《代金券》解题报告

周雨扬

北京大学

2021 年 5 月 16 日

题目大意

有 n 道菜品，第 i 道菜品花费 a_i. 可以使用价值 1 的消费券，初始时没有消费券。

如果在一道花费 a 的菜品花费 $b(b \leq a)$ 张消费券，则实际需要花 $a-b$ 元，并在用餐结束后获得 $\lfloor \frac{a-b}{c} \rfloor$ 张消费券。

Q 次对序列 a_i 修改，每次修改结束后询问按照顺序品尝 n 个菜品的最小消费。

$1 \leq n, Q \leq 3 \times 10^5, 1 \leq c \leq 10^9, 1 \leq a_i \leq 10^{12}$

算法 0

相信复杂度和 a_i 相关的做法大家都会。

$c = 1, c = 2$ 可能答案有特殊性质，出题人没有推。

算法 1

问题等价于：在购买过程中至多可以花出去多少张优惠券。

算法 1

问题等价于：在购买过程中至多可以花出去多少张优惠券。

将每一个菜价格分为 $a_i \bmod c$，$a_i - a_i \bmod c$ 两部分。

此时不难发现如果在这道菜品上使用不超过 $a_i \bmod c$ 张优惠券，其不会影响购买菜品获得的优惠券张数。

算法 1

定理 1：总存在一个最优解，使得存在一个菜品序列的后缀，使得：

- 除去后缀的第一个菜品，其余所有菜品全部使用优惠券支付。
- 所有不位于后缀的菜品，其使用的优惠券张数不超过 $a_i \bmod c$，且在满足条件的情况下使用张数尽量多。

算法 1

定理 1：总存在一个最优解，使得存在一个菜品序列的后缀，使得：

- 除去后缀的第一个菜品，其余所有菜品全部使用优惠券支付。
- 所有不位于后缀的菜品，其使用的优惠券张数不超过 $a_i \bmod c$，且在满足条件的情况下使用张数尽量多。

证明：通过调整法，如果不满足条件，则总可以通过将一张前面优惠券的使用时刻向后移动，得到一个严格不会变劣的解。

调整细节这里略去。

算法 1

此时我们对于每个菜品得到了两个参数：$b_i = a_i \bmod c$ ，$d_i = \frac{a_i - b_i}{c}$。

假设 $S_0 = 0, S_1 = -b_1, \forall i \geq 2, S_i = S_{i-1} + d_{i-1} - b_i$。

算法 1

此时我们对于每个菜品得到了两个参数：$b_i = a_i \bmod c$ ，$d_i = \frac{a_i - b_i}{c}$。

假设 $S_0 = 0, S_1 = -b_1, \forall i \geq 2, S_i = S_{i-1} + d_{i-1} - b_i$。

定理 2：假设在定理 1 中后缀的第一个元素为 $x(1 \leq x \leq n+1)$，则 $\min\limits_{0 \leq i \leq n} \{S_i\} < \min\limits_{x \leq i \leq n} \{S_i\}$

算法 1

此时我们对于每个菜品得到了两个参数：$b_i = a_i \bmod c$ ，$d_i = \frac{a_i - b_i}{c}$。

假设 $S_0 = 0, S_1 = -b_1, \forall i \geq 2, S_i = S_{i-1} + d_{i-1} - b_i$。

定理 2：假设在定理 1 中后缀的第一个元素为 $x(1 \leq x \leq n+1)$，则 $\min\limits_{0 \leq i \leq n} \{S_i\} < \min\limits_{x \leq i \leq n} \{S_i\}$

证明 2: 如果不满足，剩余的消费券不够用，无法让后面所有菜品全部使用消费券支付。

算法 1

此时我们可以枚举后缀的第一个元素 i，并且二分在第 i 道菜品花费的消费券张数。

此时不难发现，我们总是会在满足后面均用消费券支付情况下, 尽量增加当前菜品消费券使用张数, 因而可以使用二分求解, 也可以直接计算。

时间复杂度 $O(nQ\log a_i)$ 或 $O(nQ)$.

算法 2

性质：取到最优解的后缀第一个元素 i 一定满足：

- 在第 $i \sim n$ 道菜品全部使用代金券支付的时候代金券不够用。
- 在第 $i+1 \sim n$ 道菜品全部使用代金券支付的时候代金券够用。

算法 2

性质：取到最优解的后缀第一个元素 i 一定满足：

- 在第 $i \sim n$ 道菜品全部使用代金券支付的时候代金券不够用。
- 在第 $i+1 \sim n$ 道菜品全部使用代金券支付的时候代金券够用。

第二个条件可由定理 1 推出。如果第一个条件不满足，则同可以将后缀端点左移得到一个不劣的解。

算法 2

此时我们可以直接二分求出满足条件的唯一一个 i。

相关的信息均可以使用线段树维护。

时间复杂度 $O((Q+n)\log n)$