《分道扬镳》题解报告

题目大意

时间限制：7s

空间限制：5MB

给定 n 个物品，体积依次为 $v_1, v_2, \ldots, v_n$，价值依次为 $w_1, w_2, \ldots, w_n$。

你需要从中选择一些物品，保证体积和不超过 m。你将会带走你选择的物品，你的收益是这些物品的价值和。输出你的收益的最大值。

特别地，有一个每次给定的系数 k，数据保证如下性质：

- 如果你可以先选定一个元素都在 $(0, 1]$ 之间的实数序列 $p_1, p_2, \ldots, p_n$。并对于所有物品，第 i 个物品的体积和价值同时乘以 p_i，即体积变为 $p_i v_i$，价值变为 $p_i w_i$。然后你再进行选择物品，那么你的收益最多为 r。
- 在原问题中，假设你的收益最多为 l，则有 $r - l \leq k$。

数据范围

对于所有数据，保证 $1 \leq n \leq 10^4, 1 \leq m \leq 10^9, 1 \leq k \leq 20, 1 \leq v_i \leq 5000, 1 \leq w_i \leq 10^{12}$。

子任务编号	$n \leq$	$m \leq$	$k \leq$	$v_i \leq$	$w_i \leq$	数据随机	分值
1	1000	10^5	20	1000	10^{12}	否	5
2	1000	10^9	20	1000	10^{12}	否	5
3	10^4	10^9	20	333	333	否	10
4	10^4	10^9	1	2000	10^{12}	否	20
5	10^4	10^9	5	2000	2000	是	15
6	10^4	10^9	20	2000	2000	是	15
7	10^4	10^9	20	5000	10^{12}	否	30

数据随机：对于每组数据，给定 n, m, k 后，在满足 $r - l \leq k$ 的所有的数据中等概率随机选一组。

题解过程

子任务 I & II

直接暴力背包即可，记录 $dp(i,x)$ 表示前 i 个物品，体积和为 x 可以带来的最大的价值和。

考虑直接做空间会炸，因此考虑滚动数组。至此时间复杂度 $O(nm)$，空间复杂度 $O(m)$。

但发现 **子任务 II** 所需的空间还是超了一倍。因此我们令 $s=\sum_{i=1}^{n}v_i$。此时如果 $s\le 2m$。那么我们就考虑不再对选的物品进行背包，而是不选的物品。此时空间就压缩到了原来的一半。时间复杂度 $O(n^2v)$，空间复杂度 $O(nv)$。

子任务 III

由于和最终正解和 **子任务 IV** 有关，放到后续介绍。

子任务 IV

考虑 r 如何快速计算，不难发现只需要将所有物品按照 $\frac{w_i}{v_i}$ 从大到小排序即可，然后直接依次取，如果取某个物品时，体积和超出了 m，就将当前物品的缩小到和前面所有物品的体积和刚好为 m，然后结束即可，不难发现这样构造的 r 是最优的。

此时如果 r 不是整数，那么直接有 $l=\lfloor r\rfloor$，否则需要判断 l 是 $r-1$ 还是 r。

不难发现我们可以尝试构造答案是否为 r。考虑在计算 r 中最后一个加入的物品，假设其 $\frac{w_i}{v_i}$ 值为 c，那么对于 $\frac{w_i}{v_i}$ 大于 c 的物品，我们显然必须选，然后对于 $\frac{w_i}{v_i}$ 为 c 的物品，我们即需要知道是否能选出一个子集，使得其体积和配上 $\frac{w_i}{v_i}$ 大于 c 的物品，刚好为 m。

即相当于给定一个长度为 n' 的序列 $v'_1,v'_2,\dots,v'_{n'}$，和一个非负整数 m'。你需要回答是否能找到一个 v' 的子集和为 m'。

具体做法由于和正解有关，放到后续介绍。

值得一提的是，由于常数问题。利用 bitset 优化暴力背包并通过大力卡常，也可以通过此部分，时间复杂度 $O(\frac{n^2v}{32})$，空间复杂度 $O(\frac{nv}{32})$。

子任务 V & VI

这个子任务并没有一个笔者已知的做法，但是一定程度上为了体现区分度，预留给一些随机化算法和调整算法，例如模拟退火。

这两个子任务也防止有正解被卡常的，以及放过去可能存在的 $O(nv\times \mathrm{poly}(k))$ 做法。

一种赛时出现的厉害做法

考虑先按 $\frac{w}{v}$ 从大到小取，直到再取体积会超出 m。假设此时体积和为 a，价值和为 b。考虑将取了的物品的体积和价值，变为原来的相反数，然后进行背包，背包出体积和为 $m-a$ 的物品。

考虑对所有物品随机打乱，那么就只需要在 $O(\sqrt{n}v)$ 范围内调整即可。

最终时间复杂度 $O(n\sqrt{n}v)$，空间复杂度 $O(\sqrt{n}v)$。

子任务 VII

为了方便介绍正解，我们如下依次介绍 **子任务 IV** 和 **子任务 III** 对应的做法。

本题最终做法非常依赖于 这篇论文 [1]，**子任务 IV** 和 **子任务 III** 对应的做法都出自这篇论文。最终做法某种意义上算是这篇论文中背包算法的一种扩展。

