Problema dello Zaino 0-1

n oggetti $A = \{a_1, a_2, \ldots, a_n\}$

n valori $v_1, v_2, \ldots, v_n$ interi positivi

n pesi $w_1, w_2, \ldots, w_n$ interi positivi

W peso massimo sopportato dallo zaino

Problema trovare il sottoinsieme $S \subseteq A$ che rappresenti il carico più prezioso di peso $\leq W$.

1) Strategia "Greedy"

→ si selezionano gli oggetti in ordine di valore decrescente.

oggetti	peso	valore
a_1	5	10
a_2	4	5
a_3	4	6

$W = 8\ kg$

$S = \{a_1\}$ $\quad V = 10 €$

NON TROVA L'OTTIMO

$S_{ottimo} = \{a_2, a_3\}$ $\quad V = 11 €$

$T(n) = \Theta(n \log n)$

→ Si scelgono gli oggetti in ordine di valore specifico $\left(\frac{\text{valore}}{\text{peso}}\right)$

$W = 8$ kg

oggetti	peso	valore	valore/peso
a_1	5	10	2
a_2	4	5	1.25
a_3	4	6	1.50

$S = \{a_1\}$

non ha l'ottimo

Metodo "forza bruta" ⇝ Ricerca esaustiva

considero tutti i possibili sottoinsiemi di $A = \{a_1, \dots, a_n\}$

↓ sono 2^n

$S \subseteq A$ ⟺ array binario di dim n, B_S

$$B_S[i] = \begin{cases} 1 & a_i \in S \\ 0 & a_i \notin S \end{cases}$$

$A = \{a_1, a_2, \dots, a_8\}$

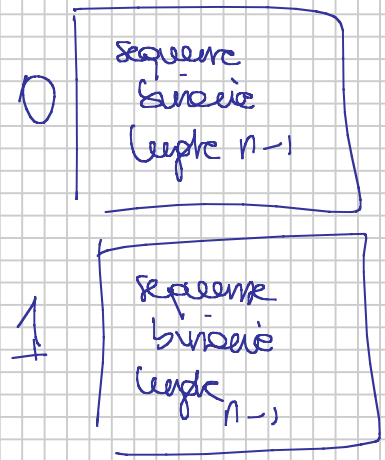
$S = \{a_2, a_3, a_7\}$

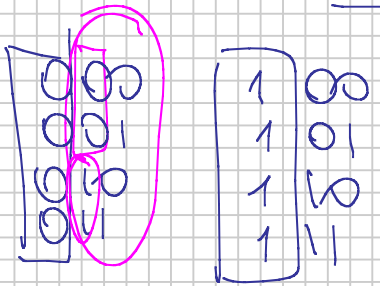
⇝ $B_S =$

1	2	3	4	5	6	7	8
0	1	1	0	0	0	1	0

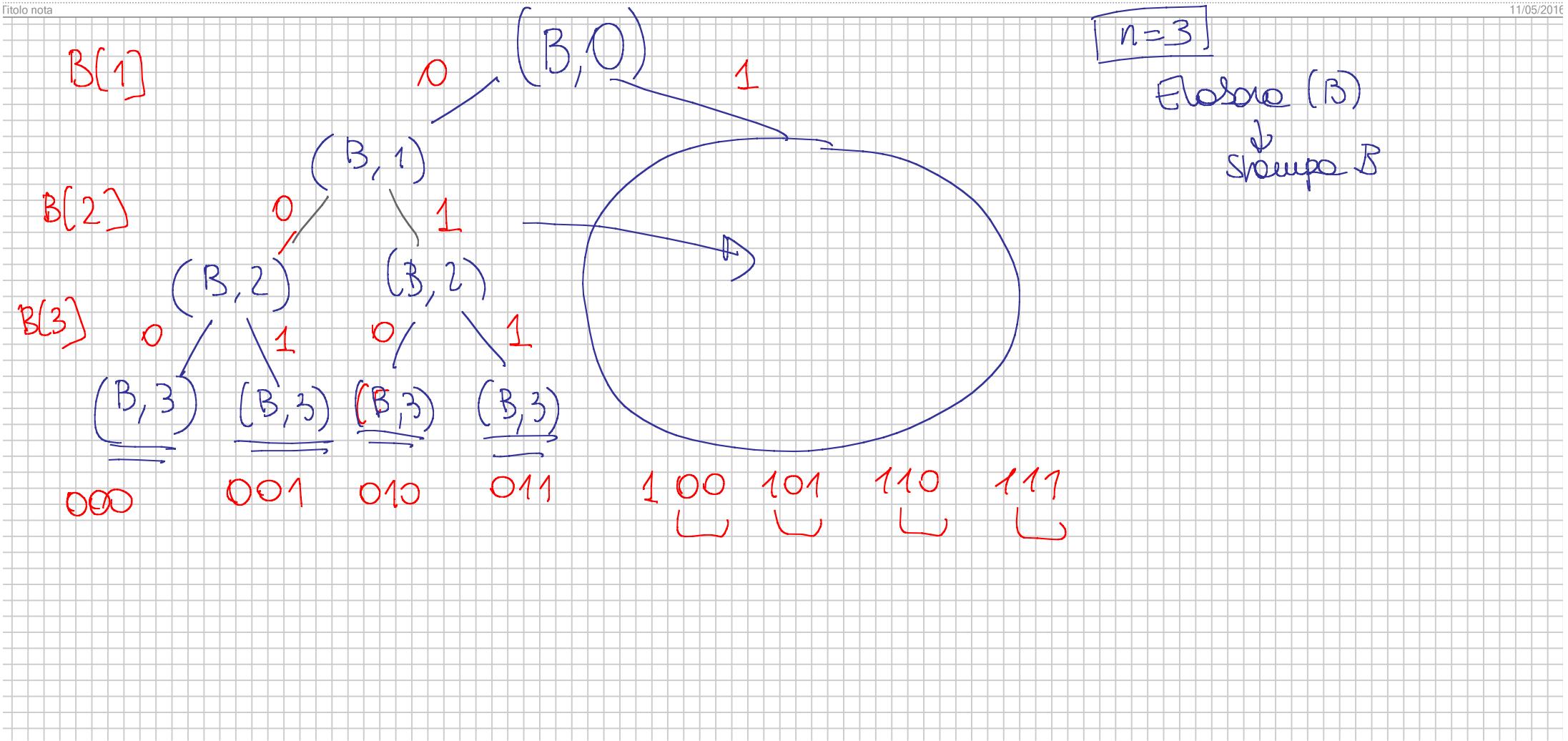
```
GeneraBinarie (B, k)
if (k == n) { Elabora (B) };
else {
    B[k+1] = 0;
    GeneraBinarie (B, k+1);
    B[k+1] = 1;
    GeneraBinarie (B, k+1);
}
```

// prima chiamata: GeneraBinarie (B, 0)





$$T(n) = \Theta\left(\underbrace{2^n}_{\uparrow} \cdot \text{costo Elabora}(B)\right)$$


B[1]
B[2]
B[3]
(B,0)
