PROBLEMA DI KNAPSACK

$\sum_{i \in S} w_i \leq L$ $\quad$ $\max \sum_{i \in S} v_i$

Soluzione: $S \subseteq O$ t.c.

$O = \{1, 2, 3, 4, 5\}$ = "oggetti"

$V = \{1, 6, 18, 22, 28\}$ = "valore"

$W = \{1, 2, 5, 6, 7\}$ = "peso"

Voglio trovare la migliore comb. di oggetti che:

- stiano nello zaino
- abbiano valore massimo

L = 11 ⇒ quantità di "peso" trasportabile.

- esempio possibili sottoinsiemi:

$\{5, 2, 1\} \rightarrow W_1 = 7+2+1 = 10$; $V_1 = 28+6+1 = 35$;

$\{3, 4\} \rightarrow W_2 = 6+5 = 11$; $V_2 = 22,28 = 40$;

FORMALIZZAZIONE:

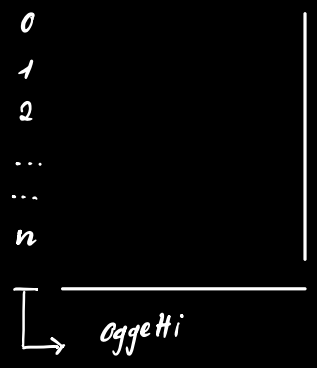
$O = \{O_1, O_2, \dots O_n\}$ $\quad V = \{V_1, V_2 \dots V_n\}$ $\quad W = \{W_1, W_2 \dots W_n\}$

L = "peso limite".

Soluzione: $S \subseteq O$ $\quad \sum_{i \in S} w_i \leq L$, $\max \sum_{i \in S} v_i$;

- Ragionamento su O_i (generico):
  - $O_i \in S \rightarrow$ aggiungere O_i crea una diramazione
    - $S_P = S_A + \overbrace{(w_i + v_i)}^{O_i}$
    - S_A
  - $O_i \notin S$

- Ha senso aggiungere O_i? Dipende da cosa ho ottenuto fino a S_A.
  - ▷ * ho ancora peso disponibile?



$M[i,J]$ = "qual è il valore massimo che posso ottenere scegliendo i primi i oggetti, x ottenere il peso J"

Come lo calcolo?

SE SONO IN POSIZIONE U AD ESEMPIO:

W_i	V_i	0	1	2	3	4	5	6	7	8	9	10	L
		0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	1	1	1	1
2	6	2	0	1	6	7	7	7	7	7	7	7	7
5	18	3	0	1	6	7	7	18	19	24	25	25	25
6	22	4	0	1	6	7	7	18	22	24	28	29	29	40
7	28	5	0	1	6	7	7	18	22	28	29	34	34	40

→ C. BASE

C. BASE

OTTIMO! = M[m,W]

o = "replico il risultato perché ho raggiunto la somma Vi migliore".

o = "prendo l'oggetto 4 per arrivare al peso 7." → "lo combino con il primo oggetto COMPATIBILE di valore migliore."

$L_7 - W_4 = 7 - 6 = 1$

Peso compatibile

o = "non prendo l'oggetto 4, allora la soluzione la trovo in un sottoinsieme dei primi 3 oggetti con peso ≤ 7".

In "caselle":

Peso O_i

O_i lo prendi? → Sì, somma V_i a $M[i-1, J-W_i]$

"peso rimanente se prendo O_i"

primi $i-1$ oggetti

migliore peso $O_{ij} + O_{i-1,J-W_i}$

→ No, → risultato = miglior valore M[i,J] prendendo i-1 oggetti con limite di peso = J.

VANTAGGIO: una volta che interviene un numero e ottengo il "meglio che posso fare" me lo trascino nella riga

* Quanto è il tempo? → $\Theta(n \cdot m)$
  - o max $(n \cdot m)$ perché è il numero di caselle da calc.
  - o min $(n \cdot m)$ perché x ogni casella ↓ ne guardo un numero finito.

n = "numero di oggetti"

m = "numero pesi possibili" = L

NP-COMPLETEZZA KNAPSACK → Sì è NP-COMPLETO.

Occhio! Non esiste un algoritmo con tempi polinomiali x 1 probl. np-completo.

PSEUDO-POLINOMIALE → $T(n)$?

"Un algoritmo è polinomiale, se lo è rispetto alla dimensione dell'input".

$\Theta(n \cdot L)$ → Se fosse 100 Pesi! 100? Quanti bit?

$2^7 \rightarrow 128$

$\log_2(n)$ → QUANTI BIT

↓

QUESTO non è POLINOMIALE!

$\Theta(n \cdot L)$ → $\{1,2,3,4,5 \dots\}$ → quanti bit?

0000
0001
0010
0011
...

→ n bit

Quindi non è polinomiale.