无价值背包

即给定一个长度为 n 的序列 $v_1,v_2,\dots,v_n$，和一个非负整数 m。你需要回答是否能找到一个 v 的子集和为 m。

下文中用 v 表示 $\max_{i=1}^{n}v_i$。

直接动态规划，记录 $f(i,x)$ 表示考虑了前 i 个物品，和为 x 是否可行，显然有转移式
$f(i,x)=f(i-1,x)\vee f(i-1,x-v_i)$。这样直接做，复杂度为 $O(n^2v)$。

原论文引出了一种 $O(nv)$ 的做法，形式化的来说，找到一个断点 p，使得 $\sum_{i=1}^{p}v_i$ 在 $[m-v,m+v]$ 之间。

然后考虑一边加入物品一边删除物品，当体积小于 m 时，尝试加入一个物品，否则尝试删除一个物品，记录 $g(i,x)$ 表示加入了前 i 个物品，体积和为 x，最多有前多少个物品可以回退，如果无解则值为 -1。

初始令 $g(p,\sum_{i=1}^{p}v_i)=p$，其他为 -1，随后不难发现有如下转移：

$\forall x \in [m-v, m+v], g(i-1,x) \to g(i,x)$

$\forall x \in [m-v, m], g(i-1,x) \to g(i,x+v_i)$

$\forall x \in [m, m+v], j \in [1, g(i,x)], j-1 \to g(i,x-v_j)$

但这样复杂度仍然是 $O(n^2v)$，实际上复杂度并没有获得优化，但是不难发现 j 实际上不需要枚举这么多，对于 $[1,g(i,x)]$ 中的所有值，实际上 $[1,g(i-1,x)]$ 已经在之前被枚举过了，因此只枚举 $(g(i-1,x),g(i,x)]$ 即可，即将第三条转移改为：

$\forall x \in [m, m+v], j \in (g(i-1,x), g(i,x)], j-1 \to g(i,x-v_i)$

不难发现，对于同一个 x，$g(i,x)$ 关于 i 单调，且 j 的枚举量为 $\sum_{i=p+1}^{n} g(i,x)-g(i-1,x)$，其实即为 $g(n,x)-g(p,x)$。不难发现 $g(n,x)$ 是 $O(n)$ 的。因此时间复杂度 $O(nv)$，滚动数组后空间复杂度 $O(v)$。

该算法即 **子任务 IV** 所需要的做法。

有价值背包

下文中用 v 表示 $\max_{i=1}^{n} v_i$，w 表示 $\max_{i=1}^{n} w_i$。

即题目给定的形式，在原有的动态规划上加一维，记录 $g(i,x,y)$ 表示考虑了前 i 个物品，体积和为 x，价值和为 y，最多有前多少个物品可以回退。

根据原做法，显然 i 和 x 的取值分别为 $O(n)$ 和 $O(v)$，关键在于 y 要取多少。

考虑对于一个体积 x，如果将物品根据 $\frac{w}{v}$ 排序，然后从大到小取，直到再取下一个体积超出 x，我们将这样带来的价值和作为下界 $\alpha(x)$，而取了下一个物品后，我们将其价值和作为上界 $\beta(x)$。显然不可能超出该上界，低于下界也的状态也没有意义，而 $\beta(x)-\alpha(x)$ 又只差一个物品的价值，因此 y 能取值的个数为 $O(w)$，因此该算法时间复杂度 $O(nvw)$。滚动数组后空间复杂度 $O(vw)$。

该算法即 **子任务 III** 所需要的做法。

更好的算法

我们仍然考虑该状态，$dp(i,x,y)$ 依然表示考虑了前 i 个物品，体积和为 x，价值和为 y，最多有前多少个物品可以回退。

在此我们将调整的范围从 $[m-v,m+v]$ 改为 $[m-2v,m]$，具体原因会在后文解释，而转移中 x 的枚举范围也有略微区别，我们不能再只枚举一半，即原本加入物品只在体积和 $[m-v,m]$ 时加，删除物品只在体积和 $[m,m+v]$ 时删，而现在所有的都需要枚举。具体来说，更改如下：

$\forall x \in [m-2v, m], y \in [\alpha(x),\beta(x)], dp(i-1,x) \to dp(i,x)$

$\forall x \in [m-2v, m-v_i], y \in [\alpha(x),\beta(x)], dp(i-1,x) \to dp(i,x+v_i)$

$\forall x \in [m-2v, m], y \in [\alpha(x),\beta(x)], j \in (dp(i-1,x), dp(i,x)] \cap \{z | v_z \le x-m+2v\}, j-1 \to dp(i,x-v_j)$

