《优惠购物》解题报告 by 沈吉颁

题目描述

小 C 要购买 n 个物品，这些物品有前置关系，必须**依次**购买（即在购买了第 i 个后才能购买第 $i+1$ 个）。

他初始有 m 张优惠券和无穷多个金币。每个物品有两个属性，价格 a_i 和优惠券的使用上限 $b_i(0 \leq b_i \leq a_i)$。

购买一个物品的流程如下：

- 选择使用 $x(0 \leq x \leq b_i)$ 张优惠券，付出 $a_i - x$ 个金币和 x 张优惠券。
- 购买完后可得到 $\lfloor \frac{a_i - x}{c} \rfloor$ 张优惠券（即一次购买中，每付出 c 个金币可以得到一张优惠券，c 为给定常数）

小 C 想求出最少花费多少个金币能购买全部物品。

数据范围

对于所有数据，$1 \leq \sum n \leq 10^6, 1 \leq m, a_i, b_i \leq 10^9, 2 \leq c \leq 10^9$。

Subtask 1 (5 pts)：$1 \leq T \leq 5, 1 \leq n \leq 10, 1 \leq m, \sum a_i, \sum b_i \leq 10$

$\text{Subtask 2 (10 pts)}$：$a_i = b_i$

$\text{Subtask 3 (10 pts)}$：$1 \leq \sum n \leq 500, 1 \leq \sum m, \sum a_i, \sum b_i \leq 500$

$\text{Subtask 4 (10 pts)}$：$1 \leq \sum n \leq 6000, 1 \leq \sum m, \sum a_i, \sum b_i \leq 6000$

$\text{Subtask 5 (10 pts)}$：$1 \leq \sum n \leq 6000$

$\text{Subtask 6 (15 pts)}$：$1 \leq \sum n \leq 2 \times 10^5, 2 \leq c \leq 20$

$\text{Subtask 7 (10 pts)}$：$1 \leq \sum n \leq 1 \times 10^6, 2 \leq c \leq 20$

$\text{Subtask 8 (15 pts)}$：$1 \leq \sum n \leq 2 \times 10^5$

$\text{Subtask 9 (15 pts)}$：$1 \leq \sum n \leq 1 \times 10^6$

时间限制：1s

空间限制：2048MB

解题过程

算法 1（暴力）

设 $f_{i,j}$ 表示购买了前 i 个物品，当前**总共**获得了 j 张优惠券（算上用过的），最大化当前最多能用掉的优惠券数量。

由 $f_{i,j}$ 可以计算出当前有的优惠券数量为 $j+m-f_{i,j}$。枚举这次使用的优惠券数量 x，则有转移：

$$f_{i,j}+x\to f_{i+1,j+\lfloor\frac{a_i-x}{c}\rfloor}$$

容易发现 dp 的复杂度为背包，复杂度即为 $O((\sum a_i)^2)$。

算法 2

设在第 i 次购买时使用了 x_i 张优惠券，考虑这种方案合法的条件：

设 s_i 表示第 i 次操作后的剩余优惠券数量，$s_i=m+\sum_{j=1}^{i}(-x_i+\lfloor\frac{a_i-x_i}{c}\rfloor)$. 若对于所有 i 都满足 $s_{i-1}-x_i\geq 0$，则方案成立。

考虑一开始一张优惠券也不用，此时最后会多出若干张优惠券。

如果在前面用了若干张优惠券，那么在过程中总获得的优惠券数量（包括中间用了的）会减少。

考虑贪心，使得减少的【总获得的优惠券数量】最少。

设 【总获得的优惠券数量】 为 S，考虑在一个物品上不断增加花的优惠券（减小价格），减少的数量可分为三个阶段：

- 花掉 $[0,a_i\bmod c]$ 的优惠券，这个阶段 S 不变。
- 上个阶段把价格变成了 c 的倍数，然后每次花掉 c 的优惠券，并且 S 减 1。
- 最后花一次 $[0,c-1]$ 的优惠券并且 S 减 1，也可能不做操作。

从操作性价比考虑，第一个阶段减少为 0，优于第二个阶段，而第二个阶段每花掉 c 个优惠券才减少一个，优于第三个阶段。

因此可以贪心，先考虑第一阶段，对于物品 i 在 $a_i\bmod c$ 内尽量使用优惠券。这样会把每个价格尽量减成 c 的倍数，如果这一步中并没有把价格减小到 c 的倍数那说明之后也不可能再减了。

然后考虑第二阶段。可以倒着贪心，设 s_i 表示第 i 次操作后的剩余优惠券数量，x_i 表示当前方案中第 i 个使用了 x_i 张优惠券。

x_i 每增加 c，则 $\forall j \geq i, s_j$ 会减少 $c+1$，由于需要保证 $s_{i-1}-x_i \geq 0$，则可以算出第 i 个可以减小的最多数量为 $\min(\lfloor \frac{b_i-x_i}{c} \rfloor, \lfloor \frac{s_{i-1}-x_i}{c} \rfloor, \min_{[j>i]} \lfloor \frac{s_{j-1}-x_j}{c+1} \rfloor) \times c$，从后往前做并记录一个后缀 $\min$ 即可。

对于第三阶段，仍然先做性价比更高的操作，也就是优先做在某个位置上用 $c-1$ 张优惠券的操作，然后是用 $c-2, c-3$.. 的操作，容易发现这样能使被操作的位置最少。

可以枚举 $j=c-1, c-2, ..., 1$，然后对于每个位置，判定再某个位置上是否能做用 j 个优惠券这个操作（也就是 s 操作后是否满足），式子和上面那部分的差不多，不难发现从后往前做，记录一个后缀 $\min$ 即可判定。

时间复杂度 $O(nc)$，瓶颈在第三部分，可以通过 Subtask 6,7。

算法 3

算法 2 是原来的 std，但吴畅同学在验题时提出了一个不一样的 $O(n \log n)$ 算法；后来我发现算法 2 也能优化到 $O(n \log n)$。

考虑优化算法 2 第三部分中找某个位置的过程。

设 $t_i = s_{i-1}-x_i, del_i = \min(b_i-x_i, t_i, \min_{[j>i]}(t_j-1))$，那上述的贪心过程相当于每次取 del_i 最大的位置，将 x_i 加上 del_i，并做修改 $t_i \to t_i - del_i, t_j \to t_j - del_i - 1 (j>i)$。直接暴力做就是 $O(n^2)$ 的。

考虑 del_i 受到的两种限制，其中 $\min(t_i, \min_{[j>i]}(t_j-1))$ 随 i 的减小而减小，也就是单调递增的。而 b_i-x_i 不具有单调性。

将所有 i 按照 b_i-x_i 排序，并依次加入。每次先向一个【可行集合】中加入若干个 b_i-x_i 相等的位置，设下一个更小的 b_i-t_i 为 lim，我们想要取掉所有 del_i 大于 lim 的位置。

由于 $\min(t_i, \min_{[j>i]}(t_j-1))$ 单调递增，则此时能取的 i 为【可行集合】和一段后缀的交集，不难发现取可行集合中最大的一定最优。于是用 set/大根堆 维护可行集合，每次取出最大的元素，判断 del_i 是否满足，如果满足就修改 t，并从可行集合中删去该元素。

t 可以用区间加，区间查 $\min$ 的线段树维护，总时间复杂度 $O(n \log n)$，可以通过所有 Subtask。

参考资料

无