

# CF 2229F Load Unbalancing —引导对话录

Claude  $\leftrightarrow$  znzryb

2026-05-25

题目链接: <https://codeforces.com/contest/2229/problem/F>

比赛: Spectral::Cup 2026 Round 2 (Codeforces Round 1100, Div. 1 + Div. 2)

限制:  $1 \leq k \leq n \leq 18$ ,  $1 \leq a_i \leq 10^9$ ,  $\sum 2^n \leq 2^{18}$ , 1.5s / 256 MB

## 第 1 步: 吃透题意

Claude

开始之前先确认我们对题意的理解一致。第二组样例的第一个 case 是:

$$n = 4, k = 2, a = [1, 2, 4, 5], \text{答案} = 8.$$

题面 note 给的最优重排是  $[1, 4, 2, 5]$ ,  $b$  演化轨迹:

$$[0, 0] \rightarrow [1, 0] \rightarrow [1, 4] \rightarrow [3, 4] \rightarrow [8, 4], \quad \max(b) = 8.$$

**问题:** 换一个角度——上面这个轨迹里, 元素 5 是**最后一个**被处理的。如果我们把同一个集合  $\{1, 2, 4, 5\}$  用**别的顺序** (比如把 5 放在前面、或者中间) 也试一试, 能不能也达到  $\max(b) = 8$ ? 或者, 能不能达到**比 8 更大的值**?

随便挑两三种重排手算一下  $b$  的最终状态告诉我。

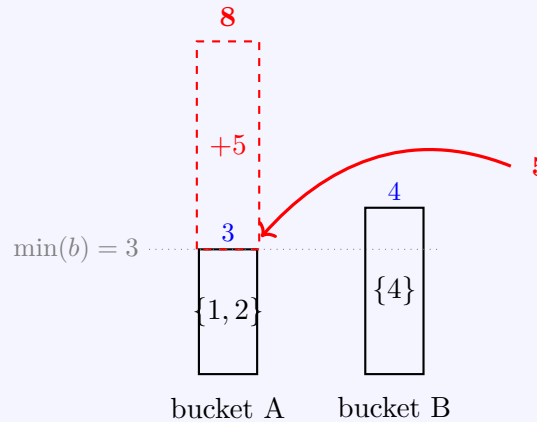
znzryb

okay, 我大概懂这个题目逻辑了, 这个答案是 8, 是因为 1, 2 搞出来 3, 4 搞出来 4, 正好 5 加在 3 上, 因此不可能更好了

Yes, 这个构造的几何描述正好抓住了关键的两件事:

- **两堆的最终高度**: 把  $\{1, 2, 4, 5\}$  切成两个子集, 一个塞  $\{1, 2\}$  累到高度 3, 另一个塞  $\{4\}$  高度 4。
- **最后一步的” 杠杆”**: 处理到 5 时,  $b = [3, 4]$ ,  $\min(b) = 3$ , 所以 5 必须加在那一堆, 得  $3 + 5 = 8$ 。

这等价于一个直观图景:



**推进一步: 复杂度预算对齐。**

题面给的约束 (重抄一下):

- $1 \leq k \leq n \leq 18$
- $1 \leq a_i \leq 10^9$
- $\sum 2^n \leq 2^{18}$  (所有 testcase 的  $2^n$  之和不超过 262144)
- 时限 **1.5 s** (“unusually low time limit”)
- 测试组数  $t \leq 10^4$

**问题:** 上面这四条放在一起, 特别是  $\sum 2^n \leq 2^{18}$  这条非常显眼的约束——你觉得它在暗示什么量级、什么风格的算法? (不用想最终算法长什么样, 只要给量级 + 风格判断就行。)

znzryb

明显是状态压缩 dp

Claude

Yes,  $\sum 2^n \leq 2^{18}$  几乎就是 bitmask DP 的明牌信号。

量级核算 (确认 1.5s 撑得住):

$$\underbrace{\sum_{\leq 2^{18}} 2^n}_{\leq 2^{18}} \times \underbrace{\text{每个 mask 的转移代价}}_{O(n) \text{ 量级}} \times \underbrace{\text{外层系数}}_{\text{常数 / log ?}}$$

