

PD per il problema del subset sum



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 $r(j, k) \in \{true, false\}$ delle somme raggiungibili, con i suoi casi base;
- ▶ eseguire a mano l'algoritmo di PD e leggere la risposta dell'istanza dell'SSP in $r(n, c)$;
- ▶ riempire la matrice s e ricostruire un sottoinsieme di peso totale c quando la risposta è sì;
- ▶ mettere in relazione $r(j, k)$ con l'algoritmo di PD del KP con $p_j = w_j$, e contare i $\Theta(nc)$ passi.

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 del subset sum e la sua soluzione come problema dello zaino 0/1 con $p_j = w_j$.
- ▶ 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 del subset sum (SSP):

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

- ▶ Scriviamo un algoritmo di PD che dà la risposta di un'istanza dell'SSP, sì o no
- ▶ Quando la risposta è sì, lo stesso algoritmo dà anche un sottoinsieme di peso totale c

Un algoritmo di PD

Introduzione: Un algoritmo di PD

- ▶ In questa sezione scriviamo un algoritmo di PD per l'SSP, in due passi: prima la risposta dell'istanza, poi un sottoinsieme di peso totale c
- ▶ L'algoritmo segue l'algoritmo di PD del KP, ma ogni cella della sua tabella memorizza un solo bit, *true* o *false*
- ▶ Lo eseguiamo a mano su un'istanza con quattro oggetti, e ne contiamo i passi

Esempio: istanza dell'SSP

Consideriamo un valore obiettivo $c = 10$ e un insieme $\mathcal{J} = \{1, 2, 3, 4\}$ di 4 oggetti ($n = 4$), con i seguenti pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
w_j	3	5	4	7

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

Sottoinsiemi e risposte

- ▶ Un sottoinsieme $S \subseteq \mathcal{I}$ si descrive con il vettore binario $(x_1, x_2, \dots, x_n)$, dove $x_j = 1$ se l'oggetto j appartiene a S e $x_j = 0$ altrimenti.
- ▶ La risposta all'istanza dell'SSP è sì se qualche vettore soddisfa

$$\sum_{j=1}^n w_j x_j = c,$$

e **no** altrimenti.

I due passi dell'algoritmo di PD

- ▶ Passo 1: calcolo della risposta
 - ▶ una formula ricorsiva dice, per ogni sottoistanza, se il suo valore obiettivo si può raggiungere
- ▶ Passo 2: calcolo di un sottoinsieme
 - ▶ quando la risposta è sì, una matrice delle decisioni dà un sottoinsieme di peso totale c

Passo 1: le sottoistanze

Per ogni $j \in \{0, 1, \dots, n\}$ e $k \in \{0, 1, \dots, c\}$, sia

$$r(j, k) \in \{true, false\}$$

la risposta della sottoistanza dell'SSP data da:

1. un valore obiettivo k ,
2. un insieme $\{1, 2, \dots, j\}$ di oggetti disponibili,
3. per ogni oggetto $h \in \{1, 2, \dots, j\}$, un peso w_h .

Passo 1: la formula ricorsiva

La formula ricorsiva per calcolare $r(j, k)$ è la seguente:

$$r(j, k) = \begin{cases} true & \text{se } k = 0, \\ false & \text{se } j = 0 \text{ e } k \geq 1, \\ r(j - 1, k) & \text{se } j, k \geq 1 \text{ e } w_j > k, \\ r(j - 1, k) \vee r(j - 1, k - w_j) & \text{se } j, k \geq 1 \text{ e } w_j \leq k. \end{cases}$$

Come si legge la formula ricorsiva

- ▶ Il simbolo $\vee$ è l'**or** logico: $r(j, k)$ è *true* se almeno uno dei due valori è *true*.
- ▶ I casi base sono:
 - ▶ $r(j, 0) = \text{true}$, perché il sottoinsieme vuoto ha peso totale 0;
 - ▶ $r(0, k) = \text{false}$ per $k \geq 1$, perché nessun oggetto è disponibile.

