接取完第一堆石子，则 $f_{l,r}=1$。否则，先手肯定只会取一个石子（因为第一堆石子越多越好），我们让 a_l--。

[AGC048D] Pocky Game 题解

这个时候该后手操作，如果 $a_l < f_{l,r-1}$，那么后手将这一堆石子取完就必胜，否则后手也只会取一个石子。于是双方就是每次都取一个石子，直到 $a_l < f_{l,r-1}$ 或 $a_r < g_{l+1,r}$。

[AGC048D] Pocky Game 题解

这个时候该后手操作，如果 $a_l < f_{l,r-1}$，那么后手将这一堆石子取完就必胜，否则后手也只会取一个石子。于是双方就是每次都取一个石子，直到 $a_l < f_{l,r-1}$ 或 $a_r < g_{l+1,r}$。

所以先手必胜的条件就是 $a_l - f_{l,r-1} > a_r - g_{l+1,r}$，所以我们可以求出 $f_{l,r} = a_r - g_{l+1,r} + f_{l,r-1} + 1$。$g_{l,r}$ 的转移同理。

[AGC048D] Pocky Game 题解

这个时候该后手操作，如果 $a_l < f_{l,r-1}$，那么后手将这一堆石子取完就必胜，否则后手也只会取一个石子。于是双方就是每次都取一个石子，直到 $a_l < f_{l,r-1}$ 或 $a_r < g_{l+1,r}$。

所以先手必胜的条件就是 $a_l - f_{l,r-1} > a_r - g_{l+1,r}$，所以我们可以求出 $f_{l,r} = a_r - g_{l+1,r} + f_{l,r-1} + 1$。$g_{l,r}$ 的转移同理。

于是我们就可以在 $\mathcal{O}(n^2)$ 的时间内求出所有 f, g，最后先手必胜的条件就是 $a_1 \geq f_{1,n}$。

[AGC023D] Go Home

一条街上有 n 栋楼，位置分别为 $X_1, X_2, \ldots, X_n$，第 i 栋楼里住着 p_i 个人。初始所有人都在 S 处的一辆车上。车是自动驾驶的，对于每一时刻，还在车上的员工都会进行投票，只能投正或负方向。班车会自动统计两个方向的票数，并且往票多的方向行驶一个单位长度，如果票一样多，那就往负方向行驶。员工们也有投票策略，每一个员工都会投能让他回家时间尽量早的方向，如果两个方向一样早，那就投负方向。如果班车到达了某一个楼，那么住在那栋楼中的所有员工都会下车。

可以证明，在上述条件下，每个员工投票的方向是能够唯一确定的，班车的运行路线也能够唯一确定。询问最后一名员工回到家，经过了多少个单位时间。$n \leq 10^5$

[AGC023D] Go Home 题解

假设 $X_1 < S < X_n$，现在只考虑第 1 和 n 栋楼。我们不妨设 $p_1 < p_n$。那么这个时候班车一定会先去第 n 栋楼，最后才会去前往第 1 栋楼。证明可以考虑反证，如果先前往 1，那么在最后一步时，只有 p_1 个人会投负方向，而其余人数一定 $> p_1$。所以会投正方向，矛盾。

[AGC023D] Go Home 题解

假设 $X_1 < S < X_n$，现在只考虑第 1 和 n 栋楼。我们不妨设 $p_1 < p_n$。那么这个时候班车一定会先去第 n 栋楼，最后才会去前往第 1 栋楼。证明可以考虑反证，如果先前往 1，那么在最后一步时，只有 p_1 个人会投负方向，而其余人数一定 $> p_1$。所以会投正方向，矛盾。

于是整个班车一定是先走到 X_n（不一定一直向右），然后再一路向左走到 p_1。所以答案就是班车到 X_n 的时间加上 $X_n - X_1$。

[AGC023D] Go Home 题解

假设 $X_1 < S < X_n$，现在只考虑第 1 和 n 栋楼。我们不妨设 $p_1 < p_n$。那么这个时候班车一定会先去第 n 栋楼，最后才会去前往第 1 栋楼。证明可以考虑反证，如果先前往 1，那么在最后一步时，只有 p_1 个人会投负方向，而其余人数一定 $> p_1$。所以会投正方向，矛盾。

于是整个班车一定是先走到 X_n（不一定一直向右），然后再一路向左走到 p_1。所以答案就是班车到 X_n 的时间加上 $X_n - X_1$。

我们发现，因为第 1 栋楼里的人知道自己到达的时间一定是班车到 X_n 的时间加上 $X_n - X_1$，所以他们的目标就变成了尽量让班车更快到 X_n，这个目标和第 n 栋楼里的人的目标相同！

[AGC023D] Go Home 题解

于是我们可以直接将第 1 栋里的人合并到第 n 栋里，然后变成一个 $n-1$ 的子问题。

[AGC023D] Go Home 题解

于是我们可以直接将第 1 栋里的人合并到第 n 栋里，然后变成一个 $n-1$ 的子问题。

于是可以直接递归下去，直到 $S \le X_1$ 或 $S \ge X_n$，这个时候可以直接计算。复杂度为 $\mathcal{O}(n)$。

[AGC056D] Subset Sum Game

给定一个长度为 n 的序列 a，保证 n 为偶数。Alice 和 Bob 轮流操作，Alice 先手，每次当前操作的人选择一个 a 中还未被删去的数删去。最终如果 Alice 选择的所有数的和 $\in [l, r]$，则 Alice 获胜，否则 Bob 获胜。求双方都采取最优策略时谁会获胜。

$n \leq 5000, a_i \leq 10^9$

[AGC056D] Subset Sum Game 题解

将 a 排序，我们可以先考虑 Bob 先手时谁会获胜。

[AGC056D] Subset Sum Game 题解

将 a 排序，我们可以先考虑 Bob 先手时谁会获胜。
Bob 可以选择让 Alice 选的数尽量小或者尽量大，如果 Bob 一开始的目标为让 Alice 选择的所有数和最小，那么每次 Bob 一定是选择还未被删去的数中最大的数。双方轮流操作下去，则 Alice 选的数之和最大可能为为 $a_1 + a_3 + \ldots + a_{n-1}$，如果这个数 $< l$，相当于 Alice 尽力让自己选的数最大但还是到不了 $[l, r]$，那么 Bob 采取上述策略一定可以获胜。同理，如果 $a_2 + a_4 + \ldots + a_n > r$，也是 Bob 获胜。

[AGC056D] Subset Sum Game 题解

将 a 排序，我们可以先考虑 Bob 先手时谁会获胜。
Bob 可以选择让 Alice 选的数尽量小或者尽量大，如果 Bob 一开始的目标为让 Alice 选择的所有数和最小，那么每次 Bob 一定是选择还未被删去的数中最大的数。双方轮流操作下去，则 Alice 选的数之和最大可能为为 $a_1 + a_3 + \ldots + a_{n-1}$，如果这个数 $< l$，相当于 Alice 尽力让自己选的数最大但还是到不了 $[l, r]$，那么 Bob 采取上述策略一定可以获胜。同理，如果 $a_2 + a_4 + \ldots + a_n > r$，也是 Bob 获胜。

否则，如果 $a_1 + a_3 + \ldots + a_{n-1} \geq l$ 且 $a_2 + a_4 + \ldots + a_n \leq r$，我们将序列分组为 $(a_1, a_2), (a_3, a_4), \ldots, (a_{n-1}, a_n)$，那么从每组任选一个数的和都在 $[l, r]$ 之间，于是 Bob 每次选一个数，Alice 都选组内另一个数即可获胜。于是这就是 Alice 获胜的充要条件。

