贪心 + 博弈 + 构造

whx1003

2024 年 12 月 15 日

写在前面

- 今天的内容会是贪心 + 博弈 + 构造.
- 内容是 (大体上) 按照知识点排布的, 也会有一些杂题
- 题目可能比较老, 希望大家没见过

Outline

1.1 概述

贪心: 即在程序的每一步骤中都使用当前的最优策略, 最终得到整个问题的最优答案.

- 传统的贪心考察方法包括各种基础算法结合, 比如二分+贪心, 按字典序贪心等.
- 近年来常见的考察方法是: 假设现在要计算某个极值, 比如计算某类序列的极大值, 然后**观察**到做某个操作之后, 一个序列的权值只增不减, 于是就可以假设在取到最大值的时候, 序列满足某个性质, 从而进一步 dp / 构造等.
- 其它还有一些固定的 trick, 如可撤销贪心, 拟阵等

1.1 概述

AGC008B. Contiguous Repainting

给定一个长为 n 的数列 $\{a_i\}$, 初始全部涂成白色. 每次操作, 你可以选择一个长为 k 的连续段全部涂成黑色或白色. 你可以做任意多次操作, 要求最大化最后涂成黑色的格子中的整数之和. 求这个和.

$1 \leq k = n \leq 10^5, |a_i| \leq 10^9$

1.1 概述

AGC008B. Contiguous Repainting

给定一个长为 n 的数列 $\{a_i\}$, 初始全部涂成白色. 每次操作, 你可以选择一个长为 k 的连续段全部涂成黑色或白色. 你可以做任意多次操作, 要求最大化最后涂成黑色的格子中的整数之和. 求这个和.

$1 \leq k = n \leq 10^5, |a_i| \leq 10^9$

结论: 除了最后一轮涂的格子必须颜色相同以外, 其它格子可以任意涂颜色.

因此枚举最后一段是哪一段, 在剩下的格子中选所有正数涂成黑色即可.

1.1 概述

打怪兽

有 n 只怪兽，初始你有 k 点血量，打第 i 个怪兽至少需要 a_i 的血量，打完第 i 个怪兽之后会掉 b_i 的血量，你可以按照任何顺序依次打所有怪兽，问能否打完所有怪兽，并给出一种方案.

$1 \le n \le 10^6, 1 \le a_i, b_i, k \le 10^9$.

1.1 概述

打怪兽

有 n 只怪兽，初始你有 k 点血量，打第 i 个怪兽至少需要 a_i 的血量，打完第 i 个怪兽之后会掉 b_i 的血量，你可以按照任何顺序依次打所有怪兽，问能否打完所有怪兽，并给出一种方案.

$1 \le n \le 10^6, 1 \le a_i, b_i, k \le 10^9$.

考虑最终击败的相邻两个怪兽 i, j.

- 如果先打 i，需要 $\max(a_i, b_i + a_j)$ 的血量.
- 如果先打 j，需要 $\max(a_j, b_j + a_i)$ 的血量.

不妨设血量 $> \max(a_i, a_j)$，则相当于先打 i 需要 $b_i + a_j$ 的血量；先打 j 需要 $b_j + a_i$ 的血量.

1.1 概述

也就是说我们先打 $b_i + a_j$ 更小的更优, 也即可以按照 $a_i - b_i$ 排序打过去即可.

坊间管这个 trick 叫做「邻项交换法」.

一个经典题是「NOIP2012」国王游戏, 以及其加强版 Luogu P2123. 皇后游戏.

1.1 概述

AGC032E. Modulo Pairing

给定一个长为 $2n$ 的数列 $\{a_i\}$ 和正整数 m. 你需要将这些数划分成 n 对 (x_i, y_i) 使得 $(x_i + y_i) \bmod m$ 的最大值最小.

$1 \le n \le 10^5, 1 \le m \le 10^9$. 保证 $a_i \in [0, m)$.

1.1 概述

AGC032E. Modulo Pairing

给定一个长为 $2n$ 的数列 $\{a_i\}$ 和正整数 m. 你需要将这些数划分成 n 对 (x_i, y_i) 使得 $(x_i + y_i) \bmod m$ 的最大值最小.

$1 \le n \le 10^5, 1 \le m \le 10^9$. 保证 $a_i \in [0, m)$.

问题的关键在于如果两个数的和 $\ge m$, 则会减去一个 m.

1.1 概述

观察 1: 如果有两组匹配 $(x,y),(z,w)$, 其中 $y>z$, 但是 $x+y<m, z+w\geq m$, 那么

- 二者最大值是 $\max(x+y, z+w-m)$.
- 交换 y,z 之后的最大值是 $\max(x+z, y+w-m)$

注意到

$$x+z<x+y,\quad y+w-m<y<y+x$$

于是交换之后不会变劣. 因此我们可以假设最终序列中是一段前缀两两匹配得到 $<m$ 的; 一段后缀两两匹配得到 $\geq m$ 的.

进一步, 由于此时前后缀等价于在做没有取模的问题, 而这个问题的最优策略是第一项和最后一项匹配, 第二项和倒数第二项匹配, ..., 于是最终方案是前缀中这样匹配, 后缀中也这样匹配.

1.1 概述

观察 2: 如果中间的分界点向左移动, 则 (在方案合法的前提下) 答案不会变劣.

假设分界点向左移动 2, 我们分类讨论:

- 在新分界点左侧的元素: 按照我们上述的匹配方式, 容易验证其匹配的权值减少了.
- 在原分界点右侧的元素: 类似可验证其匹配的权值减少了.
- 在两个分界点之间的元素, 记为 x, 注意到新的匹配 $x+y-w<x$, 于是其权值也减少了.

因此分界点越靠左越好. 但是太靠左会导致方案不合法, 于是二分这个最优分界点即可. 复杂度 $\mathcal{O}(n\log n)$.

1.2 拟阵

拟阵 (matriod) 是一个用来证明贪心正确性的工具. 其基本想法来自最小生成树的 Kruskal 算法. 在 Kruskal 算法中, 我们贪心考虑每条边,

- 如果它两端不连通, 则可以直接加入最小生成树
- 否则, 找到其所在环上边权最大的边, 然后可以用这条边替换掉它.

