Il problema dello zaino 0-1 ‘intero’

Giuseppe Prencipe — Dipartimento di Informatica — Università di Pisa

Il problema dello zaino ‘intero’

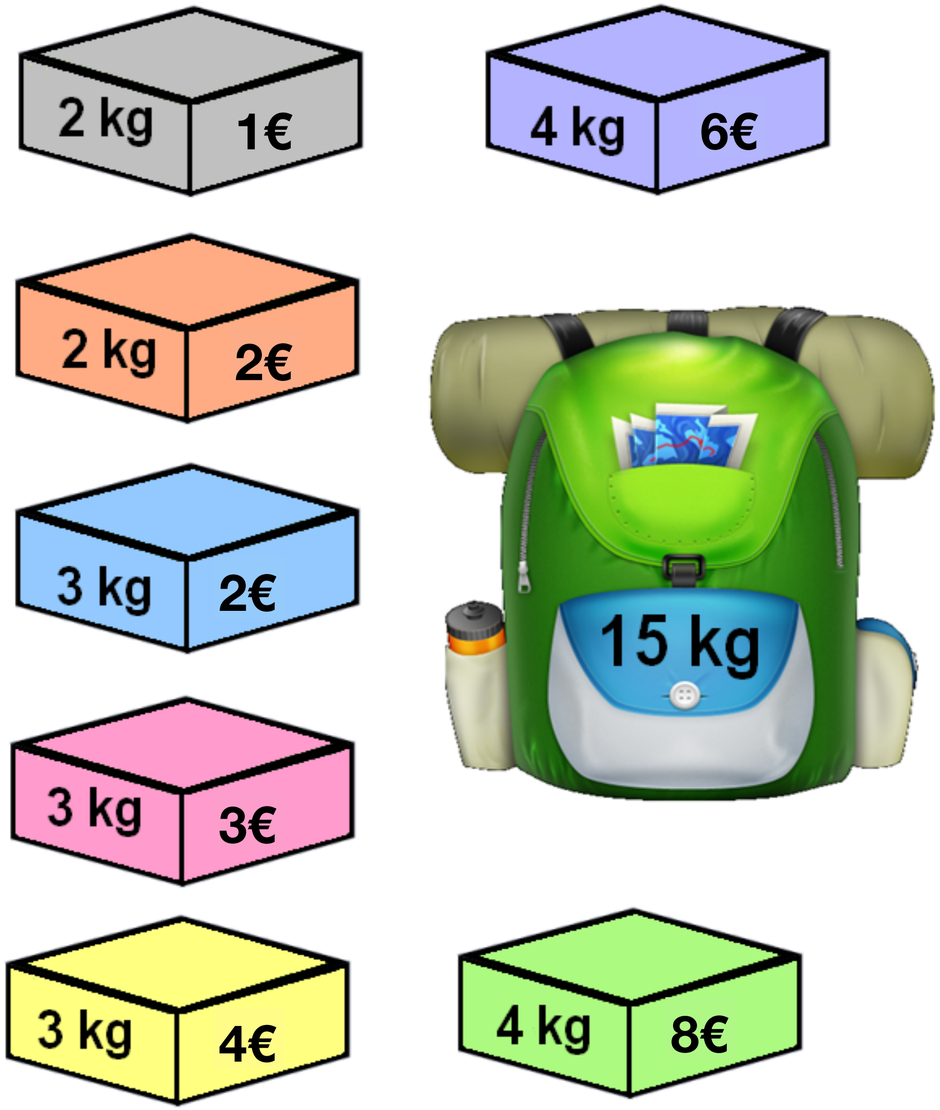
Abbiamo uno zaino e vogliamo riempirlo con un pò di oggetti

La nostra forza è (ahimè) limitata, e quindi riusciamo a trasportare al massimo un certo peso

Inoltre, ogni oggetto ha un certo valore

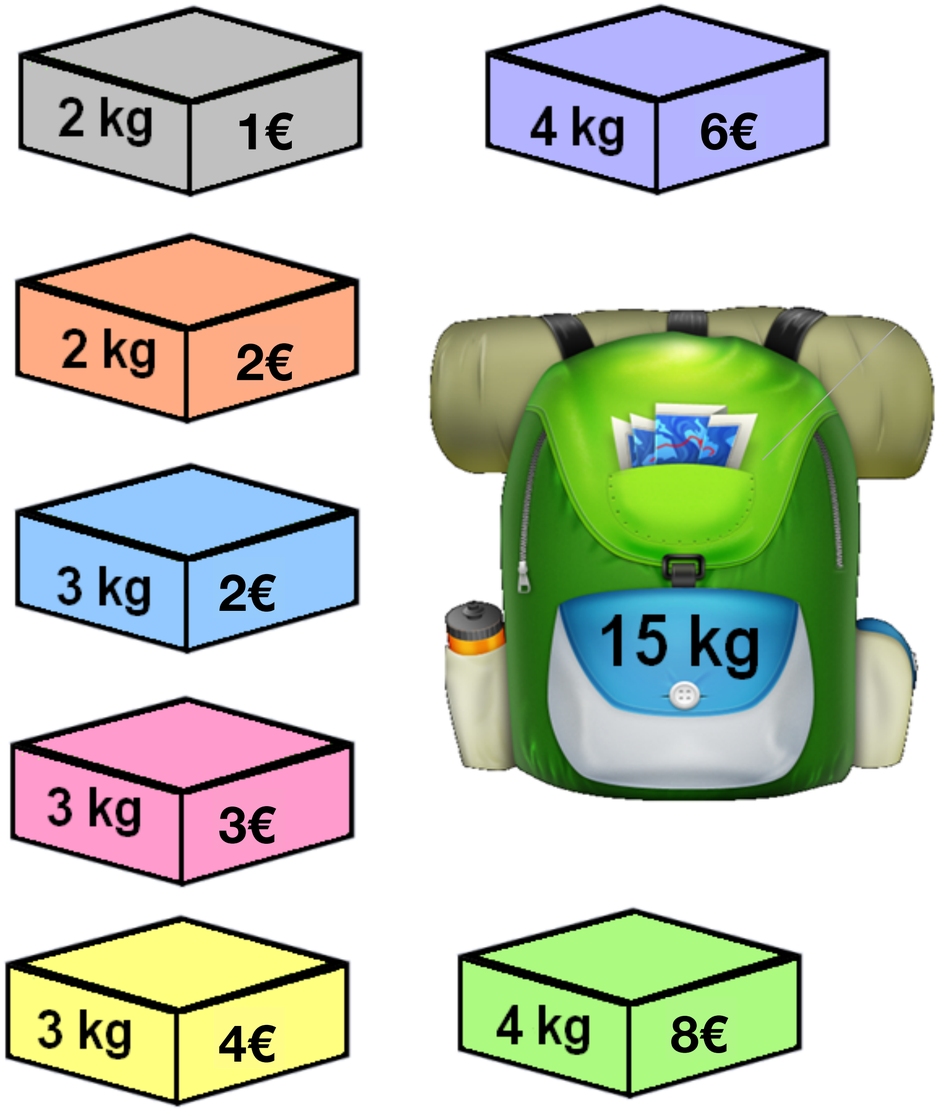
Quindi vogliamo trasportare oggetti in modo da massimizzare il valore del trasporto

Il problema dello zaino ‘intero’