每个 mask 一次  $O(n)$  转移:  $2^{18} \cdot 18 \approx 5 \times 10^6$ 。外面再套一个常数到  $\log A$  量级的系数都还在  $10^8$  以内, 1.5s 够。

## 第 2 步: 状态压缩 DP 的设计

Claude

既然走 bitmask DP, 最自然的第一刀是:

$$dp[\text{mask}] = ?$$

其中  $\text{mask} \subseteq \{0, 1, \dots, n-1\}$  表示已经处理过的下标集合。

**问题:**  $dp[\text{mask}]$  这一格里你打算存什么? 换句话说, 处理完 mask 这些元素之后, 什么信息足以决定后续的演化 + 最终答案? 先尝试给一个 naive 的 state (你脑子里第一反应那个), 不用现在就追求“能塞进数组”。

znzryb

这个, 长度为 k 的那个数组, 足以决定后续的演化

Claude

Yes,  $b$  这个 multiset 在数学上确实是 sufficient statistic——给定 mask 和当前  $b$ , 后续每一步加到  $\min(b)$  的规则就把演化锁死了, 不需要回头看历史顺序。

但来算  $b$  的规模:

- 长度  $k$ , 每个  $b_i \in [0, \sum_{j \in \text{mask}} a_j]$ 。
- $\sum a$  最大  $\sim 18 \times 10^9 = 1.8 \times 10^{10}$ 。
- 即使把桶之间的对称去掉 (看成 multiset / 升序排列的 tuple),  $b$  的可能取值数量级远远超过  $2^{18}$ 。

也就是说  $dp[\text{mask}][b]$  这种 naive state 不能直接当 dp 下标—— $b$  是  $k$  维大整数 tuple, 张不开。

**推进一步: 把  $b$  这个 multiset 再压一刀。**

既然整个  $b$  装不进 dp, 那就要从  $b$  里抽很少几个量 (理想是一个标量), 让它就足以接续 dp。

**引导问题:** 题目规则里, 下一步加  $a_i$  时只看了  $b$  的什么部分? 换言之,  $b$  里哪一个量是”过程的下一步动作锚点”? (先回答这一个量是谁, 怎么把它扣进 dp 状态我们下一步再聊。)

znzryb

这个量就是  $b$  中的最小值?

Claude

**Yes,**  $\min(b)$  就是动作锚点——下一步加  $a_i$  时去  $\min$  那个桶, 加完后  $\min$  桶变成  $\min(b) + a_i$ 。

但说句不那么动听的:  $\min(b)$  单独追踪其实并不够 sufficient——加完后新的  $\min$  是谁, 要看其它桶的值 (原  $\min + a_i$  可能超过某个原非  $\min$  桶)。如果死磕”naive simulate”路线, 状态压缩压不下来。

**关键扭转:** 我们不需要 sufficient state 来 simulate。我们只关心一个数——最终的  $\max(b)$ 。从这个数倒推就豁然开朗。

设整条处理序列里最后处理的元素是  $a_{\text{last}}$  (数值, 不是下标)。最后一步把  $a_{\text{last}}$  加到了某个  $\min$  桶上。

**引导问题:** 最终的  $\max(b)$  跟  $a_{\text{last}}$ 、跟”加之前的  $b$ ”——写一个等式 / 不等式表达它们的关系。

(不用考虑  $a_{\text{last}}$  取谁最优, 先把这个关系写出来。)

znzryb

$a_{\text{last}} + \min(b \text{ 之前}) = \max(b \text{ 之后})?$

## Claude

方向对了——你抓到了”最后那次 +” 会贡献  $\min(b_{\text{before}}) + a_{\text{last}}$  这个候选项。严格点写：

$$\max(b_{\text{after}}) = \max\left(\underbrace{\max(b_{\text{before}})}_{\text{其它桶里原本最大}}, \underbrace{\min(b_{\text{before}}) + a_{\text{last}}}_{\text{被加完的那个桶}}\right).$$

即两个候选之间取较大者。你写的等式相当于隐含假设”右项胜出”，这在  $a_{\text{last}}$  比较大时确实成立。

**推进一步：**  $a_{\text{last}}$  我们有完全的自由（自己选重排顺序）。

直觉：上式右项 =  $\min(b_{\text{before}}) + a_{\text{last}}$ ， $a_{\text{last}}$  越大这一项越大。

