12. Динамично оптимиране - I

Задача за раницата [8.2.1]

Дадена е раница с вместимост M килограма и N предмета, всеки от които се характеризира с две числа - тегло m_i и стойност c_i. Да се избере такова множество от предмети, чиято стойност е максимална, а сумата от теглата не надвишава M.

Дефинираме рекурентна целева функция:

$$F(0) = 0;\ F(i) = \max\ \{\ c_j + F(i - m_j),\ j = 1, 2, ..., N,\ m_j \le i\ \}, i > 0$$

Методът на динамичното оптимиране изисква последователно пресмятане на стойностите на $F(i)$, като за това пресмятане се използват вече пресметнатите стойности за по-малки i.

Програма на C++ за решаване на задачата.

Да разгледаме примера:

```
N = 8;
index  1  2  3  4  5  6  7  8
m[8]   3, 7, 6, 1, 2, 4, 5, 5
c[8]   5, 3, 9, 1, 1, 2, 5, 2
M = 8;

Fn[0] = 0;
Fn[1] = max{c[4]+Fn[0]} = 1;
       set[1][4]=1;    set[1] = {0,0,0,1,0,0,0}

Fn[2] = max{c[4]+Fn[1], c[5]+Fn[0]} = 1
       set[2][5]=1;    set[2] = {0,0,0,0,1,0,0}

Fn[3] = max{c[1]+Fn[0], c[4]+Fn[2], c[5]+Fn[1]} =
        max{5   +0,     1   +1,     1   +1} = 5
       set[3][1]=1;    set[3] = {1,0,0,0,0,0,0}

Fn[4] = max{c[1]+Fn[1], c[4]+Fn[3], c[5]+Fn[2], c[6]+Fn[0]} =
        max{5   +1,     1   +5,     1   +1,     2   +0} = 6
       set[4][1]=1;    set[4] = {1,0,0,1,0,0,0}

Fn[5] = max{c[1]+Fn[2],c[4]+Fn[4],c[5]+Fn[3],c[6]+Fn[1],c[7]+Fn[0],c[8]+Fn[0]} =
        max{5   +1,    1   +6,    1   +5,    2   +1,    5   +0,    2   +0} = 6
       set[5][1]=1;    set[5] = {1,0,0,0,1,0,0}
```

```
Fn[6] = max{c[1]+Fn[3],c[3]+Fn[0],c[4]+Fn[5],c[5]+Fn[4],c[6]+Fn[2],c[7]+Fn[2],c[8]+Fn[1]} =
        max{5   +5,    9   +0,    1   +5,    1   +6,    2   +1,    5    +1,   2   +1} = 9
       set[6][3]=1;    set[6] = {0,0,1,0,0,0,0}

Fn[7] = max{c[1]+Fn[4],c[2]+Fn[0],c[3]+Fn[1],c[4]+Fn[6],c[6]+Fn[5],c[7]+Fn[2],c[8]+Fn[2]} =
        max{5   +6,    5   +0,    9   +1,    1   +9,    2   +6,    5   +1,    2   +1} = 10
       set[7][3]=1;    set[7] = {0,0,1,1,0,0,0}

Fn[8] = max{c[1]+Fn[5],c[2]+Fn[1],c[3]+Fn[2],c[4]+Fn[7],c[6]+Fn[4],c[7]+Fn[3],c[8]+Fn[3]} =
        max{5   +6,    5   +1,    9   +1,    1   +10,   2   +6,    5   +1,    2   +1} = 10
       set[8][3]=1;    set[8] = {0,0,1,0,1,0,0}
```

Варианти на алгоритъма за решаване на задчата:

- с пресмятане на решението за всички стойности на капацитете
- без пресмятане на всички стойности на капацитета

Варианти на задачата за раницата:

- с определен брой предмети от всеки вид
- с неограничен брой предмети от всеки вид

"Лесна" задача

Зайче в беда

Веднъж малкото бяло зайче, гонено от един ловец попаднало в лабиринт, които имал форма на квадратна дъска N x N. В него чакал големия лош вълк, които предварително изкопал дупки, където зайчето да падне и той да го хване по-лесно. В последния момент зайчето с ужас разбрало, че може да се движи само в посока надолу и надясно и че изхода от лабиринта е чак в долния десен ъгъл на дъската.

Зайчето трябвало да разбере каква е вероятността да излезе от лабиринта без да падне в някоя дупка. За целта трябвало да изчисли броя пътища от входа до изхода на лабиринта, като успяло да се снабди с картата на този лабиринт. Картата е зададена с размер N, като местата на дупките са означени с 0, а проходимите места с 1. Напишете програма, която пресмята търсения брой пътища.

Най-дълга обща подредица [8.2.6]

Дадени са две редици (от числа или символи):

$$X = (x_1, x_2, ..., x_m) \text{ и } Y = (y_1, y_2, ..., y_n)$$

Търси се най-дълга редица $Z = (z_1, z_2, ..., z_k)$, която е подредица на X и Y едновременно. Z е подредица на X, ако Z може да бъде получена чрез премахване на (0 или няколко) членове на X.

Най-напред ще търсим само дължината на най-дългата обща подредица. Ще приложим метода на динамичното оптимиране, като $F(i, j)$ е търсената дължина за първите i члена на редицата X и първите j члена на редицата Y. Очевидно

$F(i,0) = 0$ за всяко i, $F(0, j) = 0$ за всяко j.

$F(i, j) = F(i - 1, j - 1) + 1$ за $x_i = y_j$,

$F(i, j) = \max \{F(i - 1, j), F(i, j - 1)\}$ в противен случай.

Намираме последователно стойностите на $F(i, j)$ и последната намерена стойност $F(m,n)$ е решението на задачата.

Намирането на една най-дълга подредица (може да не е една) става по същия начин, като тръгваме от последния елемент и следим откъде идва максималната стойност.

Пример:
acbcacbcaba
abacacacababa

		0	1	2	3	4	5	6	7	8	9	10	11
		""	a	c	b	c	a	c	b	c	a	b	a
0	""	**0**	0	0	0	0	0	0	0	0	0	0	0
1	a	0	**1**	1	1	1	1	1	1	1	1		1
2	b	0	**1**	1	2	2	2	2	2	2	2	2	2
3	a	0	**1**	1	2	2	3	3	3	3	3	3	3
4	c	0	**1**	2	2	3	3	4	4	4	4	4	4
5	a	0	**1**	2	2	3	4	4	4	4	5	5	5
6	c	0	**1**	2	2	3	4	5	5	5	5	5	5
7	a	0	**1**	2	2	3	4	5	5	5	6	6	6
8	c	0	**1**	2	2	3	4	5	5	6	6	6	6

9	a	0	**1**	2	2	3	4	5	5	6	7	7	7
10	b	0	**1**	2	3	3	4	5	6	6	7	8	8
11	a	0	**1**	2	3	3	4	5	6	6	7	8	9
12	b	0	**1**	2	3	3	4	5	6	6	7	8	9
13	a	0	**1**	2	3	3	4	5	6	6	7	8	9