动态规划进阶 II

275307894a

2025 年 7 月 8 日

Content

叠甲

- PPT 的名字虽然叫“动态规划进阶 II”，但是实际上只解决一类问题：大体积背包问题。
- 这也是我的集训队论文内容。这个 PPT 就是在营员交流 PPT 的基础上，增加了更多的基础内容、例题与细节，~~降低了语速得到的~~。
- 因此可能你的 OI 生涯接下来都不会碰到这类题目，但是这确实是非常有意思的一小部分。

背包问题：Recap

- 有 n 种物品，第 i 种物品体积为 w_i，价值为 v_i，共有 d_i 个。你需要找到一个选取物品装入背包的方案，使得总体积不超过背包的体积上限 M，且价值和最大。所有物品体积、价值均非负。

形式化定义

- 形式化的，你需要找到一组 x，满足

$$\forall i, x_i \in [0, d_i] \cap \mathbb{N}$$

$$\sum_{i=1}^{n} w_i x_i \leq M$$

- 目标是最大化

$$\sum_{i=1}^{n} v_i x_i$$

一般的解决方法

- 设 f_i 表示选取了体积总和为 i 时的最大价值和。加入一个物品 (w, v) 时，有转移方程

$$f_i = \max(f_i, f_{i-w} + v)$$

- 时间复杂度 $O(nM)$，空间复杂度 $O(M)$。

一般的解决方法

- 设 f_i 表示选取了体积总和为 i 时的最大价值和。加入一个物品 (w, v) 时，有转移方程

$$f_i = \max(f_i, f_{i-w} + v)$$

- 时间复杂度 $O(nM)$，空间复杂度 $O(M)$。
- 在 M 较大，但单个物品体积较小时无法接受。我们希望复杂度只与 $\log M$ 有关或和 M 无关。

洛谷 P2371 墨墨的等式

- 考虑等式 $\sum_{i=1}^{n} a_i x_i = b$，其中 a_i 给定，x 为未知数。求有多少 $b \in [l, r]$ 使得存在一组解。
- $n \leq 12, a_i \leq 5 \times 10^5, 0 \leq l \leq r \leq 10^{12}$

洛谷 P2371 墨墨的等式

- 核心观察：若对于某个 b 有解，则 $b + a_i$ 也有解，其中 i 可以是任意的。
- 也就是说，我们只需要对于每个 j，求出最小的 k 满足 $ka_1 + j$ 有解即可。
- 如果对于每对 i, j 从 i 向 $(i + a_j) \bmod a_1$ 连边权为 a_j 的边，则相当于一个最短路问题。
- 时间复杂度 $O(nA \log A)$

同余最短路，你还在最短路?

- 同余最短路建立的图实际上是有特殊性质的，这使得我们可以不用 Dijkstra 去求解。
- 首先 a 的顺序是没有关系的，所以可以按照 a 从小到大的顺序加入，每加入一次更新一次最短路，选择 a_1 作为模数。
- 在加入 a_j 时让 i 指向 $(i + a_j) \bmod a_1$，这会形成若干个环，在环上更新最短路只需要以某个方向遍历这个环两次即可。
- $O(nA)$

ARC084D Small Multiple

- 给定 n，求 n 的所有倍数中，数字和最小的一个是多少。
- $n \leq 10^5$

ARC084D Small Multiple

- 考虑一个数字是怎么诞生的：$\times 10$ 和 $+1$，其中 $+1$ 的次数就是数位和（理论上不能连续加超过 9 次，但是在这道题里不会更优）。
- 可以根据这个建立同余最短路，使用 01BFS 即可做到 $O(n)$。

AGC057D Sum Avoidance

- 给定一个正整数 S，称一个正整数集合 A 是好的，当且仅当它满足以下条件：
  - A 中元素在 $[1, S)$ 之间
  - 不能用 A 中元素多次相加得到 S
- 考虑所有好的集合中元素数量最大且字典序最小的集合 A，多次询问，求集合 A 从小到大排序后的第 k 项，或报告集合大小小于 k。
- T 组询问，$1 \le T \le 1000$，$1 \le S \le 10^{18}$。