[AGC056D] Subset Sum Game 题解

接下来考虑 Alice 先手，我们先枚举 Alice 第一次选哪一个数，就变成了 Bob 先手。然后在接下来的 $n-1$ 中，如果 Alice 能找到一个长度为 $n-2$ 的子序列，满足这个序列奇数位和 $\geq l$ 且偶数位和 $\leq r$，那么一定是 Alice 获胜，策略同上，即使 Bob 选择剩下的那个数也不影响。接下来我们要说明，如果不满足这个条件，那么一定是 Bob 获胜。

[AGC056D] Subset Sum Game 题解

接下来考虑 Alice 先手，我们先枚举 Alice 第一次选哪一个数，就变成了 Bob 先手。然后在接下来的 $n-1$ 中，如果 Alice 能找到一个长度为 $n-2$ 的子序列，满足这个序列奇数位和 $\geq l$ 且偶数位和 $\leq r$，那么一定是 Alice 获胜，策略同上，即使 Bob 选择剩下的那个数也不影响。接下来我们要说明，如果不满足这个条件，那么一定是 Bob 获胜。

如果不满足上述条件，相当于对于任意一个长度为 $n-2$ 的子序列，都满足奇数位和 $< l$ 或偶数位和 $> r$。考虑删去 $n-1$ 个数中每个数形成的子序列，那么满足奇数位和 $< l$ 的一定是一段前缀，满足偶数位和 $> r$ 的一定是一段后缀，并且一定有一个位置同时满足这两个条件，因为删除相邻两个数的子序列中，奇数位和和偶数位和至多一个会发生改变。

[AGC056D] Subset Sum Game 题解

于是 Bob 选择删去这个位置，接下来要说明对于所有奇数位和 $< l$ 且偶数位和 $> r$ 的序列 a，一定是 Bob 获胜。

[AGC056D] Subset Sum Game 题解

于是 Bob 选择删去这个位置，接下来要说明对于所有奇数位和 $< l$ 且偶数位和 $> r$ 的序列 a，一定是 Bob 获胜。

对于这样的一个序列 a，Alice 选择一个长度为 $n-2$ 的子序列，相当于删掉两个数，那么无论删掉哪两个数，剩下的子序列依然会满足奇数位和 $< l$、偶数位和 $> r$ 中的至少一个（分删的数是否同奇偶来考虑）。

[AGC056D] Subset Sum Game 题解

也就是说 Alice 在选择第一个数后，不能找到长为 $n-2$ 的子序列满足奇数位和 $\geq l$ 且偶数位和 $\leq r$，根据上面的结论，Bob 总能找到下一步要删的数使得剩下的子序列还满足奇数位和 $< l$ 且偶数位和 $> r$，这样就递归到了 $n-2$ 的子问题。当 $n=2$ 时，显然 Bob 必胜，于是就可以通过归纳法说明了这种情况下 Bob 必胜。

[AGC056D] Subset Sum Game 题解

也就是说 Alice 在选择第一个数后，不能找到长为 $n-2$ 的子序列满足奇数位和 $\geq l$ 且偶数位和 $\leq r$，根据上面的结论，Bob 总能找到下一步要删的数使得剩下的子序列还满足奇数位和 $< l$ 且偶数位和 $> r$，这样就递归到了 $n-2$ 的子问题。当 $n=2$ 时，显然 Bob 必胜，于是就可以通过归纳法说明了这种情况下 Bob 必胜。

枚举 Alice 第一次选哪个数和剩下的数中每个长为 $n-2$ 的子序列，计算奇数位和和偶数位和判断即可，复杂度 $\mathcal{O}(n^2)$。

介绍

Ad-hoc 中的 Ad-hoc。

这下是真的没有任何算法了，那就手玩各种样例找性质！

当然，做多了题也是可以发现许多 trick 的。具体的在每道题后介绍。

[AGC073B] Cyclic Jump

给定一个长度为 n 的序列 a，初始时你位于数轴原点，每次你可以选择某个 a_i，然后选择向后或向前走 a_i 的距离。合法的操作序列满足以下条件：

- 至少进行一次操作，并且最终要回到原点。
- 没有到达过负坐标区域。
- 不能连续使用同一个 a_i 并且两次方向相反。

一个合法操作序列的代价为所有到达的点中的最大坐标值，求所有合法操作序列的最小代价。

$n \le 2.5 \times 10^5, a_i \le 10^{18}$

[AGC073B] Cyclic Jump 题解

设 $f(a_1, \ldots, a_n)$ 表示数组 a 的答案，将 a 排序，因为至少要用一次操作，所以答案一定 $\geq a_1$。于是我们可以先将答案加上 a_1，然后将剩下的 $a_2, \ldots, a_n$ 减去 a_1，相当于先向右走 a_1，然后限定整个过程都在 $\geq a_1$ 的坐标，最后回到 a_1 后再走到 0，然后再处理这个子问题，即 $f(a_1, \ldots, a_n) = a_1 + f(a_1, a_2 - a_1, \ldots, a_n - a_1)$。

[AGC073B] Cyclic Jump 题解

设 $f(a_1, \ldots, a_n)$ 表示数组 a 的答案，将 a 排序，因为至少要用一次操作，所以答案一定 $\geq a_1$。于是我们可以先将答案加上 a_1，然后将剩下的 $a_2, \ldots, a_n$ 减去 a_1，相当于先向右走 a_1，然后限定整个过程都在 $\geq a_1$ 的坐标，最后回到 a_1 后再走到 0，然后再处理这个子问题，即 $f(a_1, \ldots, a_n) = a_1 + f(a_1, a_2 - a_1, \ldots, a_n - a_1)$。

考虑证明上式，分左边到右边和右边到左边来考虑：

[AGC073B] Cyclic Jump 题解

- 右边到左边，即每个新的方案都可以对应一个原来的方案。因为新方案中整个过程都是在 $\geq a_1$ 中的，所以如果某次向右走了 $a_i - a_1$，那么就相当于在原来的方案中先向左走 a_1，然后向右走 a_i；如果向左走了 $a_i - a_1$，相当于先向左走 a_i，再向右走 a_1，于是每一种新的方案都可以对应一种原来的方案。

[AGC073B] Cyclic Jump 题解

- 右边到左边，即每个新的方案都可以对应一个原来的方案。因为新方案中整个过程都是在 $\geq a_1$ 中的，所以如果某次向右走了 $a_i - a_1$，那么就相当于在原来的方案中先向左走 a_1，然后向右走 a_i；如果向左走了 $a_i - a_1$，相当于先向左走 a_i，再向右走 a_1，于是每一种新的方案都可以对应一种原来的方案。
- 左边到右边，即每个原来的方案都可以对应一个新的方案。如果是向右走了 a_i，那么可以拆成先向右走 a_1，再向右走 $a_i - a_1$；如果是向左走了 a_i，有可能走到了一个左边 $< a_1$ 的地方，就无法直接对应，但是可以发现下一步一定会向右走，如果是向右走 a_1，整个过程就是向左走 $a_i - a_1$，否则如果是向右走 a_j，那么就相当于向左走 $a_i - a_1$，然后再向右走 $a_j - a_1$。于是每一种原来的方案都可以对应一种新方案。

[AGC073B] Cyclic Jump 题解

于是我们就证明了上式，就可以递归下去做了。但是直接做可能递归轮数太多，所以可以考虑 $f(a_1,\ldots,a_n)=a_1\lfloor\frac{a_2}{a_1}\rfloor+f(a_1,a_2-a_1\lfloor\frac{a_2}{a_1}\rfloor,\ldots,a_n-a_1\lfloor\frac{a_2}{a_1}\rfloor)$，相当于辗转相除。递归终止条件为 $a_1=0$，此时答案为 0。手玩一下可以发现每做两轮操作都会让序列最小值减半，所以递归轮数为 $\mathcal{O}(\log V)$。

