Subset Sum Problem

$n = 3$

$W_1 = 1 \quad W_2 = 3 \quad W_3 = 3 \; ; \; W$

Goal: output a subset of these 3 numbers s.t. sub-sum is ~~m~~ maximized and sub-sum doesn't exceed W.

(i) $W = 7$; opt sol $= \{1, 2, 3\}$

(ii) $W = 6$; opt sol $= \{2, 3\}$

(iii) $W = 5$; opt sol $= \{1, 2\}$ or $\{1, 3\}$

Note: In general, we can't have op(sub-sum) = W

Input: n numbers; $w_1, \ldots, w_n$; $w_i \geq 0$; $W \geq 0$.

Output: a subset $S \subseteq [n]$ s.t.

(i) $\sum_{i \in S} w_i \leq W$ (ii) $w(S) = \sum_{i \in S} w_i$ is maximized

---

Q: maximize $|S|$ instead of $w(S)$ in (ii)

A: Sort the numbers in increasing order pick as many as we can without exceeding W.

Greedy solution

proof of correctness: Greedy stays ahead.

→ previous greedy approach doesn't work for our problem where we want to maximize $w(S)$.

→ Counterexample: $W = 6$

→ Greedy will pick w_1 and (w_2 / w_3)

→ $\{1, 2\}$ with a total weight of 4.

But, OPT(subset) = 6.

No known greedy algo to solve subset sum

Dynamic program for Subset Sum

Goal: output $w(S)$ for an optimal solution.

1. $w(S)$ is max
2. $w(S) \leq W$

Attempt 1:

O_j : is an optimal subset for $w_1, \dots, w_j$

$$OPT(j) = w(O_j)$$

Case 1: $j \notin O_j$ $\quad OPT(j) = OPT(j-1)$

claim optimal solution for j (i.e., O_j) is also optimal for $w_1, \dots, w_{j-1}$

Case 2: $j \in O_j$

$$OPT(j) = w_j + OPT(j')$$

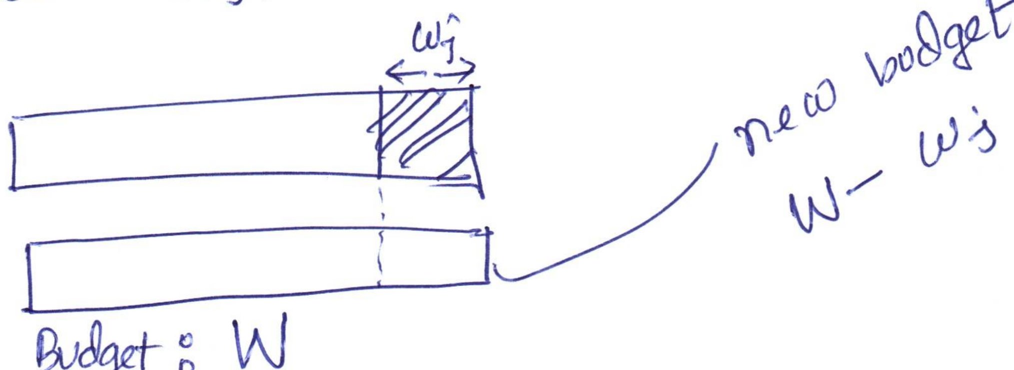
Q: $O_j \setminus \{j\}$ ⇐ what can we say about ↑?

A: $O_j \setminus \{j\}$ is an optimal solution for some number $j' < j$

↑ The above doesn't work as desired

Q: why? what's missing?

If we pick w_j, then what numbers are remaining?



Q: How should we define subproblems?

A: Keep track of both j and budget B.

$OPT(B, j)$ = weight of an optimal subset for numbers $w_1, \ldots, w_j$ and budget B.

Assume $w_j \le B$

Case 1: $j \notin$ $\underbrace{\text{optimal}(w_1, \ldots, w_j; B)}_{\text{total weight of optimal subset} = OPT(B, j)}$

$$OPT(B, j) = OPT(B, j-1)$$

Case 2: $j \in$ optimal $(w_1, \ldots, w_j; B)$

$$OPT(B, j) = w_j + OPT(B - w_j, j-1)$$

$$OPT(B, j) = \max\{w_j + OPT(B - w_j, j-1), OPT(B, j-1)\}$$

if $w_j > B$, $OPT(B, j) = OPT(B, j-1)$

Overall: (*)

$$\boxed{\begin{array}{l} \text{If } w_j > B \text{, then} \\ \quad OPT(B, j) = OPT(B, j-1) \\ \text{else, } OPT(B, j) = \max\{w_j + OPT(B - w_j, j-1), \\ \qquad\qquad\qquad OPT(B, j-1)\} \end{array}}$$

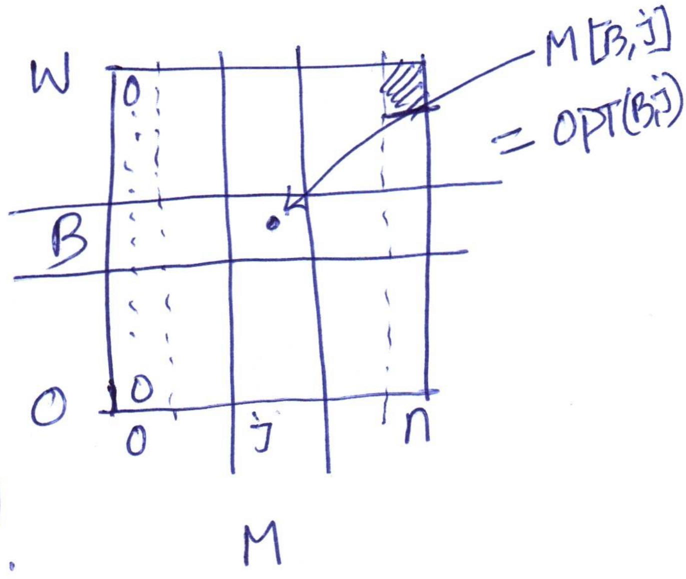
$\downarrow$ $OPT(B, j)$

w.g.s. $\rightarrow$ $M[0 \ldots n] \leftarrow OPT(n)$

$\hookrightarrow M[n]$

Q1: Which entry of M are we interested in?

$M[W, n] = OPT(W, n)$



Q2: Initial values:

$M[B, 0] \leftarrow 0 \quad \forall\, 0 \le B \le W$

Q3: How many subproblems?

$\underline{(W+1) \times (n+1)}$ subproblems

← polynomial many subproblems if $W = poly(n)$

Q4: Recurrence ⇒ (*)

Q5: An ordering among subproblems?

⇒ Column j can be computed using values in column $j-1$.

⇒ Compute the matrix M column by column.