

Il problema dello zaino (knapsack problem)

Moreno Marzolla
<http://www.moreno.marzolla.name/>

Copyright © 2010, 2011 Moreno Marzolla, Università di Bologna
(<http://www.moreno.marzolla.name/teaching/ASD2010/>)

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Problema dello zaino 0-1

- Supponiamo di avere un insieme X composto da n oggetti $\{x_1, x_2, \dots, x_n\}$
 - L'oggetto x_i ha peso $p(x_i)$ e valore $v(x_i)$
 - Esiste una unica istanza di ciascun oggetto x_i
- Disponiamo un contenitore in grado di trasportare al massimo un peso pari a P
- Vogliamo determinare un sottoinsieme $Y \subseteq X$ di oggetti, tale che
 - Il peso complessivo degli oggetti in Y sia $\leq P$
 - Il valore complessivo degli oggetti in Y sia il massimo possibile

Definizione formale

- Vogliamo determinare un sottoinsieme $Y \subseteq X$ di oggetti tale che:

$$\sum_{x \in Y} p(x) \leq P$$

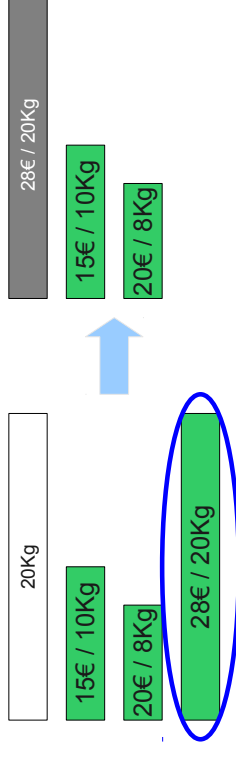
e tale da massimizzare il valore complessivo:

$$\sum_{x \in Y} v(x)$$

Approccio greedy #1

- Ad ogni passo, scelgo tra gli oggetti non ancora nello zaino quello di **valore** massimo, tra tutti quelli che hanno un peso minore o uguale alla capacità residua dello zaino
 - Nota: questo algoritmo **non** fornisce sempre la soluzione ottima

Esempio

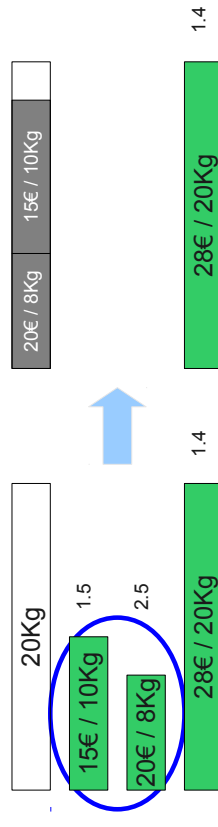


- La soluzione calcolata in questo esempio non è la soluzione ottima!

Approccio greedy #2

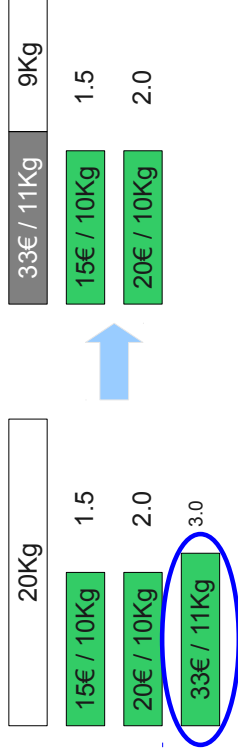
- Ad ogni passo, scelgo tra gli oggetti non ancora nello zaino quello di **valore specifico** massimo, tra tutti quelli che hanno un peso minore o uguale alla capacità residua dello zaino
 - Il **valore specifico** è definito come il valore di un oggetto diviso il suo peso

Esempio



- In questo esempio, l'algoritmo greedy #2 calcola la soluzione ottima, ma...

Esempio



- ...in questo altro esempio anche l'algoritmo greedy #2 non produce la soluzione ottima

Soluzione ottima basata sulla Programmazione Dinamica

- Questa soluzione funziona esclusivamente se i pesi sono **numeri interi**
- Sia $V(i,j)$ il **massimo valore** ottenibile da un sottoinsieme degli oggetti $\{x_1, x_2, \dots, x_j\}$ in uno zaino che ha capacità massima j
 - $i=1, 2, \dots, n$
 - $j=0, 1, \dots, P$