[AGC073B] Cyclic Jump 题解

于是我们就证明了上式，就可以递归下去做了。但是直接做可能递归轮数太多，所以可以考虑 $f(a_1,\ldots,a_n)=a_1\lfloor\frac{a_2}{a_1}\rfloor+f(a_1,a_2-a_1\lfloor\frac{a_2}{a_1}\rfloor,\ldots,a_n-a_1\lfloor\frac{a_2}{a_1}\rfloor)$，相当于辗转相除。递归终止条件为 $a_1=0$，此时答案为 0。手玩一下可以发现每做两轮操作都会让序列最小值减半，所以递归轮数为 $\mathcal{O}(\log V)$。

于是总复杂度为 $\mathcal{O}(n\log n\log V)$，当然每次只需要找最小值和次小值，所以可以做到 $\mathcal{O}(n\log V)$。

[AGC066A] Adjacent Difference

给定一个大小为 $n \times n$ 的矩阵 A 和数字 d，你可以对每个数字进行加上一个数或减去一个数的操作，使得每个数字与其相邻数字的差的绝对值 $\geq d$，并且对于每个操作值的绝对值之和不超过 $\frac{dn^2}{2}$。构造一个最终的矩阵。

$n \leq 500$

[AGC066A] Adjacent Difference 题解

人类智慧。我们可以将矩阵黑白染色，然后将所有黑色格子改为 d 的奇数倍，白色格子改为 d 的偶数倍即可满足条件。共有两种黑白染色方式，并且这两种方式的代价之和恰好为 dn^2，于是总有一种方式的代价 $\leq \frac{dn^2}{2}$。

[AGC066C] Delete AAB or BAA

给定一个由 A,B 构成的字符串 S，每次可以删掉 S 中连续的一段 AAB 或 BAA，删掉后 S 前后部分会拼起来，求最多能操作多少次。

$|S| \leq 10^6$

[AGC066C] Delete AAB or BAA 题解

对每个删除的极小段考虑（极小段相当于不能再拆分为两个独立的段分别删除），分析一个极小段能删完的必要条件是什么：

- A 的数量是 B 的两倍
- 这一段的开头或结尾是 B，因为如果开头或结尾都是 A 并且能删完的话，一定能将这段划分为两个独立的段。

[AGC066C] Delete AAB or BAA 题解

对每个删除的极小段考虑（极小段相当于不能再拆分为两个独立的段分别删除），分析一个极小段能删完的必要条件是什么：

- A 的数量是 B 的两倍
- 这一段的开头或结尾是 B，因为如果开头或结尾都是 A 并且能删完的话，一定能将这段划分为两个独立的段。

然后注意力惊人即可发现这是充分的。

[AGC066C] Delete AAB or BAA 题解

考虑证明，相当于要证如果一个序列开头或结尾为 B 且 A 的数量为 B 的两倍那么一定能删完，不妨设序列开头为 B，如果开头为 A 的话，将序列翻转即可。考虑归纳法：

[AGC066C] Delete AAB or BAA 题解

考虑证明，相当于要证如果一个序列开头或结尾为 B 且 A 的数量为 B 的两倍那么一定能删完，不妨设序列开头为 B，如果开头为 A 的话，将序列翻转即可。考虑归纳法：

- 如果序列长度为 3，那么一定可以删完。
- 否则，我们找到第一个长度 ≥ 2 的 A 连续段的位置，这一定是存在的，那么这个连续段的左边或右边一定有一个不是开头的 B，否则说明序列长度为 3。于是我们删掉这个位置，就转换为了 $len - 3$ 的子问题。

[AGC066C] Delete AAB or BAA 题解

于是我们就证明了这是充要条件，dp 是简单的。设 A 为 1，B 为 -2，s_i 表示前缀和，f_i 表示前 i 个数最少能剩多少个数。那么 j 能转移到 i 的条件为 $s_j = s_i$，a_{j+1}, a_i 中有一个为 B，简单转移即可。复杂度 $\mathcal{O}(n)$。

[AGC066C] Delete AAB or BAA 题解

于是我们就证明了这是充要条件，dp 是简单的。设 A 为 1，B 为 -2，s_i 表示前缀和，f_i 表示前 i 个数最少能剩多少个数。那么 j 能转移到 i 的条件为 $s_j = s_i$，a_{j+1}, a_i 中有一个为 B，简单转移即可。复杂度 $\mathcal{O}(n)$。

会了这道题之后，你就可以秒掉这个题：

一二三（yes）

题目描述

给你 n 个数 $a_{1\sim n}$，其中 $a_i \in \{1, 2, 3\}$，每个数还有一个权值 b_i。

你可以进行如下操作若干次：

- 选择一段区间，满足这段区间的 a_i 之和为 4 或 8，将这段区间删除，并合并剩下的序列

你希望剩下的数字的 a_i 之和尽量小，如果有多种不同的方案，求出 a_i 之和最小的条件下，b_i 的最大可能和。

你需要输出对应的 a_i 之和与 b_i 之和。在有些子任务中，你要给出方案。

[AGC066D] A Independent Set

给定一个长度为 n 的由 A,B 组成的字符串 S，保证 A 的数量 $\leq \frac{n+1}{2}$，交换 S_i 和 S_{i+1} 的代价为 x_i，求让 S 中所有 A 不相邻的最小代价。

$n \leq 10^6$

[AGC066D] A Independent Set 题解

感觉有点虚高，这个做法应该比上题自然。

[AGC066D] A Independent Set 题解

感觉有点虚高，这个做法应该比上题自然。
我们给原序列最后加一个 B，那么最终序列一定可以划分为 ABAB...AB,BBBB 这两种序列交替出现。

[AGC066D] A Independent Set 题解

感觉有点虚高，这个做法应该比上题自然。
我们给原序列最后加一个 B，那么最终序列一定可以划分为 ABAB...AB,BBBB 这两种序列交替出现。

设 A 为 1，B 为 -1，s_i 为前缀和，设 f_i 表示前 i 个字符的答案。
那么如果 $S_i = B$，则 $f_i \leftarrow f_{i-1}$，否则若 $s_i = s_j$，则 $f_i \leftarrow f_j + w(j+1, i)$，其中 $w(l, r)$ 表示将区间 $[l, r]$ 中所有 A 移动到和 l 同奇偶的位置的代价。

[AGC066D] A Independent Set 题解

感觉有点虚高，这个做法应该比上题自然。
我们给原序列最后加一个 B，那么最终序列一定可以划分为 ABAB...AB,BBBB 这两种序列交替出现。

设 A 为 1，B 为 -1，s_i 为前缀和，设 f_i 表示前 i 个字符的答案。
那么如果 $S_i = B$，则 $f_i \leftarrow f_{i-1}$，否则若 $s_i = s_j$，则 $f_i \leftarrow f_j + w(j+1, i)$，其中 $w(l, r)$ 表示将区间 $[l, r]$ 中所有 A 移动到和 l 同奇偶的位置的代价。

设 sum_i 表示从 1 交换到 i 的代价，那么设区间 $[l, r]$ 中 A 的位置分别为 $b_1, b_2, \ldots, b_k$，目标位置分别为 $c_1, c_2, \ldots, c_k$，那么 $w(l, r) = \sum_{i=1}^{k} |sum_{b_i} - sum_{c_i}|$。

[AGC066D] A Independent Set 题解

但是一个 i 可能从很多 j 转移，并且这个 $w(l,r)$ 无法快速计算。继续观察，如果 $s_j = s_i$ 且 (j,i) 中还存在一个 k 满足 $s_k = s_i = s_j$，那么这两段是可以分开计算的，所以我们每次转移只需要用最近的满足 $s_j = s_i$ 的 j 即可。

