

PD per il problema dello zaino a scelta multipla

Indice

Un algoritmo di PD
Numero di passi

Errori tipici

In sintesi

Obiettivi

Al termine di questo capitolo saremo in grado di:

- ▶ scrivere la ricorrenza $v(h, k)$ su classi e capacità, con il valore $-\infty$ per le sottoistanze senza soluzione ammissibile;
- ▶ eseguire a mano l'algoritmo di PD, provando ogni oggetto della classe h che entra nello zaino;
- ▶ riempire la matrice s degli oggetti scelti e ricostruire una soluzione ottima, un oggetto per classe;
- ▶ contare i $\Theta(nc)$ passi dell'algoritmo di PD.

Prerequisiti

- ▶ Da questo modulo, il capitolo «PD per il problema dello zaino 0/1»: la ricorrenza $v(j, k)$ del problema dello zaino 0/1 (KP) su oggetti e capacità, la matrice s delle decisioni e la ricostruzione di una soluzione.
- ▶ Da questo modulo, il capitolo «Problemi di zaino»: il problema dello zaino a scelta multipla, le sue classi $\mathcal{I}_h$ e la sua condizione di ammissibilità.
- ▶ Dal modulo *Algoritmi*: le convenzioni dello pseudocodice e la dimensione dell'input.

Il problema di questo capitolo

Il problema di questo capitolo è il
problema dello zaino a scelta multipla (MCKP):

la sua definizione e le sue ipotesi sono nella sua pagina.

- ▶ Scriviamo un algoritmo di PD che calcola il valore ottimo di un'istanza dell'MCKP
- ▶ Lo stesso algoritmo dà anche una soluzione ottima, con esattamente un oggetto in ogni classe

Un algoritmo di PD

Introduzione: Un algoritmo di PD

- ▶ In questa sezione scriviamo un algoritmo di PD per l'MCKP, in due passi: prima il valore ottimo, poi una soluzione ottima
- ▶ Gli stadi dell'algoritmo sono le classi, e in ogni classe si sceglie esattamente un oggetto
- ▶ Il valore $-\infty$ segnala le sottoistanze senza soluzione ammissibile
- ▶ Eseguiamo l'algoritmo a mano su un'istanza con sette oggetti in tre classi, e ne contiamo i passi

Esempio: istanza dell'MCKP

Consideriamo uno zaino di capacità $c = 10$ e un insieme $\mathcal{I} = \{1, 2, \dots, 7\}$ di 7 oggetti ($n = 7$), partizionato in $q = 3$ classi:

$$\mathcal{I}_1 = \{1, 2\}, \quad \mathcal{I}_2 = \{3, 4, 5\}, \quad \mathcal{I}_3 = \{6, 7\}.$$

I profitti e i pesi sono:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$
p_j	4	8	3	5	7	3	7
w_j	1	4	1	2	5	2	3

Usiamo questa istanza dell'MCKP come esempio di tutto il capitolo.

Soluzioni ammissibili e stadi

- ▶ Una soluzione ammissibile sceglie esattamente un oggetto in ogni classe, quindi si descrive con:
 - ▶ il sottoinsieme selezionato $S \subseteq \mathcal{I}$, con $|S \cap \mathcal{I}_h| = 1$ per ogni classe h , oppure
 - ▶ il vettore binario $(x_1, x_2, \dots, x_n)$ di S .
- ▶ Le classi sono gli stadi dell'algoritmo di PD:
 - ▶ la classe h si decide dopo le classi $1, 2, \dots, h-1$,
 - ▶ e la sua decisione è l'oggetto scelto in essa.

I due passi dell'algoritmo di PD

- ▶ Passo 1: calcolo del valore ottimo
 - ▶ una formula ricorsiva dà il valore ottimo di ogni sottoistanza, su classi e capacità
- ▶ Passo 2: calcolo di una soluzione ottima
 - ▶ una matrice s memorizza l'oggetto scelto in ogni classe, e da essa si legge una soluzione ottima

Passo 1: le sottoistanze

Per ogni $h \in \{0, 1, \dots, q\}$ e $k \in \{0, 1, \dots, c\}$, sia $v(h, k)$ il valore ottimo della sottoistanza dell'MCKP data da:

1. uno zaino di capacità k ,
2. le classi $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_h$,
3. per ogni oggetto $j \in \mathcal{I}_1 \cup \mathcal{I}_2 \cup \dots \cup \mathcal{I}_h$, un profitto p_j e un peso w_j ,

con $v(h, k) = -\infty$ se questa sottoistanza non ha soluzioni ammissibili.

