Задача о рюкзаке. Динамическое программирование

17 апреля 2022 г.

Во время ограбления грабитель находит гораздо больше добычи, чем он ожидал, и ему приходится решать, что взять. Общий вес его сумки (или «рюкзака») не превышает W фунтов. Можно выбрать из n предметов весом $w_1, ..., w_n$ и стоимостью в долларах $p_1, ..., p_n$. Какая самая ценная комбинация предметов, которую он может поместить в свою сумку?

Например, возьмем $W = 10$ и

№	Вес	Стоимость
1	6	30
2	3	14
3	4	16
4	2	9

Если доступно неограниченное количество каждого предмета, оптимальным выбором будет выбрать предмет 1 и два предмета 4 (всего: 48 долларов). С другой стороны, если есть по одному каждого предмета (например, грабитель проник в художественную галерею), то оптимальный рюкзак содержит предметы 1 и 3 (всего: 46 долларов). Эту задачу о рюкзаке можно сформулировать математически.

Дано n предметов, W — вместимость рюкзака, $w_j > 0, j = 1, \dots, n$ — соответствующий ему набор положительных целых весов, $p_j > 0, j = 1, \dots, n$ — соответствующий ему набор положительных целых стоимостей. Нужно найти набор бинарных величин $x = x_1, \dots, x_n$, где:

$$\begin{cases} x_j = 1, \text{если предмет с номером } j \text{ включен в рюкзак,} \\ x_j = 0, \text{иначе,} \end{cases}$$

$$j = 1, \dots, n$$

и такой что:

$$\sum_{j=0}^{n} w_j x_j \leq W$$

$$\sum_{j=0}^{n} p_j x_j \longrightarrow max$$

Введем обозначение:

$$\varphi_k(y) = max \left\{ \sum_{j=1}^{k} p_j x_j \mid \sum_{j=1}^{k} w_j x_j \le y, x_j \in \{0,1\}, j = 1, \dots, n \right\}$$

$\varphi_k(y)$ — максимальная стоимость набора среди первых k предметов, помещенного в рюкзак грузоподъемностью y. Значения $\varphi_k(y)$ можно вычислить, используя *рекуррентное соотношение Беллмана*:

$$\varphi_1(y) = \begin{cases} p_1, & \text{если } w_1 \le y, \\ 0, & \text{иначе}, \end{cases}$$

$$y = 0, \dots, W$$

$$\varphi_k(y) = \begin{cases} max \left\{ \varphi_{k-1}(y - w_k) + p_k,\ \varphi_{k-1}(y) \right\}, & \text{если } w_k \le y, \\ \varphi_{k-1}(y), & \text{иначе}, \end{cases}$$

$$y = 0, \dots, W,\ k = 2, \dots, n$$

Алгоритм строится на соотношениях, которые описаны выше. Значения функции $\varphi_k(y)$ будем записывать в массиве φ. Основная часть алгоритма состоит из двух вложенных циклов. Во внешнем цикле переменная k пробегает значения от 1 до n, внутри него цикл по переменной y. При каждом k для определения значений $\varphi_k(y)$ необходимы значения этой функции, рассчитанные на предыдущей итерации внешнего цикла. При этом значения, полученные на более ранних итерациях внешнего цикла не требуются. Внутренний цикл пробегает значения по убыванию переменной y от W до 0, то получаемые значения функции $\varphi_k(y)$, без потери нужной информации, можно хранить в одномерном массиве $\varphi[0..W]$.

Algorithm 1 Part 1

```
for y = 0 to W do
    φ[y] = 0
end for
for k = 1 to n do
    for y = W downto 0 do
        if w_k ≤ y and φ[y] < φ[y − w[k]] + p[k] then
            φ(y) = φ[y − w[k]] + p[k]
            ψ[k][y] = 1
        else
            ψ[k][y] = 0
        end if
    end for
end for
```

$\psi[k][y]$ — вспомогательный двумерный массив. Если при $w_k \le y$ выполнено $\varphi_{k-1}(y - w_k) + p_k > \varphi_{k-1}(y)$, то $\varphi(y) = \varphi(y - w_k) + p_k$ и $\psi[k][y] = 1$. Иначе $\psi[k][y] = 0$. По значениям массива ψ, будем определять входит ли k-й предмет в искомый оптимальный набор предметов.

Algorithm 2 Part 2

```
y = W
for k = n downto 1 do
    if ψ[k][y] = 1 then
        x[k] = 1
        y = y − w[k]
    else
        x[k] = 0
    end if
end for
```

Метод динамического программирование не позволяет решать задачу за полиномиальное время, потому что его сложность зависит от максимального веса. Временная сложность алгоритма решения задачи равна *O(nW)*.