这一算法的正确性可以用「图拟阵」来证明. 具体地, 我们考虑一个集合 S, 设 I 是 S 的一些子集构成的集合 (集合的集合), 如果它满足:

1. $\emptyset \in I$
2. (遗传性) 如果 $A \subseteq B \subseteq E$, 且 $B \in I$, 那么 $A \in I$.
3. (交换性) 如果 $A, B \in I$, 且 $|A| > |B|$, 那么存在 $a \in A$ 使得 $B + a \in I$.

1.2 拟阵

一个例子是图拟阵. 我们取 S 为边集, I 为所有没有环的边子集, 那么

- 空集当中没有环
- 如果 B 中没有环, 那么 B 的子集 A 中也没有环
- 如果 A, B 都没有环, 且 $|A| > |B|$, 那么可以找到 A 中的一条边, 使得 B 加上这条边之后也没有环. 否则如果 A 的每条边都和 B 构成环, 那么这条边两端都在 B 中的一个连通块中, 容易说明此时 $|A| \leq |B|$, 矛盾.

因此上述的 (S, I) 构成一个拟阵.

1.2 拟阵

定理 (带权拟阵的贪心性质).

设 (S, I) 是一个拟阵, 且 S 中的每个元素有一个权值 w_i, 那么可以按照如下方法得到 I 中元素和最大的集合: 维护一个集合 S 表示选出的元素, 初始为空集. 按照 w_i 从大到小考虑每个元素, 如果 $S + i \in I$, 则令 $S \leftarrow S + i$.

因此最小生成树的 Kruskal 算法正确性得到证明.

拟阵相关的更多内容可以见 2018 年集训队论文「浅谈拟阵的一些拓展及其应用」. 其中一个比较 non-trivial 的内容是所谓「带权拟阵交」算法. 算法比较繁琐且在 OI 中应用较少, 这里不再赘述.

1.2 拟阵

「JLOI2015」装备购买

游戏里有 n 件装备. 第 i 件装备有 m 个属性, 用向量 $(a_{i,1},...,a_{i,m})$ 表示. 第 i 个装备的价格是 c_i. 现在你想购买尽量多的装备, 使得不存在购买的某一件装备的属性是其他装备的属性的线性组合. 在这个前提下最小化总的花费.

$1 \leq n, m \leq 500$.

1.2 拟阵

「JLOI2015」装备购买

游戏里有 n 件装备. 第 i 件装备有 m 个属性, 用向量 $(a_{i,1},...,a_{i,m})$ 表示. 第 i 个装备的价格是 c_i. 现在你想购买尽量多的装备, 使得不存在购买的某一件装备的属性是其他装备的属性的线性组合. 在这个前提下最小化总的花费.

$1 \le n, m \le 500$.

相当于要找到最多的线性无关的向量, 使得权值和最小. 注意到向量关于线性无关性可以构成一个拟阵 (线性拟阵), 因此直接按照价格排序依次插入, 用线性基维护已经加入的向量的线性无关性即可.

1.3 可撤销贪心

基本形式: 在贪心时不完全放弃暂时不优的策略, 而是每次都重新考虑当前的贪心方案, 并和之前舍弃的策略作比较, 取最优策略.

- 和「模拟费用流」基本上是一个东西. 有些时候和「wqs 二分」也是一个东西.
- 费用流: 关键性质是在走完一条边之后会建一条反向边. 走这条反向边对应于可撤销贪心中的反悔操作.
- 一些历史: WC2019 讲了这个东西 (模拟费用流) 之后当年 NOI2019 就出了一个题 (「NOI2019」序列). 自此之后被 OI 圈广泛熟知.
- 可撤销贪心和模拟费用流的区别主要是是否显式地建出费用流图.

1.3 可撤销贪心

Luogu P1484. 种树

有一排 n 个数 $a_1, \ldots, a_n$，你需要从中选择 k 个两两不相邻的数，使得总和最大.

$1 \le n \le 3 \times 10^5, |a_i| \le 10^6$.

1.3 可撤销贪心

Luogu P1484. 种树

有一排 n 个数 $a_1, \ldots, a_n$，你需要从中选择 k 个两两不相邻的数，使得总和最大.

$1 \le n \le 3 \times 10^5, |a_i| \le 10^6$.

考虑按照 a_i 从大往小的顺序贪心选择 a_i. 选择一个 a_i 之后 a_{i-1} 和 a_{i+1} 就不能再选了，于是这一操作不一定优. 注意到如果撤销这一操作，则一定是 a_{i-1} 和 a_{i+1} 一起选 (不然其和不超过 a_i).

因此我们可以设置一个反悔机制，即付出 $a_{i-1} + a_{i+1} - a_i$ 的代价来撤销这一操作. 注意到撤销这一操作之后会多选一个数，且 a_{i-2}, a_{i+2} 均不能再选，于是一个撤销操作和一个数其实没有区别!

1.3 可撤销贪心

因此, 我们可以在选了 a_i 之后同时删去 a_{i-1}, a_i, a_{i+1}, 然后将 $a_{i-1} + a_{i+1} - a_i$ 放回原位继续算法.

一个可能的问题是, 如果我撤销了此前的一个撤销, 那么原来的撤销是否还可用? 这也是不能的, 因为此时 a_{i-1} 和 a_{i+1} 均已被选, 不能再选 a_i. 于是我们的算法是对的.

只需要用大根堆维护最大值所在位置即可. 复杂度 $\mathcal{O}(n \log n)$.

1.3 可撤销贪心

不知道来源题

给定 n 个区间，你需要从中选出尽量多**对**区间，使得每对区间中的两个区间不相交。每个区间只能被选择一次。

$1 \le n \le 4 \times 10^5$.

1.3 可撤销贪心

不知道来源题

给定 n 个区间，你需要从中选出尽量多**对**区间，使得每对区间中的两个区间不相交。每个区间只能被选择一次。

$1 \le n \le 4 \times 10^5$。

下面记一对匹配的区间为 (I_1, I_2)，满足 $I_1 < I_2$，且分别称为这对匹配的第一个和第二个区间。将所有区间按照左端点并依次考虑。

- 如果此前有区间的右端点小于当前的左端点，则直接匹配。
- 否则找到已匹配的区间对当中第二个区间右端点最小的一个。如果该区间的右端点小于正在考虑的区间的右端点，则将这对匹配中的第一个区间转而和当前区间匹配。

1.3 可撤销贪心

考虑一下为什么是对的. 首先注意到我们的问题实际上是一个匹配问题, 也就是说每次新增一个区间的时候要找增广路.

