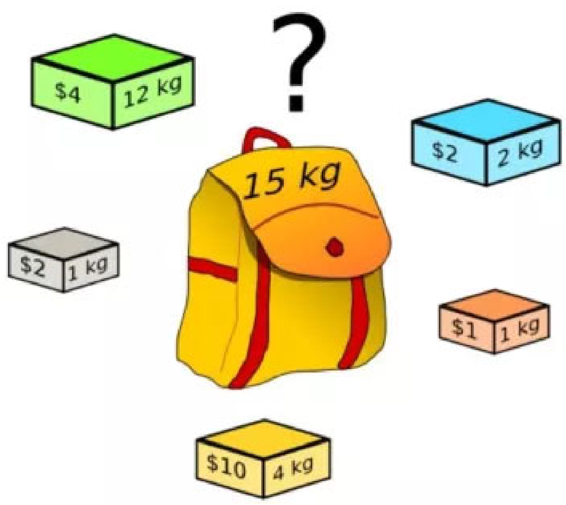
动态规划多类背包问题总结

原创 余孙婕 LOA算法学习笔记 2021-01-20 16:12

作为动态规划(DP)中的经典问题背包问题，有以下三类子问题，分别为：0-1 背包问题、完全背包问题和多重背包问题。卜老师上课提到的主要为 0-1 背包问题。背包问题顾名思义，就是将一些物品放到一个背包中，背包对物体有重量限制，而我们需要的是背包中物体的总价值尽可能地多，其本质是有约束的优化问题。

借用老师上课的图如下：



转换为有约束的优化问题的数学表示如下：

$$\max \sum x_i v_i$$
$$s.t. \begin{cases} \sum x_i w_i \le W \\ 0 \le x_i \le c_i \text{且} x_i \text{为整数} \end{cases}$$

目标函数为每类物品价值总和，约束条件1为每类物品重量和不超过背包限重，约束条件2为每类物品的选取数量约束。不同的背包问题主要差别在于每类物品选取数量的约束，也就是 c_i 。

显然，背包问题的本质是线性的有约束的优化问题，理论上也可以用线性规划解决，但这里只分析动态规划方法。

01 "0-1"背包问题

0-1背包问题中，每类物品的选取只有：选-1or 不选-0，即 $c_i = 1$ ，有约束优化问题，表示为如下：

$$\max \sum x_i v_i$$
$$s.t. \begin{cases} \sum x_i w_i \le W \\ x_i = 0 or 1 \end{cases}$$

0-1 背包问题老师上课已经讲得很多了，这里直接给出子问题和状态转移方程：

子问题：当背包容量为 j 时，前 i 个物品所能达到的最大价值。

状态转移方程：面对背包容量为 j，我们需要决策是不是需要放入第 i 个物品。如果包容量 j 放不下第 i 个物品，那么很简单，前 i 个物品所能达到的最大价值就是前 i-1个物品所能达到的最大价值；如果放得下，那么需要分析放第 i 个物品与不放第 i 个物品达到包容量 j 时的价值差异。公式表示如下

$$OPT(i,j)=\begin{cases}OPT(i-1,j), j<w_i \\ \max\{OPT(i-1,j-w_i)+v_i, OPT(i-1,j)\}, j\geq w_i\end{cases}$$

借用卜老师课件的数据我们完善一下 OPT 表格如下：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12kg/$4	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2kg/$2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
1kg/$2	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
1kg/$1	0	2	3	4	5	5	5	5	5	5	5	5	5	5	7	8
4kg/$10	0	2	3	4	10	12	13	14	15	15	15	15	15	15	15	15

最后我们得到的最优的背包价值为$15,选择的物品为 2kg/$2、1kg/$2、1kg/$1 和4kg/$10。

复杂度分析：OPT 表格中的状态数量为 W*N，W 为背包限重，N 为物品数量，状态转移复杂度为 O(1)，综合是时间复杂度为 O(W*N)，空间复杂度为 O(W*N)，类似背包问题都可以用一维数组更新代替二维数组（因为二维数组是从左往右从上到下按序生成状态值），所以空间复杂度可以优化到 O(W)。

