Lezione 12

Sommario della Lezione

Programmazione Dinamica

Passi per la risoluzione via PD

- dare una formulazione ricorsiva della soluzione al problema;

Passi per la risoluzione via PD

- dare una formulazione ricorsiva della soluzione al problema;
- far precedere ogni chiamata ricorsiva per la soluzione di sottoproblemi con un controllo, ciò al fine di verificare se il sottoproblema corrente è già stato risolto;

Passi per la risoluzione via PD

- dare una formulazione ricorsiva della soluzione al problema;
- far precedere ogni chiamata ricorsiva per la soluzione di sottoproblemi con un controllo, ciò al fine di verificare se il sottoproblema corrente è già stato risolto;
- nel caso positivo (ovvero nel caso in cui il sottoproblema è stato già risolto in precedenza) limitarsi a leggere dalla tabella la soluzione precedentemente calcolata

Passi per la risoluzione via PD

- dare una formulazione ricorsiva della soluzione al problema;
- far precedere ogni chiamata ricorsiva per la soluzione di sottoproblemi con un controllo, ciò al fine di verificare se il sottoproblema corrente è già stato risolto;
- nel caso positivo (ovvero nel caso in cui il sottoproblema è stato già risolto in precedenza) limitarsi a leggere dalla tabella la soluzione precedentemente calcolata
- nel caso negativo (ovvero nel caso in cui il sottoproblema non è stato già risolto in precedenza), risolvere il problema e memorizzare la soluzione in una opportuna entrata della tabella, per futuri usi.

PD è simile a D&I, ma...

PD è simile a D&I, ma...

```
Choose(n, r)
1. IF (r = 0||n = r) {
2.       RETURN(1)
3.       } ELSE {
4.       RETURN(Choose(n − 1, r − 1)+ Choose(n − 1, r))
         }
```

PD è simile a D&I, ma...

```
Choose(n, r)
1. IF (r = 0||n = r) {
2.       RETURN(1)
3.       } ELSE {
4.       RETURN(Choose(n − 1, r − 1)+ Choose(n − 1, r))
         }
```

```
MemChoose(n, r)
1. IF(r = 0||n = r) {
2.    RETURN(1)
3.    } ELSE {
4.       IF( T[n, r] non è definito) {
5.             T[n, r] =MemChoose(n − 1, r − 1) +MemChoose(n − 1, r)
6.          }
7.    }
8. RETURN( T[n, r])
```

Qual è la principale difficoltà per applicare PD?

Qual è la principale difficoltà per applicare PD?

Formulare il problema in termini ricorsivi!

Qual è la principale difficoltà per applicare PD?

Formulare il problema in termini ricorsivi!
In questa lezione vederemo due esempi, nel primo la formulazione ricorsiva del computo della soluzione è semplice, nel secondo occorre pensarci sù...

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X. Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1
5	1	2

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1
5	1	2
4	3	1

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1
5	1	2
4	3	1
4	2	2

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1
5	1	2
4	3	1
4	2	2
4	1	3

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1
5	1	2
4	3	1
4	2	2
4	1	3
3	3	2

Supponiamo di avere n dadi, ciascuno con m facce, ed un intero X.
Ogni faccia di ciascun dado è etichettata con un distinto intero $i \in \{1, \ldots, m\}$.
Problema: determinare qual è il numero dei possibili lanci di tutti gli n dadi che danno somma pari a X.
Ad es., supponiamo che $X = 8$, abbiamo 3 dadi, ciascuno con 6 facce etichettate con gli interi da 1 a 6. Vi sono svariati lanci dei 3 dadi che possono avere **somma totale** 8, ad es.

I dado	II dado	III Dado
6	1	1
5	2	1
5	1	2
4	3	1
4	2	2
4	1	3
3	3	2
3	2	3
⋮	⋮	⋮

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\text{Modi}(m, n, X) = \text{Modi}(m, n-1, X-1) + \tag{1}$$

Sia $\text{Modi}(m, n, X) =$ numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \end{aligned} \tag{1}$$

Sia $\text{Modi}(m, n, X) =$ numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = & \text{Modi}(m, n-1, X-1) + \\ & \text{Modi}(m, n-1, X-2) + \\ & \dots \end{aligned} \tag{1}$$

Sia $\text{Modi}(m, n, X) =$ numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
5.       M[n,X]=0
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
5.       M[n,X]=0
6.       FOR(k=1;k<m+1;k=k+1){
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned}\text{Modi}(m, n, X) =& \text{Modi}(m, n-1, X-1)+ \\ & \text{Modi}(m, n-1, X-2)+ \\ & \dots \\ & \text{Modi}(m, n-1, X-m).\end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
5.       M[n,X]=0
6.       FOR(k=1;k<m+1;k=k+1){
7.       M[n,X]=M[n,X]+CalcolaModi(m,n-1,X-k)
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
5.       M[n,X]=0
6.       FOR(k=1;k<m+1;k=k+1){
7.       M[n,X]=M[n,X]+CalcolaModi(m,n-1,X-k)
    }
}
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned} \text{Modi}(m, n, X) = &\text{Modi}(m, n-1, X-1) + \\ &\text{Modi}(m, n-1, X-2) + \\ &\dots \\ &\text{Modi}(m, n-1, X-m). \end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
5.       M[n,X]=0
6.       FOR(k=1;k<m+1;k=k+1){
7.       M[n,X]=M[n,X]+CalcolaModi(m,n-1,X-k)
    }
}
return M[n,X]
```

Sia Modi(m, n, X) = numero dei possibili lanci di tutti gli n dadi che danno somma pari a X (il numero che vogliamo calcolare)
Vale

$$\begin{aligned}\text{Modi}(m, n, X) =& \text{Modi}(m, n-1, X-1)+ \\ & \text{Modi}(m, n-1, X-2)+ \\ & \dots \\ & \text{Modi}(m, n-1, X-m).\end{aligned} \tag{1}$$