设现在增广路从当前考虑的区间 I 出发, 一直走到某对匹配的区间 (I_1, I_2), 然后这对区间一者与某个未匹配区间 I' 匹配了.

- I' 的右端点在 I 的左端点的右侧, 否则可以直接匹配 (I', I).
- 因此, $I' > I_1$, 否则 I_1 左端点在 I 左端点右侧.
- 进一步, I' 的右端点小于 I_2 的右端点, 这是由算法中调整的一步保证的.
- 因此, 只可能是 (I_1, I') 成为新的匹配. 此时, 再往增广路上回溯一步会发现与 I_2 匹配的区间的右端点在 I_2 右侧.

1.3 可撤销贪心

作为结论，如果这样的增广路存在，那么从 I 出发沿增广路方向区间右端点是递减的. 但是 I' 的右端点在 I 的左端点右侧，这导出增广路第一步的区间右端点也在 I 的左端点右侧，而这是不可能的！

因此，根本就不存在这样的增广路. 于是如果当前区间无法一步进行匹配，则不再需要用它进行匹配，而是直接加入候选集合即可. 为了保证候选集合是“最优”的，还需要进行一步反悔，也即用之前匹配过的更优的区间替换 I.

也即算法是对的. 算法可以用堆简单维护，复杂度 $\mathcal{O}(n \log n)$.

1.3 可撤销贪心

CF335F. Buy One, Get One Free

现在有 n 个物品，每个物品有一个价格 a_i. 你每次买下其中一个物品之后可以免费获得一个**价格严格更低**的物品. 问获得所有物品的最少花费是多少.

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

1.3 可撤销贪心

CF335F. Buy One, Get One Free

现在有 n 个物品，每个物品有一个价格 a_i. 你每次买下其中一个物品之后可以免费获得一个**价格严格更低**的物品. 问获得所有物品的最少花费是多少.

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

考虑将 $\{a_i\}$ 从大到小排序并依次考虑是否购买.

- 如果 a_i 两两不同，那么最优策略就是买最贵的，第 3 贵的，第 5 贵的，...，因为考虑到 a_i 的时候，就算不用免费的机会留给后面的，也不如这里直接用掉然后后面再买.

但是当 $\{a_i\}$ 有重复时会出问题，因为相同价格的物品不能互相免费.

1.3 可撤销贪心

可撤销贪心. 我们还是尽量用免费的机会. 考虑如果现在有两个价格相同的 a_i, 那么我们可以

- 购买两个 a_i.
- 撤销之前的某个 a_j 的免费机会, 然后免费获得两个 a_i.

那么这基本上就可以可撤销贪心了. 但是上面没有考虑完整: 这道题中的增广路 (撤销路) 的长度是可以 > 1 的, 也就是说, 我可以撤销之前的一次撤销.

这也是可以解决的: 一个关键观察是撤销一次撤销操作可以让我获得两个免费的机会, 这和撤销一次免费是相同的! 也就是说我可以把一次撤销看做一个价格为 $2a_i - a_j$ 的免费物品来考虑.

1.3 可撤销贪心

这里可能出现的问题是: 我撤销一个撤销之后需要将它撤销的东西加回来. 在上面的情景中, 这就是原本是通过购买 a_k 使得 a_j 免费, 然后我以 a_j 的代价撤销了这个免费并获得了两个 a_i. 接下来我如果撤销了这个撤销, 则需要把「免费获得 a_j」给加回来.

注意到此时一定有 $2a_i - a_j < a_j$, 于是我可以一开始就不删除 a_j, 因为 $2a_i - a_j$ 一定先于 a_j 考虑. 于是我的算法是没问题的.

一样用堆来维护即可. 复杂度 $\mathcal{O}(n \log n)$.

Outline

2.1 概述

OI 比赛中常见的博弈有如下的几种类型:

- 传统组合数学意义上的经典模型, 如 Nim 博弈, Nash 博弈等.
- 没什么经典模型, 但是比较考验选手对整个博弈的细致分析的, 比如仔细考虑博弈双方的最优策略等.
- 完全是套了一个博弈的壳, 其实是在考察其它内容, 比如组合数学/图论等.

其中第一种在古早时期 (大约 6 年前) 比较喜欢考, 我在役期间考的比较多的是后两种.

今天主要讲的也是后两种.

2.2 SG 函数

- 平等博弈: 在一个给定的局面下, 如果不论是谁做操作, 可能做出的决策集合是相同的, 则称为一个平等博弈.
- SG 函数: 对一个平等博弈, 定义一个局面的 SG 函数为其所有后继局面的 SG 函数值的 mex, 则首先注意到一个游戏先手必胜当且仅当 SG 函数值非零. 其次我们有如下的定理:

定理 (Sprague-Grundy).

对一个由多个平行进行的组合游戏构成的博弈, 其整体的 SG 函数值为各个子游戏的函数值的异或和.

这可以解决一大类 (传统) 博弈问题.

2.2 SG 函数

Nim Game

现在有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，每次从一个当前非空的石子堆当中取至少一个. 无法操作者输，问最后谁会获胜.

$n \le 10^5, a_i \le 10^9$.

2.2 SG 函数

Nim Game

现在有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，每次从一个当前非空的石子堆当中取至少一个. 无法操作者输，问最后谁会获胜.

$n \le 10^5, a_i \le 10^9$.

计算可知：第 i 堆石子的 SG 函数值就是 a_i.

注意到不同石子堆之间构成独立的博弈，于是整个游戏的 SG 函数值就是每堆石子的 SG 函数值的异或和，于是判断异或和是否为 0 即可.

2.2 SG 函数

POJ #2425. A Chess Game

有一个 n 个点的有向无环图，其上有 m 个棋子. Alice 和 Bob 在其上博弈，每次可以将其中一枚棋子沿出边移动一次，允许棋子重叠. Alice 先开始，不能移动则输掉游戏，问最后谁会获胜.

$n \leq 10^3, m \leq 10$.

2.2 SG 函数

POJ #2425. A Chess Game

有一个 n 个点的有向无环图, 其上有 m 个棋子. Alice 和 Bob 在其上博弈, 每次可以将其中一枚棋子沿出边移动一次, 允许棋子重叠. Alice 先开始, 不能移动则输掉游戏, 问最后谁会获胜.

$n \leq 10^3, m \leq 10$.

注意到棋子之间是独立的, 于是相当于每个棋子构成的博弈的和. 在图上递推得到每个点的 SG 函数值, 然后就可以直接异或来看整个游戏的胜负情况.