Soluzione ottima basata sulla Programmazione Dinamica

- Sia $V(i,j)$ il **massimo valore** ottenibile da un sottoinsieme degli oggetti $\{x_1, x_2, \dots, x_j\}$ in uno zaino che ha capacità massima j
 - $V(i,0) = 0$ per ogni $i=1..n$
 - Se il peso massimo è zero, il valore è sempre zero
 - $V(1,j) = v(x_1)$ se $j \geq p(x_1)$
 - C'è abbastanza spazio per l'oggetto x_1
 - $V(1,j) = 0$ se $j < p(x_1)$
 - Non c'è abbastanza spazio per l'oggetto x_1

Soluzione ottima basata sulla Programmazione Dinamica

- Per calcolare $V(i,j)$ distinguiamo due casi
 - Se $j < p(x_i)$ significa che l' i -esimo oggetto è troppo pesante per essere contenuto nello zaino. In tal caso $V(i,j) = V(i-1,j)$
 - Se $j \geq p(x_i)$ possiamo scegliere la migliore tra le seguenti possibilità
 - Inserire l'oggetto x_i nello zaino. Lo spazio residuo è $j-p(x_i)$, il valore massimo ottenibile in questa ipotesi è $V(i-1, j-p(x_i)) + v(x_i)$
 - Non inserire l'oggetto x_i nello zaino. Il valore massimo ottenibile in questa ipotesi è $V(i-1, j)$
 - Scegliendo la migliore delle possibilità, possiamo dire $V(i,j) = \max\{V(i-1, j), V(i-1, j-p(x_i)) + v(x_i)\}$

Soluzione ottima basata sulla Programmazione Dinamica

- Riassumendo

$$V(i, j) = \begin{cases} V(i-1, j) & \text{se } j < p(x_i) \\ \max\{V(i-1, j), V(i-1, j-p(x_i)) + v(x_i)\} & \text{se } j \geq p(x_i) \end{cases}$$

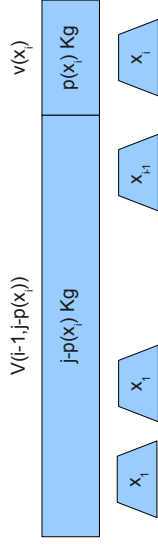


Tabella di programmazione dinamica / matrice V

$$p = [2, 7, 6, 4]$$

$$v = [12.7, 6.4, 1.7, 0.3]$$

	0	1	2	3	4	5	6	7	8	9	10
1	0.0	0.0	12.7	12.7	12.7	12.7	12.7	12.7	12.7	12.7	12.7
2	0.0	0.0	12.7	12.7	12.7	12.7	12.7	12.7	12.7	19.1	19.1
3	0.0	0.0	12.7	12.7	12.7	12.7	12.7	12.7	14.4	19.1	19.1
4	0.0	0.0	12.7	12.7	12.7	12.7	13.0	14.4	14.4	19.1	19.1

Tabella di programmazione dinamica / matrice V

$$p = [2, 7, 6, 4]$$

$$v = [12.7, 6.4, 1.7, 0.3]$$

	0	1	2	3	4	5	6	7	8	9	10
1	0.0	0.0	12.7	12.7	12.7	12.7	12.7	12.7	12.7	12.7	12.7
2	0.0	0.0	12.7	12.7	12.7	12.7	12.7	12.7	12.7	19.1	19.1
3	0.0	0.0	12.7	12.7	12.7	12.7	12.7	12.7	14.4	19.1	19.1
4	0.0	0.0	12.7	12.7	12.7	12.7	13.0	14.4	14.4	19.1	19.1

$$V(3,8) = \max\{V(2,8), V(2,8-p[3]) + v[3]\}$$

$$= \max\{12.7, 14.4\}$$

Stampare la soluzione

- Il massimo valore che è possibile inserire nello zaino rispettando il vincolo di peso massimo è contenuto nella cella $V(n,P)$ della tabella di programmazione dinamica
- Come facciamo a sapere quali oggetti fanno parte della soluzione ottima?
 - Usiamo una tabella ausiliaria booleana $K(i,j)$ che ha le stesse dimensioni di $V(i,j)$
 - $K(i,j) = \text{true}$ se e solo se l'oggetto x_i fa parte della soluzione ottima che ha valore complessivo $V(i,j)$

Tabella di programmazione dinamica / stampa soluzione ottima

```
d := p;  
i := n;  
while ( i > 0 ) do  
  if ( K(i,d) = true ) then  
    stampa "Seleziono oggetto " i  
    d := d-p(i);  
  endif  
  i := i-1;  
endwhile
```

p=[2, 7, 6, 4]
v=[12.7, 6.4, 1.7, 0.3]

	0	1	2	3	4	5	6	7	8	9	10
1	F	F	T	T	T	T	T	T	T	T	T
2	F	F	F	F	F	F	F	F	F	T	T
3	F	F	F	F	F	F	F	F	T	F	F
4	F	F	F	F	F	F	T	T	F	F	F