题目描述

考虑一种有理有据的议会席位分配方法。

假设 n 个政党的最终票分别为 V_i，分配 m 个席位，规则如下：

- 初始议会为空，每个政党有 0 个席位。
- 对于每个政党，计算 $Q_i = \frac{V_i}{S_i+1}$ 最大的政党 p, 其中 S_i 表示第 i 个政党已经获得的席位。若有多个政党 Q_i 相同，取编号最小的那个。
- 分配给政党 p 一个席位，重复上步，直到分配完 m 个席位。

现在投票还在进行，每个政党当前得票数为 v_i，还有 n 堆群众未进行投票，其中第 i 堆群众人数为 a_i，且其中每个人可以选择投给第 i 或 $i+1$ 个政党，不能弃权。特殊的，第 n 堆群众可以投给第 n 和 1 个政党。

请计算第 1 个政党最多能获得多少个席位。

数据范围

对于所有数据：$2 \le n \le 10^5, m \le 10^9, v_i, a_i \le 10^9, v_i > 0$。

Subtask 1 (10 pts)：$n, m \le 5, v_i, a_i \le 5$。

Subtask 2 (10 pts)：$n \le 50, m \le 100, v_i, a_i \le 250$。

Subtask 3 (20 pts)：$n \le 50, m \le 500$。

Subtask 4 (20 pts)：$n \le 1000, m \le 1000$。

Subtask 5 (20 pts)：$n \le 10^4$。

Subtask 6 (20 pts)：无特殊限制。

时间限制：1s。空间限制：256MB。

题解

本题是改编题。

Subtask1

dfs 即可。

Subtask2

考虑二分答案，对于第一个政党是否能取 S_1 个席位，要计算 S_i 表示在第一个政党取得第 S_1 票之前，能获得的票数。

合法即满足：$\sum_{i=1}^{n} S_i \le m$。

据题意有：$\frac{V_1}{S_1} \ge \frac{V_i}{S_i+1}$，要求的 S_i 就是满足条件的最小的 S_i。

移一下项可得：$S_i = \lceil \frac{V_i S_1}{V_1} \rceil - 1$。

然后使用 dp 来分配 a_i。

记 $f_{i,j}$ 表示，考虑完前 i 个政党，且第 i 个群体中有 j 个人投给了第 i 个政党，剩下的投给了第 $i+1$ 个政党的情况下，前 i 个政党所得最少的席位 S_i 之和，转移如下：

$$f_{i,j}=\min_{k\le a_{i-1}} f_{i-1,k}+\lceil\frac{(v_i+a_{i-1}-k+j)S_1}{V_1}\rceil-1$$

最后判断 $f_{n,0}+S_1$ 是否小于等于 m，时间复杂度 $O(na^2\log m)$。

Subtask3

考虑优化上述算法，容易发现 dp 的状态很大但值很小，考虑交换这两个东西。

记 $f_{i,j}$ 表示考虑完前 i 个政党，且前 i 个政党一共得了 j 个席位的情况下，第 i 个群体中最多有多少人投给了第 i 个政党，转移如下：

$$f_{i,j}=\max_{k\le j,f_{i-1,k}\le a_{i-1}}(\lfloor\frac{(j-k+1)V_1}{S_1}\rfloor-v_i-a_{i-1}+f_{i-1,k})$$

时间复杂度 $O(nm^2\log m)$。

Subtask4

观察一下上述式子：$\sum S_i=\sum\lceil\frac{V_iS_1}{V_1}\rceil-1$。

发现向上取整里面的分子之和

$$S_1\sum V_i=S_1(\sum v_i+\sum a_i)$$

是固定的。

于是考虑贪心，依次决策每个政党，让 V_iS_1 尽可能靠近 V_1 的倍数，毕竟向上取整了之后要 -1。

设 $r_i=V_iS_1 \bmod V_1$，贪心就是让 r_i 尽量大（特别的，0 视为最大）。

但是直接枚举 r_i 的话跟值域相关，但是 $\lceil\frac{V_iS_1}{V_1}\rceil$ 的取值是 $O(m)$ 的，可以枚举这个取值，贪心地取当前最大的余数从而找到最大值。

从 2 到 n 依次贪心即可，时间复杂度 $O(nm\log m)$。

关于贪心的证明：

首先分子之和是固定的，所以让所有政党的票数模 V_1 之和最大（0 视为 V_1），式子的值是取到最小的。

那么考虑最优的方案，如果没有使第二个政党取到最大的 r_2，只是取到了 r_2'，那么可以在这个最优方案的基础上，只调整第二个群众的给票方案，让第二个政党的余数取到 r_2.

