

# CF2225D Exceptional Segments

## 题意

给定两个整数  $n$  和  $x$ 。在序列  $[1, 2, 3, \dots, n]$  中，你需要求出满足以下条件的子区间  $[l, r]$  的个数：

1. 区间必须包含  $x$ ，即  $1 \leq l \leq x \leq r \leq n$ 。
2. 区间内所有元素的异或和为 0，即  $l \oplus (l+1) \oplus \dots \oplus r = 0$ 。

答案可能会很大，需要对 998244353 取模。

数据范围：

$$1 \leq x \leq n \leq 10^{18}, t \leq 2 \cdot 10^5。$$

## 解析

### 前置知识

处理区间异或和为 0 的问题，最核心的技巧是**异或前缀和**。

设  $S(k) = 0 \oplus 1 \oplus 2 \oplus \dots \oplus k$ 。

那么区间  $[l, r]$  的异或和为 0，等价于：

$$S(r) \oplus S(l-1) = 0 \implies S(l-1) = S(r)$$

另外，

### sub1: $n \leq 1000$ (暴力枚举 $O(n^2)$ )

直接利用前缀和数组 `s`，双重循环枚举  $l \in [1, x]$  和  $r \in [x, n]$ ，如果  $S(l-1) == S(r)$  则答案加一。

### sub2: $n \leq 10^6$ (哈希/桶优化 $O(n)$ )

由于  $l$  和  $r$  是独立的，可以先遍历  $l-1 \in [0, x-1]$ ，用一个数组或哈希表记录  $S(l-1)$  各个值出现的次数。然后再遍历  $r \in [x, n]$ ，直接去哈希表中累加  $S(r)$  出现的次数即可。

