

# CF Round 1095 (Div.2) C —Mental Monumental (Easy Version) 带教对话

Claude & zzy

2026-04-29

## 题目链接

**题面摘要：**给定数组  $a = [a_1, \dots, a_n]$  ( $0 \leq a_i \leq 10^6$ )。要求选一个数组  $b = [b_1, \dots, b_n]$ ,  $b_i \geq 1$ , 把每个  $a_i$  替换成  $a_i \bmod b_i$ 。求变换后数组的最大可能 mex。

**约束：**  $n \leq 2 \cdot 10^5$ ,  $\sum n \leq 2 \cdot 10^5$ ,  $\sum \max(a) \leq 10^6$ , 时限 4s。

## 第 1 步：理解题目 + 单元素可达集合

### Claude

先把题意吃透。样例 2:  $a = [6, 7]$ , 标准答案  $\text{mex} = 2$ 。

你能给我一组具体的  $b$ , 让变换后  $\text{mex}(c) = 2$  吗? (题面里给的  $b = [3, 3]$  是其中一个, 能不能写一个不同的?)

### zzy

比如说 5, 6 呗

### Claude

算一下  $b = [5, 6]$ :  $6 \bmod 5 = 1$ ,  $7 \bmod 6 = 1$ , 所以  $c = [1, 1]$ ,  $\text{mex} = 0$ , 不是 2。让  $\text{mex} = 2$  的  $b$  例如:

- $b = [6, 6]$ :  $6 \bmod 6 = 0$ ,  $7 \bmod 6 = 1$ , 得  $c = [0, 1]$ ,  $\text{mex} = 2$ 。
- $b = [1, 6]$ :  $c = [0, 1]$ ,  $\text{mex} = 2$ 。

### zzy

```
11 mxRem = A[i] % (A[i] / 2 + 1);
```

这个  $\text{mxRem}$  是这个东西, 通过  $\%$  可以得到的。我们直接跳过这个非常傻逼的阶段, 我们直接来考虑一下, 就是原值和这个  $\text{Rem}$  值的这个取舍。不会是使用这个二分吧, 就是二分这个答案进行 check。

### Claude

**mxRem 公式对了:**  $a \bmod (\lfloor a/2 \rfloor + 1) = \lfloor (a-1)/2 \rfloor$ , 所以对每个  $a_i \geq 1$ , 可达集合是

$$S_i = \{0, 1, \dots, \lfloor (a_i - 1)/2 \rfloor\} \cup \{a_i\}.$$

(特别地,  $a_i = 0$  的可达集合是  $\{0\}$ 。)

**二分不可行: 可行。** 设  $f(K) = \lceil \text{能否覆盖 } [0, K-1] \rceil$ , 则  $f$  单调 ( $K$  越大越难), 所以二分  $K + \text{check}$  是合法路径。不过你也可以不用二分: 从  $v = 0$  开始顺序贪心, 匹配不上就停——

停在哪个  $v$ ，答案就是  $v$ 。两种写法 check 函数本质一样。

下一个问题 (check 的入口): 对每个值  $v \in [0, K - 1]$ , 哪些  $a_i$  有资格成为  $v$  的提供者? 用关于  $a_i$  取值的条件把”有资格”刻画出来。

ZZY

我们先确认一下这个题意啊，这个操作是不是对这个每个点都必须这个进行操作??

Claude

是的，每个  $a_i$  都必须做一次 (题面”for every  $1 \leq i \leq m$ ”)。

但你可以选  $b_i > a_i$  让  $a_i \bmod b_i = a_i$ , 相当于”保持原值”。这就是为什么  $S_i$  里始终包含  $a_i$  本身。

样例 1  $a = [0, 1, 2, 3]$  配  $b = [1, 2, 3, 4]$  就是这个意思——每个  $b_i > a_i$  ( $b_0 = 1 > 0$ ), 所以  $c = a$  没动,  $\text{mex}(a) = 4$ 。

回到刚才那个问题: 对每个值  $v \in [0, K - 1]$ , 哪些  $a_i$  有资格成为  $v$  的提供者? 把条件用  $a_i$  取值写出来。

ZZY

不是，我就不理解了，你打算怎么样贪心，反正从小到大贪心肯定是不行的，你打算怎么样贪心?