```
CalcolaModi(m,n,X)
1. IF ((n==1))&&(1<=X<=m)) {
2.   return 1
3.     } ELSE {
4.     IF(M[n,X] non è definito){
5.       M[n,X]=0
6.       FOR(k=1;k<m+1;k=k+1){
7.       M[n,X]=M[n,X]+CalcolaModi(m,n-1,X-k)
    }
}
return M[n,X]
```

Complessità : $T(n, m, X) = \Theta(nmX)$

Finora abbiamo applicato Programmazione Dinamica a problemi la cui soluzione era **naturalmente** espressa in termini ricorsivi

Finora abbiamo applicato Programmazione Dinamica a problemi la cui soluzione era **naturalmente** espressa in termini ricorsivi

1. Calcolo dei numeri di Fibonacci $F_n = F_{n-1} + F_{n-2}$

Finora abbiamo applicato Programmazione Dinamica a problemi la cui soluzione era **naturalmente** espressa in termini ricorsivi

1. Calcolo dei numeri di Fibonacci $F_n = F_{n-1} + F_{n-2}$
2. Calcolo delle combinazioni $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$

Finora abbiamo applicato Programmazione Dinamica a problemi la cui soluzione era **naturalmente** espressa in termini ricorsivi

1. Calcolo dei numeri di Fibonacci $F_n = F_{n-1} + F_{n-2}$
2. Calcolo delle combinazioni $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$
3. e l'ultimo problema visto...

Finora abbiamo applicato Programmazione Dinamica a problemi la cui soluzione era **naturalmente** espressa in termini ricorsivi

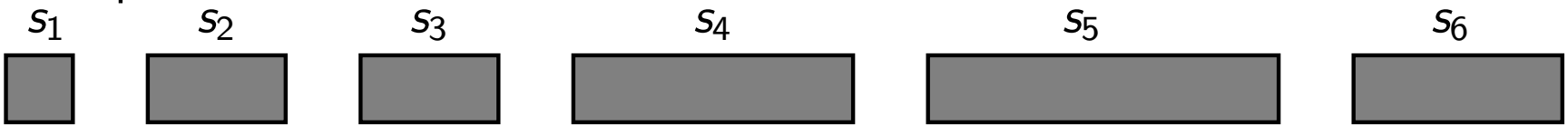
1. Calcolo dei numeri di Fibonacci $F_n = F_{n-1} + F_{n-2}$
2. Calcolo delle combinazioni $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$
3. e l'ultimo problema visto...

Per problemi che occorrono in pratica, per poter applicare la tecnica Programmazione Dinamica dovremo **noi** dare dare una formulazione ricorsiva alla sua soluzione.

Input: n oggetti di lunghezza $s_1, \ldots, s_n$, valore S
Output: `True` se e solo se esiste un sottoinsieme di tali oggetti di lunghezza totale S; `False`, altrimenti.

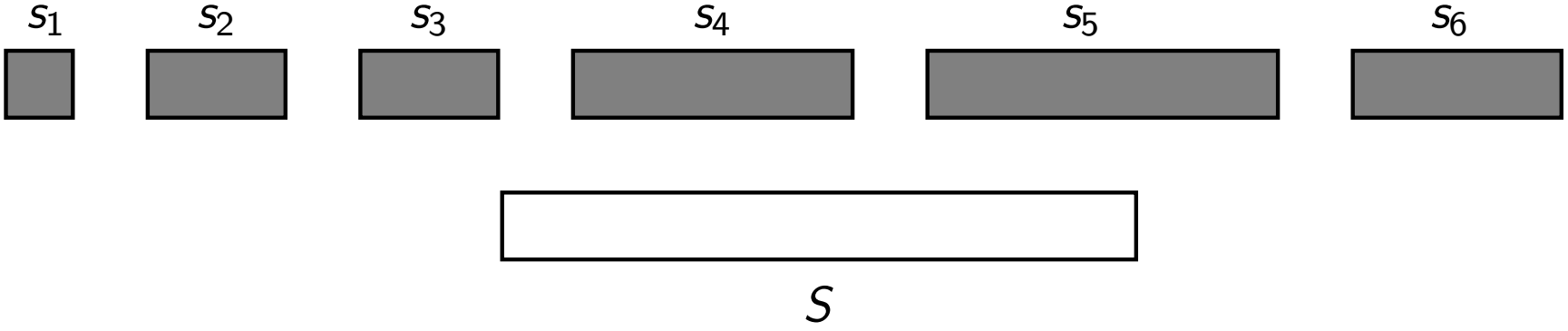
Input: n oggetti di lunghezza $s_1, \ldots, s_n$, valore S
Output: `True` se e solo se esiste un sottoinsieme di tali oggetti di lunghezza totale S; `False`, altrimenti.

Esempio:



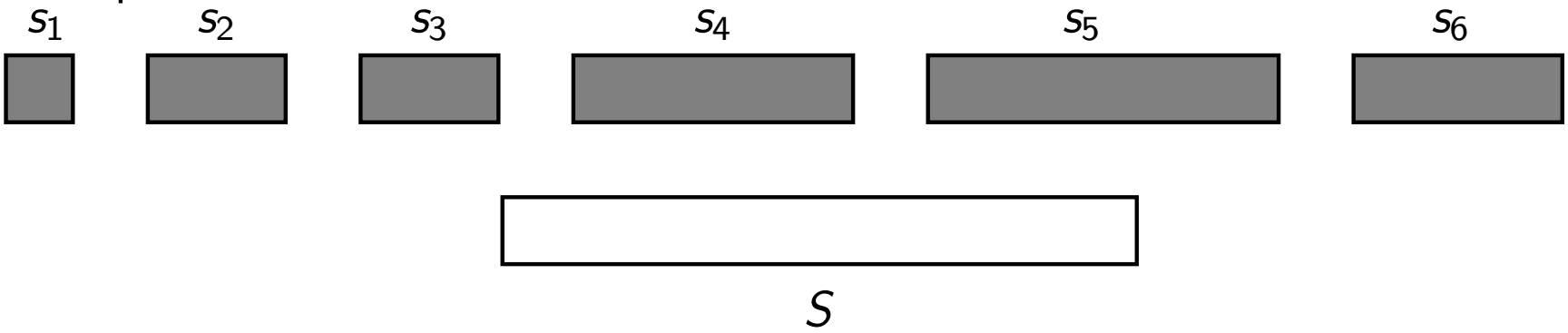
Input: n oggetti di lunghezza $s_1, \ldots, s_n$, valore S