AGC057D Sum Avoidance

- 显然集合大小为 $\lfloor\frac{S-1}{2}\rfloor$，一个构造是取所有 $>\lfloor\frac{S}{2}\rfloor$ 的数。并且我们发现 x 与 $S-x$ 一定只能选且必须选一个。
- 首先找到最小的 x 满足 $x\nmid S$，则 x 一定在答案中，并且 x 的所有倍数也都在。
- 注意到 x 很小，大概是一个 $\log S$ 级别的数。因此我们总共加入的基数不会超过 $\log S$ 个。利用同余最短路的思想，每次找到最小的一个可以加入的数加入进去即可。最后统计答案可以二分一下然后利用同余最短路的结果去算。时间复杂度 $O(T\log^3 S)$。

约定

- w_i, v_i, d_i, n, M 均与上述意义相同。
- 记 $W = \max_{i=1}^{n} w_i$, $V = \max_{i=1}^{n} v_i$。
- 记 $sum(S) = \sum_{i \in S} i$，其中 S 为可重集合。
- 所有物品均已按照性价比 $p_i = \frac{v_i}{w_i}$ 从大到小排序，且所有物品的体积价值均为正。
- $\sum_{i=1}^{n} w_i d_i > M$

$(\min,+)$ 卷积

- 有长度为 n 的序列 A 与长度为 m 的序列 B，定义其 $(\min,+)$ 卷积 $A\times B$ 的结果 C 为

$$C_i=\min_{j+k=i} A_j+B_k$$

- 其中 $0\le i\le n+m-2$。

获得图灵奖的办法

- 假设 $n = m$，学术界不存在 $O(n^{2-\delta})$ 计算 $(\min, +)$ 卷积的算法，因此在 OI 中，我们认为 $(\min, +)$ 卷积的复杂度是 $O(n^2)$ 的。
- 01 背包、完全背包、$(\min, +)$ 卷积三者可以相互规约[1]，而显然多重背包不弱于 01 背包，因此我们认为 OI 中的背包问题不存在低于平方的做法。
- 背包问题目前只存在伪多项式做法[2]，因此平方项中至少一项和值域有关。

[1] Isakovsky, https://www.cnblogs.com/isakovsky/p/17385641.html, 2023
[2] Wikipedia, https://wikipedia.org/wiki/knapsack_problem

广为人知题

- 给定 n 个物品，每个物品有体积 v_i 和价值 w_i。你需要选择体积之和不超过 V 的物品。
- 不同的是，这里的物品是可分割的。对于一个体积为 v，价值为 w 的物品，如果仅选择放入 V 体积，则会获得 $V \times \frac{v}{w}$ 的价值。
- $n \leq 10^6, v_i, w_i \leq 10^{18}$

广为人知题

- 因为这里体积是可分的，所以按照物品的性价比排序，贪心选择即可。
- 最后一个物品可以选一部分。

错误的贪心

- 对于所有物品按照性价比从大到小排序，贪心尽量选满。这个贪心在不可分的背包中并不具有正确性。
- 记 t_i 表示在这组贪心解中第 i 个物品选取了多少个。这个方案会存在一个位置 p，满足 $\forall i < p, t_i = d_i$，且 $\sum\limits_{i=p+1}^{n} w_i t_i < W$。也即，在 p 之前的所有物品全部选上，而物品 p 由于剩余体积不够未能选完，之后剩余体积就小于 W。

引理 1

引理

对于一个大小为 n 的非空可重集合 S，其存在一个非空可重子集 S'，满足

$$sum(S') \bmod n = 0$$

证明.

设 $S = \{P_1, P_2, \ldots, P_n\}$，计算 $sum_i = (\sum_{j=1}^{i} P_j) \bmod n$，特别地，$sum_0 = 0$。根据抽屉原理，一定存在一对 $l < r, l, r \in [0, n]$ 满足 $sum_l = sum_r$，则区间 $[l+1, r]$ 内的数即为要求的子集。 □

定理 1

基于这个引理，有如下定理：

定理

存在一个最优选取方案 x，满足 $\forall i, x_i \geq t_i - \frac{W^2}{w_i}$。

定理 1

证明.