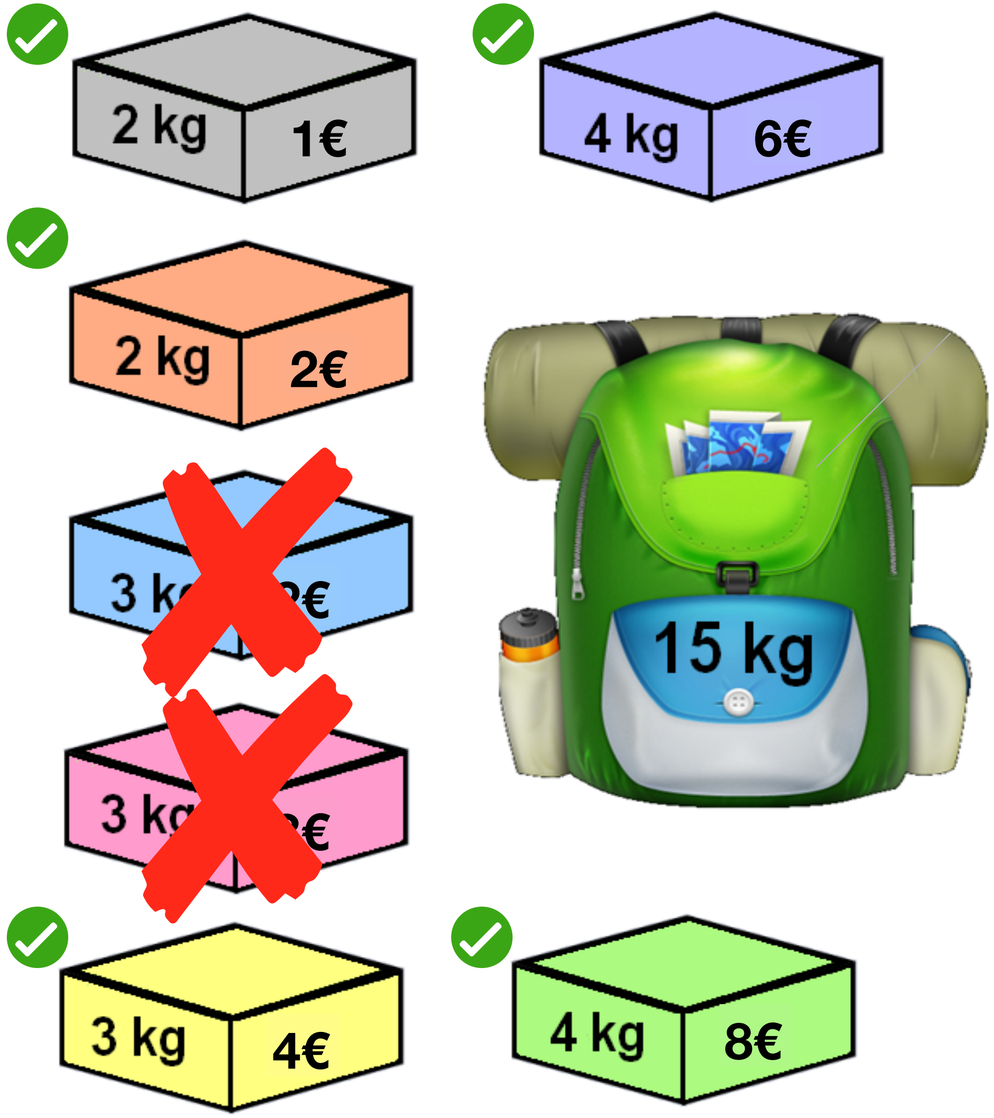


- Ogni oggetto ha peso e valore assegnati
- Massimizzare il valore
- Massimo peso trasportabile: 15 kg
- Padroneggiando ormai la Programmazione Dinamica, la usiamo per risolvere il problema!!!!

Soluzione????



Soluzione????



Il problema dello zaino ‘intero’

- Esiste la versione ‘non intera’ del problema, in cui è possibile prendere solo una parte dell’oggetto
- Nella formulazione 0-1 è possibile prendere o non prendere
  - Da qui il nome ‘intero’ o anche 0-1

Il problema dello zaino ‘intero’

- **Obiettivo**: **massimizzare il valore dello zaino** che può contenere al più peso W, scegliendo oggetti da una lista I_0, I_1, … I_{n-1}.
- Ogni oggetto ha 2 attributi:
  - **Valore** – sia v_i per l’oggetto I_i
  - **Peso** – sia w_i per l’oggetto I_i

Al solito….

- Forza brut(t)a
  - Esaminare tutte i possibili **(quanti sono?)** sottoinsiemi di n oggetti e scegliere il sottoinsieme con un peso ammissibile che massimizza il valore
  - Non ci piace, quindi proviamo con la Programmazione Dinamica

Al solito….

- Forza brut(t)a
  - Esaminare tutte i possibili 2^n sottoinsiemi di n oggetti e scegliere il sottoinsieme con un peso ammissibile che massimizza il valore
  - Non ci piace, quindi proviamo con la Programmazione Dinamica

Il problema dello zaino ‘intero’

Il problema dello zaino ‘intero’

- Proviamo a risolvere il problema in termini di sotto-problemi. Caratterizziamoli:

Il problema dello zaino ‘intero’

- Proviamo a risolvere il problema in termini di sotto-problemi.
  Caratterizziamoli:
  - Sia S_k la soluzione ottima per $\{I_0, I_1, \ldots, I_k\}$

Il problema dello zaino ‘intero’

- Proviamo a risolvere il problema in termini di sotto-problemi. Caratterizziamoli:
  - Sia S_k la soluzione ottima per $\{I_0, I_1, \ldots, I_k\}$
  - Quello che può accadere è che l’ottimo per $\{I_0, I_1, \ldots, I_{k+1}\}$ potrebbe non derivare dalla soluzione ottima per $\{I_0, I_1, \ldots, I_k\}$

Il problema dello zaino ‘intero’

- Proviamo a risolvere il problema in termini di sotto-problemi. Caratterizziamoli:
  - Sia S_k la soluzione ottima per $\{I_0, I_1, \ldots, I_k\}$
  - Quello che può accadere è che l’ottimo per $\{I_0, I_1, \ldots, I_{k+1}\}$ potrebbe non derivare dalla soluzione ottima per $\{I_0, I_1, \ldots, I_k\}$
  - In altre parole, la soluzione S_{k+1} potrebbe NON contenere elementi di S_k

Il problema dello zaino ‘intero’

Peso Massimo: 20

- L’ottimo per $\{I_0, I_1, I_2\}$ è $S_2 = \{I_0, I_1, I_2\}$
- MA l’ottimo per $\{I_0, I_1, I_2, I_3\}$ è $S_3 = \{I_0, I_2, I_3\}$.
  - Si noti che S_3 NON nasce da S_2
    - Piuttosto è costruita su $\{I_0, I_2\}$ che è l’ottimo per $\{I_0, I_1, I_2\}$ quando il peso massimo è 12

Elemento	Peso	Valore
I_0	3	10
I_1	8	4
I_2	9	9
I_3	8	11

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[?, ?],** con n il numero totale di elementi, e W il massimo peso ammissibile

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[n, W],** con n il numero totale di elementi, e W il massimo peso ammissibile
- Formulazione ricorsiva del problema può essere espressa come:

B[k,w] = ⎧ ⎫

In termini di problemi più piccoli….

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[n, W],** con n il numero totale di elementi, e W il massimo peso ammissibile
- Formulazione ricorsiva del problema può essere espressa come:

B[k,w] = ⎧
⎩

In termini di problemi più piccoli — analizziamo l’elemento k-esimo….o è parte della soluzione oppure no

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[n, W],** con n il numero totale di elementi, e W il massimo peso ammissibile
- Formulazione ricorsiva del problema può essere espressa come:

$$B[k,w] = \begin{cases} & \text{se } w_k > w \\ & \end{cases}$$

In termini di problemi più piccoli — analizziamo l’elemento k-esimo….o è parte della soluzione oppure no

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[n, W],** con n il numero totale di elementi, e W il massimo peso ammissibile
- Formulazione ricorsiva del problema può essere espressa come:

$$B[k,w] = \begin{cases} B[k-1, w] & \text{se } w_k > w \end{cases}$$

In termini di problemi più piccoli — analizziamo l’elemento k-esimo….o è parte della soluzione oppure no

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[n, W],** con n il numero totale di elementi, e W il massimo peso ammissibile
- Formulazione ricorsiva del problema può essere espressa come:

$$B[k,w] = \begin{cases} B[k-1, w] & \text{se } w_k > w \\ \text{altrimenti} \end{cases}$$

In termini di problemi più piccoli — analizziamo l’elemento k-esimo….o è parte della soluzione oppure no

Il problema dello zaino ‘intero’

- Quindi bisogna sistemare un pò le cose nella costruzione delle soluzioni ottime per i sotto-problemi
  - Sia **B[k, w]** il massimo valore della soluzione ottima S_k con peso w (quindi, per i primi k elem
  - L’obiettivo è determinare **B[n, W],** con n il numero totale di elementi, e W il massimo peso ammissibile
- Formulazione ricorsiva del problema può essere espressa come:

$$B[k,w] = \begin{cases} B[k-1, w] & \text{se } w_k > w \\ \max\{B[k-1, w], B[k-1, w - w_k] + v_k\} & \text{altrimenti} \end{cases}$$

In termini di problemi più piccoli — analizziamo l’elemento k-esimo….o è parte della soluzione oppure no

Il problema dello zaino ‘intero’ — formulazione ricorsiva

- L’ottimo S_k per i primi k elementi, per un peso massimo w, o contiene l’elemento k oppure no

$$B[k,w] = \begin{cases} B[k-1,w] & \text{se } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k] + v_k\} & \text{altrimenti} \end{cases}$$

- **Primo caso: $w_k > W$**
  - L’elemento ***k*** non può essere parte della soluzione, altrimenti il peso totale supererebbe w
- **Secondo caso: $w_k \leq W$**
  - L’elemento ***k*** potrebbe essere parte della soluzione finale

$$B[k,w] = \begin{cases} B[k-1,w] & \text{se } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k] + v_k\} & \text{altrimenti} \end{cases}$$

Il problema dello zaino ‘intero’ — soluzione

Cosa usiamo come **Tabella di Programmazione Dinamica**?

$$B[k,w] = \begin{cases} B[k-1,w] & \text{se } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k] + v_k\} & \text{altrimenti} \end{cases}$$

Il problema dello zaino ‘intero’ — soluzione

Cosa usiamo come **Tabella di Programmazione Dinamica**?

Matrice B: n x W

$$B[k,w] = \begin{cases} B[k-1,w] & \text{se } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k] + v_k\} & \text{altrimenti} \end{cases}$$

Il problema dello zaino ‘intero’ — soluzione

CASI BASE????