### sub3: $n \leq 10^{18}$ 的 $O(1)$ 数学做法

$n$  高达  $10^{18}$ ，任何遍历都会 TLE，需要利用  $S(k)$  的周期规律进行  $O(1)$  计算。

一筹莫展，尝试打表。

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int main()
4  {
5      int now = 0;
6      for (int i = 1; i <= 100; i++)
7      {
8          now = now ^ i;
9          cout << now << ", ";
10     }
11     return 0;
12 }

```

1 | 1, 3, 0, 4, 1, 7, 0, 8, 1, 11, 0, 12, 1, 15, 0, 16, 1, 19, 0, 20, ...

连续自然数的异或前缀和  $S(k)$  具有以 4 为周期的极强规律性：

- 当  $k \equiv 0 \pmod{4}$  时,  $S(k) = k$
- 当  $k \equiv 1 \pmod{4}$  时,  $S(k) = 1$
- 当  $k \equiv 2 \pmod{4}$  时,  $S(k) = k + 1$
- 当  $k \equiv 3 \pmod{4}$  时,  $S(k) = 0$

结论:  $S(l-1)$  与  $S(r)$  相等, 当且仅当它们同时等于 0, 或者同时等于 1

因此, 问题被极大地简化了:

1. 统计  $[0, x-1]$  中  $S(i) = 0$  的个数, 乘上  $[x, n]$  中  $S(i) = 0$  的个数。
2. 统计  $[0, x-1]$  中  $S(i) = 1$  的个数, 乘上  $[x, n]$  中  $S(i) = 1$  的个数。
3. 两者相加取模即可。

如何  $O(1)$  统计  $[0, M]$  中  $S(i) = 0$  或 1 的个数?

- $S(i) = 0$ : 发生于  $i = 0$  以及  $i \equiv 3 \pmod{4}$ 。个数为  $1 + (M + 1) / 4$ 。
- $S(i) = 1$ : 发生于  $i \equiv 1 \pmod{4}$ 。个数为  $(M + 3) / 4$ 。
- 利用前缀和思想,  $[x, n]$  中的个数等于  $[0, n]$  的个数减去  $[0, x-1]$  的个数。

## CF2219A Grid L

### 题意

Roger 有  $p$  个单位长度（长度为 1）的线段, 以及  $q$  个 L 型拼图（每个 L 型拼图由 2 个单位长度的线段成直角连接而成）。

他希望恰好用完这  $p$  个线段和  $q$  个 L 型拼图, 拼出一个完整的  $n \times m$  的网格。

如果可以拼出, 输出任意满足条件的  $n$  和  $m$ ; 如果无论如何都拼不出, 则输出  $-1$ 。

数据范围:

$$1 \leq t \leq 100, 1 \leq p, q \leq 10^8$$

# 解析

## 1. 网格的总线段数约束

$n \times m$  的网格，线段数量：

- 水平线段：  $n + 1$  行，每行包含  $m$  条水平线段，共计  $H = m(n + 1)$  条。
- 垂直线段：  $m + 1$  列，每列包含  $n$  条垂直线段，共计  $V = n(m + 1)$  条。

网格的总线段数为：  $S = H + V = m(n + 1) + n(m + 1) = 2nm + n + m$

要求我们恰好用完  $p$  个单位线段和  $q$  个 L 型拼图。因为每个 L 型拼图包含 2 条单位线段，所以拼图的总线段数是  $p + 2q$ 。

两者必须相等，因此我们得到核心方程：  $2nm + n + m = p + 2q$

等式两边同时乘以 2，并加上 1，可以进行因式分解：

$$4nm + 2n + 2m + 1 = 2p + 4q + 1$$

$$(2n + 1)(2m + 1) = 2p + 4q + 1$$

问题转化为了一个整数因数分解问题：我们需要找到  $2p + 4q + 1$  的两个奇数因子  $A$  和  $B$ （其中  $A \geq 3, B \geq 3$ ），那么就可以反推出  $n = (A - 1)/2$  和  $m = (B - 1)/2$ 。

## 2. L 型拼图的几何约束

我们需要考虑 L 型拼图的特殊性。题目规定：**L 型拼图由 2 个单位长度的线段成直角连接而成。**

这意味着，**每一个 L 型拼图，必定消耗恰好 1 条水平线段和 1 条垂直线段。**

既然我们要放下  $q$  个 L 型拼图，那么我们就至少需要  $q$  条水平线段和  $q$  条垂直线段。

结合前面的推导，网格中只有  $H$  条水平线段和  $V$  条垂直线段，因此必须满足：

$$q \leq H \text{ 且 } q \leq V$$

$$\text{即： } q \leq \min(m(n + 1), n(m + 1))$$

只要满足这个条件，我们总是可以在网格的角上凑出足够数量的 L 型，剩余的位置全铺满长度为 1 的单根线段即可。

# R57E My Name

## 题意

$T$  组询问，每次给一个整数  $n$ 。

求正整数三元组  $x, y, z$  的数量：

- 取值范围：  $1 \leq x, y, z \leq n$ 。