## 第 2 步：贪心策略 + 关键观察

Claude

你说的”从小到大不行”——指简单的”每个  $v$  随便挑符合条件的最小元素”。这确实有反例:  $a = [0, 5]$ , 若  $v = 0$  抢了  $a_1 = 5$ , 则  $v = 1$  死掉 ( $\text{cnt}[1] = 0$  且没剩  $\geq 3$  的)。但加一个 priority 就能救活。

先把上一题的答案给你做建材:

$$\text{cand}(v) = \{i : a_i = v\} \cup \{i : a_i \geq 2v + 1\}$$

(要么”原值”分支让  $a_i = v$  直接提供  $v$ ; 要么”Rem”分支让  $\lfloor (a_i - 1)/2 \rfloor \geq v$ , 即  $a_i \geq 2v + 1$ 。)

引导问题 (关键观察): 考虑一个  $a_i = v$  的元素, 看它的  $S_i = \{0, \dots, \lfloor (v - 1)/2 \rfloor\} \cup \{v\}$ 。处理完  $v$  之后, 这个元素还有可能被某个未来的  $v' > v$  用来覆盖吗? 为什么?

ZZY

最重要的就是覆盖范围和原值的这个取舍问题。我前面就提到了，但你没回答我。

Claude

抱歉，前面那段直接抛策略违反引导原则。重来。

用你的元素视角: 每个  $a_i$  二选一——

- **A 模式**: 贡献  $a_i$  本身。
- **B 模式**: 贡献  $[0, \lfloor (a_i - 1)/2 \rfloor]$  中某个值。

现在按  $v = 0, 1, 2, \dots$  从小到大依次”派人”覆盖。

引导问题: 当处理到  $v$  时, 所有  $v' < v$  都已经覆盖完。考虑一个  $a_i = v$  的元素, 它有两条路——

- A 模式: 贡献  $v$  (正是当前要的);

- B 模式：贡献  $\leq \lfloor (v-1)/2 \rfloor$  的某个小值。

B 模式那条路在此时此刻还有意义吗？为什么？

zzy

废话

Claude

默认你答”B 给的小值都覆盖过了，没意义”——所以  $a_i = v$  要么 A 模式贡献  $v$ ，要么彻底废。这就是 priority 1。

下一步：考虑  $\text{cnt}[v] = 0$  的情况——没人  $= v$ ，得从大元素里借一个走 B 模式补。候选是  $\{a_j : a_j \geq 2v + 1, \text{未用}\}$ ，可能不止一个。

比如  $v = 2$ ，剩余里有  $a_j = 5$  和  $a_k = 11$ （都  $\geq 5$ ）。该挑哪一个？为什么？

zzy

这个不一定吧，11 可能在后面使用其特性贡献  $[0, 5]$ ，5 也可以贡献 5。你这个例子举得很差。这个例子肯定选择 5，因为 5 能做的 11 也能做。但是如果是 5 和 9 的话，5 能做的 9 不一定能做，9 能做的 5 也不一定能做。

Claude

你这个反驳完全对，priority 2 「选最小」一刀切是错的。承认。

具体看  $a_i = 5$  vs  $a_j = 9$  在未来  $v' > v$  处理时的可用集合：

- $a_i = 5$ ：未来能填  $\{0, 1, 2, 5\}$  ——独占原值点  $v' = 5$ 。
- $a_j = 9$ ：未来能填  $\{0, 1, 2, 3, 4, 9\}$  ——涵盖  $\{3, 4\}$ ，但  $9 \neq 5$  且  $9 < 11$ ，填不了  $v' = 5$ 。

两者互不 dominate，”选最小”不一定对——若未来  $\text{cnt}[5] = 0$  且没  $\geq 11$  的元素，”留 9 用 5”直接让  $v' = 5$  死掉。

所以正确策略不是”在候选里选某种大小”——而是把” $\text{cnt}[v] = 0$  的所有  $v$ ”作为整体，和”剩余的大元素池”做一次匹配：

- $V' = \{v \in [0, K-1] : \text{cnt}[v] = 0\}$ （待补的”漏”）；
- $R =$  原数组中没被 priority 1 用掉的元素。