Output: `True` se e solo se esiste un sottoinsieme di tali oggetti di lunghezza totale S; `False`, altrimenti.

Esempio:

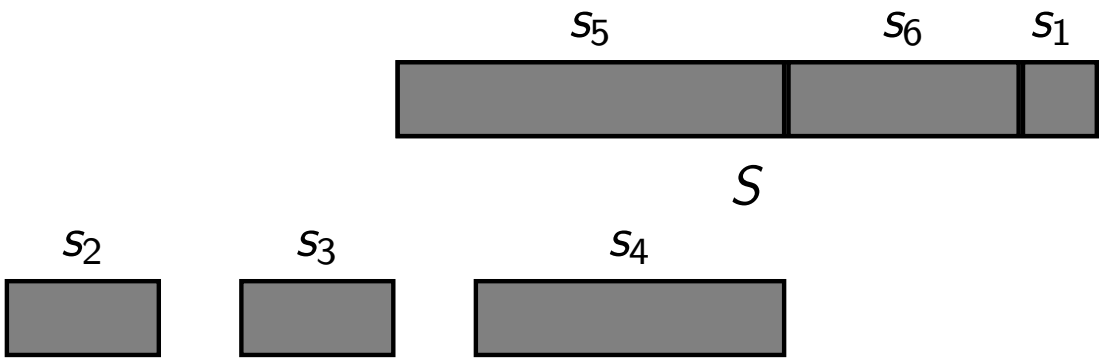


Input: n oggetti di lunghezza $s_1, \ldots, s_n$, valore S
Output: `True` se e solo se esiste un sottoinsieme di tali oggetti di lunghezza totale S; `False`, altrimenti.

Esempio:



In questo caso si vede che la risposta è `True`, in quanto la somma delle lunghezze s_1, s_5 e s_6 è esattamente pari a S.



Per motivi storici, questo problema viene chiamato Problema dello Zaino (versione semplice) in cui S rappresenta la dimensione di un ipotetico zaino e ci chiediamo se possiamo trovare un sottoinsieme di oggetti (scelti tra gli n disponibili di dimensione $s_1, \ldots, s_n$) la cui somma delle dimensioni è proprio pari a S.

Per procedere alla soluzione attraverso la PD, occorre dare una formulazione ricorsiva della soluzione mediante soluzioni a sottoproblemi dell'istanza di input di partenza.

Per procedere alla soluzione attraverso la PD, occorre dare una formulazione ricorsiva della soluzione mediante soluzioni a sottoproblemi dell'istanza di input di partenza.

Stabiliamo che un generico sottoproblema del problema di partenza è individuato da un **sottoinsieme** $s_1, \ldots, s_i$ $i \leq n$, delle lunghezze in input, e da un numero $j \leq S$ che rappresenta, concettualmente, la dimensione ipotetica di un **sottozaino** dello zaino di partenza.

Per procedere alla soluzione attraverso la PD, occorre dare una formulazione ricorsiva della soluzione mediante soluzioni a sottoproblemi dell'istanza di input di partenza.

Stabiliamo che un generico sottoproblema del problema di partenza è individuato da un **sottoinsieme** $s_1, \ldots, s_i$ $i \leq n$, delle lunghezze in input, e da un numero $j \leq S$ che rappresenta, concettualmente, la dimensione ipotetica di un **sottozaino** dello zaino di partenza.

Progettiamo un algoritmo `Zaino`(i, j) che risolve il sottoproblema individuato dalla coppia $(\{s_1, \ldots, s_i\}, j)$, per arbitrari $i = 1, \ldots, n$ e $j = 1, \ldots, S$.

Per procedere alla soluzione attraverso la PD, occorre dare una formulazione ricorsiva della soluzione mediante soluzioni a sottoproblemi dell'istanza di input di partenza.