Passo 1: la formula ricorsiva

La formula ricorsiva per calcolare $v(h, k)$ è la seguente:

$$v(h, k) = \begin{cases} 0 & \text{se } h = 0, \\ -\infty & \text{se } h \geq 1 \text{ e} \\ & \forall j \in \mathcal{J}_h: w_j > k, \\ \max_{j \in \mathcal{J}_h: w_j \leq k} \{p_j + v(h-1, k-w_j)\} & \text{se } h \geq 1 \text{ e} \\ & \exists j \in \mathcal{J}_h: w_j \leq k. \end{cases}$$

Come si legge la formula ricorsiva

- Il valore $-\infty$ segue le regole

$$p_j + (-\infty) = -\infty \quad \text{e} \quad \max\{-\infty, a\} = a :$$

una sottoistanza senza soluzioni ammissibili non dà mai un valore migliore.

- I casi base sono

$$v(0, k) = 0 \quad \text{per ogni } k :$$

senza classi, il sottoinsieme vuoto è ammissibile e ha profitto 0, qualunque sia la capacità.

Sottostruttura ottima

- ▶ La formula ricorsiva genera al più $|\mathcal{I}_h|$ **sottoproblemi indipendenti**, uno per ogni oggetto j della classe h che entra nella capacità k :
 - ▶ il sottoproblema residuo ha le classi $1, 2, \dots, h - 1$ e la capacità $k - w_j$.
- ▶ Quindi una soluzione ottima per (h, k) si costruisce combinando il miglior oggetto j della classe h con una soluzione ottima del suo sottoproblema residuo.

~> **Sottostruttura ottima verificata!**

Sottoproblemi sovrapposti

- ▶ Scelte diverse nelle classi $h + 1, h + 2, \dots, q$ possono lasciare la stessa capacità residua.
- ▶ Quindi lo stesso valore $v(h, k)$ può servire più volte durante la valutazione ricorsiva della formula.

~> **Sottoproblemi sovrapposti verificati!**

L'approccio bottom-up

- ▶ Il valore $v(h, k)$ dipende solo dai valori $v(h-1, k')$ con $k' < k$.
- ▶ Quindi l'**approccio bottom-up** calcola le righe $h = 0, 1, \dots, q$ in quest'ordine, ciascuna per k crescente, e memorizza ogni valore in una **tabella di lookup**.
- ▶ Il valore ottimo dell'istanza dell'MCKP è $v(q, c)$, e l'istanza è inammissibile se e solo se

$$v(q, c) = -\infty.$$

Passo 2: calcolo di una soluzione ottima

- ▶ Per ogni classe h e capacità k , memorizziamo $s[h][k]$:
 - ▶ l'oggetto della classe h scelto in una soluzione ottima della sottoistanza (h, k) ,
 - ▶ cioè l'oggetto che massimizza il membro destro della formula ricorsiva.
- ▶ Se $v[h][k] = -\infty$, non si sceglie alcun oggetto e $s[h][k] = 0$.

Ora: lo pseudocodice dell'**algoritmo di PD con l'approccio bottom-up**,
che memorizza la matrice s insieme alla matrice v .

Algorithm: MCKP_DP (1/2)

Input : Un'istanza dell'MCKP data da q , dalle classi $\mathcal{I}_1, \dots, \mathcal{I}_q$, dai vettori $p[1..n]$, $w[1..n]$ e dalla capacità c

Output : Il valore ottimo dell'istanza dell'MCKP e la matrice $s[0..q][0..c]$

```
1 function MCKP_DP( $q, \mathcal{I}, p, w, c$ )  
2    $v := \text{new array}[0..q][0..c]$ ;  $s := \text{new array}[0..q][0..c]$   
3   for  $k := 0$  to  $c$  do  
4      $v[0][k] \leftarrow 0$ ;  $s[0][k] \leftarrow 0$   
5   end
```

Algorithm: MCKP_DP (2/2)

function *MCKP_DP*($q, \mathcal{J}, p, w, c$) (continua)

```
6   for  $h := 1$  to  $q$  do
7       for  $k := 0$  to  $c$  do
8            $m := -\infty; t := 0$ 
9           foreach  $j \in \mathcal{J}_h$  do
10              if  $w[j] \leq k$  e  $p[j] + v[h-1][k - w[j]] > m$ 
11                 then
12                      $m \leftarrow p[j] + v[h-1][k - w[j]]; t \leftarrow j$ 
13                 end
14             end
15              $v[h][k] \leftarrow m; s[h][k] \leftarrow t$ 
16         end
17     return  $v[q][c], s$ 
18 end
```