Matrice B: n x W

$$B[k,w] = \begin{cases} B[k-1,w] & \text{se } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k] + v_k\} & \text{altrimenti} \end{cases}$$

Il problema dello zaino ‘intero’ — soluzione

```
for w = 0 to W {  // Inizializzazione
    B[0,w] = 0
}
for i = 1 to n {  // Inizializzazione
    B[i,0] = 0
}
```

Matrice B: n x W

$$B[k,w] = \begin{cases} B[k-1,w] & \text{se } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k] + v_k\} & \text{altrimenti} \end{cases}$$

Il problema dello zaino ‘intero’ — soluzione

```
for w = 0 to W {  // Inizializzazione
    B[0,w] = 0
}
for i = 1 to n {  // Inizializzazione
    B[i,0] = 0
}
for i = 1 to n {
    for w = 0 to W {
     if wi <= w {  //elemento i potrebbe essere parte della soluzione
          if vi + B[i-1,w-wi] > B[i-1,w]
               B[i,w] = vi + B[i-1,w- wi]
          else
               B[i,w] = B[i-1,w]
     }
     else B[i,w] = B[i-1,w] // wi > w
    }
}
```

Matrice B: n x W

Il problema dello zaino ‘intero’

- Esempio:
  - n = 4 (# elementi)
  - W = 5 (peso massimo)
  - Elementi (peso, valore): (2,3), (3,4), (4,5), (5,6)

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0						
1						
2						
3						
4						

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0						
1						
2						
3						
4						

```
// Inizializzazione
for w = 0 to W
    B[0,w] = 0
for i = 1 to n
    B[i,0] = 0
```

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

```
// Inizializzazione
for w = 0 to W
    B[0,w] = 0
for i = 1 to n
    B[i,0] = 0
```

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0				
2	0					
3	0					
4	0					

$i = 1$
$v_i = 3$
$w_i = 2$
$w = 1$
$w\text{-}w_i = -1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione
if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$
$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$
else
$B[i,w] = B[i\text{-}1,w]$
else **B[i,w] = B[i-1,w]** // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3			
2	0					
3	0					
4	0					

$i = 1$
$v_i = 3$
$w_i = 2$
$w = 2$
$w\text{-}w_i = 0$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3		
2	0					
3	0					
4	0					

$i = 1$
$v_i = 3$
$w_i = 2$
$w = 3$
$w\text{-}w_i = 1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	
2	0					
3	0					
4	0					

$i = 1$
$v_i = 3$
$w_i = 2$
$w = 4$
$w\text{-}w_i = 2$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0					
3	0					
4	0					

$i = 1$
$v_i = 3$
$w_i = 2$
$w = 5$
$w\text{-}w_i = 3$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), **(3,4)**, (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0				
3	0					
4	0					

$i = 2$
$v_i = 4$
$w_i = 3$
$w = 1$
$w\text{-}w_i = -2$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione
  if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$
  $B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$
  else
  $B[i,w] = B[i\text{-}1,w]$
else **$B[i,w] = B[i\text{-}1,w]$** // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), **(3,4)**, (4,5), (5,6)

i / w	**0**	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3			
3	0					
4	0					

$i = 2$
$v_i = 4$
$w_i = 3$
$w = 2$
$w\text{-}w_i = -1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i-1,w-w_i] > B[i-1,w]$

$B[i,w] = v_i + B[i-1,w- w_i]$

else

$B[i,w] = B[i-1,w]$

else **B[i,w] = B[i-1,w]** // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), **(3,4)**, (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4		
3	0					
4	0					

$i = 2$
$v_i = 4$
$w_i = 3$
$w = 3$
$w\text{-}w_i = 0$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), **(3,4)**, (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	
3	0					
4	0					

$i = 2$

$v_i = 4$

$w_i = 3$

$w = 4$

$w\text{-}w_i = 1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), **(3,4)**, (4,5), (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0					
4	0					

$i = 2$

$v_i = 4$

$w_i = 3$

$w = 5$

$w\text{-}w_i = 2$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), **(4,5)**, (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4		
4	0					

$i = 3$

$v_i = 5$

$w_i = 4$

$w = 1..3$

$w\text{-}w_i = -3..-1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i-1,w-w_i] > B[i-1,w]$

$B[i,w] = v_i + B[i-1,w- w_i]$

else

$B[i,w] = B[i-1,w]$

else **B[i,w] = B[i-1,w]** // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), **(4,5)**, (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	
4	0					

$i = 3$

$v_i = 5$

$w_i = 4$

$w = 4$

$w\text{-}w_i = 0$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

$B[i,w] = B[i\text{-}1,w]$

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), **(4,5)**, (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0					

$i = 3$
$v_i = 5$
$w_i = 4$
$w = 5$
$w\text{-}w_i = 1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione
  if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$
  $B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$
  else
  $B[i,w] = B[i\text{-}1,w]$
else **$B[i,w] = B[i\text{-}1,w]$** // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), **(5,6)**

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	

$i = 4$
$v_i = 6$
$w_i = 5$
$w = 1..4$
$w\text{-}w_i = -4..-1$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione
  if $v_i + B[i-1,w-w_i] > B[i-1,w]$
  $B[i,w] = v_i + B[i-1,w- w_i]$
  else
  $B[i,w] = B[i-1,w]$
else **$B[i,w] = B[i-1,w]$** // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), **(5,6)**

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i = 4$
$v_i = 6$
$w_i = 5$
$w = 5$
$w\text{-}w_i = 0$

if $w_i <= w$ //elemento i potrebbe essere nella soluzione

if $v_i + B[i\text{-}1,w\text{-}w_i] > B[i\text{-}1,w]$

$B[i,w] = v_i + B[i\text{-}1,w\text{-}w_i]$

else

B[i,w] = B[i-1,w]

else $B[i,w] = B[i\text{-}1,w]$ // $w_i > w$

Il problema dello zaino ‘intero’ — esempio

(2,3), (3,4), (4,5), **(5,6)**

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	**7**

FINITO!!!!

Il valore massimo per questo zaino è 7!!!!

Il problema dello zaino ‘intero’

- Questo algoritmo trova solo il massimo valore possibile dato un peso massimo
  - Valore massimo in B[n,W]

- Per conoscere con quali ***elementi*** posso raggiungere il valore massimo, dobbiamo in qualche modo tracciarli nelle tabella

Il problema dello zaino ‘intero’ — trovare gli elementi dell’ottimo

- Siano $i = n$ e $k = W$

  if $B[i, k] \neq B[i-1, k]$ then

    marca l’elemento i-esimo come parte dello zaino

    $i = i-1$, $k = k-w_i$

  else

    $i = i-1$ // L’elemento i-esimo non è nello zaino

Trovare gli elementi dell'ottimo

Elementi: Zaino:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 4
k = 5
$v_i = 6$
$w_i = 5$
B[i,k] = 7
B[i-1,k] = 7

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi: Zaino:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 4
k = 5
$v_i = 6$
$w_i = 5$
B[i,k] = 7
B[i-1,k] = 7

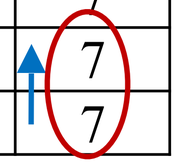
i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi: Zaino:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7



$i = 4$
$k = 5$
$v_i = 6$
$w_i = 5$
B[i,k] = 7
B[i-1,k] = 7

$i = n$, $k = W$
while i, k > 0
    if $B[i, k] \neq B[i-1, k]$ then
        marca i^{th} -esimo elemento come nello zaino
        $i = i-1$, $k = k-w_i$
    else
        $i = i-1$

Trovare gli elementi dell'ottimo

Elementi: Zaino:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 3
k = 5
$v_i = 5$
$w_i = 4$
B[i,k] = 7
B[i-1,k] = 7

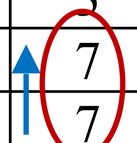
i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi: Zaino:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7



i = 3
k = 5
$v_i = 5$
$w_i = 4$
B[i,k] = 7
B[i-1,k] = 7

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi: Zaino:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 2
k = 5
$v_i = 4$
$w_i = 3$
B[i,k] = 7
B[i-1,k] = 3
$k - w_i = 2$

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

Zaino:
Elem 2

i / w	**0**	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 2
k = 5
$v_i = 4$
$w_i = 3$
B[i,k] = 7
B[i-1,k] = 3
$k - w_i = 2$

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

Zaino:
Elem 2

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i = 2$
$k = 5$
$v_i = 4$
$w_i = 3$
B[i,k] = 7
B[i-1,k] = 3
$k - w_i = 2$

$i = n$, $k = W$
while i, k > 0
    if $B[i, k] \neq B[i-1, k]$ then
        marca i^{th} -esimo elemento come nello zaino
        $i = i-1$, $k = k-w_i$
    else
        $i = i-1$

Trovare gli elementi dell'ottimo

Elementi:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

Zaino:
Elem 2

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 1
k = 2
$v_i = 3$
$w_i = 2$
B[i,k] = 3
B[i-1,k] = 0
$k - w_i = 0$

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

Zaino:
Elem 2
Elem 1

i / w	**0**	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 1
k = 2
$v_i = 3$
$w_i = 2$
B[i,k] = 3
B[i-1,k] = 0
$k - w_i = 0$

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

Zaino:
Elem 2
Elem 1

i / w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

i = 1
k = 2
$v_i = 3$
$w_i = 2$
B[i,k] = 3
B[i-1,k] = 0
$k - w_i = 0$

i = n , k = W
while i, k > 0
    if *B[i, k] ≠ B[i-1, k]* then
        marca i^{th} -esimo elemento come nello zaino
        i = i-1, k = k-w_i
    else
        i = i-1

Trovare gli elementi dell'ottimo

Elementi:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

Knapsack:
Elem 2
Elem 1

i / w	**0**	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i = 1$
$k = 2$
$v_i = 3$
$w_i = 2$
B[i,k] = 3
B[i-1,k] = 0
$k - w_i = 0$

$i = 0$, $k = 0$, e abbiamo **finito**!

