Διακριτό πρόβλημα σακιδίου (Discrete Knapsack)

Input: $|X| = n \quad c_i, a_i, b$ integers

Output: $Y \subseteq X$ s.t. $\sum_{x_i \in Y} a_i \leq b$

and *MAX profit*

Instance:

c: 10, 5, 8

a: 3, 2, 2

b=4, n=3

0-1 Knapsack

$$x_i \in \{1,0\}$$

$$Max \sum_{i=1}^{n} c_i x_i \qquad \sum_{i=1}^{n} a_i x_i \leq b$$

Instance:

max 10 x_1 **+ 5** x_2 **+ 8** x_3

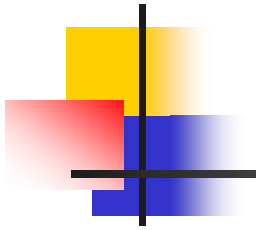
3 x_1 **+ 2** x_2 **+ 2** x_3 $\leq$ **4**

Greedy Knapsack (n, c, a, b)

c_i / a_i in non-increasing order

$$\frac{c_{j1}}{a_{j1}} \geq \frac{c_{j2}}{a_{j2}} \geq \ldots \geq \frac{c_{jn}}{a_{jn}}$$

```
Y := ∅
for i := 1 to n do
    if b ≥ a_i then
    begin
        Y := Y ∪ { x_i }
        b := b - a_i
    end
return Y
```



Dynamic Programming

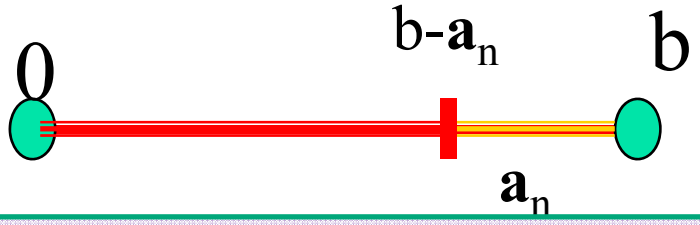
- Ορισμός υπο-προβλημάτων
- Σύνδεση βέλτιστων λύσεων (αναδρομική σχέση)
- Κατασκευή του πίνακα βέλτιστων λύσεων

