CF 1093 D2 —Unique Values (Hard)：p_2 的奇偶谓词怎么设计

2026-05-09

1 现在站在哪里

经过前两段二分，已经锁定 p_1（T 第一次出现）和 p_3（T 第三次出现）。还差 p_2，且我们知道 $p_2 \in (p_1, p_3)$。

剩余询问预算大约 11 次（前两段各 11 次），p_2 必须在 11 次询问内通过 **bit-trick 二分**锁定——也就是要找一个**对 k 单调**的 0/1 布尔谓词 $P(k)$，让 $\boldsymbol{P(k)} = \text{true} \iff \boldsymbol{k < p_2}$（或类似的单调切割），然后从 p_1 起按 2^i 加位。

下面把" 为什么之前的 `!res` 不单调" 和" 奇偶谓词怎么救场" 两件事掰开。

2 为什么 `!res` 谓词不是单调的

之前那个写法：query 集合 $[1 \dots k] \setminus \{p_1, p_3\}$，看返回的 res（恰好出现一次的值的个数）是否为 0。

设这个集合为 S，分两类值看 S 里的 count：

- T（出现 3 次的值）：positions 是 $\{p_1, p_2, p_3\}$，前两个被 deny 排除，所以 $\text{count}_T(S) = [p_2 \le k]$，要么 0 要么 1。
- 其它值 w（出现 2 次）：positions 不被 deny 影响，$\text{count}_w(S) = \text{count}_w([1..k]) \in \{0, 1, 2\}$。

res $= 0$ 等价于 **两个条件同时成立**：

(a) $p_2 > k$（T 还没进来）；

(b) $[1..k]$ 里每个 double 值的两次出现要么都进来、要么都没进来——即每个 double 在 $[1..k]$ 里" 凑成对了"。

条件 (b) **完全取决于数组的具体排列**，不是关于 k 的单调量。举个反例：$a = [1, 2, 3, 2, 1, 3, 1]$，$T = 1$，$p_1 = 1, p_2 = 5, p_3 = 7$，deny $= \{1, 7\}$。

k	1	2	3	4	5	6
$S = [1..k] \setminus \{1, 7\}$ 里的值	$\varnothing$	$\{2\}$	$\{2,3\}$	$\{2,3,1\}$	$\{2,3,1,1\}$	$\{2,3,1,1,2\}$
res	0	1	2	3	2	1

res $= 0$ 只在 $k = 0, 1$ 出现（$k = 1$ 的位置还被 deny 掉了）。bit-trick 从 $k = 0$ 起按 $\{4, 2, 1\}$ 加位，没有任何中间步能落在 res $= 0$ 上，整个二分白跑，最后 pos2 变量原地不动 $= 0$。plus$(0, 1)$ 给出"valid 集合的第一个位置 $= 2$"，但真实 $p_2 = 5$——直接错了。

结论：res $= 0$ 这个谓词在整个 k 上不是" 前段全 true、后段全 false" 那种台阶函数，它会跳来跳去，bit-trick 不能保证单调收敛。

3 救场思路：用奇偶性，不要用 `== 0`

奇偶性的好处是**每个值的贡献可以独立判断奇偶**，doubles 永远是偶贡献" 自动失声"，只剩 T 一个值在切换奇偶——然后我们把询问设计成"T 在 k 跨过 p_2 那一刻奇偶翻转"。

3.1 核心：单值贡献到 (len − res) 的奇偶表

对任意询问集合 S，定义 $\text{len} = |S|$，$\text{res} = \#\{v : \text{count}_v(S) = 1\}$。注意到

$$\text{len} - \text{res} = \sum_v \big(\text{count}_v(S) - [\text{count}_v(S) = 1]\big)$$

每个值 v 对 $\text{len} - \text{res}$ 的贡献只取决于它在 S 中的 count：

$\text{count}_v(S)$	0	1	2	3
贡献 $\text{count} - [\text{count} = 1]$	0	0	2	3
奇偶	偶	偶	偶	**奇**

关键观察：只有 $\text{count} = 3$ 那一档贡献奇数，其余三档全偶。

我们的数组里，能让某个值 $\text{count} = 3$ 的，只有那个特殊值 T（其它值总共才出现 2 次，最多 $\text{count} = 2$）。所以 $(\text{len} - \text{res}) \bmod 2$ **只反映 T 在 S 中是否凑齐 3 个 position**。

3.2 设计 query：把 T 的 count 卡在 2 或 3

回想：我们要的单调谓词是"k 是否 $\geq p_2$"。换成 T count 的语言，就是希望

- $k < p_2$：T 的 count 是**偶贡献那档**（最好就是 2）；
- $k \geq p_2$：T 的 count 是**奇贡献那档**（也就是 3）。

让 T count 在 $\{2, 3\}$ 之间切换，必须 S 里**先强制包含** $\{p_1, p_3\}$（贡献 2 个），再可选地把 p_2 包进来。p_2 是否被包进来由 k 决定——把 $[p_1 + 1 \ldots k]$ 整段塞进 S，那么 p_2 当且仅当 $k \geq p_2$ 时进来。

