Apr 24

Subset Sum Problem

Example: $n=3$; $w_1=1$, $w_2=3$, $w_3=3$; Budget W

Goal: output a subset of these 3 numbers s.t. their sum $\leq W$ and maximize such sum.

(i) $W=7 \Rightarrow$ opt soln $\Rightarrow \{1,2,3\}$
(ii) $W=6 \Rightarrow$ opt soln $=\{2,3\}$
(iii) $W=5 \Rightarrow$ opt soln $=\{1,3\}$ or $\{1,2\}$

$\Rightarrow$ In general, cannot have the sum of values in OPT $= W$.

Input: n integers; $w_1, \ldots, w_n$; $w_i > 0$;
Budget: $W \geq 0$

Output: A subset $S \subseteq [n]$ s.t.
(i) $w(S) = \sum_{i \in S} w_i \leq W$ AND (ii) max $w(S) = \sum_{i \in S} w_i$

max $|S|$ instead of $w(S)$ (for (ii))

Greedy soln: Sort the w_i's in increasing order; pick as many as possible without exceeding W.

(Pf: Ex) Prove this is optimal: Greedy stays ahead.

Q: Does the Greedy solution work for max $w(S)$.

A: No, Greedy soln does not work.
Counter example: $W = 6$; $w_1 = 1$, $w_2 = 3$, $w_3 = 3$.

Greedy solution: $\{1, 2\}$
$\to w(S) = 4$
OPT = 6

Note: No known greedy solution for Subset Sum.

Dynamic program for Subset Sum.

Goal: compute $w(S)$ for an optimal S.
$\Rightarrow \max w(S)$ s.t. $w(S) \leq W$.

Attempt 1:

let O_j be an optimal solution for $w_1, \ldots, w_j$.

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

Case: $j \notin O_j$

claim: O_j is also optimal for $w_1, \ldots, w_{j-1}$

$$\Rightarrow OPT(j) = OPT(j-1)$$

(Pf. Ex) $\to$

Case 2: $j \in O_j$

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

Hope: $O_j \setminus \{j\}$ optimal for $w_1, \ldots, w_{j'}$ for some $j' < j$.

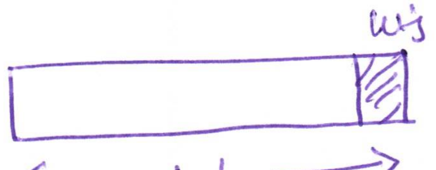
$\Rightarrow OPT(j) = w_j + OPT(j')$

Q: what's wrong in the above?
what parameter is missing ?

A: Budget is not taken into account in the above.

if $j \in O_j$, the remaining numbers are:

$w_1, \ldots, w_{j-1}$



new budget: $W - w_j$

Q: How should we define the subproblems?

A: Keep track j and budget B.

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

Assume: $w_j \le B$.

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

~~$\Rightarrow OPT(j) = OPT(j-1)$~~

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

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

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

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

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

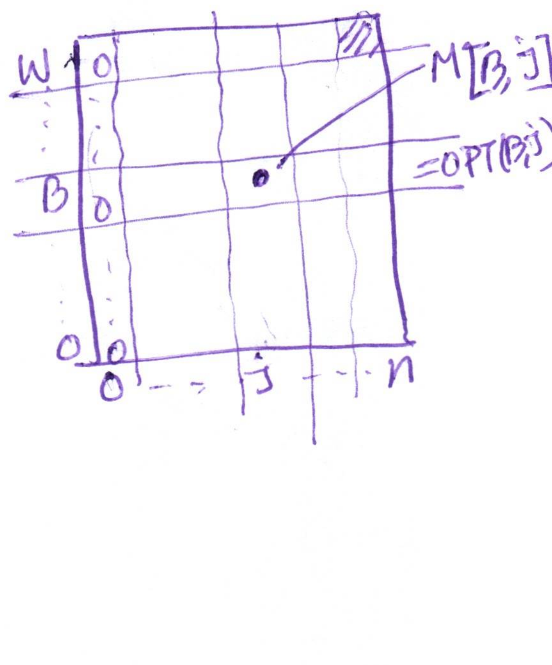
overall (*)

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

Q1: which entry in M are we interested in?

(i/p: $w_1, \ldots, w_n$; W)

A1 $M[W, n] = OPT(W, n)$



Q2: Initial value:

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

Q3: How many subproblems?

$(n+1)(W+1) = O(nW) \leftarrow$ poly if $W = \text{poly}(n)$.

Assume this for now

Q4: Recurrence? (*)

Q5: Ordering among subproblems?
knowing the values in column $j-1$ is enough to compute values in jth column.
→ Compute M column by column.

Subset Sum ($w_1, \ldots, w_n$; W)

(0) Allocate $(W+1) \times (n+1)$ matrix M. } $O(nW)$
(1) $M[B, 0] \leftarrow 0 \quad \forall\, 0 \le B \le W$ } $O(W)$
(2) for $j = 1 \ldots n$ overall: $O(nW)$

$O(nW)$ {
    for $B = 0 \ldots W$
        $O(1)$ { if $w_j > B$
            $M[B, j] = M[B, j-1]$
        else $M[B, j] = \max\{w_j + M[B - w_j, j-1], M[B, j-1]\}$
}

(3) return $M[W, n]$

Obs: $O(W)$ space if only interested in $OPT(W, n)$
$\Omega(nW)$ space if we want to compute the subset.
↑ similar algorithm that we saw for weighted interval scheduling problem.