(B,1)
(B,2)
(B,2)
(B,3)
(B,3)
(B,3)
(B,3)
0
1
0
1
0
1
0
1
000
001
010
011
100
101
110
111
n=3
Elabora (B)
Stampa B

```
Zaino(v, w, W)
Vmax = 0
Sol = nuovo array booleano di dim. n
B = nuovo array booleano di dim. n
GeneraBinario(B, 0);
// Vmax: contiene il valore del carico più pesante di peso ≤ W
// Sol: descrive il sottinsieme che rappresenta il carico più pesante
print Vmax;
print Sol;
```

// v: array di n "valori"
// w: array di n "pesi"
// W: peso max sopportabile dallo zaino

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

2^n = # sottinsiemi

n = costo di Elabora

```
Elabora(B)   // B descrive S ⊆ A
    peso = 0
    valore = 0
    for i = 1 to n {
        if (B[i] == 1) {
            peso = peso + w[i];
            valore = valore + v[i];
        }
    }
    if (peso ≤ W && valore > Vmax) {
        Vmax = valore
        for i = 1 to n { sol[i] = B[i] }
    }
```

```
//  for i = 1 to n
//      peso = peso + B[i] * w[i];
//      valore = valore + B[i] * v[i];
//
```

$T_{Elabora}(n) = \Theta(n)$

(3) Programmazione dinamica

1) Definizione dei sottoproblemi

$A = \{a_1, a_2, \dots, a_n\}$ $\qquad$ W

$0 \le i \le n$ $\qquad$ $0 \le j \le W$

Sottoproblema i, j: trovare il ~~max~~ sottinsieme dei primi i oggetti

$S \subseteq \{a_1, a_2, \dots, a_i\}$

che rappresenti il carico più pesante di peso $\le j$

$C(i,j)$ = valore del carico più pesante trovato

Matrice $(n+1) \times (W+1)$

② Sottoproblemi elementari

$\forall j \quad c[0, j] = 0$ // non ci sono oggetti da valore

$\forall i \quad c[i, 0] = 0$ // ~~se~~ lo zaino può trasportare 0 kg!