Lo zaino ottimo contiene:

Elemento 1 e Elemento 2

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
    B[0,w] = 0

for i = 1 to n
    B[i,0] = 0

for i = 1 to n
    for w = 0 to W
```

Tempo di esecuzione?

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
    B[0,w] = 0

for i = 1 to n
    B[i,0] = 0

for i = 1 to n
    for w = 0 to W
      < …. >
```

Tempo di esecuzione?

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
   B[0,w] = 0

for i = 1 to n
   B[i,0] = 0

for i = 1 to n
   for w = 0 to W
    < …. >
```

$O(W)$

Tempo di esecuzione?

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
   B[0,w] = 0
```

O(*W*)

```
for i = 1 to n
   B[i,0] = 0
```

O(*n*)

```
for i = 1 to n
   for w = 0 to W
    < …. >
```

Tempo di esecuzione?

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
   B[0,w] = 0

for i = 1 to n
   B[i,0] = 0

for i = 1 to n
   for w = 0 to W
    < …. >
```

O(*W*)

O(*n*)

Ripeti *n* volte

Tempo di esecuzione?

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
   B[0,w] = 0
```

O(*W*)

```
for i = 1 to n
   B[i,0] = 0
```

O(*n*)

```
for i = 1 to n
   for w = 0 to W
    < …. >
```

Ripeti *n* volte
O(*W*)

Tempo di esecuzione?

Il problema dello zaino ‘intero’ — Tempo

```
for w = 0 to W
   B[0,w] = 0
```

O(*W*)

```
for i = 1 to n
   B[i,0] = 0
```

O(*n*)

```
for i = 1 to n
   for w = 0 to W
    < …. >
```

Ripeti *n* volte
O(*W*)

Tempo di esecuzione?

O(*n*W*)

Il problema dello zaino ‘intero’ — Esercizio

Riempire la tabella di programmazione dinamica per questi elementi

Determinare il valore ottimo e gli elementi che determinano l’ottimo

4 kg, 6€

1 kg, 3€

6 kg, 9€

3 kg, 5€

2 kg, 4€

Il problema dello zaino frazionario

Giuseppe Prencipe — Dipartimento di Informatica — Università di Pisa

Problema dello zaino frazionario

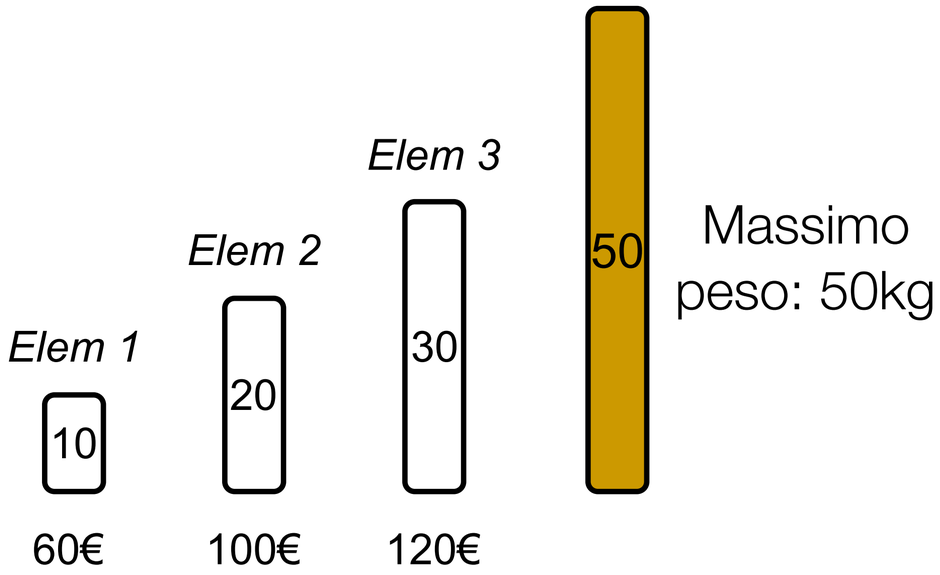
- Peso massimo: W
- Ci sono n elementi, ognuno con un valore v_i e peso w_i
- **Obiettivo**:
  - trovare x_i tali che per ogni $0 \leq x_i \leq 1, \quad i = 1, 2, .., n$

$\sum w_i x_i \leq W$ e

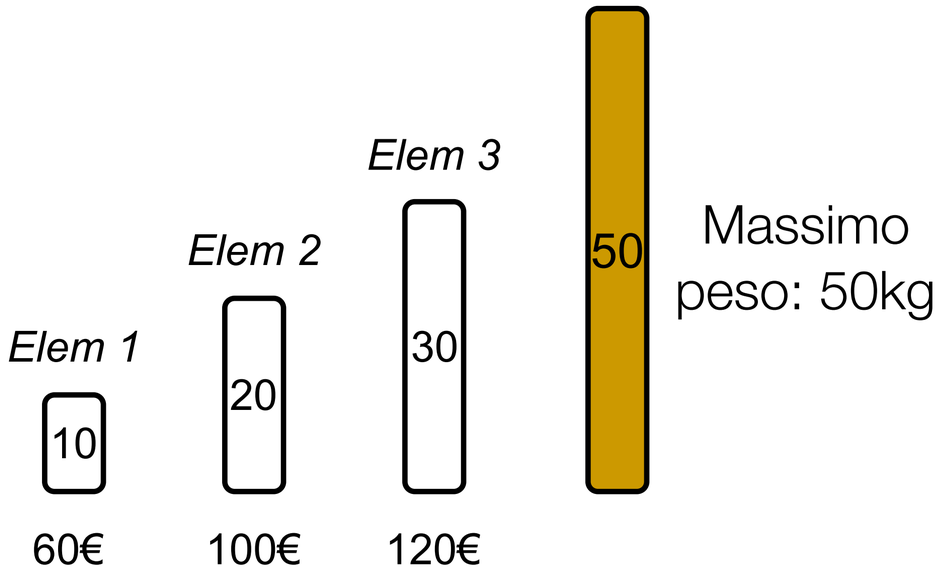
$\sum x_i v_i$ è massimo

Possiamo prendere frazioni degli oggetti

Problema dello zaino frazionario — esempio

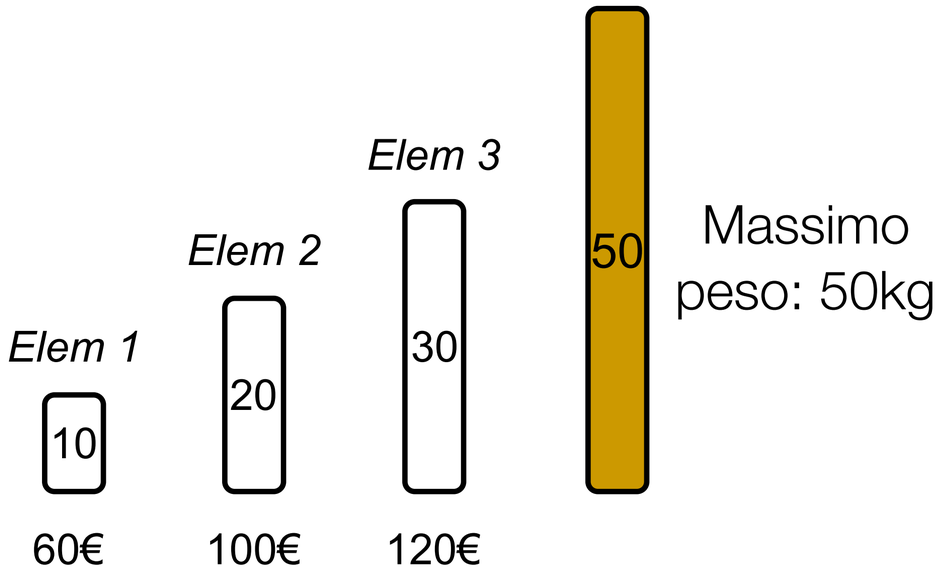


Problema dello zaino frazionario — esempio



Versione Intera

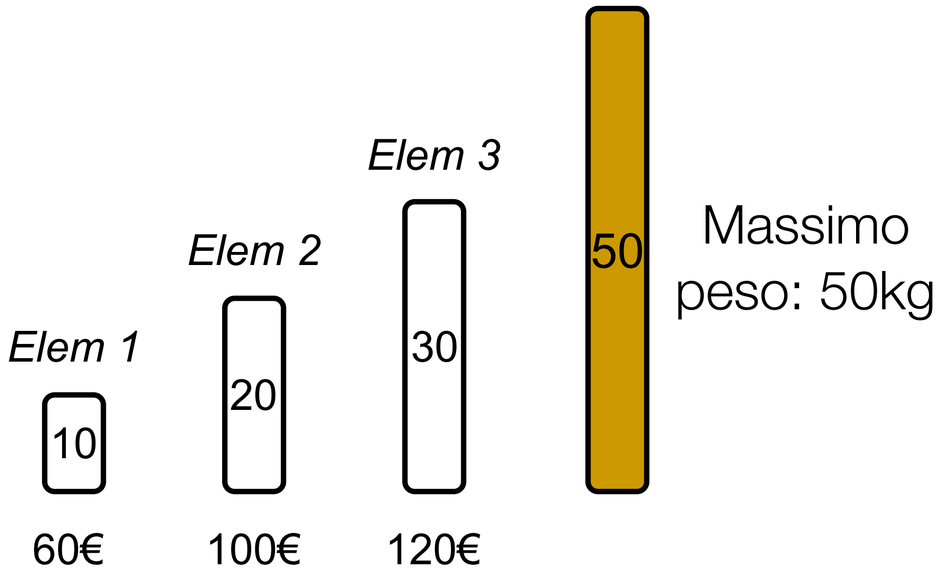
Problema dello zaino frazionario — esempio



Quali sono le possibilità da considerare (a occhio)?