所以设计的询问集合是

$$S_k = \{p_3\} \cup [p_1 \ldots k] \quad (k \in [p_1, p_3 - 1])$$

T 在 S_k 里的 count：

- $p_1 \in S_k$：恒为 `true` ($p_1 \in [p_1, k]$)，贡献 $+1$。
- $p_3 \in S_k$：恒为 `true`（显式塞进来的），贡献 $+1$。
- $p_2 \in S_k \iff p_2 \leq k$。

合计：

k 与 p_2 的关系	$\text{count}_T(S_k)$	$(\text{len} - \text{res}) \bmod 2$
$k < p_2$	2	0（偶）
$k \geq p_2$	3	1（奇）

doubles 全部是偶贡献，不影响奇偶位。∴ 谓词

$$P(k) := \big[(\text{len} - \text{res}) \bmod 2 = 0\big] \iff k < p_2$$

是台阶函数 (在 $[p_1, p_2 - 1]$ 上 `true`，$[p_2, p_3 - 1]$ 上 `false`)，bit-trick 找最大的 k 让 $P(k) =$ `true`，得 $k = p_2 - 1$，加 1 就是 p_2。

4 走一遍

继续上面那个例子：$a = [1,2,3,2,1,3,1]$，$p_1 = 1$，$p_3 = 7$。bit-trick 起点 $k_{\min} = p_1 = 1$，最大可加范围 $p_3 - 1 - p_1 = 5$，$\lfloor \log_2 5 \rfloor = 2$。

位置	1	2	3	4	5	6	7
a	1	2	3	2	1	3	1

i = 2（试加 4，候选 $k = 1 + 4 = 5$）

$$S_5 = \{7\} \cup \{1,2,3,4,5\} = \{1,2,3,4,5,7\}$$

对应值多重集 $= \{1,2,3,2,1,1\}$。count：$1 \mapsto 3$，$2 \mapsto 2$，$3 \mapsto 1$。$\mathrm{res} = 1$（只有值 3 唯一）。$\mathrm{len} = 6$，$\mathrm{len} - \mathrm{res} = 5$，奇 $\Rightarrow P(5) = \texttt{false}$，**跳**。

i = 1（试加 2，候选 $k = 1 + 2 = 3$）

$$S_3 = \{7\} \cup \{1,2,3\} = \{1,2,3,7\}$$

值 $= \{1,2,3,1\}$。count：$1 \mapsto 2$，$2 \mapsto 1$，$3 \mapsto 1$。$\mathrm{res} = 2$，$\mathrm{len} - \mathrm{res} = 2$，偶 $\Rightarrow P(3) = \texttt{true}$，**取**，$k_{\mathrm{cur}} = 3$。

i = 0（试加 1，候选 $k = 3 + 1 = 4$）

$$S_4 = \{7\} \cup \{1,2,3,4\} = \{1,2,3,4,7\}$$

值 $= \{1,2,3,2,1\}$。count：$1 \mapsto 2$，$2 \mapsto 2$，$3 \mapsto 1$。$\mathrm{res} = 1$，$\mathrm{len} - \mathrm{res} = 4$，偶 $\Rightarrow P(4) = \texttt{true}$，**取**，$k_{\mathrm{cur}} = 4$。

循环结束 $k_{\mathrm{cur}} = 4$，$p_2 = k_{\mathrm{cur}} + 1 = 5$。✓

5 代码

替换原来 line 166–182 那段：

```cpp
// pos1 and pos3 are already known; find pos2.
ll kCur = pos1;                              // P(kCur) is always true
ll range = pos3 - 1 - pos1;                  // >= 1 since pos2 lies in (pos1, pos3)
for (int i = __lg(range); i >= 0; --i) {
    ll k = kCur + (1ll << i);
    if (k >= pos3) continue;                 // out-of-range guard
    vector<ll> qs = {pos3};                  // force-include pos3
    for (ll p = pos1; p <= k; ++p)           // then include [pos1..k]
        qs.push_back(p);
    ll res = query(qs);
    ll len = (ll) qs.size();
    if ((len - res) % 2 == 0) {              // T count == 2, pos2 not yet in
        kCur = k;
    }
}
ll pos2 = kCur + 1;
cout << "! " << pos1 << " " << pos2 << " " << pos3 << endl;
```

6 询问数核账

每段二分的次数 $= \lfloor \log_2 \mathrm{range} \rfloor + 1$：

- p_3：$\mathrm{range} = 2n + 1 \le 2001$，$\le 11$ 次。

- p_1：range $= p_3 \le 2n+1$，≤ 11 次。
- p_2：range $= p_3 - 1 - p_1 \le 2n-1$，≤ 11 次。

合计 ≤ 33，正好卡到题目限制。

7 把这套思路一句话浓缩

交互 + 二分 = 找一个对 k 单调的奇偶量。doubles 自带" 偶贡献"buff 自动从奇偶位上消失，所以只需要把那个出现 3 次的" 主角" 放在贡献能从偶切到奇（$2 \to 3$）的位置上——这要求询问集合**先把 p_1, p_3 强制塞进去当 baseline**，再用 $[p_1+1, k]$ 的可变前缀决定 p_2 进不进来。