02 完全背包问题

完全背包问题中，每类物品的选择都是不收限制的，也就是说 x_i 可以取到正无穷，因此有约束的优化问题表示如下：

$$\max\sum x_i v_i$$
$$s.t.\begin{cases}\sum x_i w_i \leq W \\ x_i \geq 0\end{cases}$$

首先我们按照常规理解来做：

子问题：当背包容量为 j 时，前 i 个物品所能达到的最大价值。

状态转移方程：面对背包容量为 j，我们需要决策放入**几个**第 i 个物品。如果包容量j放不下第 i 个物品，那么很简单，前 i 个物品所能达到的最大价值就是前i-1个物品所能达到的最大价值，与 0-1 背包问题一致；如果放得下，那么需要分析**放 0,...,K 个**第 i 个物品达到包容量 j 时的价值差异。

公式表示如下：

$$OPT(i,j)=\begin{cases}OPT(i-1,j), j<w_i \\ \max\{OPT(i-1,j-kw_i)+kv_i, k=0,\ldots,K\left(\text{可计算}\frac{j}{w_i}\right)\}, j\geq w_i\end{cases}$$

我们分析下这个时候的时间复杂度，OPT 表格中的状态数量仍然为 W*N，W 为背包限重，N 为物品数量，状态转移复杂度为 O(K)，其中 $K_{ij}=\frac{j}{w_i}$，时间复杂度为 $O(W\sum_{i=1}^{N}\frac{W}{w_i})$，这显然是我们不希望看到的，我们希望仍然像 0-1 背包一样，状态转移复杂度为 O(1)。

一种方法是利用本层 OPT 状态转移数，根据子问题描述，本层的 OPT 状态转移数 $OPT(i,\bar{j}),\bar{j}=0,\ldots,j-1$ 已经包含了我们需要决策的第 i 个物品，决策的问题从选择几个第 i 个物品变为是否再次放入一个第 i 个物品。状态转

移复杂度重新降为 O(1)。

因此我们将状态转移方程描述为如下：

状态转移方程：面对背包容量为 j ，我们需要决策放入**是否再次放入**第 i 个物品。如果包容量 j 放不下第 i 个物品，那么前 i 个物品所能达到的最大价值就是前 i-1 个物品所能达到的最大价值；如果放得下，那么需要分析**再次放入一个第 i 个物品达到包容量 j 时的价值与不放第 i 个物品达到的最大价值**之间的差异。至于在放这个物品前已经放了几个第 i 个物品，由于已经在 $OPT(i, j-w_i)$ 经过了决策，因此在决策 $OPT(i, j)$ 时不需要考虑之前背包里已经存在的第 i 个物品的数目。

表达式如下：

$$OPT(i,j)=\begin{cases}OPT(i-1,j), j<w_i\\ \max\{OPT(i,j-w_i)+v_i, OPT(i-1,j)\}, j\geq w_i\end{cases}$$

同样用例题完成 OPT 表格

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12kg/$4	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2kg/$2	0	0	2	2	4	4	6	6	8	8	10	10	12	12	14	14
1kg/$2	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
1kg/$1	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
4kg/$10	0	2	4	6	10	12	14	16	20	22	24	26	30	32	34	36

最后我们得到的最优的背包价值为$36,选择的物品为 1kg/$2 选 3 件、4kg/$10 选 3 件。

复杂度分析：OPT 表格中的状态数量为 W*N，W 为背包限重，N 为物品数量，状态转移复杂度为 O(1)，综合是时间复杂度为 O(W*N)，空间复杂度为 O(W*N)，同样，该背包问题都可以用一维数组更新代替二维数组，所以空间复杂度可以优化到 O(W)。

值得注意的是：0-1 背包问题在遍历背包容量使可以正序也可以逆序，因为在生成OPT(i,j)使我们用到的只有上层状态数，而在完全背包问题下，遍历背包容量只可以正序，因为我们需要用到同层左侧的状态数，只有先生成左侧状态数，才能生成当前状态数。

