NOIp 模拟 #1

试题讲评

XOR

- 给定 L，R，计算：

$$\sum_{i=L}^{R}\sum_{j=L}^{R}(i \oplus j)$$

- 保证 $L, R \leq 10^9$

$L = 0, R = 2^K - 1$ 的做法

- 枚举 i 之后，我们枚举 j，可以让 $i \oplus j$ 的值遍历 $L \sim R$

- 因此可以 O(1) 计算答案

标准解法

- 考虑到:

$$i \oplus j = \sum_{w=0}^{30} 2^w \cdot [i\text{和}j\text{在第 }w\text{ 位不同}]$$

- 考虑枚举每一位对答案的贡献

- 我们枚举每个二进制数位，数出 L~R 中有 x 个数在这一位为 0，y 个数为 1，那它对答案的贡献为：$2^{w+1}xy$

Cont’d

- 如何快速数出有多少个数在这一位为 1 呢?
- 可以转化为两个前缀相减 count(R, w) – count(L – 1, w)
- 计算 count(n, w) 时，我们枚举满足条件的 x 和 n 最高在哪一个二进制位不同即可

STONE

- 三堆石子上玩取石子游戏，每次取出其中若干堆，并从每一堆中取出相同数量的石子

- 不能操作的人失败，判断游戏结果。

- 保证石子数不超过 300

通用的朴素解法

- 记 $f(i,j,k)$ 表示面对三堆石子数分别为 (i, j, k) 的局面，先手是否必败。
- 那么，我们判定 (i,j,k) 是先手必胜，当且仅当它能转移到一个先手必败，也就是 $f(i,j,k)=1$ 的局面
- 我们可以从小到大“筛”出所有的先手必败局面

筛法?

- 初始的时候，令所有 $f(i,j,k)=1$
- 从小到大枚举每个三元组 (i,j,k)，每发现一个 $f(i,j,k)=1$，枚举能转移到它的所有局面，把它们的 $f(i,j,k)$ 置为 0
- 显然，枚举到一个 $f(i,j,k)=1$ 时，这就表示它没有被筛掉，从而它无法转移到一个先手必败的局面，从而它本身是先手必败的局面
- 复杂度 $O(N^4)$

只有两堆石子的做法

- 大名鼎鼎的威佐夫博弈
- 通过一个简单的搜索，不难发现它的必败状态为：

- $(1,2),(3,5),(4,7),(6,10),(8,13)\ldots\ldots$

- 通过这个表，能发现什么规律？
  - 每个自然数出现一次
  - 相邻两个必败态中，石子个数之差恰好增加 1

- 这样两个现象其实是非常合理的：
  - 给定 x，应该只存在一个 y 使得 (x,y) 是先手必败态
  - 所有不同的先手必败态的 $(y-x)$ 应该互不相同

- 它们都来源于同一个事实：先手必败状态无法转移到另一个先手必败状态。

一般做法

- 我们考虑优化之前的筛法

- 根据上一页的思路，不难想到，给定 x, y 之后，使得 (x, y, z) 为先手必败态的 z 只有一个

- 不妨用$f(x, y)$ 表示这个 z

- 我们从小到大枚举一个变量 i，然后计算：
  - 有多少个 $f(x, y)$ 的值为 i

Cont’d

- 接上页，如果我们从小到大枚举 i，枚举到当前的 i 时:
  - 所有 $f(x, y) < i$ 的状态已经计算完毕，若一个 $f(x, y) = k < i$，那么代表着 $f(x+k, y), f(x, y+k), f(x+k, y+k)$ 均不可能是 i
  - 所有 $f(x, y) = i$ 的状态中，每个自然数出现不超过 1 次，且 $|x-y|$ 应该互不相同
- 根据这三个原则，我们可以在 $O(N^2)$ 的枚举中，发现所有 $f(x, y) = i$ 的状态 (x, y)。

Cont’d

- 预处理结束之后，对于一个询问 (x, y, z)，根据 $f(x, y)$ 是否等于 z 即可判定先手必胜 / 必败

- 总时间复杂度为 $O(N^3)$

Optimization

- 给定一个数组，选择 K 个不相交的连续段.
- 我们从左到右记这些连续段的和为 $s_1, s_2 \dots s_k$，你需要最大化

$$\sum_{i=1}^{k-1} |s_i - s_{i+1}|$$

- n ⩽ 30000,k ⩽ 200.

一个观察

- 本题有一个关键的性质，没有注意到的话将会寸步难行.
- 假设有两个变量 a_i, b_i，我们有：
$$\max_i(|a_i - b_i|) = \max_i(\max(a_i - b_i, b_i - a_i))$$
- 说人话：遇到**最大化绝对值之和**的题目，我们可以拆开绝对值，在拆开的两种情况中取较大值

- 在本题中，如果没有注意到这个性质，可能不得不设计一个 DP，来记录 s_i 从而求贡献.
- 有了这个观察，我们可以开始讨论一些高效算法.

K = 3

- 不妨设拆出的三个连续段为 a,b,c，我们暴力 ‘钦定’ 他们之间的大小关系.

- 钦定大小关系之后，我们可以知道每个 s_i 对答案的贡献系数。接着记录前 / 后缀的最值来求答案.

- 时间复杂度为 O(N) .

部分分做法

- 我们把每个 $s_i - s_{i+1}$ 的绝对值拆开，考虑每一项对答案的贡献.
- 显然，s_1, s_k的贡献系数为 ±1，中间的 s_i 的贡献系数为 0,±2.
  - 并且，相邻两个 2 之间至少有一个 -2.
- 容易想到它的形式为：**...-2, 0, 0, 2, 0, 0, 0, -2, 0, 2...**
- 因此可以记 f[i][j][k] 表示:
  - DP 到第 i 个位置，之前划分了 j 段，目前处于 ‘阶段’ k ∈ {1,2,3,4} 中.
- DP 枚举下一个连续段，复杂度为 $O(N^3K)$.

优化 1

- 一个观察是，除了开头和结尾，中间的数字可以都被取.
- 因此，转移枚举下一个连续段时可以约定左端点为 i + 1.
- 复杂度为 $O(N^2K)$

优化 2

- 显然同一子段的数贡献系数相等，可以不用直接枚举，而是一个个加进去.

- 时间复杂度为 O(NK).