[AGC066D] A Independent Set 题解

但是一个 i 可能从很多 j 转移，并且这个 $w(l,r)$ 无法快速计算。继续观察，如果 $s_j = s_i$ 且 (j,i) 中还存在一个 k 满足 $s_k = s_i = s_j$，那么这两段是可以分开计算的，所以我们每次转移只需要用最近的满足 $s_j = s_i$ 的 j 即可。

此时区间中所有 A 一定都是向后移动或向前移动（因为如果一个向后一个向前一定可以划分为两段），所以可以将代价写为 $|\sum_{i=1}^{k} sum_{b_i} - \sum_{i=1}^{k} sum_{c_i}|$，于是预处理两个前缀和就可以 $\mathcal{O}(1)$ 计算了。总复杂度为 $\mathcal{O}(n)$。

[AGC036E] ABC String

给定一个只包含字符 ABC 的字符串 S，你需要求出 S 最长的子序列，满足子序列中相邻的两个位置字符不同，且 ABC 的数量相等。

$|S| \leq 10^6$

[AGC036E] ABC String 题解

我们考虑一个一个删除 S 中的字符来得到最长的子序列，首先先将 S 中相同的字母缩成一段。设三个字母的出现次数分别为 C_A, C_B, C_C，不妨设 $C_A \leq C_B \leq C_C$，那么答案的上限为 C_A，于是我们要删除字符使 $C_A = C_B = C_C$，且删除的字符的数量尽量少。

[AGC036E] ABC String 题解

我们考虑一个一个删除 S 中的字符来得到最长的子序列，首先先将 S 中相同的字母缩成一段。设三个字母的出现次数分别为 C_A, C_B, C_C，不妨设 $C_A \le C_B \le C_C$，那么答案的上限为 C_A，于是我们要删除字符使 $C_A = C_B = C_C$，且删除的字符的数量尽量少。

我们先考虑 $C_A \le C_B = C_C$ 的情况，这个时候字符串肯定长成 $CB \dots BCABC \dots CBACB \dots BC \dots CB$ 这样的形式，即两个 A 之间有一串 $BC \dots BC$，这个时候我们每次删相邻两个 BC 或 CB，那么一定可以删到 $C_A = C_B = C_C$，于是此时的答案就是 C_A。

[AGC036E] ABC String 题解

如果是 $C_A \le C_B \le C_C$ 的情况，我们目标就是在删除尽量少的 A 的情况下删除 C 直到 $C_A \le C_B = C_C$。于是可以先考虑能否删除每个 C，然后如果仍然没满足条件，就删 AC 或 CA，直到 $C_A \le C_B = C_C$ 即可，这样一定是最优的。

[AGC036E] ABC String 题解

如果是 $C_A \le C_B \le C_C$ 的情况，我们目标就是在删除尽量少的 A 的情况下删除 C 直到 $C_A \le C_B = C_C$。于是可以先考虑能否删除每个 C，然后如果仍然没满足条件，就删 AC 或 CA，直到 $C_A \le C_B = C_C$ 即可，这样一定是最优的。

用一个链表来维护即可，复杂度为 $\mathcal{O}(n)$。

[AGC062D] Walk Around Neighborhood

有 n 个正偶数 D_i，一个人初始在 $(0,0)$，每次他可以选择一个未被擦除的 d_i 擦去，然后走到一个和自己曼哈顿距离为 d_i 的位置。n 次操作后，他必须回到原点 $(0,0)$。判断是否有解，如果有解，设他在第 i 次操作后在 (x_i, y_i) 位置，求出 $\max\{|x_i| + |y_i|\}$ 的最小值。

$n, D_i \leq 2 \times 10^5$

[AGC062D] Walk Around Neighborhood 题解

* 本题解为加强版，加强版中 $n, D_i \leq 10^6$，并且不保证 D_i 是偶数。

[AGC062D] Walk Around Neighborhood 题解

* 本题解为加强版，加强版中 $n, D_i \le 10^6$，并且不保证 D_i 是偶数。

将 D_i 从小到大排序，然后为了方便思考，把曼哈顿距离转化为切比雪夫距离，每次一个点就能走到一个正方形上的点。先考虑无解条件，如果 $\sum_{i=1}^{n-1} D_i < D_n$，那么一定无解，否则一定有解，且答案在 $[\frac{D_n}{2}, D_n]$ 之间。

[AGC062D] Walk Around Neighborhood 题解

* 本题解为加强版，加强版中 $n, D_i \le 10^6$，并且不保证 D_i 是偶数。

将 D_i 从小到大排序，然后为了方便思考，把曼哈顿距离转化为切比雪夫距离，每次一个点就能走到一个正方形上的点。先考虑无解条件，如果 $\sum_{i=1}^{n-1} D_i < D_n$，那么一定无解，否则一定有解，且答案在 $[\frac{D_n}{2}, D_n]$ 之间。

考虑证明，下限为 $\frac{D_n}{2}$ 是显然的，因为要走 D_n 这一步。然后我们可以构造出一组答案为 D_n 的解出来，因为 $\sum_{i=1}^{n-1} D_i \ge D_n$，所以可以先用一些 D_i 走到距离原点边长为 D_n 的正方形上。

[AGC062D] Walk Around Neighborhood 题解

接着除了最后一步 D_n，因为 $D_i < D_n$，所以可以一直在正方形边界上走，最后用一步 D_n 回去。于是就证明了答案一定在 $[\frac{D_n}{2}, D_n]$ 内，现在我们考虑如何判断一个答案是否合法。

[AGC062D] Walk Around Neighborhood 题解

接着除了最后一步 D_n，因为 $D_i < D_n$，所以可以一直在正方形边界上走，最后用一步 D_n 回去。于是就证明了答案一定在 $[\frac{D_n}{2}, D_n]$ 内，现在我们考虑如何判断一个答案是否合法。

假设当前的答案是 r，那么整个行走的过程都必须在距离原点半径为 r 的正方形内。我们尝试找到一种最优的策略，观察发现，我们的方案一定是先从原点走一些 D_i 到正方形边缘上，然后再在正方形边缘上随便走（因为 $a_i \le \frac{r}{2}$，所以一定可以一直停在正方形边上），最后再走一些 D_i 回到原点。

[AGC062D] Walk Around Neighborhood 题解

接着除了最后一步 D_n，因为 $D_i < D_n$，所以可以一直在正方形边界上走，最后用一步 D_n 回去。于是就证明了答案一定在 $[\frac{D_n}{2}, D_n]$ 内，现在我们考虑如何判断一个答案是否合法。

假设当前的答案是 r，那么整个行走的过程都必须在距离原点半径为 r 的正方形内。我们尝试找到一种最优的策略，观察发现，我们的方案一定是先从原点走一些 D_i 到正方形边缘上，然后再在正方形边缘上随便走（因为 $a_i \le \frac{r}{2}$，所以一定可以一直停在正方形边上），最后再走一些 D_i 回到原点。

因为从原点走到正方形上和走回来是等价的，所以问题就是如何判断一个集合 D_i 能否从原点走到正方形边上。

[AGC062D] Walk Around Neighborhood 题解

设 $s = \sum D_i[D_i < r]$，如果 $s \geq r$，那么一定有解。现在考虑 $s < r$ 且有 $D_i \geq r$ 时的答案，可以发现对于一个 $\geq r$ 的 D_i 最多只会用一个，因为每次走的时候要么直接无法走到半径为 r 的正方形内或者可以直接走到正方形边上。并且越小的 D_i 用不容易走出去，所以我们只会用最小的 $\geq r$ 的 D_i，记为 x。

[AGC062D] Walk Around Neighborhood 题解

设 $s=\sum D_i[D_i<r]$，如果 $s\geq r$，那么一定有解。现在考虑 $s<r$ 且有 $D_i\geq r$ 时的答案，可以发现对于一个 $\geq r$ 的 D_i 最多只会用一个，因为每次走的时候要么直接无法走到半径为 r 的正方形内或者可以直接走到正方形边上。并且越小的 D_i 用不容易走出去，所以我们只会用最小的 $\geq r$ 的 D_i，记为 x。