给每个  $v \in V'$  配一个  $r \in R$  满足  $r \geq 2v + 1$ （一一对应）。

引导问题： $V'$  和  $R$  各自排序后，怎么匹配最容易成功？

zzy

哦，我懂你意思了。

```
bool check(vector<ll> &freq, const vector<ll> &A,
           set<ll> &st, multiset<ll> &stA, ll ans) {
    freq.assign(ans+2, 0);
    st.clear(); stA.clear();
    for (int i = 0; i < ans; ++i) st.insert(i);
    for (auto a : A) {
        if (a < ans) freq[a]++;
        stA.insert(a);
    }
}
```

```

for (int i = 1; i < ans; ++i) {
    if (freq[i] > 1) {
        st.erase(i);
        stA.erase(stA.find(i));
    }
}
for (auto a : stA) {
    ll mxRem = a % (a/2 + 1);
    if (st.empty()) return true;
    if (*st.begin() <= mxRem) st.erase(st.begin());
}
return st.empty();
}

```

不过我已经写了一个代码，确认了，你的思路是有问题的。这个  $\text{freq} > 1$  的值是没问题的，但是  $\text{freq} \leq 1$  的这个节点怎么取舍也是很难的。

### Claude

你  $\text{freq} > 1$  那部分对—— $\text{cnt}[v] \geq 2$  时至少留一个  $a = v$  走 A 模式没问题。

**但  $\text{freq} == 1$  也得走 priority 1:** 理由前面那条—— $a = v$  在所有  $v' < v$  都覆盖完之后完全没用 (B 模式只能填  $\leq \lfloor (v-1)/2 \rfloor$  的小值，那些位置已处理过)，不用就废。所以 priority 1 的正确版本是：对每个  $v \in [0, K-1]$  只要  $\text{cnt}[v] \geq 1$  就拿一个  $a = v$  出来 A 模式覆盖  $v$ 。

**反例:**  $a = [3, 5, 5, 5]$ ,  $K = 4$  实际可行:

$$3 \xrightarrow{A} 3, \quad 5 \xrightarrow{B} 2, \quad 5 \xrightarrow{B} 1, \quad 5 \xrightarrow{B} 0.$$

但你的  $\text{check}(4)$  会返 false—— $\text{freq}[3] = 1$  没触发那个  $\text{erase}$ ,  $\text{stA}$  里 3 留到贪心阶段。遍历到第一个  $a = 3$  时  $\text{mxRem} = 1$ , 配  $\text{st}$  最小值 0 (B 模式), 后面  $v = 3$  没人填 ( $5 < 7 = 2 \cdot 3 + 1$ )。

**修复:**  $\text{if}(\text{freq}[i] > 1)$  改成  $\text{if}(\text{freq}[i] \geq 1)$ , 循环从  $i = 0$  起 ( $v = 0$  同样适用)。

**修复后剩下的”R 从小到大、配 st 最小未覆盖”贪心是对的——这是经典的 Hall + greedy 匹配 ( $V'$  asc 和  $R$  asc 一一对应: 第  $k$  个  $V'$  元素配一个能跨过  $2v_k + 1$  门槛的小  $R$ , 留大  $R$  给后面更大的  $v$ )。**

下一步: 改完代码, 跑题面 4 组样例看输出是不是 4 / 2 / 5 / 6。跑通后告诉我, 再看复杂度。

zzy

AC, [submission 372915486](#)。但是我还是不太懂啊, 或者说, 我有点懂, 又不是特别懂。具体卡点: priority 1 为什么严格成立?