反证，考虑最后一个不满足这个条件的位置 q，显然有 $q \le p$。因为 $< p$ 的所有物品均顶到选取上界，所以所有 $x_i > t_i$ 的位置均 $\ge p$，也即均 $> q$，说明所有相比于贪心方案多选取的物品性价比均不高于 q。

记所有多选取的物品体积集合为 S。根据 $|S|$ 分类讨论：

- 若 $|S| < W$，则新选取的物品体积和 $\le W(W-1) \le (t_q - x_q - 1)w_q$，可以多选取一个 q 物品。
- 否则，根据引理 1，S 存在一个大小不超过 w_q 的非空子集满足总和 sum 为 w_q 的倍数，则去掉这个子集并多选取 $\frac{sum}{w_q}$ 个物品 q 不会改变总体积，但是提升了总性价比。

$\square$

- 有了定理 1，我们就可以预先运行一遍贪心算法，并将每个物品选取的下界在总体积中移除。这样，剩余的体积不会超过 $W + \sum_{i=1}^{n} w_i \times \frac{W^2}{w_i} = (nW+1)W$，也即 M 被缩减到 $O(nW^2)$ 的范围。然后直接运行朴素的动态规划算法可以得到一个 $O(n^2W^2)$ 的算法。
- 对于 V 较小的情况，也可以进行类似的分析，将每种物品需要决策的物品个数控制在 $O(V^2)$ 范围内。

ARC096F Sweet Alchemy

- 有 n 个物品和 x 个特殊材料，制作第 i 个物品需要 m_i 个特殊材料。给出一个整数 d，对于每个 i $(2 \le i \le n)$ 给定 p_i $(1 \le p_i < i)$，设在材料充足的情况下制作第 i 个物品的个数为 c_i，需满足 $c_{p_i} \le c_i \le c_{p_i} + d$ 。最大化制作的物品数。
- $n \le 50$

ARC096F Sweet Alchemy

- 记 i 子树内 m_i 的和为 sum_i，子树大小为 siz_i，则可以看成每个点 i 对应一个体积为 sum_i，价值为 siz_i 的物品，最多选取 d 个。
- 设 f_i 表示价值为 i 最少需要多少体积，用上面的方法对于物品进行限制后即可做到 $O(n^4)$。

BalticOI 2022 Uplifting Excursion

- 有 $2m+1$ 种物品，重量分别为 $-m, -m+1, \ldots, m-1, m$。重量为 i 的物品有 a_i 个。
- 你需要拿走若干物品，使得这些物品重量之和恰好为 l。在此基础上，你需要拿尽可能多的物品。
- 问在物品重量之和恰好为 l 的基础上，你最多能拿多少物品。
- $m \leq 300$。

BalticOI 2022 Uplifting Excursion

- 现在有了负数。但是负数似乎并不影响上面的分析，可以一样做！
- 但是 $O(m^4)$ 的复杂度并不能接受！

BalticOI 2022 Uplifting Excursion

- 现在有了负数。但是负数似乎并不影响上面的分析，可以一样做！
- 但是 $O(m^4)$ 的复杂度并不能接受！
- 实际上调整时的体积变化量不会超过 $O(m^2)$。
- 证明与上面类似，考虑贪心的最优性。
- 复杂度 $O(m^3)$。

THUPC 2023 初赛背包

- 有 n 种物品，第 i 种物品单个体积为 v_i、价值为 c_i。
- q 次询问，每次给出背包的容积 V，你需要选择若干个物品，每种物品可以选择任意多个（也可以不选），在选出物品的体积的和恰好为 V 的前提下最大化选出物品的价值的和。你需要给出这个最大的价值和，或报告不存在体积和恰好为 V 的方案。
- $1 \le n \le 50, 1 \le v_i \le 10^5, 1 \le q \le 10^5, 10^{11} \le V \le 10^{12}$

THUPC 2023 初赛背包

- 考虑我们最初始背包的贪心是怎么做的：按照物品的性价比排序然后选取。
- 由于这里 V 足够大，可以预见的，会选择很多性价比最高的物品，然后再选取一些其它物品凑够 V 的体积。
- 因此我们选择性价比最高的物品 (v, c) 作为模数。对于第 i 个物品，从 j 向 $(j + v_i) \bmod v$ 连价值为 $c_i - c\lfloor\frac{j+v_i}{v}\rfloor$ 的边。
- 时间复杂度 $O(nV)$。