而对于 $<r$ 的 D_i，因为 $s<r$，而且在离原点越远的位置使用 x 越容易有解，所以一定会先走到离原点半径为 s 的正方形上，然后使用一次 x，于是有解的条件就是 $x-s\leq r$。

[AGC062D] Walk Around Neighborhood 题解

现在考虑枚举答案 r，我们要把所有 D_i 分成两部分使得两部分都满足上述条件。因为有一个 $\geq r$ 的 D_i 肯定比没有好，所以我们给两部分分别选出两个最小的 $\geq r$ 的 D_i，如果没有就不选，这样就可以计算出两个部分 s 的下界。于是问题就变成了能否将 $< D_i$ 的部分分成两个集合，使得两个集合的值都分别 $\geq$ 某个数。

[AGC062D] Walk Around Neighborhood 题解

现在考虑枚举答案 r，我们要把所有 D_i 分成两部分使得两部分都满足上述条件。因为有一个 $\geq r$ 的 D_i 肯定比没有好，所以我们给两部分分别选出两个最小的 $\geq r$ 的 D_i，如果没有就不选，这样就可以计算出两个部分 s 的下界。于是问题就变成了能否将 $< D_i$ 的部分分成两个集合，使得两个集合的值都分别 $\geq$ 某个数。

根据上述条件我们发现，如果一个实数 x 可以作为答案且 $x \neq \frac{D_n}{2}$，那么 $\lfloor x \rfloor$ 也一定可以作为答案。所以除了 $\frac{D_n}{2}$ 以外，其他答案一定是整数，于是我们可以直接枚举答案 r，每次将所有 $< r$ 的 D_i 加入背包，然后用一次 _Find_next 即可判断。时间复杂度 $\mathcal{O}(\frac{n^2}{\omega})$。(默认值域与 n 同阶)

[AGC062D] Walk Around Neighborhood 题解

继续思考我们发现，如果当前 $< r$ 的 D_i 的和已经很大了，那么一定可以将这些数分成两组使得这两组的和都 $\geq r$。具体来说，如果 $(\sum D_i[D_i < r]) \geq 3r$，那么向一个集合中加入 D_i 直到集合的和第一次 $\geq r$，剩下的数的和也一定 $\geq r$。

[AGC062D] Walk Around Neighborhood 题解

继续思考我们发现，如果当前 $< r$ 的 D_i 的和已经很大了，那么一定可以将这些数分成两组使得这两组的和都 $\geq r$。具体来说，如果 $(\sum D_i[D_i < r]) \geq 3r$，那么向一个集合中加入 D_i 直到集合的和第一次 $\geq r$，剩下的数的和也一定 $\geq r$。

特判掉这种情况后，就只需要做 D_i 和为 $\mathcal{O}(n)$ 的背包。于是 D_i 最多有 $\mathcal{O}(\sqrt{n})$ 种不同的值，可以使用二进制分组背包，复杂度为 $\mathcal{O}(\frac{n\sqrt{n}}{\omega})$。

[AGC059A] My Last ABC Problem

定义一个由 ABC 构成的字符串 s 的美丽度为，最少用多少次如下操作可以使得 s 中所有字符相同：

- 选择一个 s 的子串 $s_{l,\dots,r}$ 和一个 ABC 到 ABC 的双射，将 $s_{l,\dots,r}$ 中的每个字符做一次映射。

给定一个长度为 n 的字符串 s 和 q 次询问，每次给出 $[l,r]$，求 $s_{l,\dots,r}$ 的美丽度。

$n, q \le 10^5$

[AGC059A] My Last ABC Problem 题解

可以发现字符相同的段可以缩起来，则每次操作至多删除两个字符。手玩发现，对于长度 ≥ 5 的串，每次一定可以删除两个字符，而对于所有长度为 4 的串，答案一定为 2，所以如果 $|s|$ 为偶数，那么答案就是 $\frac{|s|}{2}$。

[AGC059A] My Last ABC Problem 题解

可以发现字符相同的段可以缩起来，则每次操作至多删除两个字符。手玩发现，对于长度 ≥ 5 的串，每次一定可以删除两个字符，而对于所有长度为 4 的串，答案一定为 2，所以如果 $|s|$ 为偶数，那么答案就是 $\frac{|s|}{2}$。

否则，最后一步删到长度为 3 时，答案取决于最左和最右边的两个字符是否相同，而这两个字符就是一开始的 s 最左和最右的字符，答案为 $\left\lceil \frac{|s|-1+[s_1 \neq s_{|s|}]}{2} \right\rceil$。处理出 $[s_i \neq s_{i-1}]$ 的前缀和数组即可 $\mathcal{O}(1)$ 回答询问，复杂度为 $\mathcal{O}(n+q)$。

[AGC057B] 2A + x

给定一个长度为 n 的数列 a 和正整数 x，每次你可以选择一个 i，然后令 $a_i \leftarrow 2a_i + x$，求在任意次操作后 a 的极差最小是多少。

$n \leq 10^5, a_i, x \leq 10^9$

[AGC057B] 2A + x 题解

先将 a_i 排序，首先不可能出现 a_n 操作过但某个 a_i 没操作过的情况，不然 a_n 不操作一定更优。那么要么就是 a_n 没操作过，要么就是整个序列都操作过。如果都操作过，考虑所有数同时进行操作，如果操作前序列的极差 $< x$，那么若干次操作后就可以变为 0，否则不操作一定更优。

[AGC057B] 2A + x 题解

先将 a_i 排序，首先不可能出现 a_n 操作过但某个 a_i 没操作过的情况，不然 a_n 不操作一定更优。那么要么就是 a_n 没操作过，要么就是整个序列都操作过。如果都操作过，考虑所有数同时进行操作，如果操作前序列的极差 $< x$，那么若干次操作后就可以变为 0，否则不操作一定更优。

于是只需要求出 a_n 不操作极差最小是多少，那么求出每个 a_i 能否得到 a_n，如果不能就求出操作时离 a_n 最接近的两个数 l_i, r_i 是多少。于是问题就是在每个 l_i, r_i 中选一个，求最小极差，将 l_i 排序，枚举 l 的最小值即可。复杂度 $\mathcal{O}(n \log n)$。

[AGC050F] NAND Tree

给定一棵 n 个点的树，每个点有点权 $0/1$，定义一次操作为：

- 选择一条边，将这条边的两个端点合并，新点的点权为原来两个点权的 NAND（$\mathrm{NAND}(x,y)=\neg(x\wedge y)$）。

$n-1$ 次操作后树会只剩下一个节点。求在 $(n-1)!$ 种操作顺序中，最终剩下的点的点权 1 的方案数，答案对 2 取模。

$n\le 300$

[AGC050F] NAND Tree 题解

考虑利用 $\mathrm{mod}2$ 的性质，如果我们某两次操作的两条边没有公共点，那么交换这两次操作对答案是没有影响的，所以可以抵消掉。于是我们将所有操作两两分组，要求每组内的两次操作必须有公共点，即第 $2i-1$ 次操作和第 $2i$ 次操作必须有公共点，如果 n 是偶数，我们就枚举第一次操作的边使得操作数是偶数。

[AGC050F] NAND Tree 题解

考虑利用 $\mathrm{mod}2$ 的性质，如果我们某两次操作的两条边没有公共点，那么交换这两次操作对答案是没有影响的，所以可以抵消掉。于是我们将所有操作两两分组，要求每组内的两次操作必须有公共点，即第 $2i-1$ 次操作和第 $2i$ 次操作必须有公共点，如果 n 是偶数，我们就枚举第一次操作的边使得操作数是偶数。