2.2 SG 函数

Lasker's Nim Game

现在有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，每次从一个当前非空的石子堆当中取至少一个，或者选取一个石子堆分成两个非空的石子堆. 无法操作者输，问最后谁会获胜.

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

2.2 SG 函数

Lasker's Nim Game

现在有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，每次从一个当前非空的石子堆当中取至少一个，或者选取一个石子堆分成两个非空的石子堆. 无法操作者输，问最后谁会获胜.

$n \le 10^5, a_i \le 10^9$.

只需要计算每一堆石子的 SG 函数值.

$$\mathrm{SG}(n) = \mathrm{mex}(\{\mathrm{SG}(k) \mid 0 \le k < n\} \cup \{\mathrm{SG}(k) \oplus \mathrm{SG}(n-k) \mid 0 < k < n\}).$$

n	0	1	2	3	4	5	6	7	8	9	10	11	12
$\mathrm{SG}(n)$	0	1	2	4	3	5	6	8	7	9	10	12	11

2.2 SG 函数

归纳可证: 对 $k \geq 1$,

$$\mathrm{SG}(4k) = 4k - 1, \quad \mathrm{SG}(4k+1) = 4k + 1.$$
$$\mathrm{SG}(4k+2) = 4k + 2, \quad \mathrm{SG}(4k+3) = 4k + 4.$$

2.2 SG 函数

不知道哪里来的题

一个初始有两个数 a, b. 双人博弈, 每次操作依次进行:

- 如果 $a > b$, 则交换 a, b.
- 如果 $a = 0$, 则当前玩家失败, 游戏结束.
- 玩家选择: 将 b 变为 $b \bmod a$, 或者选定一个 $k \geq 1$ 并将 b 减去 a^k.

问最后谁会获胜. $0 \leq a, b \leq 10^{18}$.

2.2 SG 函数

不知道哪里来的题

一个初始有两个数 a, b. 双人博弈, 每次操作依次进行:

- 如果 $a > b$, 则交换 a, b.
- 如果 $a = 0$, 则当前玩家失败, 游戏结束.
- 玩家选择: 将 b 变为 $b \bmod a$, 或者选定一个 $k \geq 1$ 并将 b 减去 a^k.

问最后谁会获胜. $0 \leq a, b \leq 10^{18}$.

首先递归计算 $b \leftarrow b \bmod a$ 之后的胜负情况: 如果是先手必败, 则先手直接做这个操作然后获胜. 否则如果先手必胜, 那么注意到我们有 $b - a^k \equiv b \pmod a$, 因此此时先后手都不会做取模的操作, 否则将直接失败.

2.2 SG 函数

因此问题转化为，有一堆石子共 $\left\lfloor \frac{b}{a} \right\rfloor$ 个，双方每次可以取 $1, a, a^2, \ldots$ 个，无法操作者输.

下面是 $a = 2 \sim 8, \left\lfloor \frac{b}{a} \right\rfloor = 0 \sim 15$ 的 SG 函数表：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a = 2$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0
$a = 3$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$a = 4$	0	1	0	1	2	0	1	0	1	2	0	1	0	1	2	0
$a = 5$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$a = 6$	0	1	0	1	0	1	2	0	1	0	1	0	1	2	0	1
$a = 7$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$a = 8$	0	1	0	1	0	1	0	1	2	0	1	0	1	0	1	0

2.2 SG 函数

因此问题转化为，有一堆石子共 $\left\lfloor \frac{b}{a} \right\rfloor$ 个，双方每次可以取 $1, a, a^2, \ldots$ 个，无法操作者输.

下面是 $a = 2 \sim 8, \left\lfloor \frac{b}{a} \right\rfloor = 0 \sim 15$ 的 SG 函数表：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a = 2$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0
$a = 3$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$a = 4$	0	1	0	1	2	0	1	0	1	2	0	1	0	1	2	0
$a = 5$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$a = 6$	0	1	0	1	0	1	2	0	1	0	1	0	1	2	0	1
$a = 7$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$a = 8$	0	1	0	1	0	1	0	1	2	0	1	0	1	0	1	0

结论：先手必胜当且仅当 $(n+1) \bmod (a+1)$ 为偶数.

2.2 SG 函数

AGC016F. Games on DAG

给定一个 n 个点 m 条边的 DAG, 保证每条边 $u \to v$ 都有 $u < v$. 现在 $1, 2$ 两个点各有一个石头, 每次你可以沿着 DAG 上的边移动其中一个石头, 两个石头可以重合, 不能移动则输. 现在问: 所有 2^m 种生成子图中有多少个是先手必胜的.

$2 \le n \le 15, 1 \le m \le \binom{n}{2}$.

2.2 SG 函数

AGC016F. Games on DAG

给定一个 n 个点 m 条边的 DAG, 保证每条边 $u \to v$ 都有 $u < v$. 现在 $1, 2$ 两个点各有一个石头, 每次你可以沿着 DAG 上的边移动其中一个石头, 两个石头可以重合, 不能移动则输. 现在问: 所有 2^m 种生成子图中有多少个是先手必胜的.

$2 \le n \le 15, 1 \le m \le \binom{n}{2}$.

相当于问有多少张图上 1 号点和 2 号点的 SG 函数值相同. 考虑 SG 的计算过程是一个 mex, 所以一个点 SG 值为 i 当且仅当它向 $0 \sim i-1$ 都有边, 且没有到 i 的边. 因此可以设计如下的 dp 状态: f_S 表示 SG $\le$ 某个值的所有点的集合恰为 S. 转移枚举下一个 SG 值的集合, 判断是否满足条件. 复杂度 $\tilde{\mathcal{O}}(3^n)$.

2.3 分析博弈过程

这类问题没有什么特定的解法, 基本上需要选手在场上做一些精巧的观察或者扎实的推理. 一些可能的着手点包括

- 从最简单的例子出发, 寻找必胜/必败的方法
- 尝试多推理几步, 看能不能找到规律
- 找局面中的不变量. 比如在 Nim 博弈中我们的观察就是如果当前局面异或和为 0, 则无论后手怎么操作, 先手都可以让异或和重新为 0.

下面会是一些杂题.

2.3 分析博弈过程

AGC014D. Black and White Tree

现在有一棵树. 双人博弈, Alice 先手, 交替进行: Alice 将树上一个未染色的点染成白色, Bob 将树上一个未染色的点染成黑色. 所有点都染色后, 如果有一个白点只和白点相邻, 则 Alice 获胜, 否则 Bob 获胜. 问最后谁会获胜.