CF2115E Gellyfish and Mayflower

- 给定一张 n 个点 m 条边的有向图，每条有向边 $a \to b$ 保证 $a < b$。
- 每个顶点上有一个商人，第 i 个顶点上的商人出售价格为 c_i，力量值为 w_i 的卡牌。当走到第 i 个点时，可以在第 i 个点的商人处购买任意数量的卡牌，只要你有足够的钱。
- GellyFish 携带 r 元钱，从 1 号点开始走到 p 号点。他想知道走到 p 号点时购买的卡牌具有的力量值之和最大是多少。q 次询问。
- $n \le 200, m \le 2000, c_i \le 200, q \le 2 \times 10^5, 1 \le r \le 10^9$

CF2115E Gellyfish and Mayflower

- 假设已经知道了途径的性价比最高的物品为第 i 个物品，并且 $r > c^2$，则可以跑同余最短路，单次时间复杂度 $O(mc)$。枚举性价比最高的物品，并强制不经过性价比更高的物品即可。
- 若 $r \leq c^2$，则直接跑暴力，时间复杂度 $O(mc^2)$。

从朴素的 DP 讲起

- 设一个最优选取方案为 x，则最优选取物品方案的总体积 $sumW = \sum w_i x_i$ 满足 $sumW > M - W$。
- 记 f_i 表示选取总体积恰好为 i 的物品的最大价值，只需要求出 $(M - W, M]$ 区间内的 f_i。

引理 2

- 为了快速计算一段区间的 f_i，有如下引理：

引理

设 S 为某一选取方案中选取物品体积的可重集合，则存在一种划分 S 为 S_1, S_2 的方案，满足 $|sum(S_1) - sum(S_2)| \leq W$。

- 不断将较大集合调整到较小集合即可证明。

- 因此，f_i 处的值可以由 f 的 $[\frac{i}{2}-W,\frac{i}{2}+W]$ 区间自 $(\max,+)$ 卷积得到。
- 容易归纳证明，这样递归计算 k 层之后的范围不会超出 $[\frac{M}{2^k}-2W,\frac{M}{2^k}+2W]$，对于 $\leq O(W)$ 的 i 运行朴素的 $O(W^2)$ DP 预处理 f_i 的值，就只需要 $O(\log M)$ 次 $(min,+)$ 卷积即可推出 f_i 在 $[M-W,M]$ 区间内的答案。时间复杂度 $O(W^2\log W)$。
- 这个问题在 OI 中一个可以参考的版本是 2016 USP Try-outs L[3]。

[3] https://codeforces.com/gym/101064/problem/L

QOJ #9870 Items

- 给定 n 种物品，每种物品有无限个，第 i 种物品有体积 w_i。给定 M，试判断是否存在一种恰选择 n 个物品的方案使得体积和恰好为 M。$n \leq 10^5, M \leq n^2, 0 \leq w_i \leq n$。

QOJ #9870 Items

- 记 $B=\lfloor\frac{M}{n}\rfloor$，则将所有物品体积减去 B，M 减去 nB 显然不影响答案。此时所有物品的体积在 $[-B,n-B]$ 区间内。
- 接下来我们证明如下引理：

引理

对于一个选择 n 个物品使得体积之和恰为 M 的方案，存在一种这 n 个物品的排列方式 P，使得 $\forall p,\sum_{i=1}^{p}P_i\in[0,n]$。

QOJ #9870 Items

- 证明考虑增量构造。物品数为 0 时显然成立。现在假设对于 $k \geq 1$，已经选出了满足条件的 $k-1$ 个物品并排列，体积总和为 W，希望加入第 k 个物品。接下来对于 W 和 B 的大小关系分类讨论：
  - 当 $W = B$ 时，任意选取一个物品加入体积仍符合要求。
  - 当 $W < B$ 时，选取一个体积为正的物品加入。若体积为正的物品不存在，则说明 $M \leq W$，且剩余物品体积均 ≤ 0，体积总和恰好为 $M - W$，任意选取物品加入即可。
  - 当 $W > B$ 时，选取一个体积为负的物品加入。若体积为负的物品不存在，则说明 $M \geq W$，且剩余物品体积均 ≥ 0，体积总和恰好为 $M - W$，任意选取物品加入即可。