Stabiliamo che un generico sottoproblema del problema di partenza è individuato da un **sottoinsieme** $s_1, \ldots, s_i$ $i \leq n$, delle lunghezze in input, e da un numero $j \leq S$ che rappresenta, concettualmente, la dimensione ipotetica di un **sottozaino** dello zaino di partenza.

Progettiamo un algoritmo `Zaino`(i, j) che risolve il sottoproblema individuato dalla coppia $(\{s_1, \ldots, s_i\}, j)$, per arbitrari $i = 1, \ldots, n$ e $j = 1, \ldots, S$.

Ovvero, `Zaino`(i, j) su input $(\{s_1, \ldots, s_i\}, j)$ deve restituire `True` se e solo se esiste un *sottoinsieme* dei primi i oggetti la cui somma delle lunghezze sia pari a j

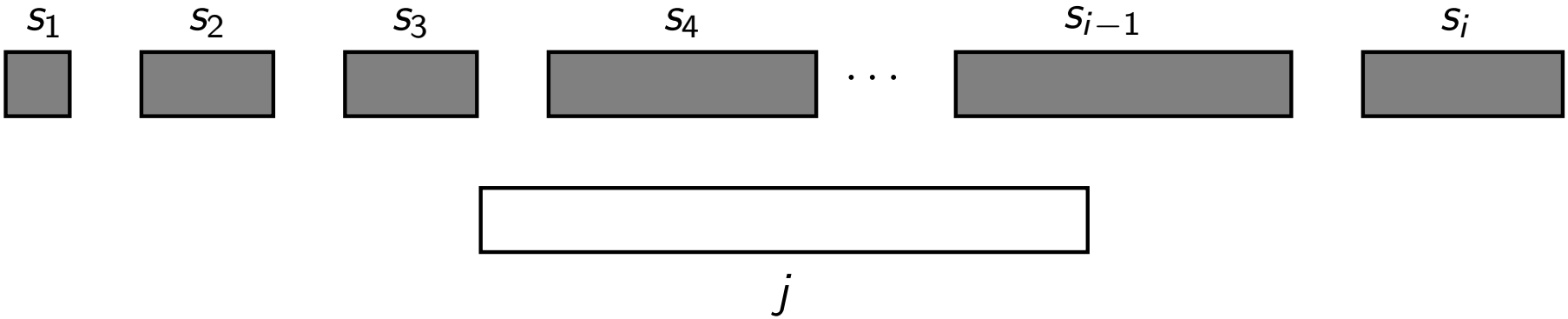
Per procedere alla soluzione attraverso la PD, occorre dare una formulazione ricorsiva della soluzione mediante soluzioni a sottoproblemi dell'istanza di input di partenza.

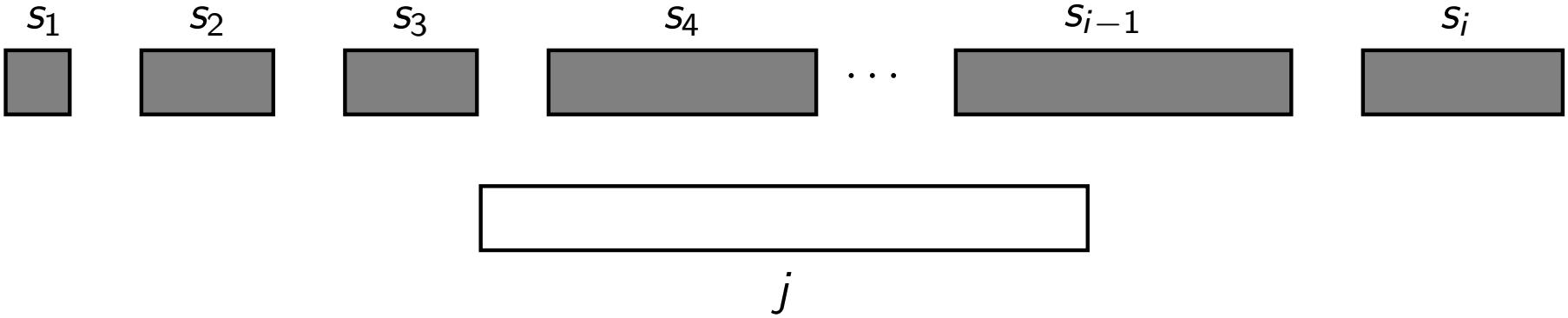
Stabiliamo che un generico sottoproblema del problema di partenza è individuato da un **sottoinsieme** $s_1, \ldots, s_i$ $i \leq n$, delle lunghezze in input, e da un numero $j \leq S$ che rappresenta, concettualmente, la dimensione ipotetica di un **sottozaino** dello zaino di partenza.

Progettiamo un algoritmo `Zaino`(i, j) che risolve il sottoproblema individuato dalla coppia $(\{s_1, \ldots, s_i\}, j)$, per arbitrari $i = 1, \ldots, n$ e $j = 1, \ldots, S$.

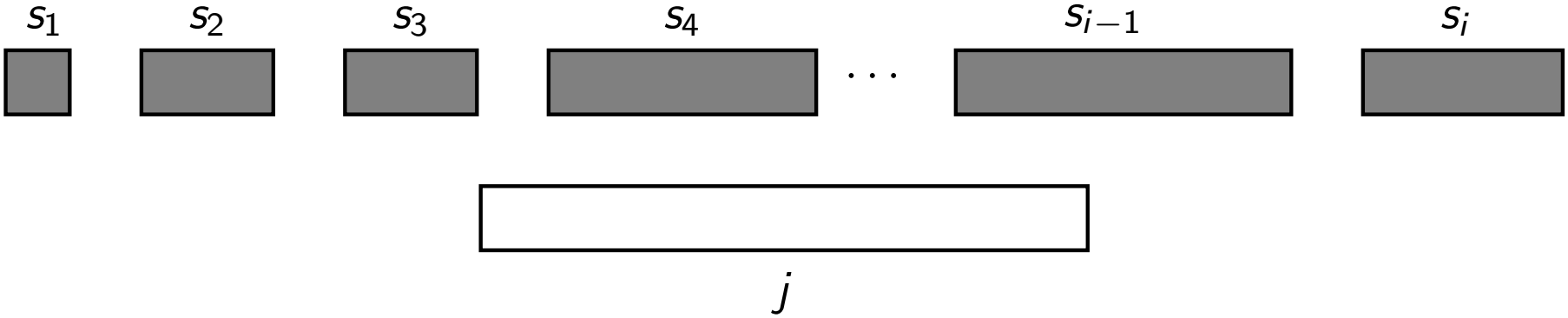
Ovvero, `Zaino`(i, j) su input $(\{s_1, \ldots, s_i\}, j)$ deve restituire `True` se e solo se esiste un *sottoinsieme* dei primi i oggetti la cui somma delle lunghezze sia pari a j

Alla fine ci servirà `Zaino`(n, S)


s_1
s_2
s_3
s_4
s_{i-1}
s_i
j

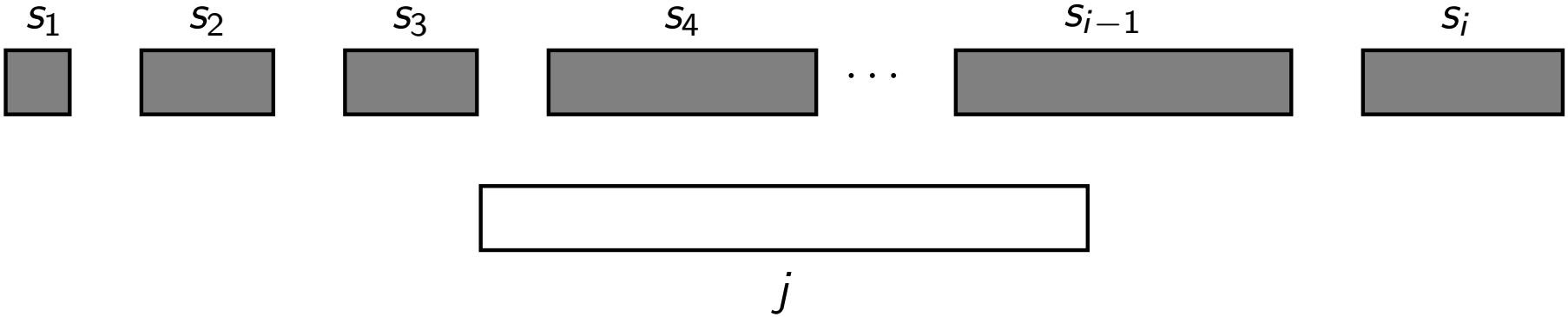


Quando `Zaino`(i, j) restituirà valore `True`?



Quando `Zaino`(i, j) restituirà valore `True`? Distinguiamo i due casi:

1. o **usiamo** l'oggetto i-esimo (e quindi la sua lunghezza) s_i per arrivare alla lunghezza totale j,

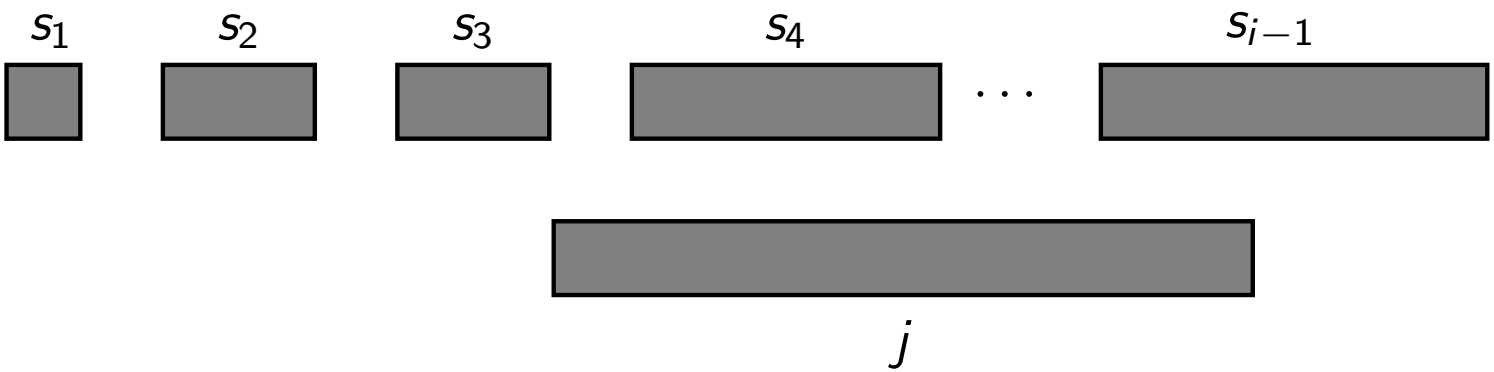


Quando `Zaino`(i, j) restituirà valore `True`? Distinguiamo i due casi:

1. o **usiamo** l'oggetto i-esimo (e quindi la sua lunghezza) s_i per arrivare alla lunghezza totale j,