Versione Intera

Problema dello zaino frazionario — esempio

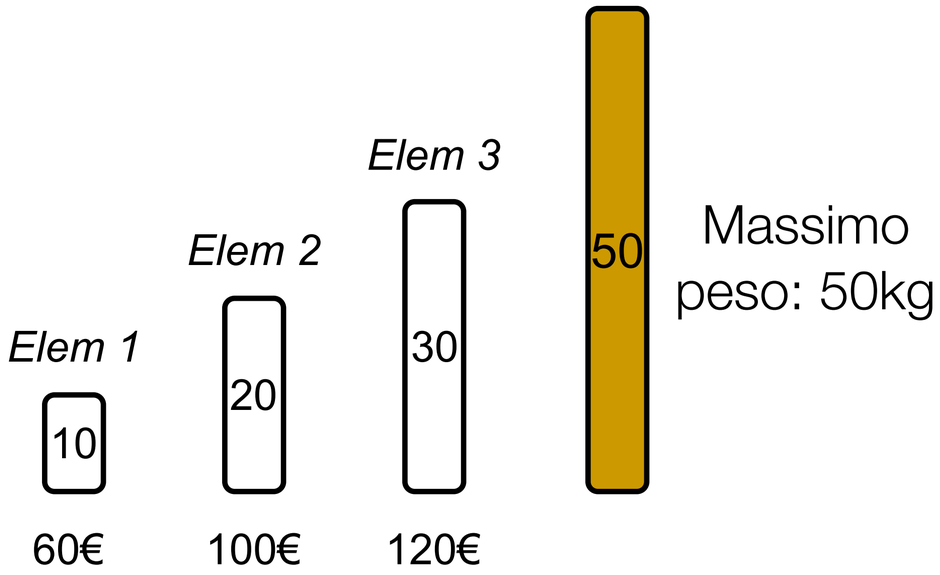


Massimo peso: 50kg

Quali sono le possibilità da considerare (a occhio)?

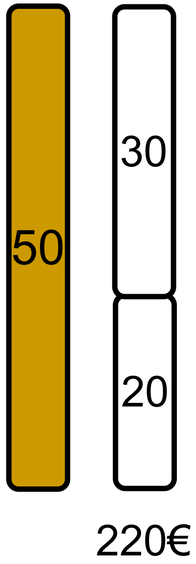
Versione Intera

Problema dello zaino frazionario — esempio

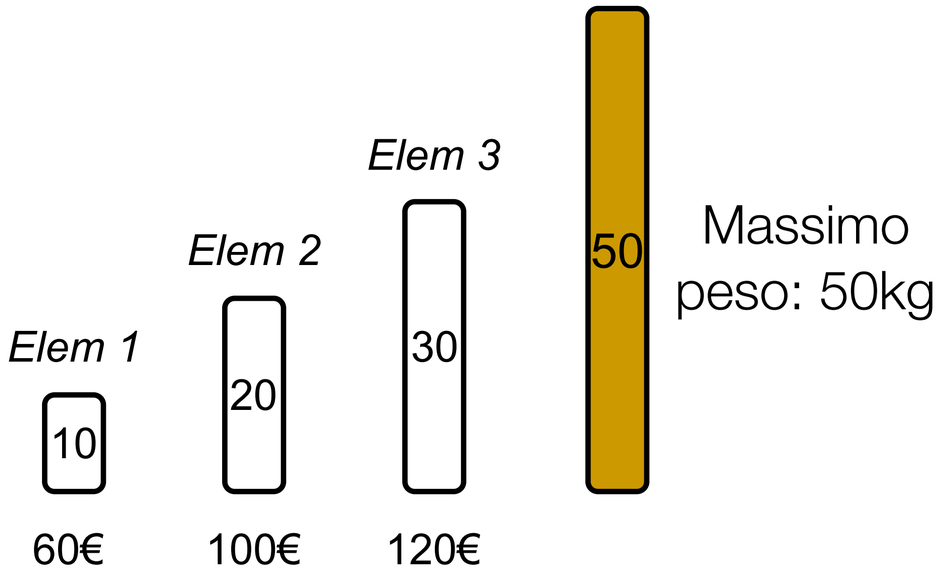


Quali sono le possibilità da considerare (a occhio)?

Versione Intera



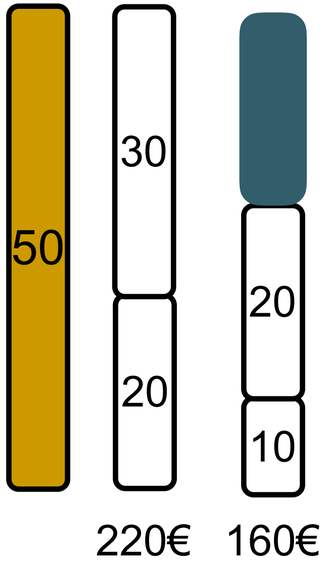
Problema dello zaino frazionario — esempio



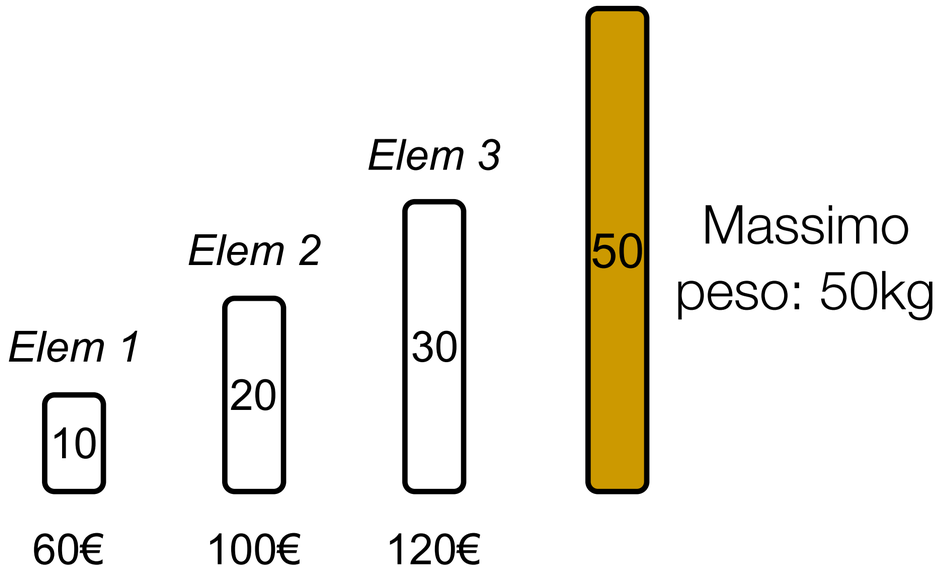
Massimo peso: 50kg

Quali sono le possibilità da considerare (a occhio)?