③ Regola ricorsiva

$\boxed{w_i > j} \Rightarrow$ l'i-esimo oggetto non può essere valore

$$c[i, j] = c[i-1, j]$$

$\boxed{w_i \leq j}$

$$c[i,j] = \max \{ c[i-1, j] \,,\; v_i + c[i-1,\, j-w_i] \}$$

$c[i-1, j]$: non prendo a_i

$v_i + c[i-1, j-w_i]$: prendo a_i

④ Soluzione:

$$V_{max} = c[n, W]$$

Algoritmo in pseudo codice

```
Zaino PD (n, v, w, W)
① {  c = nuova matrice (n+1) x (W+1);
② {  for j=0 to W
         c[0,j] = 0;
```

```
for i = 1 to n
    c[i, 0] = 0;
```

i

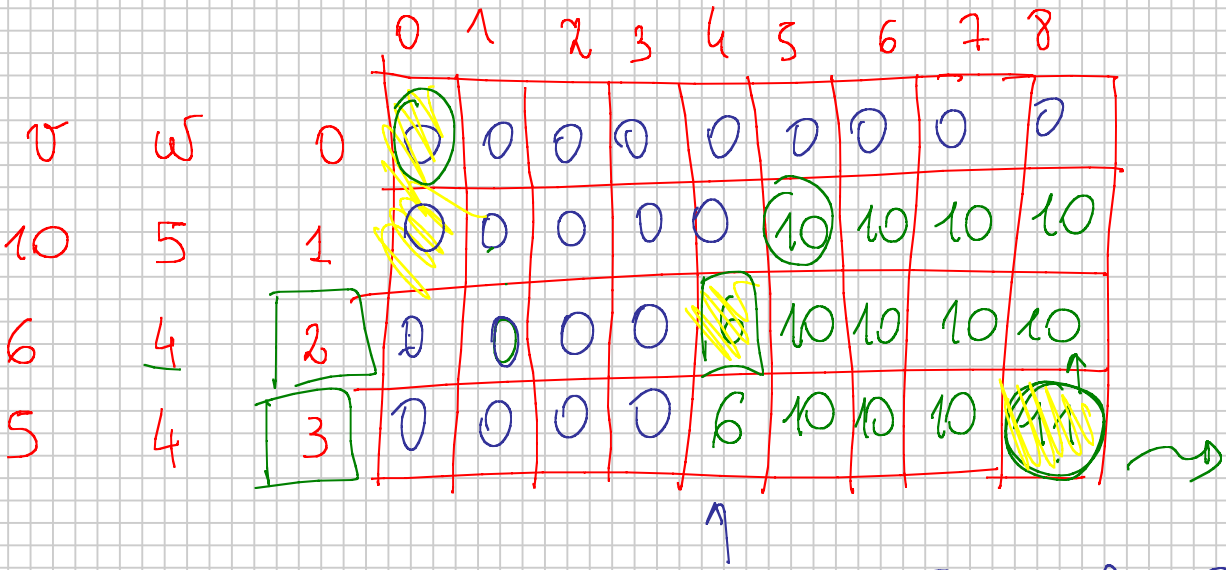
```
for i = 1 to n {
    for j = 1 to W {
        c[i, j] = c[i-1, j];
        if (j ≥ w[i]) {
            m = c[i-1, j-w[i]] + v[i];
            if (m > c[i, j]) c[i, j] = m;
        }
    }
}
return c[n, W];
```

(3)

(4)

$$T(n, W) = \Theta(n \cdot W)$$

$\Theta(1)$



$\max\{10, 6\}$

se non rubo Q_3:
$C[2,8] = 10$

se rubo Q_3:
$5 + C[2,4] =$
$5 + 6 = 11$

se rubo Q_3: $5 + C[2,0] = 5$

se non rubo Q_3: $C[2,4] = 6$

Soluzione: $\{a_2, a_3\}$

Soluzione enumerativa $\longleftrightarrow$ Soluzione PD

$T_e(n, W) = \Theta(n \cdot 2^n)$

$T_{PD}(n, W) = \Theta(n * W)$

esponenziale in n

→ polinomiale in n, polinomiale nel VALORE di W

Istanza di input: array dei pesi e dei valori di dim n

W lo scrivo con $\log W$ bit

Algoritmo pseudopolinomiale

polinomiale in n, esponenziale in $\log W$

Se $W = O(n^c)$ c = costante

$$T_{PD}(n, W) = \Theta(n \cdot W) = \Theta(n \cdot n^c) = \Theta(n^{c+1})$$

$\Rightarrow$ l'algoritmo è polinomiale

Se invece $W = \Theta(2^n)$

$$T_{PD}(n, W) = \Theta(n \cdot 2^n) \rightarrow \text{è esponenziale}$$

Esempi

$n = 3$

$W = 1000$ kg

PD

sottoproblemi considerati:

$(n+1) * (W+1) = 4004$

sottoproblemi (senza contare quelli elementari)

$n * W = 3000$

S.E.

sottoproblemi

$2^n = 8$ sottoproblemi

$n = 20$

$W = 4$ kg

PD

sottoinsiemi (non elementari)

$n * W = \underline{\underline{80}}$ (105 capienza)

SE

sottoinsiemi

$2^{20} \simeq 10^6$