2. oppure **non usiamo** l'oggetto i-esimo.

Nel caso 2., se l'oggetto i-esimo **non viene usato** per arrivare alla lunghezza totale j, allora `Zaino`(i, j) restituirà valore `True` se e solo se `Zaino`$(i-1, j)$ restituisce valore `True`.

Nel caso 2., se l'oggetto i-esimo **non viene usato** per arrivare alla lunghezza totale j, allora $\texttt{Zaino}(i, j)$ restituirà valore `True` se e solo se $\texttt{Zaino}(i-1, j)$ restituisce valore `True`. Detto in altri termini, la soluzione la cerchiamo usando **solo** le lunghezze $s_1, \ldots, s_{i-1}$. Ovvero, siamo in una situazione del tipo di sotto riportata:



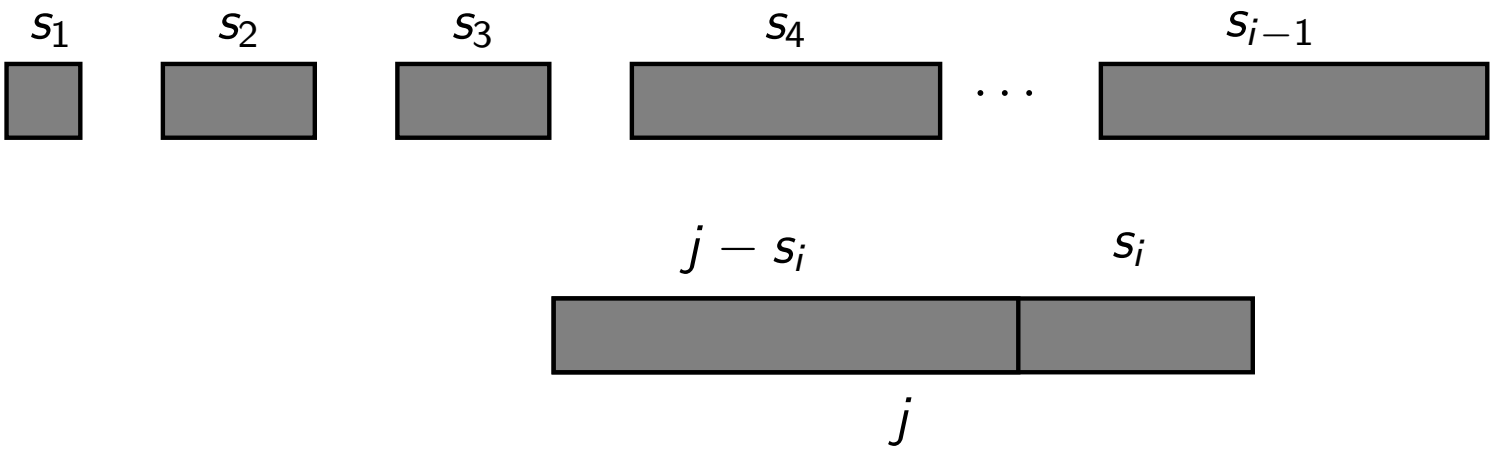
Se invece l'oggetto i-esimo s_i **viene usato** per arrivare alla lunghezza totale j, allora `Zaino`(i, j) restituirà valore `True` se e solo se `Zaino`$(i-1, j-s_i)$ restituirà valore `True`.

Se invece l'oggetto i-esimo s_i **viene usato** per arrivare alla lunghezza totale j, allora $\texttt{Zaino}(i, j)$ restituirà valore `True` se e solo se $\texttt{Zaino}(i-1, j-s_i)$ restituirà valore `True`.

Detto in altri termini, siamo **prima** riusciti a trovare un sottoinsieme dei primi $i-1$ oggetti di lunghezza **totale** $j - s_i$ con cui, **aggiungendo** poi l'i-esimo oggetto, arriviamo ad una lunghezza totale pari a $(j - s_i) + s_i = j$.

Se invece l'oggetto i-esimo s_i **viene usato** per arrivare alla lunghezza totale j, allora $\texttt{Zaino}(i, j)$ restituirà valore `True` se e solo se $\texttt{Zaino}(i-1, j-s_i)$ restituirà valore `True`.

Detto in altri termini, siamo **prima** riusciti a trovare un sottoinsieme dei primi $i-1$ oggetti di lunghezza **totale** $j - s_i$ con cui, **aggiungendo** poi l'i-esimo oggetto, arriviamo ad una lunghezza totale pari a $(j - s_i) + s_i = j$. Cioè , siamo nella situazione:



Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

`Zaino`(i, j) %ritorna `True` se e solo se con un sottoinsieme di $s_1, \ldots, s_i$ si può riempire tutto j

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

`Zaino`(i, j) %ritorna `True` se e solo se con un sottoinsieme di $s_1, \ldots, s_i$ si può riempire tutto j

1. `IF`$(i == 0)$ {

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