Come funziona MCKP_DP

- ▶ La variabile m parte da $-\infty$, non da 0: si deve scegliere esattamente un oggetto della classe h , quindi il valore di una scelta vuota non è disponibile.
- ▶ Il ciclo più interno prova ogni oggetto j della classe h che entra nello zaino, conserva in m il valore massimo trovato finora e in t l'oggetto che lo dà.
- ▶ Se nessun oggetto della classe h dà un valore maggiore di $-\infty$, la cella mantiene $v[h][k] = -\infty$ e $s[h][k] = 0$.
- ▶ In caso di pareggio si conserva l'oggetto incontrato per primo in $\mathcal{J}_h$, perché m viene aggiornato solo quando il valore è strettamente maggiore.

Esempio: esecuzione dell'algoritmo di PD (1/10)

Consideriamo l'istanza dell'MCKP dell'esempio precedente, con capacità $c = 10$ e classi $\mathcal{J}_1 = \{1, 2\}$, $\mathcal{J}_2 = \{3, 4, 5\}$, $\mathcal{J}_3 = \{6, 7\}$:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$
p_j	4	8	3	5	7	3	7
w_j	1	4	1	2	5	2	3

- ▶ Per ogni passo dell'algoritmo di PD con $h \geq 1$, le tabelle seguenti riportano i valori di h e k , le celle $v[h][k]$ e $s[h][k]$ e l'operazione eseguita.
- ▶ La riga $h = 0$ è $v[0][k] = 0$ e $s[0][k] = 0$ per ogni k .

Esempio: esecuzione dell'algoritmo di PD (2/10)

h	k	$v[h][k]$	$s[h][k]$	Azione
1	0	$-\infty$	0	nessun oggetto di $\mathcal{J}_1$ entra: $-\infty$
1	1	4	1	$\max\{p_1 + v[0][0]\}$ $= \max\{4\} = 4$
1	2	4	1	$\max\{p_1 + v[0][1]\}$ $= \max\{4\} = 4$
1	3	4	1	$\max\{p_1 + v[0][2]\}$ $= \max\{4\} = 4$
1	4	8	2	$\max\{p_1 + v[0][3], p_2 + v[0][0]\}$ $= \max\{4, 8\} = 8$
1	5	8	2	$\max\{p_1 + v[0][4], p_2 + v[0][1]\}$ $= \max\{4, 8\} = 8$

Esempio: esecuzione dell'algoritmo di PD (3/10)

h	k	$v[h][k]$	$s[h][k]$	Azione
1	6	8	2	$\max\{p_1 + v[0][5], p_2 + v[0][2]\}$ $= \max\{4, 8\} = 8$
1	7	8	2	$\max\{p_1 + v[0][6], p_2 + v[0][3]\}$ $= \max\{4, 8\} = 8$
1	8	8	2	$\max\{p_1 + v[0][7], p_2 + v[0][4]\}$ $= \max\{4, 8\} = 8$
1	9	8	2	$\max\{p_1 + v[0][8], p_2 + v[0][5]\}$ $= \max\{4, 8\} = 8$
1	10	8	2	$\max\{p_1 + v[0][9], p_2 + v[0][6]\}$ $= \max\{4, 8\} = 8$

Esempio: esecuzione dell'algoritmo di PD (4/10)

h	k	$v[h][k]$	$s[h][k]$	Azione
2	0	$-\infty$	0	nessun oggetto di $\mathcal{J}_2$ entra: $-\infty$
2	1	$-\infty$	0	$\max\{p_3 + v[1][0]\}$ $= \max\{-\infty\} = -\infty$
2	2	7	3	$\max\{p_3 + v[1][1], p_4 + v[1][0]\}$ $= \max\{7, -\infty\} = 7$
2	3	9	4	$\max\{p_3 + v[1][2], p_4 + v[1][1]\}$ $= \max\{7, 9\} = 9$
2	4	9	4	$\max\{p_3 + v[1][3], p_4 + v[1][2]\}$ $= \max\{7, 9\} = 9$
2	5	11	3	$\max\{p_3 + v[1][4], p_4 + v[1][3], p_5 + v[1][0]\}$ $= \max\{11, 9, -\infty\} = 11$