$n \leq 10^5$.

2.3 分析博弈过程

AGC014D. Black and White Tree

现在有一棵树. 双人博弈, Alice 先手, 交替进行: Alice 将树上一个未染色的点染成白色, Bob 将树上一个未染色的点染成黑色. 所有点都染色后, 如果有一个白点只和白点相邻, 则 Alice 获胜, 否则 Bob 获胜. 问最后谁会获胜.

$n \leq 10^5$.

首先注意到 Alice 不会染叶子, 否则 Bob 只要把这个叶子的父亲染黑就寄了; 相反, 如果两个叶子共用一个父亲, 那么 Alice 将这个父亲染了就赢.

2.3 分析博弈过程

因此考虑这样的一个博弈顺序: Alice 将某个叶子的父亲染白, Bob 将对应的叶子染黑, 反复过程, 会发现获得了原树的一个匹配. 因此:

- 如果原树有完美匹配, 那么每次 Alice 染一个点, Bob 只需要染匹配点, 这样 Alice 必败.
- 否则, 按照我们之前的方案进行, 一定某一次有两个叶子共用父亲, 这样 Alice 必胜.

于是检查原树有没有完美匹配即可.

2.3 分析博弈过程

CodeChef DESTRUCT

有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，Alice 先手，交替进行. 当 Alice 操作时，由 Bob 选取一个非空的石子堆，然后 Alice 从中拿走任意多个石子. 同理，当 Bob 操作时，由 Alice 选取一个非空石子堆，然后 Bob 从中拿走任意多个. 如果轮到某人操作时没有石子了那么这个人输. 问谁赢.

多组数据，$T \leq 200, n \leq 1000, a_i \leq 10^9$.

2.3 分析博弈过程

CodeChef DESTRUCT

有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，Alice 先手，交替进行. 当 Alice 操作时，由 Bob 选取一个非空的石子堆，然后 Alice 从中拿走任意多个石子. 同理，当 Bob 操作时，由 Alice 选取一个非空石子堆，然后 Bob 从中拿走任意多个. 如果轮到某人操作时没有石子了那么这个人输. 问谁赢.

多组数据，$T \leq 200, n \leq 1000, a_i \leq 10^9$.

观察 1: 相邻两次操作可以看作是，拿走若干个石子，然后指定下一个人取的石子堆.

2.3 分析博弈过程

CodeChef DESTRUCT

有 n 堆石子，第 i 堆有 a_i 个. 双人博弈，Alice 先手，交替进行. 当 Alice 操作时，由 Bob 选取一个非空的石子堆，然后 Alice 从中拿走任意多个石子. 同理，当 Bob 操作时，由 Alice 选取一个非空石子堆，然后 Bob 从中拿走任意多个. 如果轮到某人操作时没有石子了那么这个人输. 问谁赢.

多组数据，$T \leq 200, n \leq 1000, a_i \leq 10^9$.

观察 1: 相邻两次操作可以看作是，拿走若干个石子，然后指定下一个人取的石子堆.

观察 2: 如果当前这一堆的大小 > 1，那么先手可以选择直接把这一堆取完或者把这一堆取到只剩一个并让后手取，从而是必胜的.

因此可以枚举第一次 Bob 指定的堆，然后根据有没有非 1 的堆以及 1 的个数的奇偶性判断一下即可.

2.3 分析博弈过程

AGC026F. Manju Game

有 n 个盒子摆成一排，第 i 个盒子的权值是 a_i. 两人轮流操作，每次操作的方法如下：

- 设上一个人选择的盒子是 i. 如果 i 存在，并且 $i-1, i+1$ 中至少有一个还没被选择过的盒子，那么就在 $i-1, i+1$ 中选一个未被选过的盒子，取走其中的权值.
- 否则，可以任选一个未被选过的盒子，取走其中的权值.
- 如果每个盒子都被选过了则结束游戏.

两人希望最大化自己拿到的权值. 求两人最终拿到的权值.

$n \leq 3 \times 10^5$.

2.3 分析博弈过程

首先注意到先手可以选边上的球来获得一个权值. 称这个策略为基本策略.

- 如果 n 是偶数, 发现不管先手选在哪里, 后手可以选一个合适的方向使得往这个方向选完之后是先手结束, 然后后手可以在剩下的球中选择一个基本策略使得先手的收益不超过一开始就是用基本策略得到的收益. 因此最佳策略就是基本策略.
- 如果 n 是奇数, 首先发现先手不会选偶数位置的球, 否则类似可知收益不超过基本策略. 如果选的是奇数位置, 后手选择一边拿完之后又回到先手, 然后重复这个操作, 直到某一次先手用基本策略选完.

因此最终的答案即所有偶数位置的和, 加上中间某个区间中的奇数权值减偶数权值. 我们二分答案, 这样相当于有一些区间是赢的, 另外一些区间是输的. 最终递推可知, 这个情况合法当且仅当赢的区间可以覆盖整个区间. 这是可以简单 $\mathcal{O}(n)$ 判断的, 于是总复杂度 $\mathcal{O}(n \log n)$.

Outline

3.1 概述

基本没什么特殊的做法.

可能有用的技巧:

- 手玩较小的情况找规律.
- 通过样例找规律.
- 抓住题目中的特殊性质寻找答案.

3.1 概述

AGC052A. Long Common Subsequence

给定三个长为 $2n$ 的 01 串 s_1, s_2, s_3，其中每个都恰有 n 个 0 和 n 个 1. 你需要找到一个长为 $2n+1$ 的 01 串使得它同时是 s_1^2, s_2^2, s_3^2 的子序列.

$1 \le n \le 10^5$.

3.1 概述

AGC052A. Long Common Subsequence

给定三个长为 $2n$ 的 01 串 s_1, s_2, s_3, 其中每个都恰有 n 个 0 和 n 个 1. 你需要找到一个长为 $2n+1$ 的 01 串使得它同时是 s_1^2, s_2^2, s_3^2 的子序列.

$1 \le n \le 10^5$.

只需要输出 n 个 0, 然后 n 个 1, 最后 1 个 0 即可. 因为每个 s_i^2 的第 n 个 0 和第 $2n$ 个 0 之间都恰有 n 个 1.