Sottostruttura ottima

- ▶ Un sottoinsieme di $\{1, 2, \dots, j\}$ con peso totale k o:
 - ▶ non contiene l'oggetto j , e allora è un sottoinsieme di $\{1, 2, \dots, j-1\}$ con peso totale k , oppure
 - ▶ contiene l'oggetto j , e allora gli altri oggetti formano un sottoinsieme di $\{1, 2, \dots, j-1\}$ con peso totale $k - w_j$.
- ▶ Quindi la formula ricorsiva genera al più 2 **sottoproblemi indipendenti**, e un sottoinsieme per (j, k) si costruisce da un sottoinsieme di uno di essi aggiungendo o no l'oggetto j .

~> **Sottostruttura ottima verificata!**

Sottoproblemi sovrapposti e approccio bottom-up

- ▶ Decisioni diverse sugli oggetti possono portare alla stessa coppia (j, k) , e quindi lo stesso valore $r(j, k)$ può servire più volte durante la valutazione ricorsiva della formula.

⇒ **Sottoproblemi sovrapposti verificati!**

- ▶ Il valore $r(j, k)$ dipende solo dai valori $r(j - 1, k')$ con $k' \leq k$.
- ▶ Quindi l'**approccio bottom-up** calcola le righe $j = 0, 1, \dots, n$ in quest'ordine, ciascuna per k crescente, e memorizza ogni valore in una **tabella di lookup**.

L'SSP e l'algoritmo di PD del KP

- ▶ L'SSP si può risolvere anche con l'algoritmo di PD del KP, con $p_j = w_j$ per ogni oggetto j .
- ▶ Il suo valore $v(j, k)$ è il massimo peso totale, non superiore a k , di un sottoinsieme di $\{1, 2, \dots, j\}$, e

$$r(j, k) = \text{true} \iff v(j, k) = k.$$

- ▶ La ricorrenza per $r(j, k)$ memorizza un solo bit per cella, e le sue operazioni logiche sostituiscono le somme e i confronti della ricorrenza del KP.

Passo 2: calcolo di un sottoinsieme

- ▶ Quando la risposta è sì, un sottoinsieme di peso totale c si ricostruisce dalla matrice s :
 - ▶ $s[j][k] = 1$ se l'oggetto j è usato per raggiungere il valore *true* di $r[j][k]$,
 - ▶ $s[j][k] = 0$ altrimenti.
- ▶ L'algoritmo di PD preferisce non usare l'oggetto j : $s[j][k] = 1$ solo se $r[j-1][k]$ è *false* e $r[j-1][k-w_j]$ è *true*.

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

Algorithm: SSP_DP (1/2)

Input : Un'istanza dell'SSP data da n , dal vettore $w[1..n]$ e dal valore obiettivo c

Output: La risposta $r[n][c]$ dell'istanza dell'SSP e la matrice $s[0..n][0..c]$

```
1 function SSP_DP( $n, w, c$ )
2    $r := \text{new array}[0..n][0..c]$ ;  $s := \text{new array}[0..n][0..c]$ 
3    $r[0][0] \leftarrow \text{true}$ ;  $s[0][0] \leftarrow 0$ 
4   for  $k := 1$  to  $c$  do
5      $r[0][k] \leftarrow \text{false}$ ;  $s[0][k] \leftarrow 0$ 
6   end
```

Algorithm: SSP_DP (2/2)

function *SSP_DP*(n, w, c) (continua)

```
7   for  $j := 1$  to  $n$  do
8       for  $k := 0$  to  $c$  do
9            $r[j][k] \leftarrow r[j-1][k]$ ;  $s[j][k] \leftarrow 0$ 
10          if  $r[j][k] = \text{false}$  e  $w[j] \leq k$  e
               $r[j-1][k-w[j]] = \text{true}$  then
11               $r[j][k] \leftarrow \text{true}$ ;  $s[j][k] \leftarrow 1$ 
12          end
13      end
14  end
15  return  $r[n][c], s$ 
16 end
```