## 第 3 步: priority 1 的交换论证

### Claude

**Claim:** 若存在覆盖  $[0, K-1]$  的方案  $M$ , 且  $\text{cnt}[v] \geq 1$  (数组里有  $a_i = v$ ), 则存在覆盖同样范围的方案  $M'$  使得  $a_i$  走 A 模式覆盖  $v$ 。

**关键引理:** 若  $a_j \geq 2v + 1$  且  $a_i = v$ , 则  $S_i \subseteq S_j$  (整体包含)。

证明:

- $S_i = \{0, \dots, \lfloor (v-1)/2 \rfloor\} \cup \{v\}$ , 所有元素  $\leq v$ 。

- $S_j \supseteq \{0, \dots, \lfloor (a_j - 1)/2 \rfloor\}$ , 而  $a_j \geq 2v + 1 \Rightarrow \lfloor (a_j - 1)/2 \rfloor \geq v$ 。
- 所以  $\{0, 1, \dots, v\} \subseteq S_j$ , 整段包含  $S_i$ 。 □

**交换步:** 在  $M$  里看  $v$  是谁覆盖的:

- 若已是某  $a_j = v$  走 A 模式覆盖  $v$  —— 跳过。
- 否则是某  $a_j > v$  走 B 模式 (必须  $a_j \geq 2v + 1$ )。此时另一个  $a_i = v$  在  $M$  里的输出是某  $c_i \in S_i$ 。 **交换  $a_i$  与  $a_j$  的角色:**
  - $a_i$  改输出  $v$  (A 模式, 合法  $v \in S_i$ );
  - $a_j$  改输出  $c_i$  (引理给出  $c_i \in S_i \subseteq S_j$ , 合法)。

交换后  $c$  数组的**多重集不变** ( $v$  和  $c_i$  在两个位置上对换), 所以  $[0, K - 1]$  仍被覆盖。

**推广:** 对每个  $\text{cnt}[v] \geq 1$  的  $v$  **独立** 做一次交换 (互不影响——每次交换只改两个元素的输出且多重集不变), 最终得到  $M'$  满足 priority 1。 □

**这论证为什么“严格”:** 它说明“无论原最优解长啥样, 都能调整成 priority 1 形式”——所以贪心强制走 priority 1 不可能错过最优解。这是 exchange argument 的标准套路: 构造性地把任何方案调整到 we 希望的“规范形式”。

看完这段, 哪步还接不上? 或者其他几个卡点 (R/V' 匹配、二分单调性、单元素可达集合) 想看哪个继续?

zzy

呃, 显然, 我不喜欢看数学证明, 你就用自然语言描述一下, 但又不失严谨性。

Claude

**核心比喻: 专才与通才。**

把数组里每个  $a_i$  想成一个“工人”, 他只能做某几种“活”——就是他能输出的值。

- $a_i = v$  是**专才**: 只会做  $v$  这一件“主活”, 加上一些更简单的杂活 (值  $\leq \lfloor (v - 1)/2 \rfloor$ )。
- $a_j \geq 2v + 1$  是**通才**: 除了能做自己的“原值活”  $a_j$ , 还能做所有  $\leq \lfloor (a_j - 1)/2 \rfloor \geq v$  的杂活——这一段**整整覆盖了专才能做的全部活** (包括  $v$  自己)。

**专才会做的, 通才一定会做** (除了通才自己的原值  $a_j$ , 那是通才独有的)。

**核心主张:** 任何最优分工方案里, 如果当前是“通才在做  $v$ 、专才在做别的或闲着”——**对调他们**: 让专才做  $v$  (他完全能做), 让通才接管专才原本的活 (通才覆盖了专才所有能力, 也合法)。对调之后**总输出不变**, 覆盖到的值集合一字不差, 所以仍然最优。但现在专才在做他的“主活”  $v$ , priority 1 自动满足。

**严谨性来自三点:**

1. **可对调:** 通才能力  $\supseteq$  专才能力——通才接专才的活合法。
2. **不破坏其他覆盖:** 对调只换两人的输出, 其他工人没动, 整体输出的**多重集仅仅“换了序”**, 覆盖到的值集合一模一样。
3. **每个  $v$  独立处理:** 对  $v_1$  做的对调不影响  $v_2$  (多重集不变), 所以对所有  $\text{cnt}[v] \geq 1$  的  $v$  都做一遍, 最后 priority 1 同时满足。

所以——**贪心强制走 priority 1 不会错过任何最优解**, 因为最优解总能“调整成” priority 1 的样子。

这段是不是顺多了? 看完还有不接的地方吗?