Versione Intera



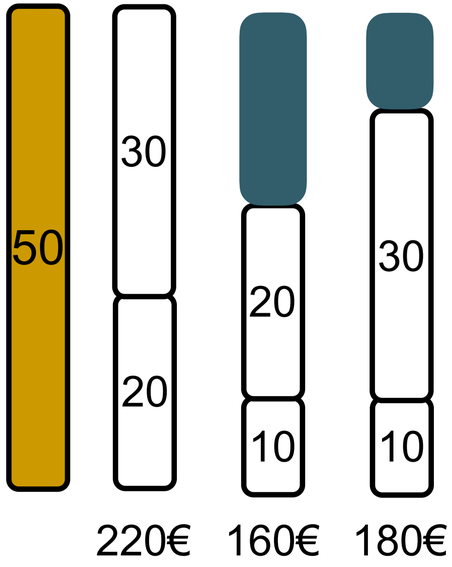
Problema dello zaino frazionario — esempio



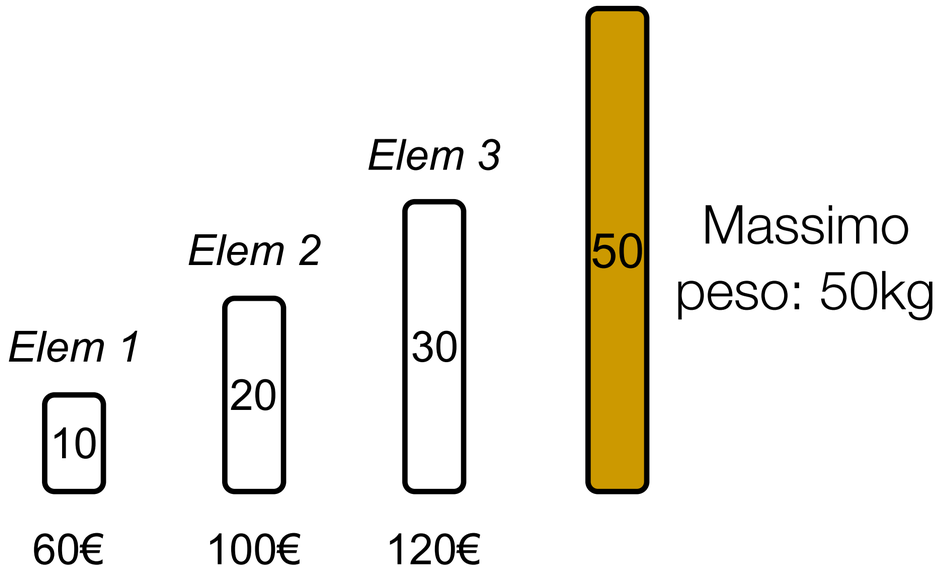
Massimo peso: 50kg

Quali sono le possibilità da considerare (a occhio)?

Versione Intera

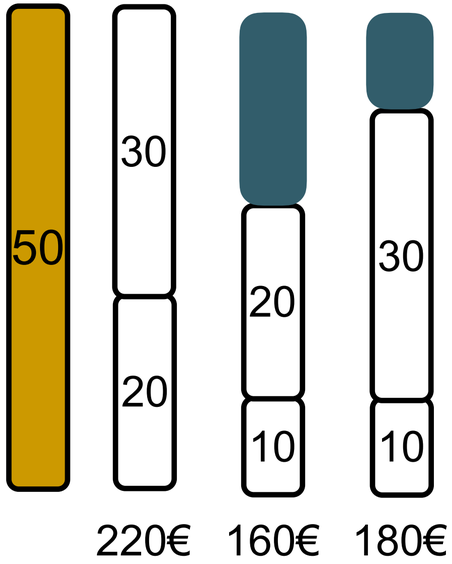


Problema dello zaino frazionario — esempio



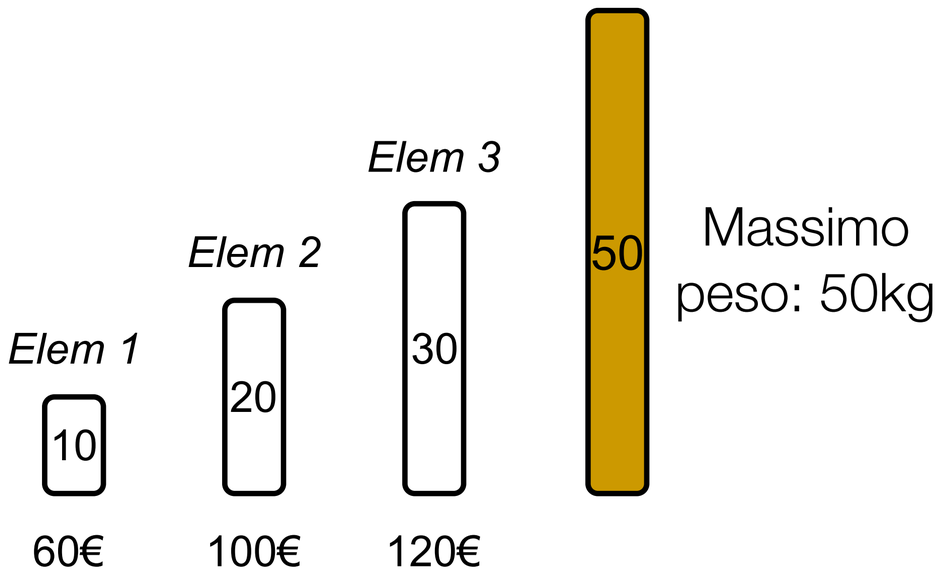
Massimo peso: 50kg

Versione Intera



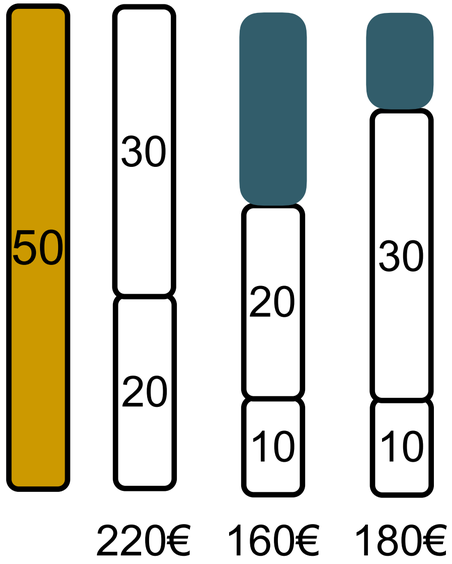
Versione Frazionaria

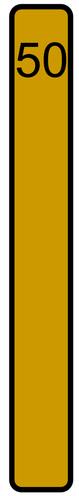
Problema dello zaino frazionario — esempio



Massimo peso: 50kg

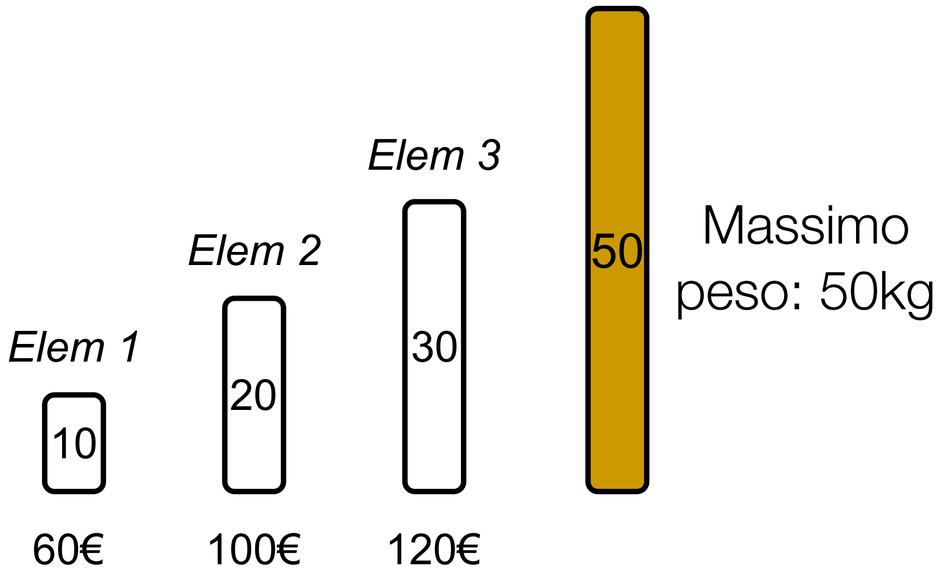
Versione Intera





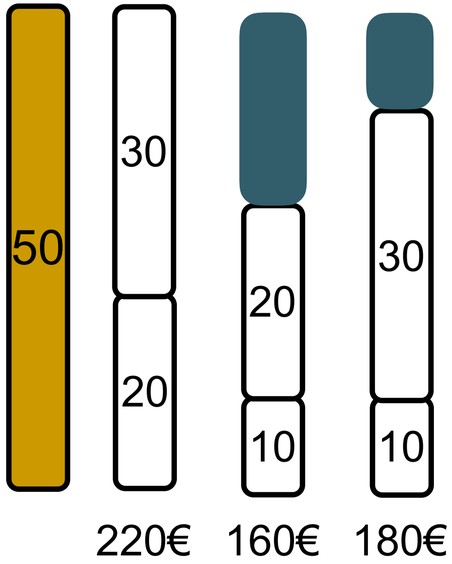
Versione Frazionaria

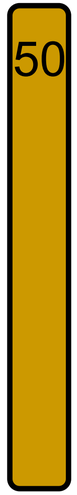
Problema dello zaino frazionario — esempio