Esempio: esecuzione dell'algoritmo di PD (5/10)

h	k	$v[h][k]$	$s[h][k]$	Azione
2	6	13	4	$\max\{p_3+v[1][5], p_4+v[1][4], p_5+v[1][1]\}$ $= \max\{11, 13, 11\} = 13$
2	7	13	4	$\max\{p_3+v[1][6], p_4+v[1][5], p_5+v[1][2]\}$ $= \max\{11, 13, 11\} = 13$
2	8	13	4	$\max\{p_3+v[1][7], p_4+v[1][6], p_5+v[1][3]\}$ $= \max\{11, 13, 11\} = 13$
2	9	15	5	$\max\{p_3+v[1][8], p_4+v[1][7], p_5+v[1][4]\}$ $= \max\{11, 13, 15\} = 15$
2	10	15	5	$\max\{p_3+v[1][9], p_4+v[1][8], p_5+v[1][5]\}$ $= \max\{11, 13, 15\} = 15$

Esempio: esecuzione dell'algoritmo di PD (6/10)

h	k	$v[h][k]$	$s[h][k]$	Azione
3	0	$-\infty$	0	nessun oggetto di $\mathcal{J}_3$ entra: $-\infty$
3	1	$-\infty$	0	nessun oggetto di $\mathcal{J}_3$ entra: $-\infty$
3	2	$-\infty$	0	$\max\{p_6 + v[2][0]\}$ $= \max\{-\infty\} = -\infty$
3	3	$-\infty$	0	$\max\{p_6 + v[2][1], p_7 + v[2][0]\}$ $= \max\{-\infty, -\infty\} = -\infty$
3	4	10	6	$\max\{p_6 + v[2][2], p_7 + v[2][1]\}$ $= \max\{10, -\infty\} = 10$
3	5	14	7	$\max\{p_6 + v[2][3], p_7 + v[2][2]\}$ $= \max\{12, 14\} = 14$

Esempio: esecuzione dell'algoritmo di PD (7/10)

h	k	$v[h][k]$	$s[h][k]$	Azione
3	6	16	7	$\max\{p_6 + v[2][4], p_7 + v[2][3]\}$ $= \max\{12, 16\} = 16$
3	7	16	7	$\max\{p_6 + v[2][5], p_7 + v[2][4]\}$ $= \max\{14, 16\} = 16$
3	8	18	7	$\max\{p_6 + v[2][6], p_7 + v[2][5]\}$ $= \max\{16, 18\} = 18$
3	9	20	7	$\max\{p_6 + v[2][7], p_7 + v[2][6]\}$ $= \max\{16, 20\} = 20$
3	10	20	7	$\max\{p_6 + v[2][8], p_7 + v[2][7]\}$ $= \max\{16, 20\} = 20$

Esempio: esecuzione dell'algoritmo di PD (8/10)

Questa è la matrice v dopo l'esecuzione dell'algoritmo di PD:

$v[h][k]$	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$h = 0$	0	0	0	0	0	0
$h = 1$	$-\infty$	4	4	4	8	8
$h = 2$	$-\infty$	$-\infty$	7	9	9	11
$h = 3$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	10	14

$v[h][k]$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$h = 0$	0	0	0	0	0
$h = 1$	8	8	8	8	8
$h = 2$	13	13	13	15	15
$h = 3$	16	16	18	20	20

Esempio: esecuzione dell'algoritmo di PD (9/10)

Questa è la matrice s dopo l'esecuzione dell'algoritmo di PD:

$s[h][k]$	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$h = 0$	0	0	0	0	0	0
$h = 1$	0	1	1	1	2	2
$h = 2$	0	0	3	4	4	3
$h = 3$	0	0	0	0	6	7

$s[h][k]$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$h = 0$	0	0	0	0	0
$h = 1$	2	2	2	2	2
$h = 2$	4	4	4	5	5
$h = 3$	7	7	7	7	7

Esempio: esecuzione dell'algoritmo di PD (10/10)

- ▶ Per esempio, $v[2][1] = -\infty$: ogni scelta di un oggetto in $\mathcal{I}_1$ e di uno in $\mathcal{I}_2$ pesa almeno

$$w_1 + w_3 = 2.$$

- ▶ Il valore ottimo dell'istanza è

$$v[3][10] = 20,$$

raggiunto con l'oggetto $s[3][10] = 7$ della classe 3.