- 特殊性质：  $x^2 - y^2 = z^3$ 。

数据范围：

- 20pts:  $T \leq 10, n \leq 1000$
- 40pts:  $T \leq 10, n \leq 10^5$

- 100pts:  $T \leq 10^5, n \leq 10^6$

## 解析

### sub1: $T \leq 10, n \leq 1000$

由于  $n$  非常小，最直观的想法是直接枚举其中的两个变量，算出第三个变量来验证。

为了避免开根号带来的精度问题，我们可以枚举  $y$  和  $z$ （范围均为  $[1, n]$ ），然后计算  $val = y^2 + z^3$ 。如果  $val$  是一个完全平方数，且其平方根  $x \leq n$ ，那么我们就找到了一个合法的三元组。

**时间复杂度：** 单次询问  $O(n^2)$ 。对于  $T = 10, n = 1000$ ，总计算量在  $10^7$  级别，可以轻松通过。

### sub2: $T \leq 10, n \leq 10^5$

当  $n$  为  $10^5$ ， $O(n^2)$  显然超时，考虑优化。

将原式变形：

$$x^2 - y^2 = z^3 \implies (x - y) \times (x + y) = z^3$$

令  $A = x - y$ ， $B = x + y$ ，可知  $A < B$ 。

则：

- $x = \frac{A+B}{2}$
- $y = \frac{B-A}{2}$

观察到性质： $A$  和  $B$  必须同奇偶。

同时，由于  $x^2 = y^2 + z^3 > z^3$ ，我们可以推导出  $x > z^{3/2}$ 。这意味着在任何合法的三元组中， $x$  始终是最大的元素。所以  $z < x^{2/3} \leq n^{2/3}$ 。

当  $n = 10^5$  时， $z$  的上限仅为  $100000^{2/3} \approx 2154$ 。

**做法：** 枚举  $z \in [1, n^{2/3}]$ ，然后暴力找出  $z^3$  的所有约数  $A$ （只需枚举到  $\sqrt{z^3}$ ），算出对应的  $B = \frac{z^3}{A}$ 。检查  $A, B$  是否同奇偶，以及计算出的  $x$  是否  $\leq n$ 。

**时间复杂度：** 单次询问约  $O(n^{2/3} \cdot \sqrt{z^3})$ ，大幅降低，但仍无法通过。

**做法：**

直接枚举  $z \in [1, n^{2/3}]$ ，对  $z$  进行质因数分解，利用 DFS 快速求出  $z^3$  的所有约数  $A$ 。然后算出  $B = z^3 / A$ ，验证奇偶性并统计答案。

**时间复杂度：** 单次询问  $O(n^{2/3} \cdot \text{约数个数})$ 。足以通过 Subtask 2。

### sub3: $n \leq 1000000, T \leq 100000$

此时询问次数  $T$  高达 100000，任何“每次询问单独计算（在线）”的算法都会超时。我们必须做到  $O(1)$  响应查询。

继续利用 Subtask 2 的核心结论： $x$  永远是三元组中的最大值。

那么“三元组全部  $\leq n$ ”的条件等价于“ $x \leq n$ ”。只要  $x$  不越界，整个三元组必然合法。

基于全局最大可能的数据范围  $n = 1000000$ ，我们可以算出  $z$  的全局绝对上限：

$$z < (1000000)^{2/3} = 10000$$

既然全局  $z$  最多只有 10000 种可能，我们完全可以在程序一开始，把所有合法的  $(x, y, z)$  组合全部算出来！

# CF2215A Interval Mod

## 题意

给定一个长度为  $n$  的数组  $a$ ，以及一个长度限制  $k$  和两个取模参数  $p$  与  $q$ （保证  $p < q$ ）。

你可以执行任意次操作：每次选择一个长度至少为  $k$  的区间，将区间内的所有元素对  $p$  或对  $q$  取模。

求在执行任意次操作后，整个数组元素总和的最小值。

数据范围：

- $1 \leq t \leq 10^4$
- $1 \leq k \leq n \leq 10^5, 1 \leq p < q \leq 10^9$
- $1 \leq a_i \leq 10^9$

## 解析

为了逐步引导出正解，我们可以设计以下部分分（Subtasks）：

- **Subtask 1 (20 pts):**  $k = 1$ 。区间长度没有限制，我们可以对每个元素独立进行操作。
- **Subtask 2 (20 pts):**  $k = n$ 。每次操作只能对整个数组全局进行，不能区分各个元素。
- **Subtask 3 (60 pts):** 无特殊限制（ $1 \leq k \leq n \leq 10^5$ ）。

### sub1: $k = 1$

当  $k = 1$  时，每个元素互不干扰。由于取模操作只会让数字变小或不变（ $x \bmod m \leq x$ ），且  $p < q$ ，因此对于任何一个数字，其最优的最终归宿只有两种：

- 只对  $p$  取模：最终值为  $A_i = a_i \bmod p$
- 先对  $q$  取模，再对  $p$  取模：最终值为  $B_i = (a_i \bmod q) \bmod p$ 。由于  $p < q$ ，对  $p$  取模后的数一定小于  $q$ ，所以再对  $q$  取模没有意义