Come funziona SSP_DP

- ▶ La matrice $r[0..n][0..c]$ è la **tabella di lookup** delle risposte, e la matrice $s[0..n][0..c]$ memorizza le decisioni.
- ▶ Le righe 3–5 riempiono la riga $j = 0$: senza oggetti è raggiungibile solo il valore obiettivo 0.
- ▶ Per ogni coppia (j, k) , l'algoritmo di PD copia prima la risposta senza l'oggetto j , e poi usa l'oggetto j solo se questa risposta è *false* e l'oggetto j rende raggiungibile il valore k .
- ▶ Infine, la risposta dell'istanza dell'SSP viene restituita come $r[n][c]$, insieme alla matrice s .

Esempio: esecuzione dell'algoritmo di PD (1/9)

- ▶ Consideriamo l'istanza dell'SSP dell'esempio precedente, con valore obiettivo $c = 10$ e pesi $w = (3, 5, 4, 7)$.
- ▶ Per ogni passo dell'algoritmo di PD con $j \geq 1$, le prossime tabelle riportano:
 - ▶ i valori di j e k ,
 - ▶ le celle $r[j][k]$ e $s[j][k]$,
 - ▶ l'operazione eseguita;

T sta per *true* e F per *false*.

- ▶ La riga $j = 0$ è $r[0][0] = T$ e $r[0][k] = F$ per $k \geq 1$, con $s[0][k] = 0$.

Esempio: esecuzione dell'algoritmo di PD (2/9)

j	k	$r[j][k]$	$s[j][k]$	Azione
1	0	T	0	$w_1 = 3 > 0$: $r[0][0] = T$
1	1	F	0	$w_1 = 3 > 1$: $r[0][1] = F$
1	2	F	0	$w_1 = 3 > 2$: $r[0][2] = F$
1	3	T	1	$r[0][3] \vee r[0][0] = F \vee T = T$
1	4	F	0	$r[0][4] \vee r[0][1] = F \vee F = F$
1	5	F	0	$r[0][5] \vee r[0][2] = F \vee F = F$
1	6	F	0	$r[0][6] \vee r[0][3] = F \vee F = F$
1	7	F	0	$r[0][7] \vee r[0][4] = F \vee F = F$
1	8	F	0	$r[0][8] \vee r[0][5] = F \vee F = F$
1	9	F	0	$r[0][9] \vee r[0][6] = F \vee F = F$
1	10	F	0	$r[0][10] \vee r[0][7] = F \vee F = F$

Esempio: esecuzione dell'algoritmo di PD (3/9)

j	k	$r[j][k]$	$s[j][k]$	Azione
2	0	T	0	$w_2 = 5 > 0$: $r[1][0] = T$
2	1	F	0	$w_2 = 5 > 1$: $r[1][1] = F$
2	2	F	0	$w_2 = 5 > 2$: $r[1][2] = F$
2	3	T	0	$w_2 = 5 > 3$: $r[1][3] = T$
2	4	F	0	$w_2 = 5 > 4$: $r[1][4] = F$
2	5	T	1	$r[1][5] \vee r[1][0] = F \vee T = T$
2	6	F	0	$r[1][6] \vee r[1][1] = F \vee F = F$
2	7	F	0	$r[1][7] \vee r[1][2] = F \vee F = F$
2	8	T	1	$r[1][8] \vee r[1][3] = F \vee T = T$
2	9	F	0	$r[1][9] \vee r[1][4] = F \vee F = F$
2	10	F	0	$r[1][10] \vee r[1][5] = F \vee F = F$