Ricostruzione di una soluzione ottima

- ▶ Calcolata la matrice s , se $v[q][c] > -\infty$ si ricostruisce una soluzione ottima:
 - ▶ scorrendo le classi da $h = q$ fino a $h = 1$,
 - ▶ partendo da $k = c$ e da $x_j = 0$ per ogni oggetto j :
$$j \leftarrow s[h][k], \quad x_j \leftarrow 1, \quad k \leftarrow k - w_j.$$
- ▶ Ogni classe dà esattamente un oggetto, quindi la ricostruzione seleziona q oggetti.

Algorithm: MCKP_Reconstruct

Input : Il numero q di classi, la capacità c , il vettore $w[1..n]$ e la matrice $s[0..q][0..c]$, con $v[q][c] > -\infty$

Output: Una soluzione ottima dell'istanza dell'MCKP, come vettore binario

```
1 function MCKP_Reconstruct( $q, c, w, s$ )
2    $x :=$  new array[1.. $n$ ], con  $x[j] = 0$  per ogni  $j$ 
3    $k := c$ 
4   for  $h := q$  downto 1 do
5      $j \leftarrow s[h][k]; x[j] \leftarrow 1$ 
6      $k \leftarrow k - w[j]$ 
7   end
8   return  $x$ 
9 end
```

Esempio: calcolo di una soluzione ottima (1/2)

Dopo l'esecuzione dell'algoritmo di PD sull'istanza dell'esempio precedente, partiamo da $h = 3$ e $k = 10$:

Iterazione	h	k	$j = s[h][k]$	Selezione	Capacità
—	—	—	—	—	$k = 10$
1	3	10	7	$x[7] \leftarrow 1$	$k \leftarrow 10 - 3 = 7$
2	2	7	4	$x[4] \leftarrow 1$	$k \leftarrow 7 - 2 = 5$
3	1	5	2	$x[2] \leftarrow 1$	$k \leftarrow 5 - 4 = 1$

Esempio: calcolo di una soluzione ottima (2/2)

- ▶ Una soluzione ottima dell'istanza dell'MCKP dell'esempio è

$$S = \{2, 4, 7\},$$

cioè il vettore

$$x = (0, 1, 0, 1, 0, 0, 1).$$

- ▶ Il suo peso totale è $4 + 2 + 3 = 9 \leq c$.
- ▶ Il suo valore ottimo è $v(3, 10) = 20$.

Quanti passi esegue l'algoritmo di PD?

Contiamo quante volte viene eseguita
ogni riga dello pseudocodice di MCKP_DP.

Numero di passi (1/3)

- ▶ Indichiamo con $S(n, q, c)$ il numero di passi eseguiti da MCKP_DP nel caso peggiore.
- ▶ Il ciclo più interno scorre gli oggetti della classe h , quindi su tutte le classi viene eseguito

$$|\mathcal{I}_1| + |\mathcal{I}_2| + \cdots + |\mathcal{I}_q| = n$$

volte per ogni capacità k .

Numero di passi (2/3)

Supponendo che la condizione alla riga 10 sia sempre vera:

- ▶ Le righe 2 e 17 vengono eseguite 1 volta ciascuna.
- ▶ Le righe 3 e 4 vengono eseguite $c + 1$ volte ciascuna.
- ▶ La riga 6 viene eseguita q volte.
- ▶ Le righe 7, 8 e 14 vengono eseguite $q(c + 1)$ volte ciascuna.
- ▶ Le righe 9, 10 e 11 vengono eseguite $n(c + 1)$ volte ciascuna.

Numero di passi (3/3)

Quindi il numero totale di passi nel caso peggiore è

$$S(n, q, c) = 3n(c + 1) + 3q(c + 1) + q + 2c + 4.$$

Poiché ogni classe contiene almeno un oggetto, $1 \leq q \leq n$, e

$$3n(c + 1) \leq S(n, q, c) \leq 6n(c + 1) + n + 2c + 4.$$

Entrambi i bound sono $\Theta(nc)$, quindi $S(n, q, c) = \Theta(nc)$.

Il numero di passi dell'algoritmo di PD

L'algoritmo di PD per l'MCKP esegue

$$S(n, q, c) = \Theta(c(|\mathcal{J}_1| + |\mathcal{J}_2| + \cdots + |\mathcal{J}_q|)) = \Theta(nc)$$

passi nel caso peggiore.