3.2 分析可行条件

很多构造问题中，很容易分析出哪些条件是不存在构造的，也即可以得到存在构造方案的一个必要条件. 从这一必要条件入手可以做一点文章.

3.2 分析可行条件

CF804E. The same permutation

给定正整数 n, 考虑长为 n 的单位排列 $(1, 2, ..., n)$. 你需要构造一种所有 $\binom{n}{2}$ 个对换的排列顺序, 使得依次执行这些对换之后可以重新得到单位排列, 或者输出无解.

$1 \leq n \leq 1000$.

3.2 分析可行条件

CF804E. The same permutation

给定正整数 n, 考虑长为 n 的单位排列 $(1, 2, ..., n)$. 你需要构造一种所有 $\binom{n}{2}$ 个对换的排列顺序, 使得依次执行这些对换之后可以重新得到单位排列, 或者输出无解.

$1 \le n \le 1000$.

注意到每次对换会导致逆序对奇偶性改变, 于是存在构造至少需要 $\binom{n}{2}$ 为偶数, 也即 $n = 4k$ 或 $n = 4k + 1$.

- 当 $n = 4k$ 时, 我们将所有元素四个一组分组, 然后人类智慧 / 搜索一下可以得到每组组内的对换的方案和两组组间的对换的方案, 于是可以进行构造.
- 当 $n = 4k + 1$ 时, 在 $1, 2, ..., n - 1$ 的基础上我们还需要加入 n.

3.2 分析可行条件

注意到

$$(i,n)(i,j)(j,n)=(i,j),$$

于是找到所有 $(2i-1,2i)$ 的位置，让它变成上面的构造即可.

如果你想知道前面的 $n=4k$ 的构造:

- 组内: $(1,2)(3,4)(2,3)(1,4)(1,3)(2,4)$.
- 组间:

$$(1,a)(2,b)(3,c)(4,d)(1,b)(2,c)(3,d)(4,a)$$
$$(1,d)(2,a)(3,b)(4,c)(1,c)(2,d)(3,a)(4,b)$$

3.2 分析可行条件

AGC046E. Permutation Cover

给定正整数 n 和一个数列 $a_1, ..., a_n$. 你需要构造一个数列, 值域在 $[1, n]$ 之间, 且 i 恰出现 a_i 次. 且对数列中的每个元素, 都存在一个长为 n 的子串包含它, 并且这个子串是 $1, ..., n$ 的一个排列. 你需要输出满足条件的字典序最小的排列, 或输出无解.

$1 \le n \le 1000, \sum a_i \le 1000.$

3.2 分析可行条件

AGC046E. Permutation Cover

给定正整数 n 和一个数列 $a_1, \ldots, a_n$. 你需要构造一个数列, 值域在 $[1, n]$ 之间, 且 i 恰出现 a_i 次. 且对数列中的每个元素, 都存在一个长为 n 的子串包含它, 并且这个子串是 $1, \ldots, n$ 的一个排列. 你需要输出满足条件的字典序最小的排列, 或输出无解.

$1 \le n \le 1000, \sum a_i \le 1000.$

相当于最终序列中, 所有是 $1, \ldots, n$ 的排列的子串可以覆盖整个序列. 注意到有解的时候 a_i 的极差不能太大. 具体地, 一个贪心方案是形如 A, B, A, A, B, A, 其中 A 是较大的 a_i, B 是较小的 a_i, 这样相当于可以复用较小的 a_i. 于是可以猜测合法当且仅当

$$2 \min a_i \ge \max a_i.$$

记 $\arg\min a_i = x, \arg\max a_i = y$, 则我们按照 y 划分最终的序列, 则每个 y 两侧的两段至少有一个 x, 从而确实需要 $2a_x \ge a_y$.

3.2 分析可行条件

反过来, 当满足条件时, 可以按照我们之前的构造, 每一个 A, B, A 单元中可以选择每个 a_i 减少 1 或者减少 2, 于是一定有解.

考虑如何让字典序最小. 在已经构造好的序列基础上, 每次可以枚举新加入多少个元素构成一个排列, 则首先可以唯一确定是哪些元素,

- 如果操作后 $2\min a_i \geq \max a_i$ 则直接从小到大放新加的元素即可.
- 如果操作后 $2\min a_i \leq \max a_i + 2$ 则一定不合法.
- 如果操作后 $2\min a_i = \max a_i + 1$, 此时需要考虑新加入的元素的摆放顺序. 由于 $2\min a_i = \max a_i + 1$ 时序列是无解的, 于是现在的唯一一种可能就是我们下一个排列中复用了当前的 $a_i = \min a_i$ 的位置并新增了 $a_i = \max a_i$ 的部分. 于是简单判断即可.

总复杂度 $\mathcal{O}\left(n^2 \sum a_i\right)$.

3.2 分析可行条件

CF1053E. Euler tour

一棵 n 个点的树的欧拉序定义为，从根节点出发按某个顺序 DFS，记录经过的所有结点，得到的长为 $2n-1$ 的结点序列.

现在给定一个欧拉序，但是有部分位置缺失. 你需要还原一个可能的欧拉序，或者输出无解.

$1 \le n \le 5 \times 10^5$.

3.2 分析可行条件

CF1053E. Euler tour

一棵 n 个点的树的欧拉序定义为, 从根节点出发按某个顺序 DFS, 记录经过的所有结点, 得到的长为 $2n-1$ 的结点序列.

现在给定一个欧拉序, 但是有部分位置缺失. 你需要还原一个可能的欧拉序, 或者输出无解.

$1 \le n \le 5 \times 10^5$.

考虑一下存在解的必要条件:

- 如果序列首尾都存在, 那它们应当一样
- 如果 $a_l = a_r$, 那么它们之间是一棵子树, 从而 $r-l$ 为偶数且 $[l,r]$ 只有至多 $\frac{r-l}{2}+1$ 种权值.
- 所有满足 $a_l = a_r$ 的区间可以包含, 不能相交.

3.2 分析可行条件

事实上这就是有解的充要条件. 考虑一下如何构造.

- 如果存在 $a_l = a_r$, 那么 $[l, r]$ 对应一棵子树, 只需要递归构造之后将 $[l, r]$ 缩成一个 a_l 即可. 于是可以假设序列中没有重复的元素.
- 如果区间中出现了少于 $\frac{r-l}{2} + 1$ 种元素, 则可以用未出现的元素填到恰 $\frac{r-l}{2} + 1$ 种.
- 如果存在形如 xy空 或者 空yx 的子串, 则可以将其填为 xyz, 然后缩成一个点 x, 于是可以假设序列中没有相邻的非空元素.
- 此时序列一定形如 $a_l, x,$ 空$, y, \ldots,$ 空$, z, a_l$ 此时可以将所有空位置全部填为 a_l.