**引导问题：**从  $\{a_0, a_1, \dots, a_{n-1}\}$  里， $a_{\text{last}}$  取谁最优？给出你的猜测 + 一个简短理由。（先不用严格证明，给直觉版即可。）

## znzryb

alast 取最大的？我是这样子感觉的

## Claude

**Yes,**  $a_{\text{last}} = \max(a)$  就是最优。这是 editorial 的 Observation 1。

**直觉：**右项  $\min(b_{\text{before}}) + a_{\text{last}}$  随  $a_{\text{last}}$  单调增；只要它能 dominate 左项  $\max(b_{\text{before}})$ ，越大越好。 $a_{\text{last}}$  取  $\max(a)$  时这一项给出最强的 dominate。

**严格化：**用 exchange argument。设最优重排里  $\max(a)$  不在末位，记  $\max(a) = x$  在位置  $i < n$ ，末位元素 =  $y$ 。设最终最大桶在交换前贡献了 ans。

- 把  $x$  从原位置  $i$  取出，把  $y$  从末位取出，再把  $x$  放到末位、 $y$  顶替位置  $i$ 。
- 这等价于”末位放  $x$ ”，重新走流程。
- 可证：新策略的  $\max(b_{\text{final}}) \geq \text{ans}$ 。Editorial 的小证明分两 case ( $y$  的原桶仍是 min /  $x$  的原桶变成 min)，都凑出  $\geq \text{ans}$ 。

证明细节不重要，关键是结论： $a_{\text{last}} = \max(a)$  一定可以放在最后位置而不损失最优性。所以我们从一开始就锁定它。

**推进一步：**既然  $a_{\text{last}} = \max(a)$  锁死，原问题被转化成什么？

记  $x = \max(a)$ ， $a' = a \setminus \{x\}$ （剩下  $n - 1$  个元素）。

$$\max(b)_{\text{final}} = \max(\max(b'), \min(b') + x),$$

其中  $b'$  = 用  $a'$  处理完后的  $b$  状态。

**引导问题：**既然  $x = \max(a)$  比  $a'$  里每个元素都至少不小，那  $\min(b') + x$  跟  $\max(b')$  比，哪一项更可能“胜出”成为最终的  $\max$ ？为什么？（先答这一个，下一步再聊原问题转化成什么子问题。）

znzryb

$\min(b')+x$ ? 我猜的

Claude

Yes,  $\min(b') + x$  会胜出。

**Claim:**  $\max(b') - \min(b') \leq \max(a') \leq x$ 。

**每步**  $\Delta = \max(b) - \min(b)$  的演化判定表（要把  $a_i$  加到  $\min$  桶上）：

| Case | 条件                     | 新 max        | 新 min            | 新 $\Delta$        |
|------|------------------------|--------------|------------------|-------------------|
| 1    | $\min + a_i \leq \max$ | max          | 第二低桶 $\geq \min$ | $\leq$ 原 $\Delta$ |
| 2    | $\min + a_i > \max$    | $\min + a_i$ | 第二低桶 $\geq \min$ | $\leq a_i$        |

任意时刻  $\Delta$  要么不增、要么被新加的  $a_i$  兜底；初始  $\Delta = 0$ 。所以处理完  $a'$  后  $\Delta \leq \max(a') \leq x$ ，即

