Nov 21

Subset Sum Problem

Input: n integers $w_1, \ldots, w_n$ $\quad w_i > 0$

Bound $W \geq 0$

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

(i) $\sum_{i \in S} w_i \leq W$

(ii) maximizes $\sum_{i \in S} w_i \overset{def}{=} w(S)$

For example: $n = 3$ $\quad w_1 = 1, w_2 = 3, w_3 = 3$

$W = 7 \Rightarrow$ soln $\{1,2,3\}$

$W = 6 \Rightarrow$ soln $= \{2,3\}$

Simpler Q: max $|S|$ instead of $w(S)$

Q1: Greedy algo for max $|S|$

Q2: Can you think of a greedy soln for max $w(S)$

A1: Sort by increasing w_i's & pick as many as possible ~~bef~~ w/o exceeding W

Ex: Prove this is optimal ("greedy stays ahead")

Q2: Use above algo for general case [max $w(S)$]

↳ Counter example $w_1 = 1, w_2 = w_3 = 3$ $\quad W = 6$

→ Go in descending order: Counter-exampl $w_1 = \cancel{2}\,3, w_2 = 3$
$w_3 = \cancel{4}\,5$
$W = 6$

Note: No known greedy algo for this problem.

Goal: Design a Dynamic Programming algo

Let $\mathcal{O}_j$ be optimal subset for $1..j$

$OPT(j) = w(\mathcal{O}_j)$ Revised goal: Compute $OPT(n)$

Case 1: $n \notin \mathcal{O}_n \Rightarrow OPT(n) = OPT(n-1)$

Claim: $\mathcal{O}_n$ is an optimal subset of $1..n-1$

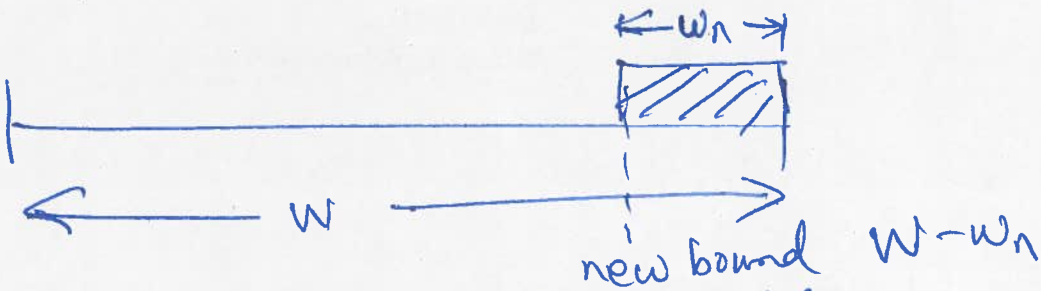
Pf idea: If not & say $\mathcal{O}'$ is better soln for $1..n-1$

$\Rightarrow \mathcal{O}'$ is a better soln (than $\mathcal{O}_n$) $1..n$

Case 2: $n \in \mathcal{O}_n$. What can we say about $\mathcal{O}_n \setminus \{n\}$

Hope: $OPT(n) = w_n + OPT(n')$ $n' < n$

eg $= w_n + OPT(n-1)$



Soln: Change defn & add a variable

$OPT(j, W')$ ← optimal subset for $1..j$ with bound W'.

$OPT(n, W) = w_n + OPT(n-1, W-w_n)$ ← if $w_n \le W$

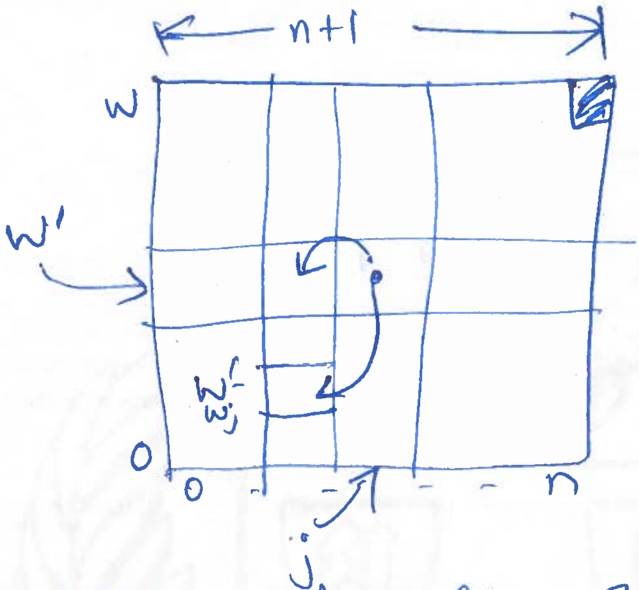
Case 1: $n \notin \mathcal{O}_n$: $OPT(n, W) = OPT(n-1, W)$

$\Rightarrow OPT(n, W) = \max\{ w_n + OPT(n-1, W-w_n), OPT(n-1, W)\}$

General recursion:

If $w_j > W'$ $\Rightarrow$ $OPT(j, W') = OPT(j-1, W')$

else $OPT(j, W') = \max \{ w_j + OPT(j-1, W'-w_j), OPT(j-1, W') \}$



$M[W'][j] = OPT(j, W')$

Q1) How many sub-problems? $(n+1)(W+1)$

Q2) What entry to want? $OPT(n, W)$

Q3) Ordering? Go column by column.