03 多重背包问题

多重背包问题提供了每种物品的数量限制，是介于 0-1 背包问题和完全背包问题中间的问题，有约束优化问题表示如下：

$$\max\sum x_i v_i$$
$$s.t.\begin{cases}\sum x_i w_i \leq W\\ 0\leq x_i\leq c_i\end{cases}$$

我们同样可以按照常规方法去做：

子问题：当背包容量为 j 时，前 i 个物品所能达到的最大价值。

状态转移方程：面对背包容量为 j ，我们需要决策放入**几个**第 i 个物品。如果包容量 j 放不下第 i 个物品，前 i 个物品所能达到的最大价值就是前i-1个物品所能达到的最大价值；如果放得下，那么需要分析**放 0,..., c_i 个**第 i 个物品达到包容量 j 时的价值差异。

公式表示如下：

$$OPT(i,j)=\begin{cases}OPT(i-1,j), j<w_i\\ \max\{OPT(i-1,j-kw_i)+kv_i, k=0,...,c_i\,(\text{由题目给出})\}, j\geq w_i\end{cases}$$

同样分析下时间复杂度，OPT 表格中的状态数量仍然为 W*N，W 为背包限重，N 为物品数量，状态转移复杂度为 $O(c_i)$，时间复杂度为 $O(W\sum_{i=1}^{N}c_i)$

https://www.kancloud.cn/kancloud/pack/70127 背包问题九讲中有对多重背包问题的优化，可以将时间复杂度降低至 $O(W\sum_{i=1}^{N}log\ c_i)$。基本的策略是将第 i 种物品划分成若干件，划分标准在 blog 里有详细的说明。这里用 blog 中的一个例子，当 $c_i=13$ 时，物品被划分为系数是 1,2,4,6 的四件物品，当你想取任意数目的 $x_i(\le c_i)$ 时，都可以用这四个系数组合而成。这样完成了将第 i 种物品分成了 $O(log\ c_i)$ 种物品的 0-1 背包问题，复杂度从需要决策放入 0...13 件第 i 种物品变为决策是否放入第 $i_{0\ldots log\ c_i}$ 种物品。

同样用例题完成 OPT 表格，我们设置物品数量限制c=(3,3,2,3,2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12kg/$4	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2kg/$2	0	0	2	2	4	4	6	6	6	6	6	6	6	6	6	6
1kg/$2	0	2	4	4	6	6	8	8	10	10	10	10	10	10	10	10
1kg/$1	0	2	4	5	6	7	8	9	10	11	12	13	13	13	13	13
4kg/$10	0	2	4	5	10	12	14	15	20	22	24	25	26	27	28	29

最后我们得到的最优的背包价值为$29,选择的物品为 2kg/$2 选 2 件、1kg/$2 选 2 件、1kg/$1 选 1 件、4kg/$10 选 2 件。

04 混合背包问题

混合背包问题将上述三种背包问题进行混合，即有的物品只能放一件，有的可以放多件，有的可以放无穷多件。很显然背包问题涉及的子问题都是完全一致的，即 OPT(i,j)中的状态数表示的含义是完全一致的。因此只需要按类选择不同的状态转移方程就行，如下所示：

$$OPT(i,j)=\begin{cases}OPT(i-1,j),j<w_i\\ \max\{OPT(i-1,j-w_i)+v_i,OPT(i-1,j)\},j\ge w_i,0-1\text{背包}\\ \max\{OPT(i,j-w_i)+v_i,OPT(i-1,j)\},j\ge w_i,\text{完全背包}\\ \max\{OPT(i-1,j-kw_i)+kv_i,k=0,\ldots,c_i\},j\ge w_i,\text{多重背包}\end{cases}$$

05 结语

背包问题作为动态规划中的基础问题，延伸出了许许多多多种多样的变形。通过对背包问题这一基础问题的学习和总结，或许能帮助我们加深对动态规划的理解，也能对那些看上去比较难的算法问题提供解题思路。