$T(n, L) = \Theta(2^n \cdot (\log_2 n))$

n L

* FORMALIZZAZIONE DELLA SOLUZIONE:

① Variabile:

$M[i,J]$ = "massimo valore che ottengo considerando i primi i oggetti e limite di peso J".

② Caso base:

$$M[0][J] = 0 \quad \forall\ 0 \le J \le L \qquad \wedge \qquad M[i,0] = 0 \quad \forall\ 0 \le i \le n$$

③ Caso passo

$$M[i,J] = \begin{cases} \text{se } J - W_i < 0 & M[i,J] = M[i-1,J] \\ \text{se } J - W_i \geq 0 & \max\{ M[i-1,J] \ ; \ V_i + M[i-1, J-W_i] \} \end{cases}$$

$J - W_i$ → Peso "RIMANENTE" / "COMPATIBILE"

④ Soluzione

$M[n, L]$

* Algoritmo bottom-up

```
For  J=0  to  L   M[0,J] = 0
For  i=0  to  n   M[i,0] = 0

For (i=1 to n){
    For (J=1 to L) {
        M[i,J]   ~>  Θ(1)
```

→ $\Theta(n \cdot L)$

Return $(M[n, L])$; $\rightarrow \Theta(1)$

* il **tempo** SEMBRA polinomiale → ma è pseudo-polinomiale.
* lo **spazio**: $S = \Theta(n \cdot L)$

POSSO FARLO CON 2 RIGHE: $2 \times L$ } Computaz. prec. + Comp. corr.

$S = \Theta(L)$

LEZIONE 2 KNAPSACK:

ISTANZA

$X_n = \{1, \dots, n\}$

$\forall i \in \{1 \dots n\}$

V_i = "valore dell'oggetto i";

W_i = "peso dell'oggetto i";

C = "capacità dello zaino";

SOLUZIONE

$S \subseteq X_n$ t.c:

$$V(S) = \max_{A \subseteq X_n} \{V(A)\} \wedge W(A) \leq C.$$

FUNZIONI UTILI

V → la funzione che restituisce il valore di un $A \subseteq X_n$

$$\forall A \subseteq X_n \; ; \quad V(A) = \sum_{i \in A} V_i \quad \text{se } A \neq \emptyset \text{ (o altrimenti)}$$

W → la funzione che restituisce il peso di un $A \subseteq X_n$. Definito come:

$$\forall A \subseteq X_n \; ; \quad W(A) = \sum_{i \in A} W_i \quad \text{se } A \neq \emptyset \text{ (o altrimenti).}$$

DEFINIZIONE DEI SOTTO-PROBLEMI:

Istanza $X_i = \{1 \dots i\}$ $\quad i \in \{0 \dots n\}$

"prefisso"

$X_0 = \emptyset$

C = Capacità residua $\quad C \in \{0 \dots c\}$

Soluzione: $S_{i,c}$ → "massimo valore ottenibile con (i) elementi (un sottoinsieme A di X_i) e (C) costo".

$$V(S_{i,c}) = \max_{A \subseteq X_i} \{V(A)\} \quad \text{e} \quad W(A) \leq C.$$

CASO BASE {i}=elemento di un insieme, J= intero.

$S_{i,0} = \emptyset \qquad V(S_{i,0}) = 0$

$S_{\emptyset,J} = \emptyset \qquad V(S_{\emptyset,J}) = 0$

Caso passo → Assumiamo di aver già risolto tutto

- Ho 3 possibilità:

1) i è tale che $w_i > c$

   Allora: $i \notin S_{i,c}$

   La soluzione: $S_{i-1,c}$ ⟧→ $A \subseteq X_n$ che non comprende i

   $V(S_{i,c}) = V(S_{i-1,c})$

2) Se $w_i \leq c$ posso avere:

   2a) Mi conviene prendere i nella soluzione.
   Inoltre è comp

   $i \in S_{i,c}$ $S_{i,c} = S_{i-1,c-w_i} \cup \{i\}$

   Soluzione: $V(S_{i,c}) = V(S_{i-1,c-w_i}) + v_i$

   2b) $i \notin S_{i,c}$ $S_{i,c} = S_{i-1,c}$

   $V_{S_{i,c}} = V(S_{i-1,c})$

PER STABILIRE QUAL È TRA 2a e 2b → max ← tra prendere o no i.

$$S_{i,c} = \begin{cases} S_{i-1,c} & \text{se } w_i > c \\ S_{i-1,c-w_i} \cup \{v_i\} & \text{se } w_i < c \wedge \underbrace{opt_{i-1,c-w_i} + v_i}_{\max} \geq OPT_{i-1,c} \end{cases}$$

$S_{i-1,c}$ se $w_j \leq c$

Confronto tra "prendere oggetto i" e "non prenderlo" nella soluz.