Nov 14

Dynamic Program for Subset Sum problem

Goal: Compute $W(S) = \sum_{j \in S} w_j$ (for an optimal S)

$\rightarrow$ max $W(S)$ s.t. $W(S) \leq W$

Attempt 1:

Q_j : be an optimal solution: $w_1, \ldots, w_j$

$OPT(j) = W(Q_j)$

Case 1: $j \notin Q_j \Rightarrow OPT(j) = OPT(j-1)$

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

↳ Ex

Case 2: $j \in Q_j$

Q: What can we say about $Q_j \setminus \{j\}$

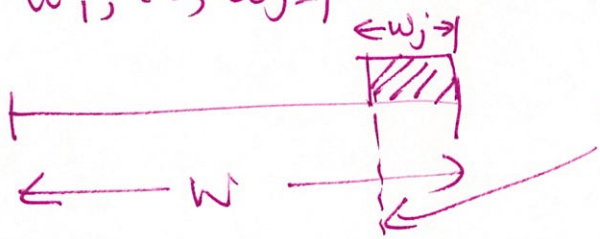
Hope! $Q_j \setminus \{j\}$ is optimal for $w_1, \ldots, w_{j'}$ for some $j' < j$

If so, $OPT(j) = w_j + OPT(j')$.

Q: What goes wrong in the above? Something is "missing" in sub-problems?

Q1: What number remain if we pick w_j?

↳ $w_1, \ldots, w_{j-1}$



Q2: How should we define sub-problems:

A: Keep track of j and budget B

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

Assume: $w_j \le B$

Case 1: $j \notin$ optimal $(w_1, \ldots, w_j ; B)$

total weight = $OPT(B,j)$

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

(Pf: Ex.)

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

Overall:

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

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

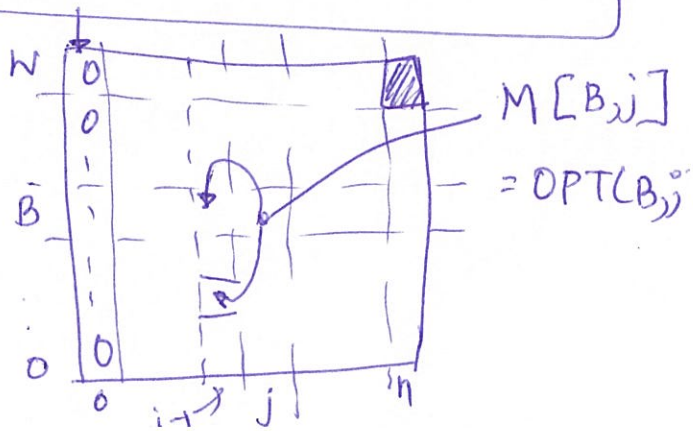
Q1) What entry of the matrix M are we interested in?

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

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

Q2) Initial values:

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



Q3) How many sub-problems do we have?

$(n+1)(W+1) = O(nW) \rightarrow$ poly only IF $W = \text{poly}(n)$

assume this for now

Q4) Recurrence (A4) (*)

Q5) Ordering among sub-problems?

A5) knowing values in $j-1$ column is enough to compute the j^{th} column.

→ Compute the M column by column

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

0. Allocate an $(W+1) \times (n+1)$ matrix } $O(nW)$
1. $M[B,0] \leftarrow 0$ + $0 \le B \le W$ } $O(W)$
2. $O(nW)$ { for $j = 1 \ldots n$
   for $B = 0 \ldots W$
   $O(1)$ { if $w_j > B$
   $M[B,j] \leftarrow M[B,j-1]$
   else
   $M[B,j] \leftarrow \max\{ w_j + M[B-w_j, j-1],$
   $M[B,j-1] \}$

Overall: $O(nW)$

3. return $M[W,n]$

Obs: $O(W)$ space enough if we are only interested in $OPT(W,n)$

$\Omega(nW)$ space if we also want to compute the actual subset

↑ Similar recursive algo as we saw in Weighted interval scheduling