$$\min(b') + x \geq \max(b')$$

恒成立。回到主公式：

$$\max(b)_{\text{final}} = \max(\max(b'), \min(b') + x) = \min(b') + x.$$

右项无条件胜。

**推进一步：**既然原问题 =  $\min(b') + x$  最大化， $x$  是常数，那么完全等价于：

用  $n - 1$  个元素  $a'$ （按“每步加到  $\min$  桶”规则）装  $k$  个桶，最大化最终  $\min(b')$ 。

这是一个干净的子问题，跟原问题的纠缠（ $\max$  跟  $\min$  混在一起）已经解开了。

**引导问题：**这个子问题（“ $k$ -桶  $\max$ - $\min$ ”）你之前见过吗？如果让你不考虑过程规则、只看最终结果——你能把它再翻译成更纯粹的集合划分语言吗？

（提示：忽略“每步加到  $\min$ ”的过程约束，只看“ $k$  个桶各自最终装了哪些元素”。）

znzryb

那说白了，就是把原集合（除了  $\max a$ ）划分成和尽可能相等的几个集合是吧

Claude

方向对，严格点是”**最小子集和最大化**”（不一定相等；无法均分时这是更精确的表述）：

把  $a'$  划分成  $k$  个不相交子集  $S_1, \dots, S_k$ ，最大化  $\min_j \text{sum}(S_j)$ 。

**为什么过程约束可以丢**：editorial 给了一个 greedy 构造——任意一种划分  $\{S_j\}$ （每个  $\text{sum}(S_j) \geq v$ ），都能找到一个合法重排 + tie-breaking，让最终每桶就是  $S_j$ 。具体：每次该往某桶加时，从对应  $S_j$  里挑**剩余最大**的元素。所以”过程约束”在”最终能否达到  $\min(b') \geq v$ ”维度上无副作用，**划分本身就是充要条件**。

**推进一步**：**max-min partition 的经典套路**。

直接对”min 子集和”做 DP 不太顺手（DP 状态里要存”当前已划分子集的最小和”很别扭）。但 max-min 这类目标函数有一个**教科书级的转换**。

**引导问题**：max-min 类问题（最小化 max / 最大化 min）的经典套路是什么？把”求 max-min”转成什么**更好处理**的形式？

znzryb

这种平均化问题一眼二分啊

Claude

**Yes**, 标准套路：**二分答案  $v$  + 写  $\text{check}(v)$** 。把”求 max-min”改成”**验证某个  $v$  是否可达**”——后者是个 yes/no 判定，处理起来比目标函数直接 DP 容易得多。

二分上界： $\min(b') \leq \sum a'/k \leq \sum a'$ ，所以  $v \in [0, \sum a']$ ，二分 log 在  $\log(1.8 \times 10^{10}) \approx 34$  次量级。

**推进一步**：**设计  $\text{check}(v)$** 。

把 max-min 问题脱壳： **$\text{check}(v)$  要回答 yes/no 的判定命题是什么？写出来 + 怎么用 bitmask DP 实现？**

（如果你之前练过” $n \leq 20$  划分子集”这种题型，直接出 DP 状态；如果觉得卡壳就先只回答判定命题。）

znzryb

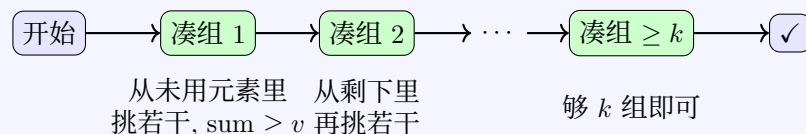
这个 bitmask dp 有点难实现啊，因为，如果看成一个物品有  $k$  种状态的话，就不太做的出来了

## Claude

你卡的点正是这个 DP 的精华——别按”每个物品分到  $k$  个桶之一”编码状态。那等价于  $dp[\text{mask}]$ [每个桶的子集 / 累计] 的多维状态，确实爆。  
关键观察：题目里  $k$  个桶是匿名等价的（不区分 id，只关心最终 multiset）。所以  $\text{check}(v)$  的等价形式不是”把  $a'$  划分进  $k$  个有 id 的桶”，而是：

能否贪心地从  $a'$  里一个一个凑出  $k$  个不相交子集，每个  $\text{sum} \geq v$ ？

把”凑子集”想成一个线性的串行过程：



每一时刻，状态可以由两个标量描述：

- 已封口的组数  $\text{cnt}$  ——已经凑出几个  $\text{sum} \geq v$  的合格组。
- 当前未封口组的累加和  $\text{cur}$  ——还差  $v - \text{cur}$  就能再封一组。

加上  $\text{mask}$ （哪些元素已用）， $dp[\text{mask}] = (\text{cnt}, \text{cur})$  就够了。

引导问题：基于这个流水线视角，给出  $dp[\text{mask}]$  的：

- 转移方程（从  $\text{mask} \setminus \{i\}$  拉过来，或从  $\text{mask}$  推到  $\text{mask} \cup \{i\}$ ）
- 多个候选 state 之间怎么比”哪个更优”（贪心字典序？）
- 最终  $\text{check}(v) = ?$

（写一段伪代码 / 公式都行。）