```
Zaino(i, j)      %ritorna True se e solo se con un sottoinsieme
                          di s_1, ..., s_i si può riempire tutto j
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
```

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

```
Zaino(i, j)      %ritorna True se e solo se con un sottoinsieme
                          di s_1, ..., s_i si può riempire tutto j
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
          } ELSE {
7. IF(s_i ≤ j) {
```

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

```
Zaino(i, j)      %ritorna True se e solo se con un sottoinsieme
                          di s_1, ..., s_i si può riempire tutto j
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
         } ELSE {
7. IF(s_i ≤ j) {
8.    RETURN Zaino(i − 1, j)||Zaino(i − 1, j − s_i)
```

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

```
Zaino(i, j)      %ritorna True se e solo se con un sottoinsieme
                          di s_1, ..., s_i si può riempire tutto j
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
         } ELSE {
7. IF(s_i ≤ j) {
8.    RETURN Zaino(i − 1, j)||Zaino(i − 1, j − s_i)
9.    } ELSE {
10. RETURN Zaino(i − 1, j)
     }   }
```

Ma noi non sappiamo **a priori** se usare o no l'oggetto i-esimo! Come ce la caviamo?

Proviamo **entrambe** le alternative (ovvero proviamo a riempire uno zaino di capacità j **usando** l'oggetto i-esimo e proviamo **anche** a riempire lo zaino **senza usare** l'oggetto i-esimo, restituendo `True` se riusciamo a riempirlo con **almeno** una delle due alternative).

```
Zaino(i, j)      %ritorna True se e solo se con un sottoinsieme
                          di s_1, ..., s_i si può riempire tutto j
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
          } ELSE {
7. IF(s_i ≤ j) {
8.    RETURN Zaino(i − 1, j)||Zaino(i − 1, j − s_i)
9.    } ELSE {
10. RETURN Zaino(i − 1, j)
     }    }
```

Complessità : $T(n) = 2T(n-1) + d$ se $n > 1$

Ma abbiamo già visto che la soluzione all'equazione di ricorrenza

$$T(n) = \begin{cases} c & \text{se } n = 1, \\ 2T(n-1) + d & \text{se } n > 1 \end{cases}$$

è $T(n) = \Theta(2^n)$

Ma abbiamo già visto che la soluzione all'equazione di ricorrenza

$$T(n) = \begin{cases} c & \text{se } n = 1, \\ 2T(n-1) + d & \text{se } n > 1 \end{cases}$$

è $T(n) = \Theta(2^n)$

Usiamo allora la tecnica della Memization, ovvero usiamo una tabella $t[\cdot, \cdot]$ dove memorizziamo soluzioni a sottoproblemi e, prima di ciascuna chiamata ricorsiva, **verificare** se la soluzione che cerchiamo in quella chiamata ricorsiva, è stata o meno già computata in precedenza.

Ma abbiamo già visto che la soluzione all'equazione di ricorrenza

$$T(n) = \begin{cases} c & \text{se } n = 1, \\ 2T(n-1) + d & \text{se } n > 1 \end{cases}$$

è $T(n) = \Theta(2^n)$

Usiamo allora la tecnica della Memization, ovvero usiamo una tabella $t[\cdot, \cdot]$ dove memorizziamo soluzioni a sottoproblemi e, prima di ciascuna chiamata ricorsiva, **verificare** se la soluzione che cerchiamo in quella chiamata ricorsiva, è stata o meno già computata in precedenza.

Se la soluzione è stata già computata, non effettuiamo la chiamata ricorsiva e ci limitiamo a ritornare il valore della soluzione già calcolata e memorizzata precedentemente nella tabella.