对于某两次操作，因为有公共点，所以可以写成 $x-y-z$ 的样子。如果 $a_x=a_z$，那么两次操作交换后不影响答案，可以抵消。如果 $a_x\neq a_z$，可以发现两次操作交换和不交换得到的结果一定是 0 和 1，这是因为 $\mathrm{NAND}(\mathrm{NAND}(y,x),z)=x$。于是我们可以每次选择两个操作（不考虑操作间的顺序）进行合并，要求 $a_x\neq a_z$，然后将 x,y,z 缩成一个自由点，自由点表示这个点取 $0/1$ 时都是一种方案。

[AGC050F] NAND Tree 题解

接下来考虑有自由点参与的合并，可以发现，如果 y 是自由点那么一定会抵消，如果 x, z 都是自由点，那么也抵消了！所以如果有自由点参与合并，那么一定是在 x 或 z 并且另一个点不是自由点，而且合并后 x, y, z 又会缩成一个自由点。那么如果树上已经有一个自由点了，又合并出了一个新的自由点，则最终这两个自由点一定会相遇，那么贡献一定为 0。

[AGC050F] NAND Tree 题解

接下来考虑有自由点参与的合并，可以发现，如果 y 是自由点那么一定会抵消，如果 x, z 都是自由点，那么也抵消了！所以如果有自由点参与合并，那么一定是在 x 或 z 并且另一个点不是自由点，而且合并后 x, y, z 又会缩成一个自由点。那么如果树上已经有一个自由点了，又合并出了一个新的自由点，则最终这两个自由点一定会相遇，那么贡献一定为 0。

所以我们一定只会在第一次操作时合并出一个自由点，之后每次操作都是这个自由点作为 x 或 z 参与合并。那么在最终只剩一个点时，这个点一定是自由点，取 1 时就是一种方案。于是问题就变成了，有多少种方案满足，一开始选择三个点进行合并，之后每次选择包含上次合并出来的点的三个点进行合并。

[AGC050F] NAND Tree 题解

可以发现这相当于是一个每次选择两个点的拓扑序计数，把开始三个点也算上，那么就是求有多少个拓扑序 $c_1, \ldots, c_n$，满足所有 c_{2i-1} 都是 c_{2i} 的父亲。开始三个点还要求 $a_x \neq a_z$，为了防止记重，我们还要钦定 $a_{c_1} = 1, a_{c_3} = 0$。然后如果你注意力惊人，可以发现 $a_{c_3} = 0$ 和 c_{2i-1} 是 c_{2i} 的父亲这两个条件都可以忽略，即只需要以所有 $a_u = 1$ 的点作为根进行拓扑序计数再求和即可。

[AGC050F] NAND Tree 题解

可以发现这相当于是一个每次选择两个点的拓扑序计数，把开始三个点也算上，那么就是求有多少个拓扑序 $c_1, \ldots, c_n$，满足所有 c_{2i-1} 都是 c_{2i} 的父亲。开始三个点还要求 $a_x \neq a_z$，为了防止记重，我们还要钦定 $a_{c_1} = 1, a_{c_3} = 0$。然后如果你注意力惊人，可以发现 $a_{c_3} = 0$ 和 c_{2i-1} 是 c_{2i} 的父亲这两个条件都可以忽略，即只需要以所有 $a_u = 1$ 的点作为根进行拓扑序计数再求和即可。

为什么这是对的，先考虑 c_{2i-1} 是 c_{2i} 的父亲这个条件，如果某个拓扑序存在 i 不满足这个条件，那么我们可以找到第一个 i，则交换 c_{2i-1} 和 c_{2i} 依然合法，所以可以抵消。再考虑 $a_{c_3} = 0$ 这个条件，如果 $a_{c_3} = 1$，那么在以 c_1, c_3 为根时都会计算一遍，也会抵消。

[AGC050F] NAND Tree 题解

所以直接对每个 $a_u = 1$ 的点为根做拓扑序计数即可，因为拓扑序个数为 $\frac{n!}{\prod siz_u!}$，所以我们求出所有 $x!$ 中 2 的因子个数即可。算上一开始的枚举第一次操作，总复杂度为 $\mathcal{O}(n^3)$。当然简单换根可以做到 $\mathcal{O}(n^2)$。

[AGC028E] High Elements

给定一个 $1 \sim n$ 的排列 p，求一个字典序最小的 01 串 s，满足：将 s 中 0 在 p 中对应下标的数和 1 对应的数分别拉出来构成两个序列，这两个序列的 $premax$ 个数相等。或者报告无解。

$n \leq 2 \times 10^5$

[AGC028E] High Elements 题解

因为要字典序最小，所以考虑逐位确定，每次相当于给定一个前缀的划分方式，求是否存在后面的划分方式满足条件。假设前面已经分成了两个序列 a, b，设这两个序列的 $premax$ 个数分别为 c_a, c_b，最大值分别为 h_a, h_b。后面的全局 $premax$ 个数为 s，这 s 个数一定会造成贡献，剩下的一些非 $premax$ 的数可能会造成贡献。

[AGC028E] High Elements 题解

因为要字典序最小，所以考虑逐位确定，每次相当于给定一个前缀的划分方式，求是否存在后面的划分方式满足条件。假设前面已经分成了两个序列 a, b，设这两个序列的 $premax$ 个数分别为 c_a, c_b，最大值分别为 h_a, h_b。后面的全局 $premax$ 个数为 s，这 s 个数一定会造成贡献，剩下的一些非 $premax$ 的数可能会造成贡献。

可以发现，一个非 $premax$ 的位置放在它前一个 $premax$ 所在序列中贡献为 0，那么如果剩下的非 $premax$ 位置对 a, b 序列都有贡献，那么可以交换两个对 a, b 有贡献的非 $premax$ 位置，这样两边答案都减 1。一直调整下去，一定存在一边全是由 $premax$ 造成贡献，不妨设是 a 数组。

[AGC028E] High Elements 题解

假设 b 数组中有 k 个全局 $premax$，有 m 个非 $premax$ 有贡献，那么有等式：

$$c_a + s - k = c_b + k + m$$

[AGC028E] High Elements 题解

假设 b 数组中有 k 个全局 $premax$，有 m 个非 $premax$ 有贡献，那么有等式：

$$c_a + s - k = c_b + k + m$$

移项得 $2k + m = c_a + s - c_b$，右边部分是一个常数。于是我们可以将 $premax$ 赋权值 2，非 $premax$ 位置赋权值 1，那么就相当于在后面选一个权值和恰好为 $c_a + s - c_b$ 的上升子序列，并且开头要大于 h_b。由于权值只有 $1, 2$，所以如果有一个权值和为 x 的上升子序列，那么一定存在和为 $x - 2$ 的上升子序列，所以我们只需要求所有权值和为奇数或偶数的上升子序列中权值和最大是多少。

[AGC028E] High Elements 题解

于是我们先用两棵线段树维护出整个序列中以 $> x$ 的数开头，并且权值和为奇数/偶数的上升子序列中权值和最大是多少，然后每次删去当前的数，就可以查询这个后缀的答案，相当于单点修改，区间查 $\max$。复杂度为 $\mathcal{O}(n \log n)$。

[AGC014F] Strange Sorting

给定一个 $1 \sim n$ 的排列 p，定义一次操作为将排列中所有 $premax$ 按原来顺序移到排列末尾，求经过多少次操作后 p 会变得有序。

$n \leq 2 \times 10^5$。

[AGC014F] Strange Sorting 题解

可以发现 1 这个数对 $premax$ 没有影响，假设我们先求出了排列去除 1 的答案，考虑求出加上 1 后的贡献。如果我们能处理这个子问题，那么就可以依次枚举 i 从 n 到 1，然后求出 i 对 $[i+1,n]$ 的子序列的贡献，求和即可。