Esempio: esecuzione dell'algoritmo di PD (4/9)

j	k	$r[j][k]$	$s[j][k]$	Azione
3	0	T	0	$w_3 = 4 > 0$: $r[2][0] = T$
3	1	F	0	$w_3 = 4 > 1$: $r[2][1] = F$
3	2	F	0	$w_3 = 4 > 2$: $r[2][2] = F$
3	3	T	0	$w_3 = 4 > 3$: $r[2][3] = T$
3	4	T	1	$r[2][4] \vee r[2][0] = F \vee T = T$
3	5	T	0	$r[2][5] \vee r[2][1] = T \vee F = T$
3	6	F	0	$r[2][6] \vee r[2][2] = F \vee F = F$
3	7	T	1	$r[2][7] \vee r[2][3] = F \vee T = T$
3	8	T	0	$r[2][8] \vee r[2][4] = T \vee F = T$
3	9	T	1	$r[2][9] \vee r[2][5] = F \vee T = T$
3	10	F	0	$r[2][10] \vee r[2][6] = F \vee F = F$

Esempio: esecuzione dell'algoritmo di PD (5/9)

j	k	$r[j][k]$	$s[j][k]$	Azione
4	0	T	0	$w_4 = 7 > 0: r[3][0] = T$
4	1	F	0	$w_4 = 7 > 1: r[3][1] = F$
4	2	F	0	$w_4 = 7 > 2: r[3][2] = F$
4	3	T	0	$w_4 = 7 > 3: r[3][3] = T$
4	4	T	0	$w_4 = 7 > 4: r[3][4] = T$
4	5	T	0	$w_4 = 7 > 5: r[3][5] = T$
4	6	F	0	$w_4 = 7 > 6: r[3][6] = F$
4	7	T	0	$r[3][7] \vee r[3][0] = T \vee T = T$
4	8	T	0	$r[3][8] \vee r[3][1] = T \vee F = T$
4	9	T	0	$r[3][9] \vee r[3][2] = T \vee F = T$
4	10	T	1	$r[3][10] \vee r[3][3] = F \vee T = T$

Esempio: esecuzione dell'algoritmo di PD (6/9)

Questa è la matrice r dopo l'esecuzione dell'algoritmo di PD, colonne $k = 0, 1, \dots, 5$:

$r[j][k]$	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$j = 0$	T	F	F	F	F	F
$j = 1$	T	F	F	T	F	F
$j = 2$	T	F	F	T	F	T
$j = 3$	T	F	F	T	T	T
$j = 4$	T	F	F	T	T	T

Esempio: esecuzione dell'algoritmo di PD (7/9)

Questa è la matrice r dopo l'esecuzione dell'algoritmo di PD, colonne $k = 6, 7, \dots, 10$:

$r[j][k]$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
$j = 0$	F	F	F	F	F
$j = 1$	F	F	F	F	F
$j = 2$	F	F	T	F	F
$j = 3$	F	T	T	T	F
$j = 4$	F	T	T	T	T

Esempio: esecuzione dell'algoritmo di PD (8/9)

Questa è la matrice s dopo l'esecuzione dell'algoritmo di PD, colonne $k = 0, 1, \dots, 5$:

$s[j][k]$	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$j = 0$	0	0	0	0	0	0
$j = 1$	0	0	0	1	0	0
$j = 2$	0	0	0	0	0	1
$j = 3$	0	0	0	0	1	0
$j = 4$	0	0	0	0	0	0

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

Questa è la matrice s dopo l'esecuzione dell'algoritmo di PD, colonne $k = 6, 7, \dots, 10$:

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

- ▶ Poiché $r[4][10] = T$, la risposta all'istanza dell'SSP è sì.
- ▶ Per esempio, $s[4][10] = 1$ perché $r[3][10] = F$ e $r[3][3] = T$: l'oggetto 4 serve per raggiungere il valore obiettivo 10.

Ricostruzione di un sottoinsieme

- ▶ Calcolata la matrice s , se $r[n][c]$ è *true* si ricostruisce un sottoinsieme di peso totale c :
 - ▶ scorrendo gli oggetti da $j = n$ fino a $j = 1$,
 - ▶ partendo da $k = c$ e da $x_j = 0$ per ogni oggetto j :
$$x_j \leftarrow s[j][k], \quad k \leftarrow k - s[j][k] w_j.$$
- ▶ La ricostruzione termina con $k = 0$, perché ogni passo mantiene $r[j - 1][k]$ uguale a *true*.