容易验证上面的构造是合法的. 复杂度 $\mathcal{O}(n)$.

3.3 构造组合操作

可以尝试用给定的基本操作构造一些具有比较好性质的组合操作, 然后用这些组合操作来构造最终方案.

3.3 构造组合操作

AGC006E. Rotate 3x3

有一个 $3 \times n$ 的矩阵，(i, j) 位置是 $i + 3(j - 1)$. 每次操作你可以把一个 3×3 的矩阵旋转 $180°$，你需要判断是否存在一个操作序列，得到给定的矩阵.

$5 \leq n \leq 10^5$.

3.3 构造组合操作

AGC006E. Rotate 3x3

有一个 $3 \times n$ 的矩阵，(i, j) 位置是 $i + 3(j - 1)$. 每次操作你可以把一个 3×3 的矩阵旋转 $180°$，你需要判断是否存在一个操作序列，得到给定的矩阵.

$5 \leq n \leq 10^5$.

首先操作是不改变每一列的构成和其所处位置的奇偶性的，于是可以先把不合法的情况判掉. 然后可以发现，每次操作形如 $(x, y, z) \to (-z, -y, -x)$. 那么首先可以通过交换来让每一列归位，然后就是要处理奇偶性.

3.3 构造组合操作

我们构造一个组合操作，效果是让两个距离为 2 的列同时取负：

$$(1,2,3,4,5) \to (-3,-2,-1,4,5) \to (-3,-2,-5,-4,1)$$
$$\to (5,2,3,-4,1) \to (5,2,-1,4,3) \to (1,-2,-5,4,3) \to (1,-2,3,-4,5).$$

类似可以构造同时让 $1,3$ 或 $3,5$ 取负的方案. 反复做这个操作我们可以在奇数列和偶数列分别恰有偶数个负数时达成目标.

注意到 奇数列中负数个数 与 偶数列的逆序对个数 之和的奇偶性是定值，于是我们不可能在保持偶数列位置不变的同时使奇数列中恰一个元素取负，从而上面的条件就是充要条件. 于是只要统计奇偶性即可.

3.3 构造组合操作

CF715D. Create a Maze

有一个 $n \times m$ 的迷宫，相邻两个位置之间可能会有墙阻隔导致无法通过. 定义一个迷宫的难度为从 $(1,1)$ 走到 (n,m)，每次只能向下或向右走的方案数，你需要构造一个方案数恰为 T，且 $n,m \leq 50$，墙总数不超过 300 的迷宫.

$T \leq 10^{18}$.

3.3 构造组合操作

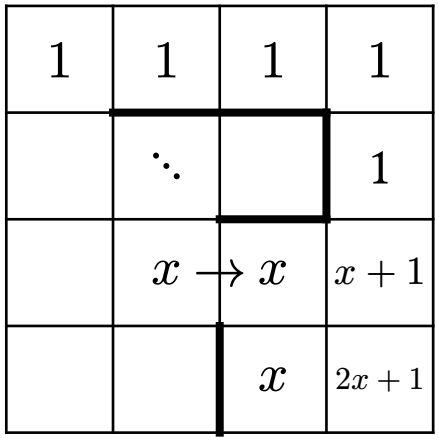
CF715D. Create a Maze

有一个 $n \times m$ 的迷宫，相邻两个位置之间可能会有墙阻隔导致无法通过. 定义一个迷宫的难度为从 $(1,1)$ 走到 (n,m)，每次只能向下或向右走的方案数，你需要构造一个方案数恰为 T，且 $n,m \leq 50$，墙总数不超过 300 的迷宫.

$T \leq 10^{18}$.

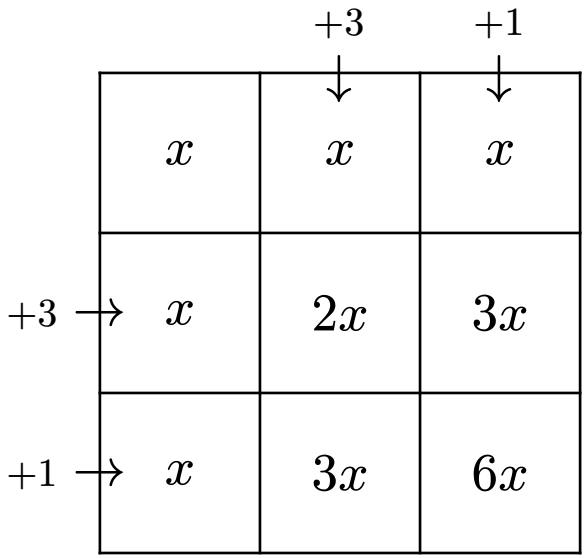
注意到一个 2×2 的网格，如果走到左上的方案数为 x，那么走到右下的方案数就是 $2x$. 于是如果可以构造 $+1$ 操作就可以直接用二进制构造出方案了.

考虑我们可以在第一行维护一个方案数 1，然后在合适的时候向下引入 1，从而达到 $+1$ 的目的



3.3 构造组合操作

实际写一下的话会发现这个做法差一点常数，所以还需要进一步改进成 3×3 的网格，每次乘以 6，具体如下图



.

3.4 找中间状态

对一类构造操作方案的题目，如果观察到操作是可逆的，则可以想办法找一个比较好的中间状态，然后构造初始状态和结束状态到这个中间状态的操作序列.

3.4 找中间状态

CF1276E. Four Stones

有四块石子，初始在 $a_1, \ldots, a_4$ 处，你需要将它们移动到 $b_1, \ldots, b_4$ 处（可以相差一个排列，也即只需要 $a'_{p_i} = b_i$，对某个排列 p）。每次操作可以选定 $x \neq y$，然后将 x 沿着 y 对称一次。也即令 $(a_x, a_y) \mapsto (2a_y - a_x, a_y)$。给出一个不超过 1000 步的移动序列，或者输出无解。

$|a_i| \leq 10^{18}$。

3.4 找中间状态

CF1276E. Four Stones

有四块石子, 初始在 $a_1, \ldots, a_4$ 处, 你需要将它们移动到 $b_1, \ldots, b_4$ 处 (可以相差一个排列, 也即只需要 $a'_{p_i} = b_i$, 对某个排列 p). 每次操作可以选定 $x \neq y$, 然后将 x 沿着 y 对称一次. 也即令 $(a_x, a_y) \mapsto (2a_y - a_x, a_y)$. 给出一个不超过 1000 步的移动序列, 或者输出无解.