现在我们的目标即要给出一种 $\alpha(x)$ 和 $\beta(x)$ 的构造，并满足其差为 $O(k)$。

为了方便说明，我们加入无限个体积为 1，价值为 0 的物品。

考虑建立直角坐标系，两轴分别为体积和价值，这样每个物品可以看作一个向量，我们将物品按照 $\frac{w}{v}$ 排序，并从原点开始，从大到小首尾相连拼接，不难发现会得到一个凸包，我们将这个函数称之为 $f(x)$。此时也不难发现 $r=f(m)$。

考虑背包的过程，拿出在当前状态下，除了确认不选的所有物品，将物品按照 $\frac{w}{v}$ 排序，并从原点开始，从大到小首尾相连拼接，也会得到一个凸包。我们将这个函数称之为 $g(x)$。

我们可能还需要详细的解释，什么叫确认不选呢？假设其在可能删除的前半部分，如果确认了将其删除，则我们称其为确认不选。如果在可能加入的后半部分，如果确认了不加入，则我们称之为确认不选。这个 $g(x)$ 只是对于某一个 dp 状态来说的，因为一个 dp 状态本质上就是一种物品的选择。而从初始状态，经过 dp 转移，变换到结果状态，实际上也可以理解为 g 函数的一种改变，并最后取 $g(m)$。

随后我们引入两个结论：

- 对于任意时刻的 g，都满足：对于 $x_1<x_2$，有 $f(x_1)-g(x_1) \le f(x_2)-g(x_2)$，即 $f(x)-g(x)$ 单调不降。
- 对于任意 x，$f(x)-g(x)$ 在转移的过程中，单调不降。

证明如下：

- 由于 g 实际上是从 f 删除一些线段得到。对于 f 和 g 的转折点，即不可导的端点来说是平凡的，剩余因此实际上固定 x 之后，对体积轴过 x 的向量对应的物品来说，f 对应的 $\frac{w}{v}$ 不会比 g 更小。即 $f'(x) \ge g'(x)$。因此 $f'(x)-g'(x) \ge 0$，因此 $f(x)-g(x)$ 单调不降。
- 不难发现在 g 的变换过程中，删除一个向量时，代替这个位置的向量实际上是后面的向量，其 $\frac{w}{v}$ 只会更小，因此固定 x 后，$g(x)$ 在转移的过程中单调不增，又由于 $f(x)$ 是定值，因此 $f(x)-g(x)$ 在转移的过程中，单调不降。

根据上述两条结论，我们可以证明 $\beta(x)$ 直接取 $f(x)$，而 $\alpha(x)$ 只需要取 $\beta(x)-k$ 即可。上界显然，而对于下界，若在一时刻的某一体积，其价值和离上界差了 k 以上，由于 $f(x)-g(x)$ 关于体积，时间都单调，在最终结果的 $f(m)-g(m)$ 中，也一定会大于 k，因此并不需要在意。

至此所有 $\beta(x)-\alpha(x)$ 的界被限紧到了 $O(k)$，也可以轻松通过此题。

简单的做法

后来笔者发现有一种更为简单的证法，且直接将调整范围用作 $[m-v,m+v]$ 也可以被证明是正确的。

考虑动态规划的过程，我们令 $a_i=\sum_{j=1}^{i}v_j$。对于一个当前所在的状态 (x,y)，即体积和为 x，价值和为 y：

- 如果我们加入一个物品 (v_i,w_i)，那么由于此时后续的物品全都没有被加入，此时一定有 $x\le a_{i-1}$。
- 如果我们删除一个物品 (v_i,w_i)，那么由于此时前面的物品全都没有被删除，此时一定有 $x\ge a_i$。

又由于我们提前将物品排过序，借用上次使用的 $f(x)$ 函数，由于 f 是凸的，因此固定一个正实数 d，对于任意非负实数 $x_1<x_2$，$f(x_1+d)-f(x_1)\ge f(x_2+d)-f(x_2)$。

回到我们的问题，加入物品时，由于我们有 $f(a_{i-1}+v_i)-f(a_{i-1})=w_i$，所以易得：

- $f(x+v_i)-(y+w_i)=(f(x+v_i)-f(x))-w_i+f(x)-y\ge((f(a_{i-1}+v_i)-f(a_{i-1}))-w_i)+f(x)-y=f(x)-y$

即 $f(x+v_i)-(y+w_i)\ge f(x)-y$，对于删除物品同理，因此在调整的过程中 $f(x)-y$ 的值只会越来越大，因此超出 k 的部分无意义，于是所有 $\beta(x)-\alpha(x)$ 的界被限紧到了 $O(k)$。

时间复杂度：$O(nvk)$

空间复杂度：$O(n+vk)$

特别感谢

特别感谢 IOI2024 国家集训队队员 **宋佳兴** 对本题提供常数上的指导。

感谢 IOI2025 国家集训队队员 **叶李蹊，翟鹏昊，陈诺，赵海鲲** 和 IOI2024 国家集训队队员 **黄洛天，咸浩哲** 参与本题的验题工作。

参考资料

[1] Linear Time Algorithms for Knapsack Problems with Bounded Weights - David Pisinger