做法：答案就是  $\sum \min(A_i, B_i)$ 。

### sub2: $k = n$

当  $k = n$  时，必须对整个数组同时操作。经过推导，全局最优的操作序列只能是：全员选状态 A（全体只对  $p$  取模），或者全员选状态 B（全体先对  $q$  取模，再对  $p$  取模）。

做法：答案为  $\min(\sum A_i, \sum B_i)$ 。

### sub3

当  $1 < k < n$  时，每个元素依然只有上述两种最优归宿（状态 A 或 状态 B）。因为我们可以所有操作的最后，对整个  $[1, n]$  区间执行一次全局对  $p$  取模（长度  $n \geq k$  总是合法的），这保证了每个元素的最终态要么是  $A_i$ ，要么是  $B_i$ 。

问题转化为：给每个元素分配状态 A 或状态 B，使得最终总和最小。但受到区间长度约束，哪些 A/B 的分配序列是合法的？

存在结论：一个由 A 和 B 组成的状态序列是合法的，当且仅当序列中【至少存在一段长度  $\geq k$  的连续相同状态】（即全 A 或全 B）。

因为：一个元素最终是状态 A 还是状态 B，完全由覆盖它的【第一次有效操作】决定，一旦决定，终生锁死。

- 如果某个元素第一次被框入的是 p 操作，它就永久锁死为状态 A。后续无论再怎么框它，都对它无效了。
- 如果第一次被框入的是 q 操作，它就变成了状态 B（只要我们保证最后全员进行一次 p 操作兜底）。

假设  $k = 3$ ，希望构造出一个复杂的序列：A B B A A A B A。

初始状态：\_ \_ \_ \_ \_ \_ \_ \_ （全都是原始数字）

第一步：基础（长度  $\geq k$  的 A）

在中间这三个位置执行 p 操作（状态 A）。

当前状态：\_ \_ \_ A A A \_ \_ （这三个 A 已经被永久锁死）

第二步：向左扩张

让左边挨着的一个元素变成 B。

我们可以选定一个长度为 3 的操作区间，覆盖 [未填, A, A]，对它们执行 q 操作（状态 B）。

因为后面两个 A 是锁死的，所以 q 操作对它们无效，只有最左边那个没填的元素变成了 B。

当前状态：\_ \_ B A A A \_ \_

结论 只要存在这样一个长度  $\geq k$  的“同化区”，由于该区域在最后一次被区间操作时自身合法，它就可以作为跳板，掩盖并向外扩张，从而构造出剩余的任意复杂状态分布。

实现步骤

1. 贪心地让每个元素独立选择最小的代价： $M = \sum \min(A_i, B_i)$ 。
2. 定义把元素  $i$  强行改为状态 A 的额外代价为  $DA_i = A_i - \min(A_i, B_i)$ 。
3. 定义把元素  $i$  强行改为状态 B 的额外代价为  $DB_i = B_i - \min(A_i, B_i)$ 。
4. 我们需要找到一个长度为  $k$  的区间，将其强行全部变为状态 A 或状态 B，使得产生的“额外代价”最小。
5. 前缀和优化：构建  $DA$  和  $DB$  的前缀和数组  $\text{suma}$  和  $\text{sumb}$ 。
6. 遍历所有长度为  $k$  的区间起点  $i \in [0, n - k]$ ，利用前缀和在  $O(1)$  时间内查询出该区间全部变成 A 或 B 的额外代价，取其中的最小值  $\text{minn}$ 。
7. 最终答案即为  $M + \text{minn}$ 。

# R58E 排序

## 题意

给定一个长度为偶数  $n$  的排列  $a$ 。你每次可以进行如下操作：

1. 将序列分为左半部分  $l(a_1 \dots a_{n/2})$  和右半部分  $r(a_{n/2+1} \dots a_n)$ 。
2. 构造一个长度为  $n$  的操作字符串，其中包含  $n/2$  个 'l' 和  $n/2$  个 'r'。
3. 按照字符串的指示，依次从  $l$  或  $r$  的队首弹出元素放入新数组  $b$ 。
4. 将  $a$  更新为  $b$ 。

目标是在  $2n$  次操作内，使  $a$  变为单调递增序列  $1, 2, \dots, n$ 。

数据范围：

- 20pts:  $a = [1, 3, 5, \dots, 2, 4, 6 \dots]$ ,  $n \leq 1000, k \leq 2n$ 。
- 100pts:  $n \leq 1000, k \leq 2n^{**}$ 。

## 解析

**特殊性质：**  $a = [1, 3, 5, \dots, 2, 4, 6 \dots]$ ,  $n \leq 1000, k \leq 2n$

显然  $L = [1, 3, 5, \dots]$ ,  $R = [2, 4, 6, \dots]$ ，只需要交叉合并就可以通过一次操作合并成有序序列。

**100pts:**  $n \leq 1000, k \leq 2n$

正难则反，把乱序变成有序很难控制每一次的走向，不妨反过来思考：**如何把“有序的数组”还原回“乱序的原数组”？**

逆向操作的本质是：面对当前数组，你写下一个包含 l 和 r 的字符串。标记为 l 的数字会被统一扔到左半区，标记为 r 的数字会被统一扔到右半区。

**关键联想：基数排序**

假设没有“l 和 r 必须数量相等”的限制，要把  $1 \dots n$  变成任意排列，只需要看目标位置的二进制：

- 第 0 位是 0 的扔左边，是 1 的扔右边；
- 第 1 位是 0 的扔左边，是 1 的扔右边；
- .....