Δυναμικός Προγραμματισμός/0-1 Knapsack

$f_n(b)$: τιμή βέλτιστης λύσης με το πολύ n αντικείμενα

$f_{n-1}(b)$: τιμή βέλτιστης λύσης με το πολύ n-1 αντικείμενα

$$f_n(b) = f_{n-1}(b - \mathbf{a}_n) + \mathbf{c}_n$$

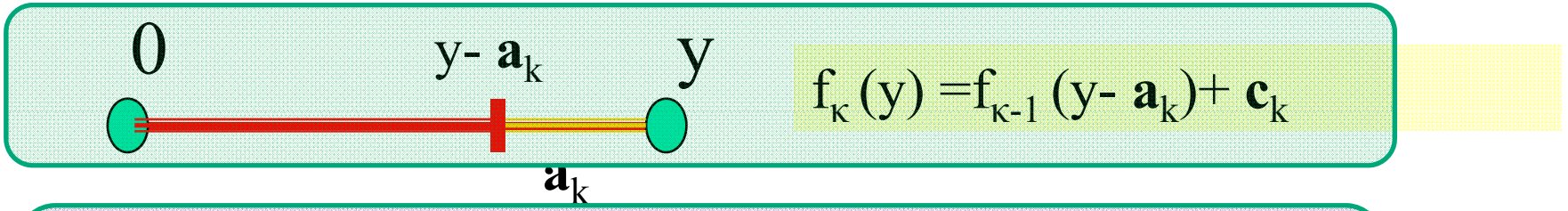


$$f_n(b) = \max\{f_{n-1}(b),\ f_{n-1}(b - \mathbf{a}_n) + \mathbf{c}_n\}$$

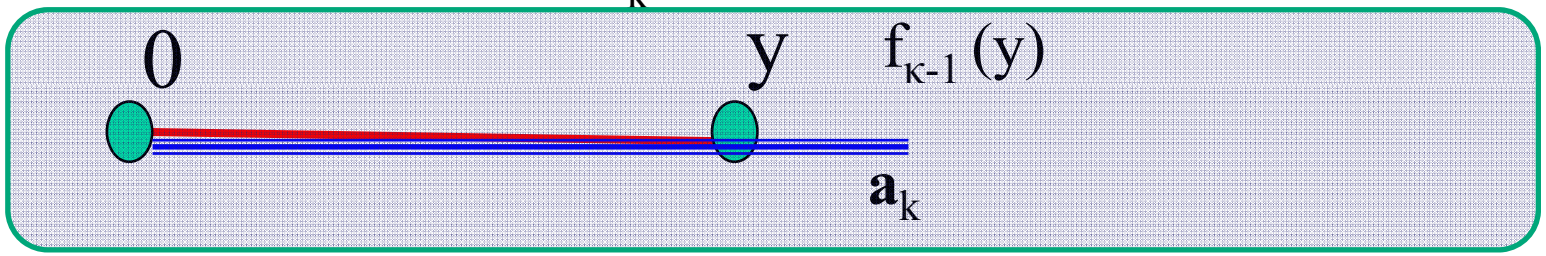
Δυναμικός Προγραμματισμός/0-1 Knapsack/ n x (b+1) υποπροβλήματα

