

Zaino Intero (oppure zaino 0-1)

È molto simile allo zaino frazionario, con l'unica differenza che non permette di prendere frazioni di cosa. Quindi l'oggetto o si prende oppure no.

Dobbiamo prendere un numero di oggetti tale che S sia massimale.

Capacità C

Volume v_0, \dots, v_{n-1}

Valore w_0, \dots, w_{n-1}

Volume e Valore dell'unesimo oggetto

Vogliamo una soluzione $S \subseteq \{0, \dots, n-1\}$ t.c. $\sum_{i \in S} v_i \leq C$ e $\sum_{i \in S} w_i$ massimo

oggetti a disposizione

Devono stare nello zaino

Devono essere quelli con valore più alto.

Abbiamo un problema $P_{n,C}$, ovvero con n oggetti e capacità C .

Teorema Sottostuttura Ottima:

Sia S soluzione ADO per $P_{n,C}$

ammissibile
ottimale

Abbiamo 2 casi possibili, l'oggetto n c'è o non c'è

1) $n-1 \in S$

2) $n-1 \notin S$

1) Il resto dello zaino deve essere riempito nel modo migliore possibile



$$S = \{n-1, i_1, i_2, \dots, i_k\}$$

questi oggetti devono stare nello zaino, quindi la loro capacità deve essere \leq della capacità dello zaino. Quindi abbiamo:

$$\sum_{j=1}^k v_j \leq C - v_{n-1}$$

$$\sum_{j=1}^k w_j \text{ massimo tra } S \subseteq \{0, 1, \dots, n-2\}$$

$n-1$ non è presente perché stiamo considerando solo gli oggetti: i_1, \dots, i_k

Allora possiamo dire di avere:

$$S' = S \setminus \{n-1\} \text{ per } P_{n-1, C-v_{n-1}}$$

Perché S' è ASO?

È immediato dimostrare che S' è ammissibile perché non abbiamo superato la capacità dello zaino.

L'ottimalità va dimostrata.

Per assurdo diciamo che esiste

$$S^* \text{ sol ASO di } P_{n-1, C-v_{n-1}} \text{ con } \text{val}(S^*) > \text{val}(S')$$

"ovvero il problema con $n-1$ elementi (uno in meno confronto la partenza) e capacità $C-v_{n-1}$ (quindi togliendo il volume dell'oggetto)

Il di fatto S^* è una copia di S' ma con valore maggiore.

$$\text{Quindi gode di: } \text{val}(S^*) \leq C - v_{n-1} \text{ e } n-1 \notin S^*$$

Se riaggiungiamo $n-1$ ad S^* otteniamo: $\text{val}(S^* \cup \{n-1\}) \leq C$, quindi $S^* \cup \{n-1\}$

è ammissibile anche per $P_{n, C}$ " ovvero il problema originale, perché ho riaggiunto il nodo mancante.

Se ora guardiamo i valori otteniamo: $\text{val}(S^* \cup \{n-1\}) = \text{val}(S^*) + w_{n-1}$

Ma questo é impossibile perché otteniamo: $val(s^*) + w_{n-1} > val(s') + w_{n-1}$

└──────────┘
 └─> ovvero S

Ma S é per definizione
Ottimale

2) $n-1 \notin S$

← Come puoi vedere manca $n-1$ a differenza di prima, quindi è immediato dire che la soluzione è valida anche per $n-1$ oggetti.

Quindi abbiamo:

S è soluzione AAO per $P_{n-1,c}$ da notare come la capacità resti invariata perché non abbiamo aggiunto nessun nuovo elemento.

Se è ammissibile perché non ho riempito lo zaino con più oggetti della capacità.

Dim per l'ottimalità:

per assurdo esiste un S^* sol di P_{n-1}, C con $\text{val}(S^*) > \text{val}(S)$

S^* è ammissibile anche per $P_{h,c}$ ma otteniamo $\text{val}(S^*) > \text{val}(S)$, quindi è impossibile.

Come puoi notare anche in questo caso la soluzione viene costituita da una delle 2 soluzioni più piccole.

Pseudo codice:

$\forall c \text{ val}(0, c) = 0$ // per ogni capacità metto il valore dell'insieme degli oggetti:

$\emptyset \geq 0$. Ovvero metto la prima riga a 0.

for $n=1$ to N // n è il numero di elementi presenti
nello zaino, e non è il numero dell'elemento!

for $c=0$ to $CapZaino$

m : insieme senza l'elemento che stiamo considerando adesso

if $(c \geq vol_{n-1})$ // occhio che è n minuscolo,

$val(n, c) = \max(val(n-1, c), val(n-1, c - vol_{n-1}) + w_{n-1})$ // oggetto attuale

else $val(n, c) = val(n-1, c)$ // Se non ho più spazio uso l'ultimo val calcolato

return $val(N, CapZaino)$;

Esempio

$C=3$

Vol: 1, 3, 2

val: 4, 9, 2

dim insieme oggetti	0	1	2	3
$\emptyset \rightarrow 0$	0	0	0	0
$\{0\} \rightarrow 1$	0	4	4	4
$\{0,1\} \rightarrow 2$	0	4*	4*	9
$\{0,1,2\} \rightarrow 3$	0	4*	4	9

// con insieme vuoto hai valori a zero

* non ho spazio, quindi else

risultato

Costo: $O(nC)$

num oggetti: \uparrow
Capacità dello zaino: \uparrow

Occhio però che non ha costo lineare ma Pseudopolinomiale,

perché entra come $\log_2 C$ ma diventa 2^C , quindi esponenziale.

In fatti all'incremento di C si incrementa anche il numero di colonne della tabella.

Questo avviene principalmente quando C domina.

Lo zaino infatti è anche considerato un Problema Difficile, capirai più avanti.

Se lo implementi ricorsivamente hai un'ottimizzazione su input piccoli.

Volendo puoi usare solo 2 righe invece di n , sovrascrivendole ogni volta, però in questo modo non puoi ottenere gli oggetti che compongono la soluzione.

Come ottenere gli oggetti che fanno parte della soluzione?

1) Calcolo la tabella dei valori

2) $\text{numOgg} = n$; $\text{cap} = C$
while ($\text{numOgg} > 0$) and ($\text{cap} > 0$)
- se $\text{Val}[\text{numOgg}, \text{cap}] > \text{Val}[\text{numOgg}-1, \text{cap}]$
• $S = S \cup \{\text{numOgg}-1\}$ ← questo ci fa parte della soluzione
• $\text{cap} = \text{cap} - S_{\text{numOgg}-1}$
• $\text{numOgg} = \text{numOgg}-1$;
- else
• $\text{numOgg} = \text{numOgg}-1$;

num: per prendere l'id dell'oggetto.

Funzionamento:

din insieme oggetti Q →	0	1	2	3
{ } → 0	0	0	0	0
{0} → 1	0	4	4	4
{0,1} → 2	0	4	4	9
{0,1,2} → 3	0	4	4	9

$\text{numOgg} = 3$ $\text{cap} = 3$

$9 > 9$? No, passo alla penultima riga

$9 > 4$? Sì, quindi:

$S = \{2\}$, $\text{cap} = 0$ stop iterazioni

Vol: 1, 3, 2

val: 4, 9, 2