# Задача о рюкзаке. Динамическое программирование

Иванцов Глеб Богданович

Москва — 2022

# Содержание

# 1 Введение

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.

# 2 Постановка задачи

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

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

$$j = 1, \ldots, n$$

и такой что:

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

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

# 3 Описание алгоритма

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 $\varphi$. 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

```
1:  for y = 0 to W do
2:      φ[y] = 0
3:  end for
4:  for k = 1 to n do
5:      for y = W downto 0 do
6:          if w_k ≤ y and φ[y] < φ[y − w[k]] + p[k] then
7:              φ(y) = φ[y − w[k]] + p[k]
8:              ψ[k][y] = 1
9:          else
10:             ψ[k][y] = 0
11:         end if
12:     end for
13: 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

**Algorithm 2** Part 2

1: $y = W$
2: **for** $k = n$ downto 1 **do**
3:  **if** $\psi[k][y] = 1$ **then**
4:  $x[k] = 1$
5:  $y = y - w[k]$
6:  **else**
7:  $x[k] = 0$
8:  **end if**
9: **end for**

массива $\psi$, будем определять входит ли $k$-й предмет в искомый оптимальный набор предметов.

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

# 4 Листинг программы на python

```
def solver(self):
    A = [0 for i in range(self.capacity + 1)]
    B = [[0 for i in range(self.capacity + 1)] for j in range(self.number)]
    for k in range(self.number):
        for y in reversed(range(self.capacity + 1)):
            if self.weights[k] <= y \
                    and A[y] < A[y - self.weights[k]] + self.profits[k]:
                A[y] = A[y - self.weights[k]] + self.profits[k]
                B[k][y] = 1
            else:
                B[k][y] = 0

    packed_items = []
    y = self.capacity
    for k in reversed(range(self.number)):
        if B[k][y] == 1:
            packed_items.append(k)
            y = y - self.weights[k]

    computed_value = 0
    packed_weights = []
    for k in packed_items:
        computed_value += self.profits[k]
        packed_weights.append(self.weights[k])

    return packed_items[::-1], packed_weights[::-1], computed_value
```

# 5 Тесты

```
glebivantsov@192 Knapsack-problem % python3 test_runner.py
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0004007816
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0000581741
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0002849102
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0001010895
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0002458096
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0003230572
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 0.0032701492
ok
test (knapsack_tests.Test) ... knapsack_tests.Test.test: 34.4831180573
ok

----------------------------------------------------------------------
Ran 8 tests in 34.488s

OK
```

Тест 7
$W = 750$

| № | Вес | Стоимость |
|---|---|---|
| 1 | 70 | 135 |
| 2 | 73 | 139 |
| 3 | 77 | 149 |
| 4 | 80 | 150 |
| 5 | 82 | 156 |
| 6 | 87 | 163 |
| 7 | 90 | 173 |
| 8 | 94 | 184 |
| 9 | 98 | 192 |
| 10 | 106 | 201 |
| 11 | 110 | 210 |
| 12 | 113 | 214 |
| 13 | 115 | 221 |
| 14 | 118 | 229 |
| 15 | 120 | 240 |

Время выполнения теста в секундах: 0.003351
Индексы упакованных предметов:
[0, 2, 4, 6, 7, 8, 13, 14]
Веса упакованных предметов:
[70, 77, 82, 90, 94, 98, 118, 120]
Стоимость упакованных предметов: 1458

Тест 8
$W = 6404180$

| № | Вес | Стоимость |
|---|---|---|
| 1 | 382745 | 825594 |
| 2 | 799601 | 1677009 |
| 3 | 909247 | 1676628 |
| 4 | 729069 | 1523970 |
| 5 | 467902 | 943972 |
| 6 | 44328 | 97426 |
| 7 | 34610 | 69666 |
| 8 | 698150 | 1296457 |
| 9 | 823460 | 1679693 |
| 10 | 903959 | 1902996 |
| 11 | 853665 | 1844992 |
| 12 | 551830 | 1049289 |
| 13 | 610856 | 1252836 |
| 14 | 670702 | 1319836 |
| 15 | 488960 | 953277 |
| 16 | 951111 | 2067538 |
| 17 | 323046 | 675367 |
| 18 | 446298 | 853655 |
| 19 | 931161 | 1826027 |
| 20 | 31385 | 65731 |
| 21 | 496951 | 901489 |
| 22 | 264724 | 577243 |
| 23 | 224916 | 466257 |
| 24 | 169684 | 369261 |

Время выполнения теста в секундах: 34.483
Индексы упакованных предметов:
[0, 1, 3, 4, 5, 9, 10, 12, 15, 21, 22, 23]
Веса упакованных предметов:
[382745, 799601, 729069, 467902, 44328, 903959, 853665, 610856, 951111, 264724, 224916, 169684]
Стоимость упакованных предметов: 13549094