Algorithm: SSP_Reconstruct

Input : Il numero n di oggetti, il valore obiettivo c , il vettore $w[1..n]$ e la matrice $s[0..n][0..c]$, con $r[n][c] = \text{true}$

Output: Un sottoinsieme di peso totale c , come vettore binario

```
1 function SSP_Reconstruct( $n, c, w, s$ )
2    $x := \text{new array}[1..n]$ 
3    $k := c$ 
4   for  $j := n$  downto 1 do
5      $x[j] \leftarrow s[j][k]$ 
6      $k \leftarrow k - x[j] w[j]$ 
7   end
8   return  $x$ 
9 end
```

Esempio: calcolo di un sottoinsieme

Dopo l'esecuzione dell'algoritmo di PD sull'istanza dell'esempio precedente, partiamo da $j = 4$ e $k = 10$:

Iterazione	j	k	$s[j][k]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	Azione
—	—	—	—	0	0	0	0	$k = 10$
1	4	10	1	0	0	0	1	$x[4] \leftarrow 1, k \leftarrow 10 - 7 = 3$
2	3	3	0	0	0	0	1	$x[3] \leftarrow 0, k = 3$
3	2	3	0	0	0	0	1	$x[2] \leftarrow 0, k = 3$
4	1	3	1	1	0	0	1	$x[1] \leftarrow 1, k \leftarrow 3 - 3 = 0$

La risposta all'istanza dell'SSP dell'esempio è sì: il vettore $(1, 0, 0, 1)$ seleziona il sottoinsieme $\{1, 4\}$, il cui peso totale è $3 + 7 = c$.

Quanti passi esegue l'algoritmo di PD?

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

Numero di passi (1/2)

Indichiamo con $S(n, c)$ il numero di passi eseguiti da SSP_DP nel caso peggiore. Contiamo quante volte viene eseguita ogni riga dello pseudocodice, supponendo che la condizione alla riga 10 sia sempre vera:

- ▶ Le righe 2 e 3 vengono eseguite 1 volta ciascuna.
- ▶ La riga 4 viene eseguita c volte, e la riga 5 c volte.
- ▶ La riga 7 viene eseguita n volte.
- ▶ Le righe 8, 9, 10 e 11 vengono eseguite $n(c + 1)$ volte ciascuna.
- ▶ La riga 15 viene eseguita 1 volta.

Numero di passi (2/2)

Quindi il numero totale di passi nel caso peggiore è

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

Di conseguenza,

$$S(n, c) = \Theta(nc),$$

poiché

$$\begin{aligned}\lim_{n, c \rightarrow \infty} \frac{4n(c + 1) + n + 2c + 3}{nc} &= \lim_{n, c \rightarrow \infty} \left(4 + \frac{4}{c} + \frac{1}{c} + \frac{2}{n} + \frac{3}{nc} \right) \\ &= 4 \in \mathbb{R}_{>0}.\end{aligned}$$

Il numero di passi dell'algoritmo di PD

L'algoritmo di PD per l'SSP esegue

$$S(n, c) = \Theta(nc)$$

passi nel caso peggiore,
lo stesso numero dell'algoritmo di PD del KP.

- ▶ La ricostruzione esegue $\Theta(n)$ passi, uno per ogni oggetto.
- ▶ Il numero di passi contiene il valore obiettivo c , un numero dell'input, e non solo la dimensione dell'input: è **pseudo-polinomiale**.

