取石子 解题报告

-1 是随手加上的，~~稍有常识的人都能看出~~根本不可能无解。

算法 1

直接输出 0，可以通过第一个子任务。

算法 2

简单的状压 DP，只需要记录当前每堆石子的剩余个数。

算法 3

考虑 $a_i = 0$ 的情况。

首先我们加一堆编号为 n 的石子 $(0, 10^9 n, 10^9 nt)$，这样可以使每次操作必定取满 k 个。

记 sb_i 为 b_i 的前缀和。

令 $f[i][j]$ 表示：考虑编号为 $[0, i]$ 的石堆，且游戏进行 j 轮，且每次取石子都取满 k 个的最少操作次数；$g[i][j]$ 表示：考虑编号为 $[0, i]$ 的石堆，且游戏进行 $j + 0.5$ 轮（即额外进行一轮取石子操作），每次取石子都取满 k 个，且编号为 $[0, i)$ 的石堆都被取完的最少操作次数。显然答案即为 $f[n][t]$。

从小到大枚举 i, j，每次考虑石堆 i 的影响。分两种情况转移：

- 存在一组方案使得能在不取第 i 堆石子的情况下通关（即 $j \cdot b_i \le c_i$ 且 $f[i-1][j] \ne \infty$）。这种情况下，转移为：
  - $f[i][j] \leftarrow f[i-1][j]$
  - $g[i][j] \leftarrow \lceil \frac{j \cdot sb_{i-1}}{k} \rceil$

  其中第二种转移只在 $\lceil \frac{j \cdot sb_{i-1}}{k} \rceil \cdot k \le j \cdot sb_i$ 时进行。

  显然 $\lceil \frac{j \cdot sb_{i-1}}{k} \rceil$ 是取完 $[0, i)$ 需要的最少操作次数；而由于方案的存在性，我们可以在前 j 轮中，用不超过 $\lfloor \frac{j \cdot sb_{i-1}}{k} \rfloor$ 次操作通关，只需要在第 $j + 0.5$ 轮时补一些操作即可。因为规定了每次一定取满 k 个，所以这个下界是一定能达到的。
- 否则，我们枚举最后一次取 i 的时间 r，那么我们的策略一定是这样的：
  - 在前 $r + 0.5$ 轮中取完 $[0, i)$ 的石子。这一部分的答案显然为 $g[i][r]$。
  - 用这一轮操作从 i 中取走石子。我们先计算此时 i 中剩余的石子数 $m = r \cdot sb_i - k \cdot g[i][r]$，为了让 i 的石子数不超过上限，我们至少还需要 $x = \lceil \frac{\max(0, m + (j-r) \cdot b_i - c_i)}{k} \rceil$ 次操作。如果 $x \cdot k > m$，则无解。
  - 在剩下的 $j - r$ 轮中取石子。由于 r 是最后一次取 i 的时刻，因此我们只需要考虑 $[0, i)$ 这些石堆了。对于 $f[i][j]$，操作次数显然为 $f[i-1][j-r]$；对于 $g[i][j]$，与情况 1 类似，可以得到最少操作次数为 $\lceil \frac{(j-r) \cdot sb_{i-1}}{k} \rceil$。

  因此，我们的转移为：

  - $f[i][j] \leftarrow g[i][r] + x + f[i-1][j-r]$
  - $g[i][j] \leftarrow g[i][r] + x + \lceil \frac{(j-r) \cdot sb_{i-1}}{k} \rceil$

  第二种转移仍然需要判断取的石子数是否不超过总数。

时间复杂度为 $O(nt^2)$，可以通过子任务 3。

算法 4

考虑将算法 3 进行扩展，不难发现区别仅在于转移 2 的第三步，此时所有石堆的个数都是 0。因此我们只需要将状态扩展为 $f[i][j][z](z \in \{0,1\})$，表示每堆石子初始时有 $z \cdot a_i$ 个，在第三步时用 $f[i-1][j-r][0]$ 进行转移即可。$g[i][j][z]$ 的扩展类似。只需要在计算剩余石子个数时，额外考虑 a_i 的影响。

总结

本题改编自 Codeforces 1007E，是一道较难的动态规划题目。选手不难想到本题的做法与动态规划有关，但要得到较高的分数，必须根据题目的性质，设计合适的 DP 状态。本题思维难度较高，同时代码实现上略有细节，难度适中，可以较好地考察选手的动态规划能力。出（搬）题人相信，这道良心的题目，可以给拼搏于 IOI 之路上的你，提供一个有力的援助。