[AGC014F] Strange Sorting 题解

可以发现 1 这个数对 $premax$ 没有影响，假设我们先求出了排列去除 1 的答案，考虑求出加上 1 后的贡献。如果我们能处理这个子问题，那么就可以依次枚举 i 从 n 到 1，然后求出 i 对 $[i+1,n]$ 的子序列的贡献，求和即可。

假设 $[2,n]$ 的答案是 t，如果 $t=0$，那么就是看 1 是否在整个序列的开头，如果不是那么就会执行一次操作。

[AGC014F] Strange Sorting 题解

可以发现 1 这个数对 $premax$ 没有影响，假设我们先求出了排列去除 1 的答案，考虑求出加上 1 后的贡献。如果我们能处理这个子问题，那么就可以依次枚举 i 从 n 到 1，然后求出 i 对 $[i+1,n]$ 的子序列的贡献，求和即可。

假设 $[2,n]$ 的答案是 t，如果 $t=0$，那么就是看 1 是否在整个序列的开头，如果不是那么就会执行一次操作。

否则，我们设 f 表示 $[2,n]$ 在最后一次操作时序列开头是多少，显然有 $f>2$，因为如果 $f=2$，那么最后一次操作一定会让 2 移到非开头位置，此时肯定没有排好序。那么最后一次操作时，$1,2,f$ 的相对位置可能有 $(1,f,2),(f,1,2),(f,2,1)$，如果是 $(f,1,2)$ 那么最后一次操作 1 也会排好，其余两种情况则会多排一轮。

[AGC014F] Strange Sorting 题解

现在问题就是如何判断最后一次操作时 $(1, 2, f)$ 的相对关系，可以发现 $(1, f, 2), (f, 2, 1)$ 都会有 1 的贡献，而 $(f, 1, 2)$ 没有贡献，即与 $(1, f, 2)$ 循环同构即会有贡献，否则就没有。

[AGC014F] Strange Sorting 题解

现在问题就是如何判断最后一次操作时 $(1, 2, f)$ 的相对关系，可以发现 $(1, f, 2), (f, 2, 1)$ 都会有 1 的贡献，而 $(f, 1, 2)$ 没有贡献，即与 $(1, f, 2)$ 循环同构即会有贡献，否则就没有。

那么我们可以大胆猜测任何一次操作都不会改变 $(1, 2, f)$ 的循环同构顺序，那么只需要判断最开始排列 $(1, 2, f)$ 的顺序即可。

~~人类如何想出这一步？~~

[AGC014F] Strange Sorting 题解

现在问题就是如何判断最后一次操作时 $(1,2,f)$ 的相对关系，可以发现 $(1,f,2),(f,2,1)$ 都会有 1 的贡献，而 $(f,1,2)$ 没有贡献，即与 $(1,f,2)$ 循环同构即会有贡献，否则就没有。

那么我们可以大胆猜测任何一次操作都不会改变 $(1,2,f)$ 的循环同构顺序，那么只需要判断最开始排列 $(1,2,f)$ 的顺序即可。

~~人类如何想出这一步？~~

考虑证明上述结论，首先有一个结论是 f 除了最后一次操作其余时刻一定不会被操作（即不是 $premax$），因为如果被操作了那么 f 前一定会有个 $<f$ 的数，那么之后 f 一定不会成为开头。可以直接枚举 $(1,2,f)$ 的 6 种全排列考虑：

[AGC014F] Strange Sorting 题解

- 如果相对顺序为 $(f, 1, 2)$ 或 $(f, 2, 1)$，因为 f 不是 $premax$，所以 $1, 2$ 也不是，即所有数都不会被操作。

[AGC014F] Strange Sorting 题解

- 如果相对顺序为 $(f,1,2)$ 或 $(f,2,1)$，因为 f 不是 $premax$，所以 $1,2$ 也不是，即所有数都不会被操作。
- 如果相对顺序为 $(1,f,2)$ 或 $(2,f,1)$，那么后两个数不是 $premax$，无论第一个数是否是 $premax$ 都不会改变循环同构。

[AGC014F] Strange Sorting 题解

- 如果相对顺序为 $(f,1,2)$ 或 $(f,2,1)$，因为 f 不是 $premax$，所以 $1,2$ 也不是，即所有数都不会被操作。
- 如果相对顺序为 $(1,f,2)$ 或 $(2,f,1)$，那么后两个数不是 $premax$，无论第一个数是否是 $premax$ 都不会改变循环同构。
- 如果相对顺序为 $(1,2,f)$ 或 $(2,1,f)$，可以发现当第二个数是 $premax$ 时第一个数一定是 $premax$，也都不会改变循环同构。

[AGC014F] Strange Sorting 题解

- 如果相对顺序为 $(f,1,2)$ 或 $(f,2,1)$，因为 f 不是 $premax$，所以 $1,2$ 也不是，即所有数都不会被操作。
- 如果相对顺序为 $(1,f,2)$ 或 $(2,f,1)$，那么后两个数不是 $premax$，无论第一个数是否是 $premax$ 都不会改变循环同构。
- 如果相对顺序为 $(1,2,f)$ 或 $(2,1,f)$，可以发现当第二个数是 $premax$ 时第一个数一定是 $premax$，也都不会改变循环同构。

[AGC014F] Strange Sorting 题解

- 如果相对顺序为 $(f,1,2)$ 或 $(f,2,1)$，因为 f 不是 $premax$，所以 $1,2$ 也不是，即所有数都不会被操作。
- 如果相对顺序为 $(1,f,2)$ 或 $(2,f,1)$，那么后两个数不是 $premax$，无论第一个数是否是 $premax$ 都不会改变循环同构。
- 如果相对顺序为 $(1,2,f)$ 或 $(2,1,f)$，可以发现当第二个数是 $premax$ 时第一个数一定是 $premax$，也都不会改变循环同构。

综上，我们可以直接用一开始 $(1,2,f)$ 的顺序来确定是否有贡献，于是直接枚举 i 从 n 到 1，记录此时的 t,f 即可。复杂度 $\mathcal{O}(n)$。

[ARC206E] Rectangle Coloring

给定一个 $n \times n$ 的棋盘，称在棋盘边上但不在角落上的格子为好格子，给定每个好格子的权值（共 $4n-8$ 个）。初始时所有格子都是白色，定义一次操作为：

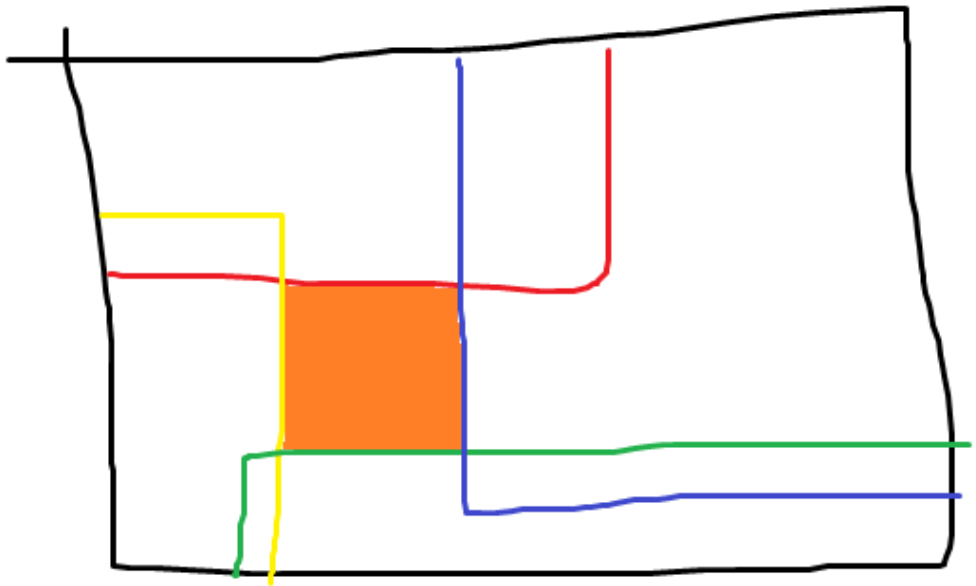
- 选择两个不同的未被选择过的好格子，然后将这两个格子确定的矩形染黑，代价为这两个格子的权值之和。