QOJ #9870 Items

- 记 $f^k(i)$ 表示恰好选取了 k 个物品，体积总和是否可以为 i。根据上述定理，f^k 只保留 $[-n, n]$ 区间内的答案不影响正确性，则容易用 FFT $O(n \log n)$ 做到 $f^k \to f^{2k}, f^k \to f^{k+1}$，进行 $O(\log n)$ 轮分治即可求出正确答案。

能不能更进一步?

- 上述做法仅能求出一段长为 $O(W)$ 区间的 DP 值，接下来我们将进行精细分析，使得能在同样的时间复杂度内对于所有 $i \in [1, W^2]$ 求出相应的 DP 值。
- 从前文的分析可以看出，对于背包问题的优化与同余有比较大的联系。

定理 2

- 我们先来介绍一个关于线性丢番图方程[4]的定理：

定理

给定 n 个数 $0 < a_1 < \cdots < a_n$，记 $g = \gcd(a_1, \ldots, a_n)$，$G(a, n)$ 表示最大的不能表示为 $\sum_{i=1}^{n} k_i a_i (k_i \in \mathbb{N})$ 的 g 的倍数。则有

$$G(a, n) \leq 2a_{n-1} \left\lfloor \frac{a_n}{gn} \right\rfloor - a_n$$

- 受篇幅所限，且证明过程[5]和本文关系不大，略去。

[4] Wikipedia, https://en.wikipedia.org/wiki/Diophantine_equation

[5] P. Erdös and R. L. Graham, On a linear diophantine problem of Frobenius, 1972

优化的一个方向

- 不失一般性地，以下分析假设没有两个物品体积相同。
- 记 x 为一个选取方案。考虑一个位置 p，记 S 为所有 $> p$ 的被选取物品体积的可重集合。若存在 $\emptyset \neq S' \subset S$ 使得存在一个长为 p 的自然数序列 k 满足 $\sum_{i=1}^{p} k_i w_i = sum(S')$，则可以将 S' 替换成一个性价比更高的选取方案而总体积不变，那么 x 就不是最优选取方案。

定理 3

- 记 $g = \gcd(w_1, \ldots, w_p)$，显然所有能被这段前缀表示出来的数一定是 g 的倍数。在一个集合中选取一个子集为 g 的倍数使我们想到引理 1，对其进行扩展可以得到如下定理：

定理

对于一个大小为 m 的可重集合 S 及一常数 $n(m \geq n)$，S 存在一个大小至少为 $m-n+1$ 的子集 S' 使得

$$sum(S') \bmod n = 0$$

- 根据引理 1 的方法，不断选出集合直至剩余集合大小 $< n$ 即可证明。

定理 4

- 最后就可以得出对于选取体积的限制：

定理

记 x 为一个最优划分方案，则 $\forall p \in [1, n)$，满足

$$\sum_{i=p+1}^{n} x_i w_i \le \left\lfloor \frac{3W^2}{p} \right\rfloor$$

证明.

反证，假设存在一个 p 满足

$$\sum_{i=p+1}^{n} x_i w_i > \left\lfloor \frac{3W^2}{p} \right\rfloor$$

记 $g = \gcd(w_1, \ldots, w_p)$。因为 $w_1, \ldots, w_p$ 互不相同，所以 $g \le \lfloor \frac{W}{p} \rfloor$。
现考虑在 $> p$ 选取的物品体积的可重集合 S 中选出一个和最大且 $\bmod g = 0$ 的集合 S'，这至多需要去除 $g-1$ 个数，故 S' 内数之和 $> \lfloor \frac{2W^2}{p} \rfloor$。由定理 2，$w_1, \ldots, w_p$ 最大不能表示出来的数 $< \lfloor \frac{2W^2}{pg} \rfloor \le \lfloor \frac{2W^2}{p} \rfloor$，因此，$S'$ 内的数之和一定能被表示出来，即存在一种方案将 S' 所对应的物品替换成前 p 种物品且体积不变，与 x 为最优方案矛盾。 $\square$