Massimo peso: 50kg

Versione Intera

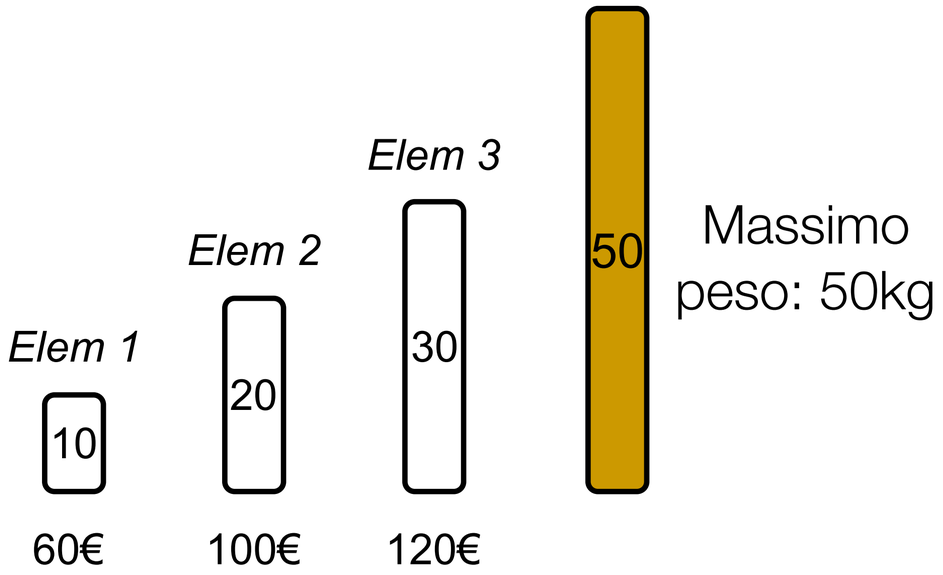




Versione Frazionaria

Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

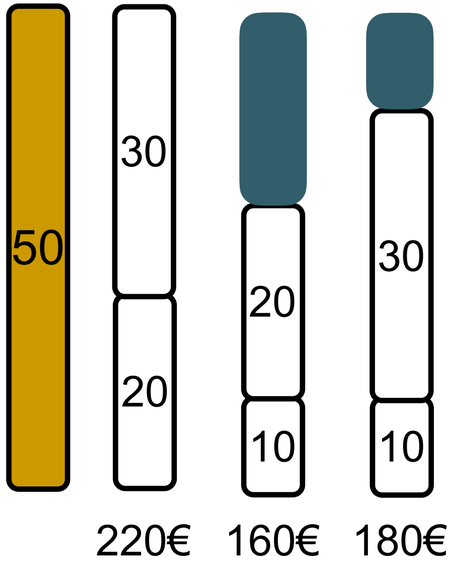
Problema dello zaino frazionario — esempio



Al supermercato quale sceglieresti?

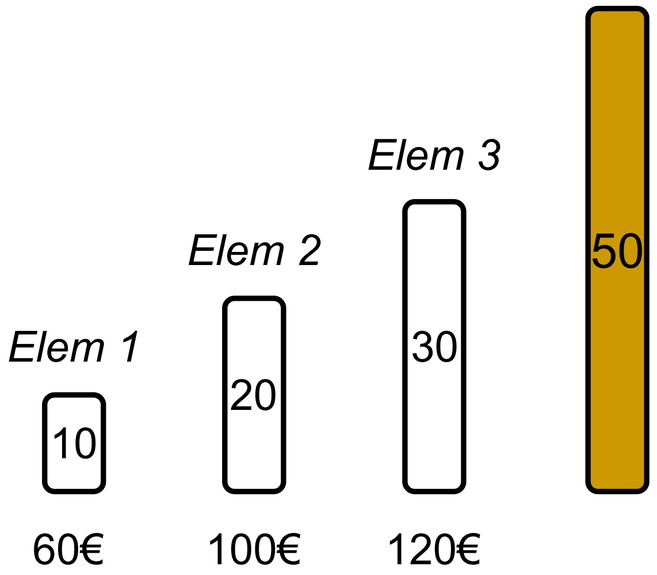
Versione Frazionaria

Versione Intera



Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

Problema dello zaino frazionario — esempio

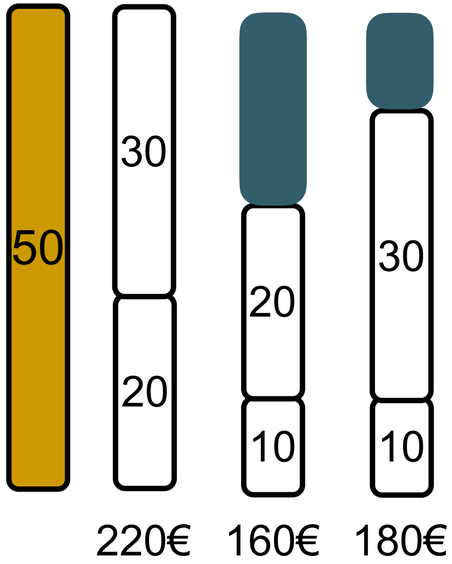


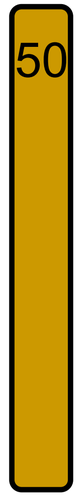
Massimo peso: 50kg

Al supermercato quale sceglieresti?

Elemento con il miglior rapporto costo al kg!

Versione Intera

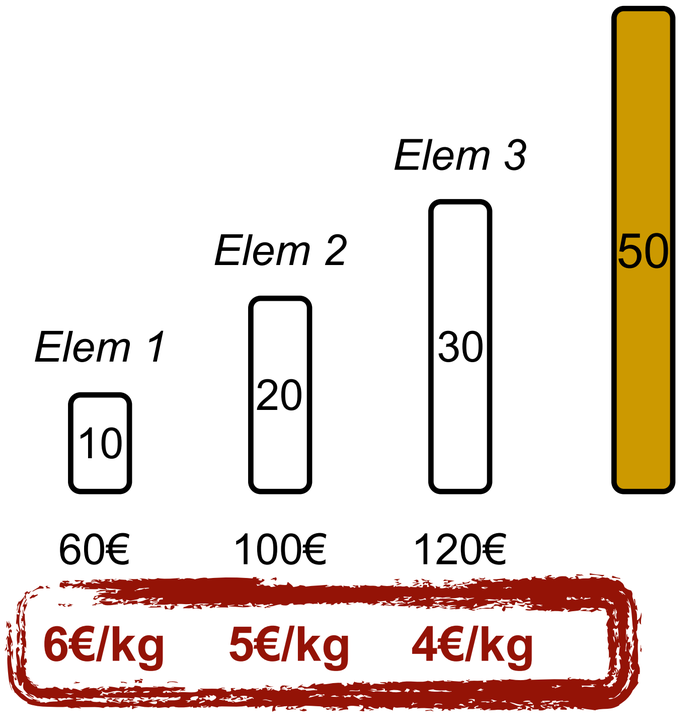




Versione Frazionaria

Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

Problema dello zaino frazionario — esempio

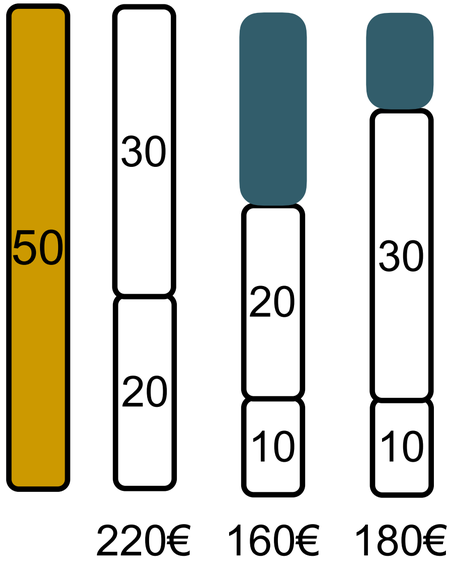


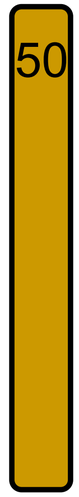
Massimo peso: 50kg

Al supermercato quale sceglieresti?

Elemento con il miglior rapporto costo al kg!

Versione Intera

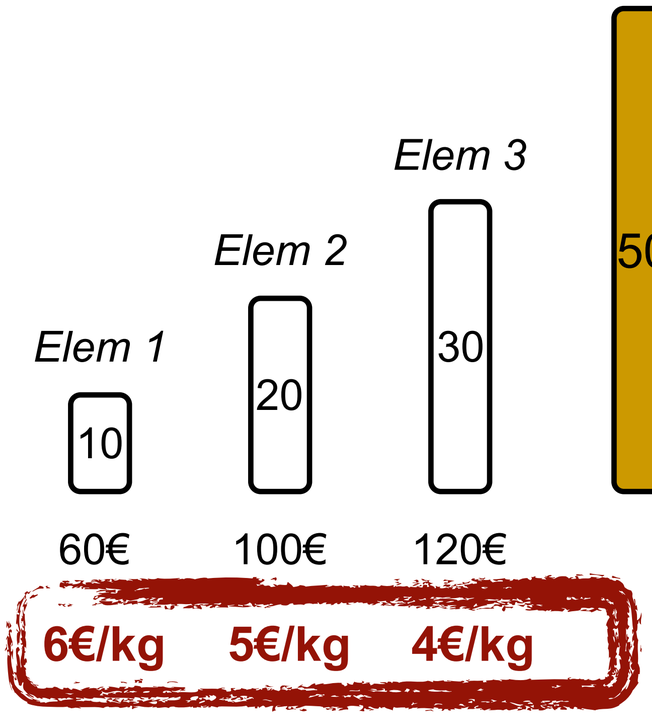




Versione Frazionaria

Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

Problema dello zaino frazionario — esempio

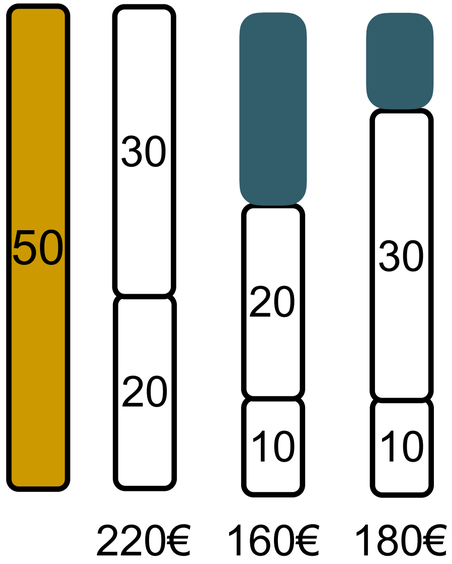


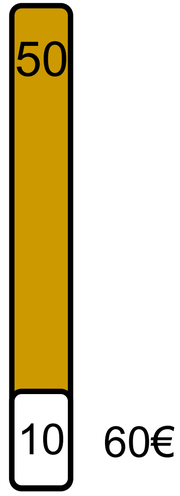
Massimo peso: 50kg

Al supermercato quale sceglieresti?