$f_κ(y)$: τιμή βέλτιστης λύσης με το πολύ κ αντικείμενα

$f_{κ-1}(y)$: τιμή βέλτιστης λύσης με το πολύ κ-1 αντικείμενα



$$f_κ(y) = f_{κ-1}(y - a_k) + c_k$$



$$f_κ(y) = \max\{f_{κ-1}(y),\ f_{κ-1}(y - a_k) + c_k\}$$

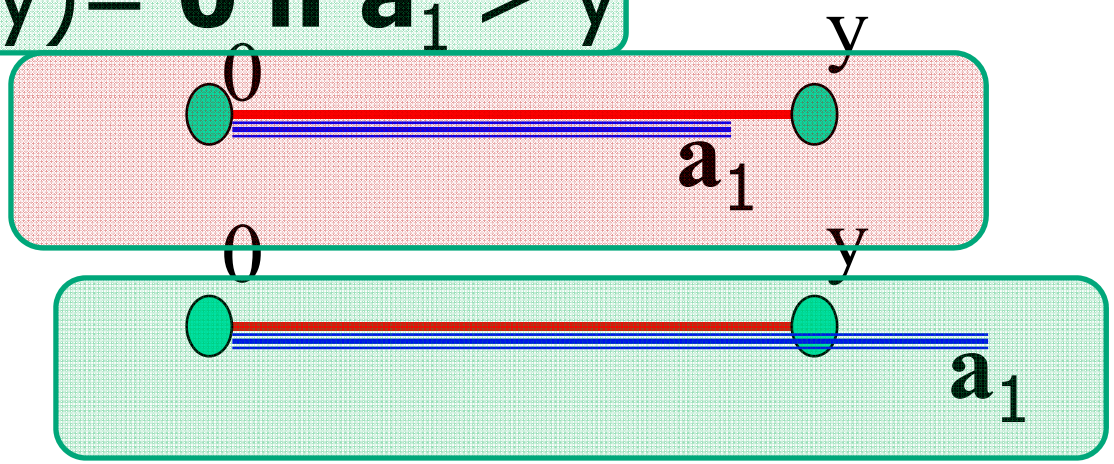
Δυναμικός Προγραμματισμός/0-1 Knapsack

Αρχική συνθήκη

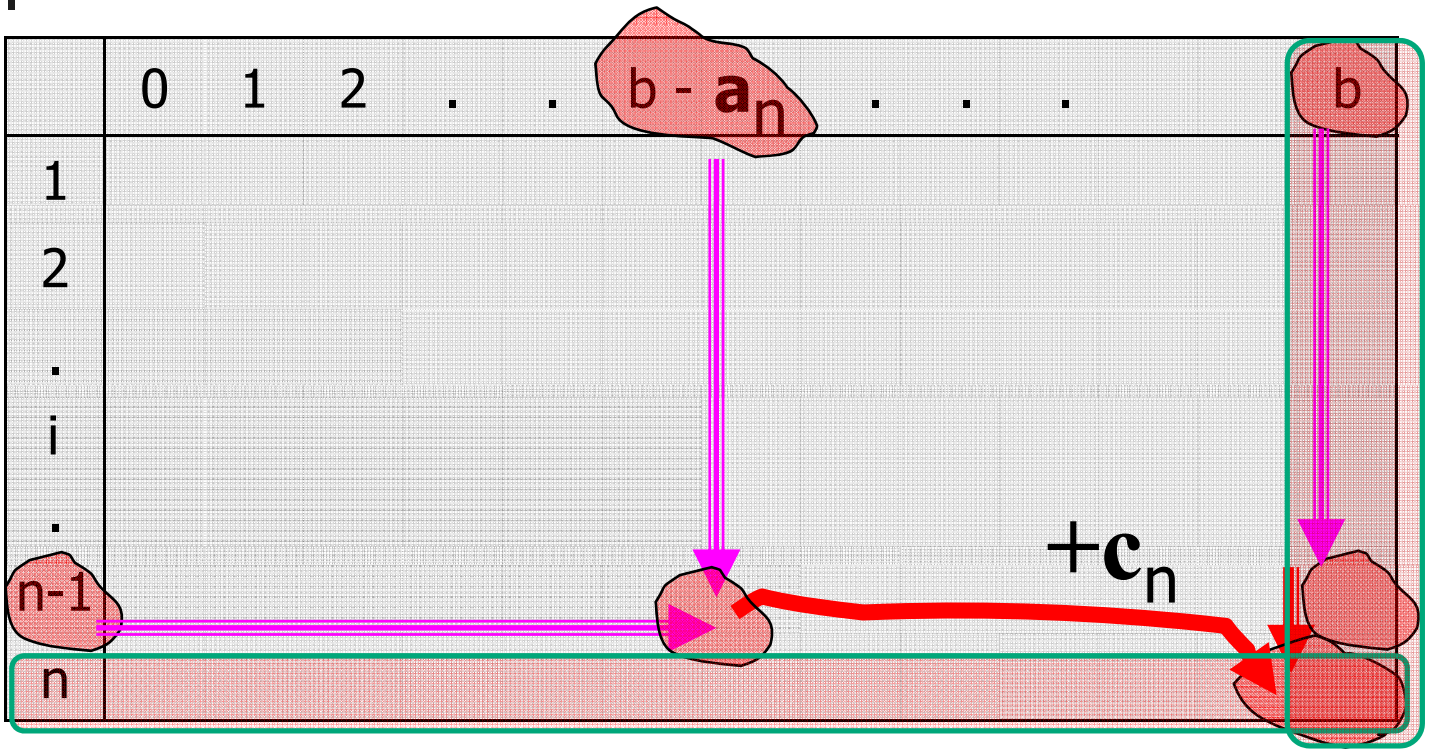
$f_1(y)$: τιμή βέλτιστης λύσης με ένα αντικείμενο

$f_1(y) = c_1$ **if** $a_1 \leq y$

$f_1(y) = 0$ **if** $a_1 > y$



Δυναμικός Προγραμματισμός (παράδειγμα) / υποπροβλήματα



$f_n(b)$: τιμή βέλτιστης λύσης με το πολύ n αντικείμενα

$f_{n-1}(b)$: τιμή βέλτιστης λύσης με το πολύ n-1 αντικείμενα

Δυναμικός Προγραμματισμός/αναδρομική σχέση

Παράδειγμα

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} \alpha_j x_j \leq b \\ & x_j \in \{0,1\} \end{aligned}$$

$$f_k(y) = \max\{\sum_{j=1}^{k} c_j x_j \mid \sum_{j=1}^{k} \alpha_j x_j \leq y, \quad x_j \in \{0,1\}, j = 1,\ldots,k\}$$

$$f_k(y) = \begin{cases} f_{k-1}(y) & \text{αν } y \leq \alpha_k \\ \max\{f_{k-1}(y), c_k + f_{k-1}(y-\alpha_k)\} & \text{διαφορετικά} \end{cases}$$

Δυναμικός Προγραμματισμός

$$\begin{cases} 0 \le y \le b \\ 1 \le k \le n \\ f_k(0) = 0 \end{cases}$$

βέλτιστη λύση $= f_n(b)$

- εύρεση λύσης:

$$x_k(y) = \begin{cases} 0 & \text{αν } f_k(y) = f_{k-1}(y) \\ 1 & \text{διαφορετικά} \end{cases}$$

Δυναμικός Προγραμματισμός (παράδειγμα) / εύρεση της τιμής / κατασκευή πίνακα

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

	0	1	2	3	4	5	6	7	8	**9**	10	11	12
1	0	0	0	0	0	0	0	0	0	**20**	**20**	**20**	**20**

Δυναμικός Προγραμματισμός (παράδειγμα)/ εύρεση της τιμής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	**20**	**20**	**20**	**20**
2	0	0	0	0	0	0	0	0	16	20	20	20	20

+16

+16

Δυναμικός Προγραμματισμός (παράδειγμα)/ εύρεση της τιμής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	**20**	**20**	**20**	**20**
2	0	0	0	0	0	0	0	0	16	20	20	20	20
3	0	0	0	0	0	0	**11**	**11**	16	20	20	20	20