- La ricostruzione esegue $\Theta(q)$ passi, uno per ogni classe.

Riepilogo (1/2)

Passo 1: il valore ottimo:

- ▶ gli stadi sono le classi:
$$v(h, k) = \max_{j \in \mathcal{J}_h: w_j \leq k} \{p_j + v(h-1, k-w_j)\}$$
- ▶ casi base $v(0, k) = 0$; $v(h, k) = -\infty$ se nessun oggetto della classe h entra
- ▶ sottostruttura ottima e sottoproblemi sovrapposti: bottom-up, riga per riga, in una tabella di lookup
- ▶ il valore ottimo è $v(q, c)$; l'istanza è inammissibile se e solo se $v(q, c) = -\infty$

Riepilogo (2/2)

Passo 2: una soluzione ottima:

- ▶ $s[h][k]$ è l'oggetto della classe h che massimizza la formula ricorsiva, 0 se $v[h][k] = -\infty$
- ▶ la soluzione si legge all'indietro a partire da (q, c) :
 $j \leftarrow s[h][k], x_j \leftarrow 1, k \leftarrow k - w_j$
- ▶ sull'esempio: $S = \{2, 4, 7\}$, con peso totale 9 e valore ottimo 20

Numero di passi:

- ▶ $S(n, q, c) = \Theta(nc)$ per MCKP_DP, $\Theta(q)$ per la ricostruzione

Errori tipici

Introduzione: Errori tipici

- ▶ Raccogliamo gli errori più frequenti con l'algoritmo di PD per l'MCKP
- ▶ Due riguardano il valore $-\infty$ e la regola «esattamente un oggetto per classe», due i casi base e la ricostruzione
- ▶ Controllali prima di eseguire a mano l'algoritmo di PD

Errori tipici (1/2)

Osservazione 1: Far partire m da 0 invece che da $-\infty$

Con $m = 0$ una classe senza oggetti che entrano nello zaino verrebbe saltata, e la soluzione non conterrebbe esattamente un oggetto per classe.

Osservazione 2: Aggiungere l'opzione «nessun oggetto della classe h »

Il termine $v(h-1, k)$ della ricorrenza del KP qui non ha posto: ogni classe dà esattamente un oggetto.

Errori tipici (2/2)

Osservazione 3: Porre $v(0, k) = -\infty$ per $k \geq 1$

Senza classi il sottoinsieme vuoto è ammissibile per ogni capacità: la capacità non deve essere riempita.

Osservazione 4: Ricostruire un'istanza inammissibile

Se $v(q, c) = -\infty$ l'istanza non ha soluzioni ammissibili, e $s[q][c] = 0$ non dà alcun oggetto da selezionare.

Esattamente un oggetto per classe:

- ▶ m parte da $-\infty$, non da 0
- ▶ nessuna opzione «nessun oggetto della classe h »: il termine $v(h-1, k)$ del KP qui non ha posto

Casi base e ricostruzione:

- ▶ $v(0, k) = 0$ per ogni k : la capacità non deve essere riempita
- ▶ si ricostruisce una soluzione solo quando $v(q, c) > -\infty$

In sintesi

Introduzione: In sintesi

- ▶ Raccogliamo le idee chiave del capitolo sull'algoritmo di PD per l'MCKP
- ▶ Il quadro riassuntivo raccoglie il risultato sull'esempio del capitolo

Riepilogo (1/2)

- ▶ Gli stadi dell'algoritmo di PD sono le classi:

$$v(h, k) = \max_{j \in \mathcal{J}_h: w_j \leq k} \{p_j + v(h-1, k - w_j)\}, \quad \text{con } v(0, k) = 0.$$

- ▶ Il valore $-\infty$ segnala le sottoistanze senza soluzione ammissibile, e l'istanza è inammissibile se e solo se $v(q, c) = -\infty$.
- ▶ La matrice s memorizza l'oggetto scelto in ogni classe, e una soluzione ottima si legge all'indietro a partire da (q, c) , un oggetto per classe.
- ▶ L'algoritmo di PD esegue $\Theta(c \sum_{h=1}^q |\mathcal{J}_h|) = \Theta(nc)$ passi.

Riepilogo (2/2)

Quadro riassuntivo dell'esempio dell'MCKP ($c = 10$, tre classi):

Metodo	Lower bound	Upper bound	Origine
Algoritmo di PD	20	20	valore ottimo, soluzione $S = \{2, 4, 7\}$