经过  $\lceil \log_2 n \rceil$  次操作，每个数字就会精准地落到它该去的位置。

但是问题是限制 l 和 r 的数量相等，但是  $n$  是偶数，设二进制位数  $m = \lceil \log_2 n \rceil$ ，对于  $[1, \frac{n}{2}]$  中的每个数  $i$ ，我们把它映射成  $i - 1$ ，对于  $(\frac{n}{2}, n]$  中的每个数  $i$ ，我们把它映射成  $2^m - i + \frac{n}{2}$ ，容易发现  $i$  和  $n - i + 1$  的异或值恰好为  $2^m - 1$ ，等价于我们把这  $n$  个数配成了  $\frac{n}{2}$  对，从而保证了 l 和 r 的数量相等。

时间复杂度： $\mathcal{O}(n \log n)$ ，操作次数： $\lceil \log_2 n \rceil$  次

## R58F 回路

### 题意

给定一个包含  $n$  个城市的有向图，城市编号为 0 到  $n-1$ 。对于任意城市  $i$ ，存在两条出边（传送门）：

- **A 类边：** 指向  $2i \bmod n$
- **B 类边：** 指向  $(2i+1) \bmod n$

**目标：**判断图中是否存在一条从 0 出发、恰好经过所有城市一次并回到 0 的回路（即哈密顿回路）。若存在，输出任意一种合法的传送门选择序列。

数据范围：

- 40pts:  $n \leq 20$
- 100pts:  $n \leq 10^6$

## 解析

### 特判

先观察原图的一个硬性数学规律：**奇数 n 几乎必定无解。**

假设存在一条合法的哈密顿回路，回路中包含了 0 到 n-1 的所有顶点。我们对回路中“所有起点的编号之和”与“所有终点的编号之和”在模 n 意义下建立等式：

$$\sum_{x=0}^{n-1} x \equiv \sum_{x=0}^{n-1} (2x + c) \pmod{n}$$

这里的  $c \in \{0, 1\}$  代表传送门的类型。设起点和为 S，化简可得：

$$S \equiv 2S + \sum c \pmod{n}$$

$$\sum c \equiv -S \equiv -\frac{n(n-1)}{2} \pmod{n}$$

如果 n 是大于 1 的奇数，那么  $\frac{n-1}{2}$  是整数，这意味着  $\frac{n(n-1)}{2}$  必定是 n 的倍数。

因此  $\sum c \equiv 0 \pmod{n}$ 。

因为 c 只能取 0 或 1，所以回路中使用的 B 类传送门数量要么全是 0 个（全 A），要么全是 n 个（全 B）。

- **全走 A：**  $0 \rightarrow 0$  形成自环，无法遍历其他点。
- **全走 B：**  $n-1 \rightarrow 2n-1 \bmod n = n-1$  形成自环，同样断层。

**结论：**除了特例  $n = 1$  外，只要 n 为奇数，直接输出 NO。这一步剪枝对于后续所有做法都是必须的。

### sub1: $n \leq 20$

对于非常小的数据规模，可以直接使用暴力搜索。

- 开辟一个 `visited` 数组记录当前哪些城市已经被访问过。
- 从 0 号城市开始，每次递归分两个分支：尝试走 A 类边或 B 类边。
- 如果到达新城市已经被访问过，则回溯。
- 当递归深度达到 n 时，检查最后一步所在的城市是否能够通过 A 或 B 传送到 0。如果可以，说明找到了哈密顿回路。

**复杂度：**最坏时间复杂度为指数级，但在  $n \leq 20$  且奇数直接判否的情况下，加上 `visited` 剪枝，足以拿满 40 分。

### sub2: $n \leq 10^6$

在一般有向图中寻找哈密顿回路是一个著名的 NP-Hard 问题，数据规模达到  $1e6$ ，暴力搜索绝对会超时。观察题目给定的**特殊连边规则**。

观察偶数 n（设  $n = 2k$ ）的连边规律：



- 城市  $u$  的出边指向:  $2u \bmod n$  和  $(2u+1) \bmod n$
- 城市  $u+k$  的出边指向:  $2(u+k) \bmod n = 2u \bmod n$  和  $2(u+k)+1 \bmod n = (2u+1) \bmod n$

发现, 相差  $k$  的两个城市, 它们的去向完全一致! 暗示原图  $G$  实际上是另一个图  $G'$  的线图。(原图的“边”抽象成了新图的“点”。)

如何转化?

1. 构造一个只有  $k$  个顶点 (即  $n/2$  个顶点) 的新图  $G'$ , 顶点编号为  $0$  到  $k-1$ 。
2. 在  $G'$  中, 对于任意顶点  $u$ , 连出两条有向边:
  - 一条边指向  $(2u) \bmod k$ , 边编号为  $2u$ 。
  - 另一条边指向  $(2u+1) \bmod k$ , 边编号为  $2u+1$ 。
3. 在新图  $G'$  中, 有  $k$  个顶点和  $2k$  条边。每一条边的编号, 恰好完美对应了原图  $G$  中的  $n$  个城市编号

结论: 求原图  $G$  ( $n$  个点) 中遍历所有点恰好一次的哈密顿回路, 等价于求新图  $G'$  ( $k$  个点,  $n$  条边) 中遍历所有边恰好一次的欧拉回路

还原传送门类型: 遍历求得的路径序列  $path$ 。如果当前走到的城市 (即边编号) 指向的下一个城市编号是偶数, 说明用了 A 类门; 如果是奇数, 说明用了 B 类门。