Riepilogo (1/2)

Passo 1: la risposta:

- ▶ $r(j, k) = r(j - 1, k) \vee r(j - 1, k - w_j)$ se $w_j \leq k$, e
 $r(j, k) = r(j - 1, k)$ se $w_j > k$
- ▶ casi base $r(j, 0) = \text{true}$ e $r(0, k) = \text{false}$ per $k \geq 1$; la risposta è $r(n, c)$
- ▶ sottostruttura ottima e sottoproblemi sovrapposti: bottom-up, riga per riga, in una tabella di lookup

Relazione con il KP:

- ▶ con $p_j = w_j$, $r(j, k) = \text{true}$ se e solo se $v(j, k) = k$
- ▶ un bit per cella, operazioni logiche al posto di somme e confronti

Riepilogo (2/2)

Passo 2: un sottoinsieme:

- ▶ $s[j][k] = 1$ solo se $r[j-1][k]$ è *false* e $r[j-1][k-w_j]$ è *true*
- ▶ il sottoinsieme si legge all'indietro a partire da (n, c) :
 $x_j \leftarrow s[j][k], k \leftarrow k - s[j][k] w_j$
- ▶ sull'esempio: il sottoinsieme $\{1, 4\}$, di peso totale $3 + 7 = 10$

Numero di passi:

- ▶ $S(n, c) = \Theta(nc)$ per SSP_DP, $\Theta(n)$ per la ricostruzione
- ▶ il numero di passi dipende dal valore c : è pseudo-polinomiale

Errori tipici

Introduzione: Errori tipici

- ▶ Raccogliamo gli errori che si fanno più spesso con l'algoritmo di PD per l'SSP
- ▶ Due riguardano i casi base, due la risposta e la ricostruzione
- ▶ Controllateli prima di eseguire a mano l'algoritmo di PD

Errori tipici (1/2)

Osservazione 1: Porre $r(j, 0) = \text{false}$

Il valore obiettivo 0 è sempre raggiungibile con il sottoinsieme vuoto; con $r(j, 0) = \text{false}$ nessun valore diventerebbe mai true.

Osservazione 2: Porre $r(0, k) = \text{true}$ per $k \geq 1$

Senza oggetti è raggiungibile solo il valore obiettivo 0.

Errori tipici (2/2)

Osservazione 3: Leggere la risposta nella cella sbagliata

La risposta è $r(n, c)$; un valore true in un'altra cella dell'ultima riga dice che è raggiungibile un valore obiettivo diverso.

Osservazione 4: Ricostruire quando la risposta è no

Se $r(n, c)$ è false, nessun sottoinsieme ha peso totale c , e la matrice s non dà nulla da ricostruire.

Riepilogo

Casi base:

- ▶ $r(j, 0) = \text{true}$: il sottoinsieme vuoto ha peso totale 0
- ▶ $r(0, k) = \text{false}$ per $k \geq 1$: senza oggetti è raggiungibile solo 0

Risposta e ricostruzione:

- ▶ la risposta è nella cella $r(n, c)$, non in un'altra cella dell'ultima riga
- ▶ si ricostruisce un sottoinsieme solo quando $r(n, c)$ è *true*

In sintesi

Introduzione: In sintesi

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

Riepilogo (1/2)

- ▶ La ricorrenza $r(j, k) = r(j - 1, k) \vee r(j - 1, k - w_j)$ dice se il valore obiettivo k è raggiungibile con gli oggetti $1, 2, \dots, j$.
- ▶ I casi base sono $r(j, 0) = \text{true}$ e $r(0, k) = \text{false}$ per $k \geq 1$; la risposta è $r(n, c)$.
- ▶ La matrice s memorizza se l'oggetto j è usato, e un sottoinsieme di peso totale c si legge all'indietro a partire da (n, c) .
- ▶ L'algoritmo di PD esegue $\Theta(nc)$ passi, come l'algoritmo di PD del KP con $p_j = w_j$.

Riepilogo (2/2)

Quadro riassuntivo dell'esempio dell'SSP ($c = 10$, quattro oggetti):

Metodo	Risposta	Origine
Algoritmo di PD	sì	$r(4, 10) = \text{true}$, sottoinsieme $\{1, 4\}$