求将所有格子染黑的最小代价。

$4 \le n \le 5 \times 10^4$

[ARC206E] Rectangle Coloring 题解

因为每次操作至多覆盖一个角落，所以我们至少需要 4 次操作，每条边至少需要选两个格子。那第一想法就是每条边选权值最小的两个格子，考虑什么情况下不合法，如下图：



(橙色格子是没有被覆盖到的，这张图对称的情况也不合法)

[ARC206E] Rectangle Coloring 题解

于是我们先考虑所有每条边选 8 个点的合法情况的最小值。然后考虑选了 >8 个点的情况，那么就相较于上图，就要至少再增加两个点，显然这两个点不能在同一条边上，而如果增加到邻边上使得矩阵被覆盖完了，那么一定可以删掉两个点，于是这种情况不优。

[ARC206E] Rectangle Coloring 题解

于是我们先考虑所有每条边选 8 个点的合法情况的最小值。然后考虑选了 >8 个点的情况，那么就相较于上图，就要至少再增加两个点，显然这两个点不能在同一条边上，而如果增加到邻边上使得矩阵被覆盖完了，那么一定可以删掉两个点，于是这种情况不优。

最后还有可能加两个点到对边上，可以发现无论这两个点加到哪，都一定可以覆盖完整个矩阵，所以答案就是对边前 3 小值 $+$ 另一组对边前 2 小值。两种情况都可以简单计算，总复杂度为 $\mathcal{O}(n)$。

P9870 [NOIP2023] 双序列拓展

序列 a 的拓展 b 为将 a 中每个数替换为任意正整数个这个数后的序列，给定两个序列 $(x_1, \ldots, x_n), (y_1, \ldots, y_m)$，你需要分别找到一个 x, y 的长度为 $l_0 = 10^{100}$ 的拓展 f, g，使得 $\forall 1 \le i, j \le l_0, (f_i - g_i)(f_j - g_j) > 0$，求是否存在这样两个序列。

另外还给定 q 次修改，每次会将 f, g 中一些位置修改为其他数，在最开始和每次修改完输出是否有解。修改之间相互独立。

$n, m \le 5 \times 10^5, q \le 60, x_i, y_i \le 10^9$

P9870 [NOIP2023] 双序列拓展 题解

考虑一个单次 $\mathcal{O}(nm)$ 的暴力做法，首先不妨设 $x_1 < y_1$，那就是要求所有 $f_i < g_i$。我们设一个矩阵 a，其中 $a_{i,j} = [x_i < y_j]$，那么有解的条件就是存在一条从 $(1, 1)$ 向右或向下或向右下走到 (n, m) 的全是 1 的路径。

P9870 [NOIP2023] 双序列拓展 题解

考虑一个单次 $\mathcal{O}(nm)$ 的暴力做法，首先不妨设 $x_1 < y_1$，那就是要求所有 $f_i < g_i$。我们设一个矩阵 a，其中 $a_{i,j} = [x_i < y_j]$，那么有解的条件就是存在一条从 $(1, 1)$ 向右或向下或向右下走到 (n, m) 的全是 1 的路径。

接下来我们可以考虑特殊性质，即 x_n 是最小值且 y_m 是最大值，如果存在一个 $y_j \le x_n$ 那么就有一列 0，显然无解，$x_i \ge y_m$ 同理。所以有 x_n 小于所有 y_j，y_m 大于所有 x_i，即矩阵 a 的最后一行和最后一列都是 1。

P9870 [NOIP2023] 双序列拓展 题解

考虑一个单次 $\mathcal{O}(nm)$ 的暴力做法，首先不妨设 $x_1 < y_1$，那就是要求所有 $f_i < g_i$。我们设一个矩阵 a，其中 $a_{i,j} = [x_i < y_j]$，那么有解的条件就是存在一条从 $(1,1)$ 向右或向下或向右下走到 (n,m) 的全是 1 的路径。

接下来我们可以考虑特殊性质，即 x_n 是最小值且 y_m 是最大值，如果存在一个 $y_j \le x_n$ 那么就有一列 0，显然无解，$x_i \ge y_m$ 同理。所以有 x_n 小于所有 y_j，y_m 大于所有 x_i，即矩阵 a 的最后一行和最后一列都是 1。

那么我们就是要从 $(1,1)$ 走到最后一行或最后一列，接下来考虑 $x_{1\sim n-1}$ 的最小值 x_p 和 $y_{1\sim m-1}$ 的最大值 y_q。如果 x_p 小于所有 $y_{1\sim m-1}$，那么相当于矩阵第 p 全是 1。

P9870 [NOIP2023] 双序列拓展 题解

那么有解条件就是能走到第 p 行或者第 m 列，我们从 (n, m) 递归到了 (p, m) 的子问题。同理如果 y_q 大于所有 $x_{1\sim n-1}$，那么就可以递归到 (n, q)。而如果上述两个条件都不满足，相当于同时存在一行和一列全 0，则一定无解。于是我们可以直接递归处理，单次复杂度为 $\mathcal{O}(n+m)$。

P9870 [NOIP2023] 双序列拓展 题解

那么有解条件就是能走到第 p 行或者第 m 列，我们从 (n,m) 递归到了 (p,m) 的子问题。同理如果 y_q 大于所有 $x_{1\sim n-1}$，那么就可以递归到 (n,q)。而如果上述两个条件都不满足，相当于同时存在一行和一列全 0，则一定无解。于是我们可以直接递归处理，单次复杂度为 $\mathcal{O}(n+m)$。

接下来考虑原问题，还是考虑所有 x 中的最小值 x_p 和 y 中的最大值 y_q，那么还是要求 x_p 要小于所有 y_j，y_q 要大于所有 x_i，相当于矩阵中间有一行和一列全为 1。于是我们就可以分为 $(1,1)$ 走到第 p 行或第 q 列、第 p 行或第 q 列走到 (n,m) 两个子问题来考虑，这两个子问题就是特殊性质，和上面一样的做法。于是就做完了，处理出前缀和后缀 min,max 的位置，复杂度为 $\mathcal{O}(q(n+m))$。

其他题目

最后还有一些其他 Ad-hoc 题目，一般是通过打表来找规律再推性质，这里不方便讲，就只放一下题目链接，大家可以自己下来去做：

- [AGC061A] Long Shuffle
- [ARC207B] Balanced Neighbors 2
- [AGC061B] Summation By Construction

Contents

3

总结 & 结语

Ad-hoc 的一些技巧

Ad-hoc 主要还是自己打表、找性质，但是也是有一些 trick 的，比如：

- 一直找必要条件，然后证充分性。
- 递归到一个子问题上解决，使用归纳法。
- 如果是每次可以删去某个符合条件的区间，然后将两边拼起来，可以考虑一个区间能删完的充要条件。
- 求出答案的某个上界，然后证明每次都可以取到上界，或者只有最后规模很小的时候会有一些偏差。
- 如果是模 2，可以用交换或者其他方法将操作进行一一对应，然后就可以抵消。
- .....

结语

今天主要给大家带来了一些字符串和一些，如果有疑问，可以随时来找我问。

Ad-hoc 在近几年来考得越来越频繁，可见其重要性，要想提升这方面的水平主要也是多刷题。

然后这里挂一个题单的链接：link (id:875445)

希望大家在听完这些题目后，能有自己的领悟、提升！

谢谢大家！