```
MemZaino(i, j)
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
```

```
MemZaino(i, j)
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
```

```
MemZaino(i, j)
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
          } ELSE {
7. IF(t[i, j] non è definito) {
```

```
MemZaino(i, j)
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
          } ELSE {
7. IF(t[i, j] non è definito) {
8.    IF(s_i ≤ j) {
9.       t[i, j] = MemZaino(i − 1, j)||MemZaino(i − 1, j − s_i)
```

```
MemZaino(i, j)
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
          } ELSE {
7. IF(t[i, j] non è definito) {
8.    IF(s_i ≤ j) {
9.       t[i, j] = MemZaino(i − 1, j)||MemZaino(i − 1, j − s_i)
10.       } ELSE {
11.       t[i, j] = MemZaino(i − 1, j)
```

```
MemZaino(i, j)
1. IF(i == 0) {
3.    IF(j == 0) {
4.       RETURN True
5.       } ELSE {
6.       RETURN False
          } ELSE {
7. IF(t[i, j] non è definito) {
8.    IF(s_i ≤ j) {
9.       t[i, j] = MemZaino(i − 1, j)||MemZaino(i − 1, j − s_i)
10.       } ELSE {
11.       t[i, j] = MemZaino(i − 1, j)
       }   }    }
12. RETURN t[i, j]
```

Complessità

Ogni entrata della tabella $t[\cdot, \cdot]$ viene computata una ed una sola volta, ed **ogni** entrata richiede tempo costante per essere computata.

Complessità

Ogni entrata della tabella $t[\cdot, \cdot]$ viene computata una ed una sola volta, ed **ogni** entrata richiede tempo costante per essere computata.

Essendo il numero di entrate della tabella $t[\cdot, \cdot]$ pari ad nS, otteniamo che il numero totale di operazioni elementari eseguite da `MemZaino`(n, S) è $\Theta(nS)$

Possiamo dare una versione iterativa dell'algoritmo in cui memorizziamo $\texttt{Zaino}(i, j)$ in una tabella $t[i, j]$, dove $t[i, j] = \texttt{True}$ se e solo se

- **o** vale che $t[i - 1, j] = \texttt{True}$,

Possiamo dare una versione iterativa dell'algoritmo in cui memorizziamo $\texttt{Zaino}(i, j)$ in una tabella $t[i, j]$, dove $t[i, j] = \texttt{True}$ se e solo se

- **o** vale che $t[i-1, j] = \texttt{True}$,
- **oppure** $t[i-1, j-s_i]$ ha senso (ovvero $j - s_i \geq 0$) ed è stato posto in precedenza a $\texttt{True}$

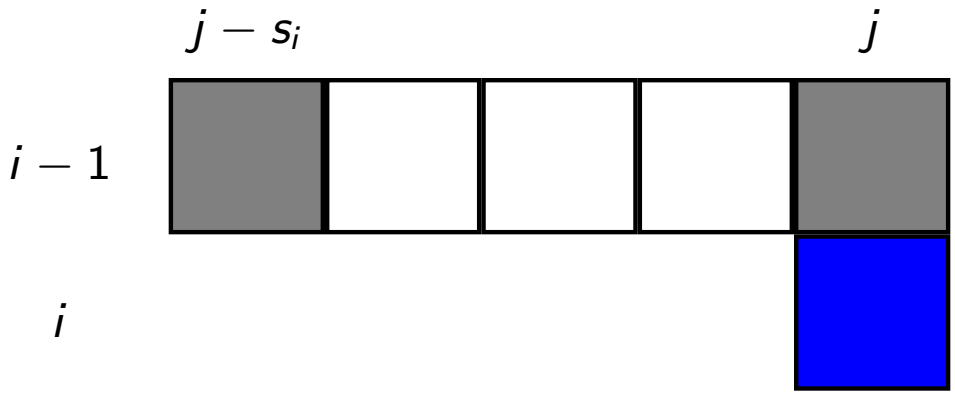
Possiamo dare una versione iterativa dell'algoritmo in cui memorizziamo $\texttt{Zaino}(i, j)$ in una tabella $t[i, j]$, dove $t[i, j] = \texttt{True}$ se e solo se

- **o** vale che $t[i-1, j] = \texttt{True}$,
- **oppure** $t[i-1, j-s_i]$ ha senso (ovvero $j - s_i \geq 0$) ed è stato posto in precedenza a $\texttt{True}$

Ciò viene fatto col seguente codice

$t[i, j] = t[i-1, j]$
$\texttt{IF}(j - s_i \geq 0)\ \{$
$\quad t[i, j] = (t[i, j] || t[i-1, j-s_i])$
$\quad \}$

Mostriamo come vengono riempite le entrate della tabella $t[\cdot, \cdot]$, dove l'entrata di colore blu viene calcolata sulla base dei valori delle due entrate di colore grigio, poste nella riga precedente.



per calcolare $t[i, j]$ occorre che la riga $i - 1$ sia stata giá calcolata, quindi calcoleremo la matrice $t[\cdot, \cdot]$ riga per riga, da sinistra a destra, e dall'alto in basso

L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

`IterZaino`$(s_1, \ldots, s_n, S)$
1. $t[0,0] =$`True`

L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

```
IterZaino(s_1, ..., s_n, S)
1. t[0,0] =True
2. FOR(j = 1; j < S + 1; j = j + 1) {
3.       t[0,j] =False
```

L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

```
IterZaino(s_1, ..., s_n, S)
1. t[0,0] =True
2. FOR(j = 1; j < S + 1; j = j + 1) {
3.       t[0,j] =False
         }
4. FOR(i = 1; i < n + 1; i = i + 1) {
5.    FOR(j = 0; j < S + 1, j = j + 1) {
```

L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