那么又根据分子之和不变，调整之后的第三个政党的余数 r'_3 要么 $r'_3 = r'_3 - (r_2 - r'_2)$，要么 $r'_3 = r'_3 - (r_2 - r'_2) + V_1$，可见调整之后的余数之和不会更劣。

那么这样顺次考虑下去，最优方案都能调整到我们的贪心结果上。

Subtask5

上面的贪心瓶颈在于找最大余数那里，可以转化成这么一个问题：

求 $kx + b \bmod p (0 \le x \le n)$ 的最大值。

考虑二分答案 mid，转换成计数问题：

$$\sum_{i=0}^{n} [(ki + b) \bmod p < mid]$$

$$= \sum_{i=0}^{n} [\lfloor \frac{ki + b - mid}{p} \rfloor + 1 = \lfloor \frac{ki + b}{p} \rfloor]$$

$$= \sum_{i=0}^{n} \lfloor \frac{ki + b}{p} \rfloor - \sum_{i=0}^{n} \lfloor \frac{ki + b - mid}{p} \rfloor$$

这个可以用类欧 $O(\log n)$ 做。

m 和值域视为同阶，那么就做到了 $O(n \log m^3)$ 的复杂度。

Subtask6

二分的 log 是可以去掉的！可以直接求最值！

下面设：

$$f(a, b, c, n) = \max_{i=0}^{n} \{(ai + b) \bmod c\}$$

$$g(a, b, c, n) = \min_{i=0}^{n} \{(ai + b) \bmod c\}$$

其中，$a, b, n \ge 0, c > 0$。

- 简单的边界情况：

$$f(a, b, c, 0) = g(a, b, c, 0) = f(0, b, c, n) = g(0, b, c, n) = b \bmod c$$

- 简单但是极其重要的第一个递归式：

$$f(a, b, c, n) = f(a \bmod c, b \bmod c, c, n)$$

$$g(a, b, c, n) = g(a \bmod c, b \bmod c, c, n)$$

接下来是阴间的部分 (考虑交换 a, c):

$$\begin{aligned} f(a,b,c,n) &= \max_{i=0}^{n}\{(ai+b) \bmod c\} \\ &= \max_{j=0}^{\lfloor (an+b)/c \rfloor}\{\lfloor \frac{cj+c-b-1}{a} \rfloor \cdot a + b - cj\} \\ &= c-1-\min_{j=0}^{\lfloor (an+b)/c \rfloor}\{cj+c-b-1-\lfloor \frac{cj+c-b-1}{a} \rfloor \cdot a\} \\ &= c-1-g(c,c-b-1,a,\lfloor (an+b)/c \rfloor) \end{aligned}$$

然而上面的推导有一个重大 (yinjian) 的漏洞，枚举每一个 j 时，对应到的所有 i 不一定能够都满足 $0 \le i \le n$。

事实上，这种情况会发生当且仅当 $j=0$ 或 $j=\lfloor (an+b)/c \rfloor$ 的时候，而在 f 函数求解的时候，唯一可能出现问题的，恰好是 $j=\lfloor (an+b)/c \rfloor$。

因此我们就可以得到 $f \to g$ 的最终的转移式：

$$f(a,b,c,n) = \max((an+b) \bmod c, c-1-g(c,c-b-1,a,\lfloor (an+b)/c \rfloor - 1))$$

我们开辟了一条新路，那么 g 的反过来推导变得容易了。

$$\begin{aligned} g(a,b,c,n) &= \min_{i=0}^{n}\{(ai+b) \bmod c\} \\ &= \min_{j=0}^{\lfloor (an+b)/c \rfloor}\{\lfloor \frac{cj+a-b-1}{a} \rfloor \cdot a + b - cj\} \\ &= a-1-\max_{j=0}^{\lfloor (an+b)/c \rfloor}\{cj-b-1-\lfloor \frac{cj-b-1}{a} \rfloor \cdot a\} \\ &= a-1-f(c,-b-1,a,\lfloor (an+b)/c \rfloor) \end{aligned}$$

但是同样要注意交换求和号的时候会在首尾两端不满足一一对应关系！这次是 $j=0$ 会出问题，于是更正如下：

$$g(a,b,c,n) = \min(b, a-1-\max_{j=1}^{\lfloor (an+b)/c \rfloor}\{cj-b-1-\lfloor \frac{cj-b-1}{a} \rfloor \cdot a\}$$

$$g(a,b,c,n) = \min(b, a-1-\max_{j=0}^{\lfloor (an+b)/c \rfloor - 1}\{cj+c-b-1-\lfloor \frac{cj+c-b-1}{a} \rfloor \cdot a\}$$

因此我们就可以得到 $g \to f$ 的一个转移式：

$$g(a,b,c,n) = \min(b, a-1-f(c,c-b-1,a,\lfloor (an+b)/c \rfloor - 1))$$

利用这两个递归式，f 和 g 的计算都可以做到与辗转相除计算 $\gcd(a,c)$ 级别相同的复杂度。所以最终时间复杂度 $O(n \log m^2)$。

其他

关于卡常，外层的二分可以算一下上下界，是 $O(n)$ 级别的。

关于数据，在值域或者 n 很小的时候基本随不出强的数据 QAQ，随便写个假贪心都能过。。。只能手工造了。