+11

+11

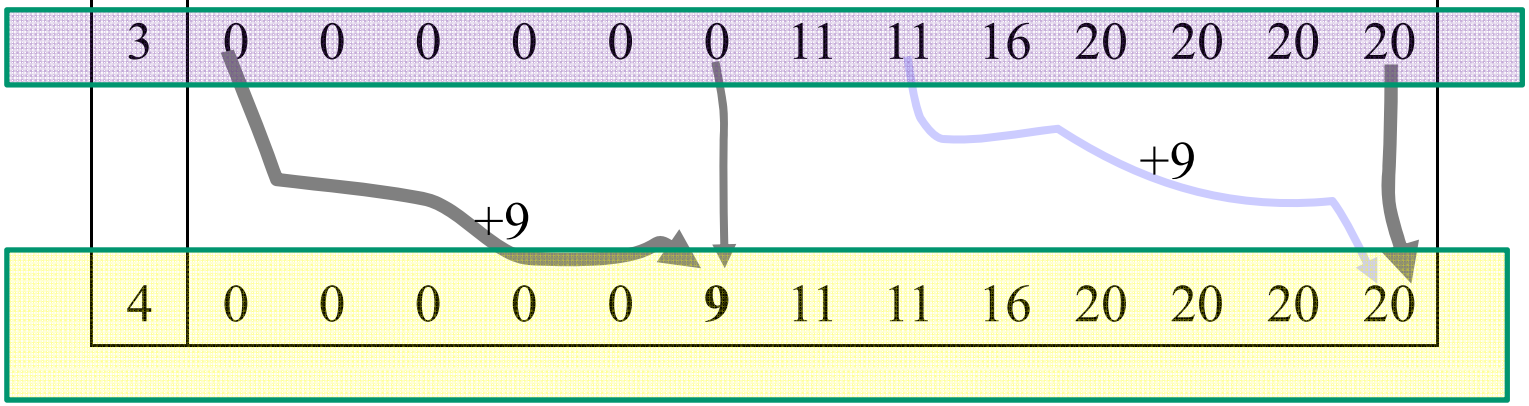
Δυναμικός Προγραμματισμός (παράδειγμα)/ εύρεση της τιμής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	**20**	**20**	**20**	**20**
2	0	0	0	0	0	0	0	0	16	20	20	20	20
3	0	0	0	0	0	0	11	11	16	20	20	20	20
4	0	0	0	0	0	**9**	11	11	16	20	20	20	20



Δυναμικός Προγραμματισμός (παράδειγμα) / εύρεση της τιμής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	**20**	**20**	**20**	**20**
2	0	0	0	0	0	0	0	0	16	20	20	20	20
3	0	0	0	0	0	0	11	11	16	20	20	20	20
4	0	0	0	0	0	9	11	11	16	20	20	20	20
								+7		+7			
5	0	0	0	0	7	9	11	11	16	20	20	20	**23**

Δυναμικός Προγραμματισμός (παράδειγμα) / εύρεση της τιμής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	**20**	**20**	**20**	**20**
2	0	0	0	0	0	0	0	0	**16**	20	20	20	20
3	0	0	0	0	0	0	**11**	**11**	16	20	20	20	20
4	0	0	0	0	0	**9**	11	11	16	20	20	20	20
5	0	0	0	0	**7**	**9**	11	11	16	20	20	20	**23**
6	0	**1**	**1**	**1**	7	9	11	**12**	16	20	**21**	**21**	23

initialization;

$f_1(y)$

for κ=2 to n do

for y=0 to b do

$$f_\kappa(y) = \max\{f_{\kappa-1}(y),\ f_{\kappa-1}(y-\mathbf{a}_k)+\mathbf{c}_k\}$$

Απομνημόνευσε αν το κ αντικείμενο επελέγη (στο πίνακα $x_\kappa(y)$ (1 αν ναι / 0 αν όχι))

endfor

endfor

Δυναμικός Προγραμματισμός (παράδειγμα) / εύρεση της δομής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

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

$x_6(12) = 0$

$x_5(12) = 1$

$x_4(8) = 0$

$x_2(8) = 1$

Δυναμικός Προγραμματισμός (παράδειγμα)/εύρεση της δομής

$$\max \quad z = 20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6$$

$$9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \le 12$$

$$x_j \in \{0,1\}$$

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

Δυναμικός Προγραμματισμός / αλγόριθμος σακιδίου/εύρεση δομής

```
k=n, y=b
while k>0 do
    if  x_κ(y) = 1  then
        print (k)
        y=y - a_κ
    endif
    k=k-1
endwhile
```

$y=y-a_2$

$y=y-a_5$

$y=b$

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

k-1

k=n

Δυναμικός Προγραμματισμός / 0-1 knapsack

- Πολυπλοκότητα:
  - $O(nb)$ χρόνος
  - $O(b)$ μνήμη
  - (pseudo-polynomial)