```
IterZaino(s_1, ..., s_n, S)
1. t[0,0] =True
2. FOR(j = 1; j < S + 1; j = j + 1) {
3.       t[0,j] =False
         }
4. FOR(i = 1; i < n + 1; i = i + 1) {
5.    FOR(j = 0; j < S + 1, j = j + 1) {
6.       t[i,j] = t[i - 1,j]
```

L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

```
IterZaino(s_1, ..., s_n, S)
1.  t[0,0] =True
2.  FOR(j = 1; j < S + 1; j = j + 1) {
3.        t[0,j] =False
          }
4.  FOR(i = 1; i < n + 1; i = i + 1) {
5.     FOR(j = 0; j < S + 1, j = j + 1) {
6.        t[i,j] = t[i - 1,j]
7.        IF(j - s_i ≥ 0) {
8.           t[i,j] = (t[i,j] ∨ t[i - 1,j - s_i])
```

L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

```
IterZaino(s_1, ..., s_n, S)
1. t[0,0] =True
2. FOR(j = 1; j < S + 1; j = j + 1) {
3.       t[0,j] =False
         }
4. FOR(i = 1; i < n + 1; i = i + 1) {
5.    FOR(j = 0; j < S + 1, j = j + 1) {
6.       t[i,j] = t[i - 1,j]
7.       IF(j - s_i ≥ 0) {
8.          t[i,j] = (t[i,j] ∨ t[i - 1,j - s_i])
            }
         }
      }
9. RETURN(t[n,S])
```

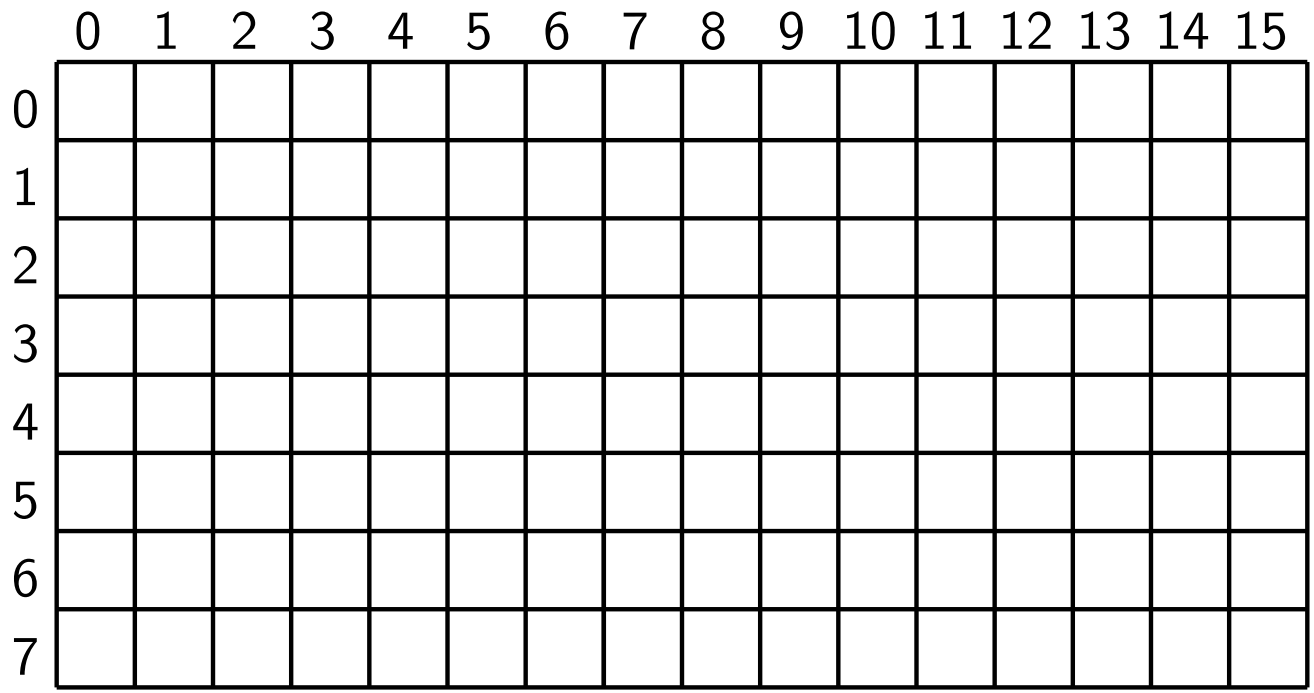
L'algoritmo completo di Programmazione Dinamica, versione iterativa, è

```
IterZaino(s_1, ..., s_n, S)
1.  t[0,0] =True
2.  FOR(j = 1; j < S + 1; j = j + 1) {
3.        t[0,j] =False
          }
4.  FOR(i = 1; i < n + 1; i = i + 1) {
5.     FOR(j = 0; j < S + 1, j = j + 1) {
6.        t[i,j] = t[i - 1,j]
7.        IF(j - s_i ≥ 0) {
8.           t[i,j] = (t[i,j] ∨ t[i - 1,j - s_i])
             }
          }
      }
9.  RETURN(t[n,S])
```

Complessità : $\Theta(nS)$

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$



Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T															
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F														
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F													
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F												
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F											
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F										
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F									
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F								
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F							
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F						
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F					
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F				
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F			
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F		
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1																
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T															
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T														
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F													
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F												
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F											
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F										
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F									
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F								
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F							
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F						
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F					
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F				
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F			
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F		
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2																
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T															
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T														
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T													
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T												
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F											
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F										
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F									
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F								
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F							
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F						
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F	F					
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F	F	F				
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F	F	F	F			
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F	F	F	F	F		
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	
3																
4																
5																
6																
7																

Esempio sull'input

$s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 4, s_5 = 5, s_6 = 2, s_7 = 4, S = 15$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
3																
4																
5																
6																
7																