$|a_i| \leq 10^{18}$.

特判 a 全部相同或 b 全部相同的情况.

首先注意到跳跃操作不改变 gcd, 于是至少需要前后的序列的 gcd 相同. 此外, 注意到移动的时候不改变 a_x 的奇偶性, 于是 a 当中的奇数个数和 b 中的应当相同.

实际上这也是充要条件. 下面我们假设 $\gcd = 1$ 并构造一个合法的移动序列.

3.4 找中间状态

注意到构造是可逆的，于是我们可以尝试找中间状态. 我们考虑这样的一个中间状态：所有 a_i 的极差 ≤ 1，也即存在 x 使得 $a_i \in [x, x+1]$. 于是我们只需要构造

- 任意状态到中间状态的方案：主要考虑如何**快速**减小 $\Delta := \max a_i - \min a_i$.
  - 设 a_i 递增，则如果 $\left[a_1 + \frac{\Delta}{4}, a_4 - \frac{\Delta}{4}\right]$ 中有 a_2 或 a_3，那么翻转之后 $\Delta' \leq \frac{3}{4}\Delta$.
  - 否则，注意到将某个 a_i 先沿着 a_j 翻转，再沿着 a_k 翻转可以让其增加 $2\left(a_k - a_j\right)$. 考虑将 $a_2 - a_1, a_4 - a_3$ 中较小的一个每次增加两倍的较大的一个，这样类似 Fibonacci 数列可知只需要 $\mathcal{O}(\log)$ 次就可以增大到 $\frac{\Delta}{4}$，然后执行上一步即可.

  于是只需要 $\mathcal{O}\left(\log^2\right)$ 次操作就可以完成这一步.

3.4 找中间状态

- 中间状态之间的转移. 注意到我们总可以通过将所有 a_i 沿着 a_4 对称两侧来使整体平移 2Δ. 那我们可以将此时的 a_2 沿着 a_4 对称, a_3 沿着 a_1 对称, 让 $\Delta' \in [2\Delta, 3\Delta]$, 然后重复这个操作直到 Δ 充分大, 然后向 y 的方向平移, 然后逐步撤销膨胀操作并继续缩短. 因此这部分是 $\mathcal{O}(\log)$ 的.

因此最终只需要 $\mathcal{O}(\log^2)$ 次操作就可以完成. 对一类构造操作方案的题目, 如果观察到操作是可逆的, 则可以想办法找一个比较好的中间状态, 然后构造初始状态和结束状态到这个中间状态的操作序列.

Outline

4. 题单

- 贪心: AGC010C,「ROI 2018 Day 2」无进位加法,「LibreOJ Round #9」Menci 的序列,「JSOI2007」建筑抢修,「APIO / CTSC2007」数据备份
- 博弈: AGC002E, AGC005E, AGC010D, AGC010E, AGC029D,「ZJOI 2009」取石子游戏,「SDOI 2011」黑白棋,「SNOI 2020」取石子
- 构造: CF341E, AGC030C, CF1495C, AGC029C, CF1227G, AGC020D,「JOISC 2020 Day1」汉堡肉, CF538G

后面还有几道找不到题号的, 我直接把题面和题解放上来了

4. 题单

heike

现在有 n 个员工，在 T 天内进行工作. 第 i 个员工的可工作时间为 $[l_i, r_i]$，如果工作可以产生 v_i 的收益. 要求每一天至多有一个人工作，且每个人在 T 天内至多工作一次，问最大收益.

m 组询问，每次询问去掉一个员工或加入一个新员工，查询上面的问题的答案；询问之间独立.

$1 \le T \le 300, 1 \le n, m \le 10^6$.

4. 题单

heike

现在有 n 个员工, 在 T 天内进行工作. 第 i 个员工的可工作时间为 $[l_i, r_i]$, 如果工作可以产生 v_i 的收益. 要求每一天至多有一个人工作, 且每个人在 T 天内至多工作一次, 问最大收益.

m 组询问, 每次询问去掉一个员工或加入一个新员工, 查询上面的问题的答案; 询问之间独立.

$1 \le T \le 300, 1 \le n, m \le 10^6$.

注意到问题是一个二分图费用流, 是显然满足拟阵性质的, 于是可以将所有员工按权值排序, 然后能加就加.

4. 题单

考虑如何判断能不能加. 设 S 是所有员工的一个子集, 考虑一下什么时候 S 中的员工都可以工作. 注意到这相当于问二分图最大匹配是不是满的, 于是根据 Hall 定理, S 合法当且仅当任意区间有交的人数都不超过区间长度.

因此原问题可以直接 $\mathcal{O}(T^3+n)$ 维护哪些区间是满的并计算. 考虑一下多组询问怎么做. 根据拟阵的性质, 当新加入一个人的时候, 一定是顶替掉原来的一个权值最小的人; 删除一个人时, 一定是从原来没有加入的人当中选一个权值最大的加入. 这都是可以简单预处理的. 总复杂度 $\mathcal{O}(n+m+T^3)$.

4. 题单

特趣米诺

考虑俄罗斯方块的 7 种方块, 即 ZSJLTOI. 给定 n, m, 你需要用这些方块恰填满一个 $n \times m$ 的矩形, 且每种方块的个数相同.

$1 \leq n \times m \leq 10^6$.

4. 题单

特趣米诺

考虑俄罗斯方块的 7 种方块, 即 ZSJLTOI. 给定 n, m, 你需要用这些方块恰填满一个 $n \times m$ 的矩形, 且每种方块的个数相同.

$1 \leq n \times m \leq 10^6$.

首先 $n \times m$ 必须是 28 的倍数. 进一步, 由于黑白染色 T 字形会让两种颜色的差为 2, 所以 $n \times m$ 必须是 56 的倍数. 然后显然 $n, m \leq 2$ 也做不了.

我们断言, 其它情况都能做.

构造时我们可以用人类智慧或打表构造 $7 \times 8, 14 \times 4, 28 \times 6, 56 \times 3, 56 \times 5$ 的 5 种, 然后容易发现其他情况都可以由它们组合得到. 具体细节留给选手思考.