Elemento con il miglior rapporto costo al kg!

Versione Intera

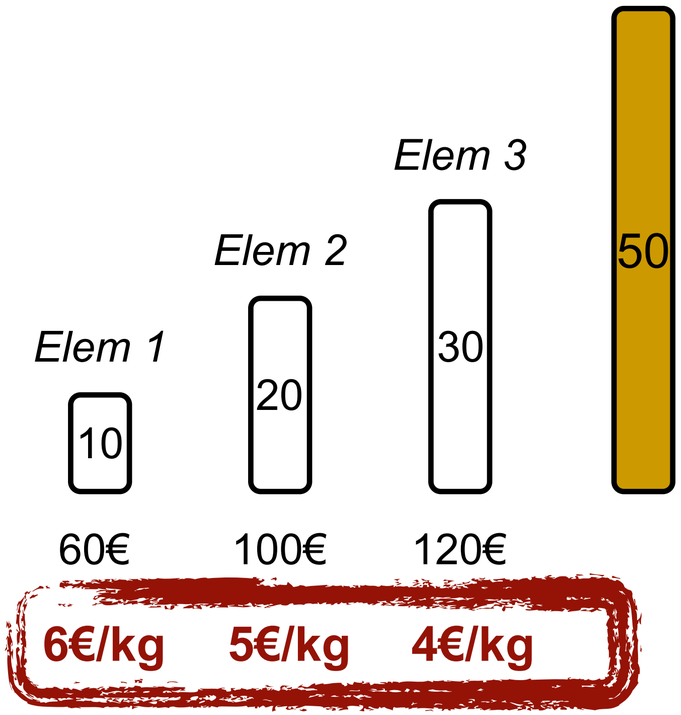




Versione Frazionaria

Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

Problema dello zaino frazionario — esempio

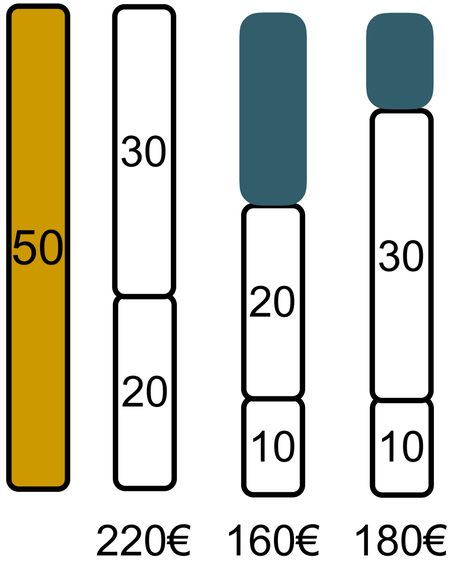


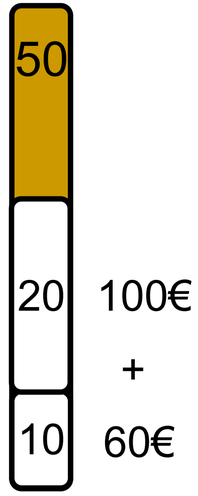
Massimo peso: 50kg

Al supermercato quale sceglieresteste?

Elemento con il miglior rapporto costo al kg!

Versione Intera

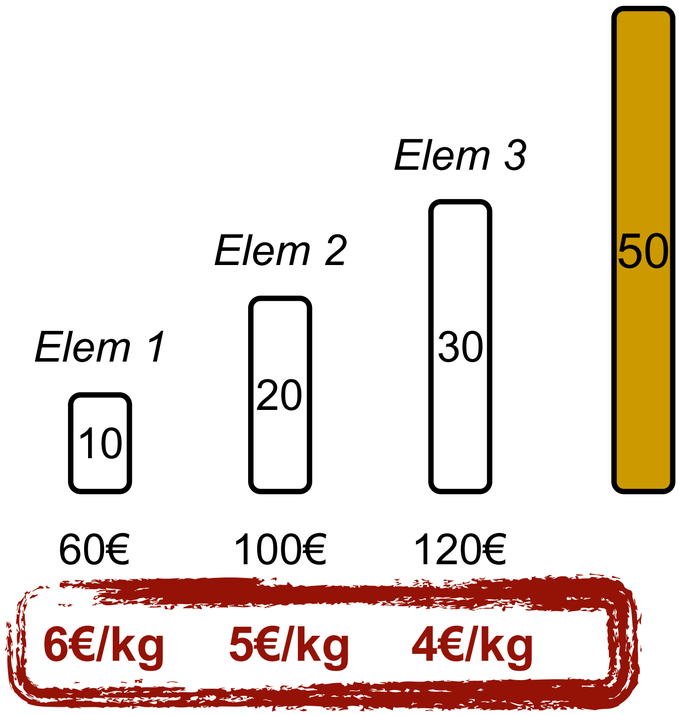




Versione Frazionaria

Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

Problema dello zaino frazionario — esempio

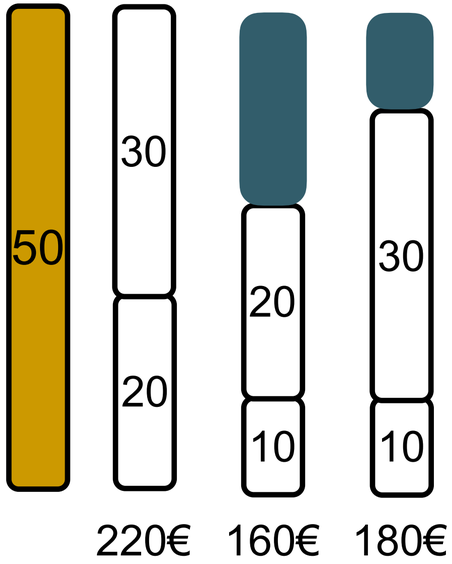


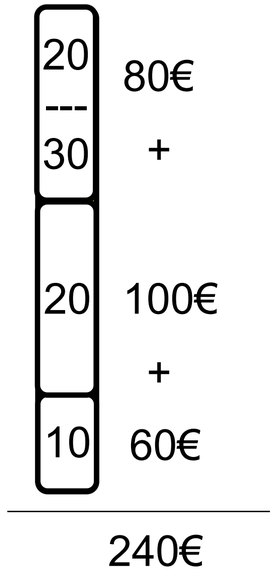
Massimo peso: 50kg

Al supermercato quale sceglieresteste?

Elemento con il miglior rapporto costo al kg!

Versione Intera





Versione Frazionaria

Strategia Greedy: scegliere l'ottimo locale….quale è l'ottimo locale?

Problema dello zaino frazionario

Strategia Greedy:

- Scegli elemento con il massimo valore per peso v_i/w_i

Problema dello zaino frazionario

Strategia Greedy:

- Scegli elemento con il massimo valore per peso v_i/w_i
- Se la quantità di quell'elemento termina, ed lo zaino non è anora al massimo peso possibile: prendi quanto possibile dal prossimo elemento con il migliore valore per peso

Problema dello zaino frazionario

Strategia Greedy:

- Scegli elemento con il massimo valore per peso v_i/w_i
- Se la quantità di quell'elemento termina, ed lo zaino non è anora al massimo peso possibile: prendi quanto possibile dal prossimo elemento con il migliore valore per peso
- Tipicamente si ordinano gli elementi in base al valore per peso:

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$$

Problema dello zaino frazionario

Zaino Frazionario (W, $v[n]$, $w[n]$)

1. While $w > 0$ e ci sono elementi da aggiungere
2. scegli elemento con il massimo v_i/w_i
3. $x_i \leftarrow \min(1, w/w_i)$
4. rimuovi elemento i dalla lista
5. $w \leftarrow w - x_i w_i$

- w – spazio rimanente nello zaino ($w = W$)
- **Tempo**: $\Theta(n)$ se gli elementi sono già ordinati, altrimenti $\Theta(n \lg n)$