最终做法

- 因此，从后往前加入物品，加入物品 i 时只需要更新前 $\lfloor \frac{3W^2}{i} \rfloor$ 个 DP 状态，这样的复杂度为 $O(\sum_{i=1}^{W} \frac{3W^2}{i}) = O(W^2 \log W)$，最后得到的 DP 数组即为 $i = 1, \dots, W^2$ 的答案。
- 相比于分治做法，这个做法的代码难度和常数均显著降低。

完全的二进制拆分

- 在 W 较小的情形下，优化多重背包问题常采用的思路有两种：单调队列与二进制拆分。对于前者，我目前暂未找到拓展的方法。
- 对于后者，考虑进行更为完全的拆分，也即，对于一个物品 (w, v, d)，令 $k = \lfloor \log_2(d+1) \rfloor$，先对于所有 $0 \le i < k$，拆出物品 $(2^i w, 2^i v, 1)$，然后对于所有剩下的物品，也根据二进制表示拆分成 $(2^i w, 2^i v, 1)$ 的形式。容易证明，所有可能的选取次数均可以用这 $O(\log d)$ 个物品组合出来。

类数位 DP 的 01 背包

- 现在转化为了 01 背包问题。从大到小对于每个 k，考虑所有形如 $(2^k w, 2^k v)$ 的物品，维护 DP 数组 f_i 表示剩余体积还有 $i2^k$ 的最大价值。当 $k \to k-1$ 时，先修改体积的变化，然后用所有符合条件的物品更新 DP 状态。
- 直接这样做的时间复杂度还是 $O(nM)$ 的，但是可以发现，对于每个物品，拆到每个 k 上不超过 2 个。因此对于每个 k，所有 2 的幂次小于 k 的物品体积之和不超过 $2^k \times 2\sum_{i=1}^{n} w_i$。那么 dp 状态只需要保留前 $2\sum_{i=1}^{n} w_i$ 个传入 $k-1$，时间复杂度优化为 $O(n\sum w_i \log W)$。

体积较大但是价值较小的情况

- 对于 $\sum w_i$ 较大，但是 $\sum v_i$ 较小的情形，可以将 DP 状态中的值域与状态互换。具体的，设 f_i 表示价值为 $i2^k$ 的最小体积，但是此时如何保留状态并不确定。一个相对简单的办法是二分答案 ans，这样在 $k \to k-1$ 时就只需要保留 $[\frac{ans}{2^k} - 2\sum v_i, \frac{ans}{2^k}]$ 区间内的状态，但是这样的复杂度是 $O(n\sum v_i \log^2 V)$ 的，与 $\sum w_i$ 较小的情况并不统一。

引理 2

- 记 $V = \max v_i$。为了去除二分带来的 $\log V$，我们考虑利用贪心算法。有如下引理：

引理 (贪心算法的 V 最优性)

记贪心算法得到的最大权值和为 P，实际答案为 Q，则 $P > Q - V$。

- 结合性价比贪心过程易证。

- 则预先运行一遍贪心算法得到答案 P，在 $k \to k-1$ 时只需要保留 $[\frac{P}{2^k} - 2\sum v_i, \frac{P+V}{2^k}]$ 区间内的状态，复杂度降至 $O(n\sum v_i \log V)$。
- 此外，还有另一种不基于贪心的做法。在 $k \to k-1$ 时，找到最大的 q 使得 $f_q \leq W$，则所有 $i > q$ 的部分显然均可舍弃。对于 $i < q - 2\sum v_i$ 的部分，类似之前的分析也可以得出是不必要的，因此只需要保留 $[q - 2\sum v_i, q]$ 区间内的状态，时间复杂度也为 $O(n\sum v_i \log V)$。
- 值得一提的是，可以构造出需要保留 $O(\sum v_i)$ 级别状态的数据，因此这个算法在保留状态的界上是紧的。

At Last

Thanks for listening!