Ottimizza la somma

Supponete di avere in input un vettore di n interi positivi distinti $V[1 \dots n]$ e un valore W. Scrivere un algoritmo che:

1. restituisca il massimo valore $X = \sum_{i=1}^{n} x[i]V[i]$ tale che $X \leq W$ e ogni $x[i]$ è un intero non negativo;
2. stampi il vettore x.

Ad esempio, per $V[\,] = \{18, 3, 21, 9, 12, 24\}$ e $W = 17$, una possibile soluzione ottima è $x = [0, 2, 0, 1, 0, 0]$ da cui deriva $X = 15$.
Discutere correttezza e complessità.

Spoiler alert!

Ottimizza la somma

E' possibile notare che questo problema è un caso particolare dello Zaino senza limiti di scelta, quindi – a livello di compito – è possibile semplicemente chiamare il codice che abbiamo discusso a lezione, passando il vettore D sia come peso che come profitto.

---

int bestSum(**int**[ ] V, **int** n, **int** W)

---

return knapsack(V, V, n, W)

La complessità è $O(nW)$.

Ottimizza la somma

Risolviamo invece il problema "da capo". Sia $DP[i][w]$ il massimo valore che posso ottenere seguendo le regole di cui sopra avendo a disposizione i primi i oggetti e un valore massimo w.

$$DP[i][w] = \begin{cases} -\infty & i \geq 0 \wedge w < 0 \\ 0 & i = 0 \vee w = 0 \\ \max\{DP[i-1][w], DP[i][w - V[i]] + V[i]\} & \text{altrimenti} \end{cases}$$

Ottimizza la somma

int bestSum(**int**[ ] V, **int** n, **int** W)

% Crea un vettore DP inizializzato a -1
int[ ][ ] DP = **new int**$[0 \ldots n][1 \ldots W] = \{-1\}$
return bsRec(V, n, W, DP)

int bsRec(**int**[ ] V, **int** i, **int** w, **int**[ ][ ] DP)

if $w < 0$ **then**
  return $-\infty$
if $i == 0$ **or** $w == 0$ **then**
  return 0
if $DP[i][w] < 0$ **then**
  $DP[i][w] =$
    $\max(\text{bsRec}(V, i-1, w, DP), \text{bsRec}(V, i, w - V[i], DP) + V[i])$
return $DP[i][w]$