Dynamic Programming (Contd.)

Interval Scheduling

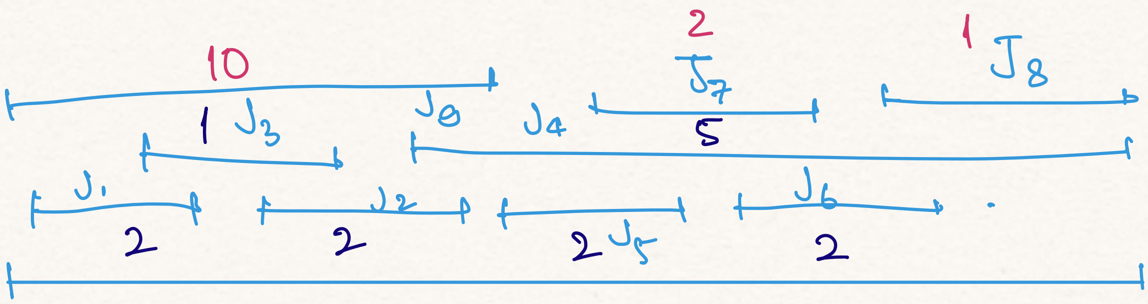
P: $J_1, J_2, \ldots, J_n \leftarrow$ Jobs
$s_1 \quad s_2 \quad\quad s_n \leftarrow$ Start times
$f_1 \quad f_2 \quad\quad f_n \leftarrow$ End times.
$w_1 \quad w_2 \quad\quad w_n \leftarrow$ Weights

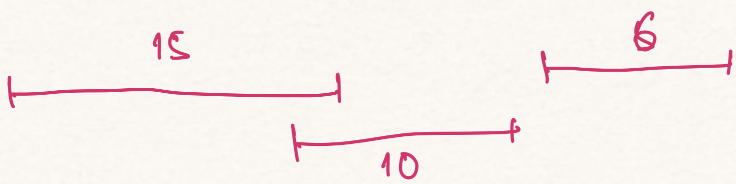
~~+ve~~ non-negative.

Want:
Maximum number of non-overlapping jobs
Earliest finish time.

Want: $\max\limits_{S \subseteq P} \left(\sum\limits_{i \in S} w_i \right)$

S contains non-overlapping jobs





1. Sort the jobs on the basis of their finish times (start).

$J_1, \ldots, J_n$
$s_1 \;\cdots\; s_n$
$f_1 \;\cdots\; f_n \leftarrow f_1 \leq f_2 \leq \cdots \leq f_n$.
$w_1 \;\cdots\; w_n$

Case-1: Job $J_n \in$ optimal subset O
Case-2: Job $J_n \notin$ optimal subset O

Case-1: $J_n \in O$:

- P_n : Set of all jobs that do not overlap with J_n.
- Find the wt of optimal subset in P_n.
- Report w_n + wt of optimal subset in P_n.

Case-2: $J_n \notin O$:

- Report the wt. of the optimal subset in $P \setminus \{J_n\}$

OptimalSubsetWt$(J_1, \ldots, J_n)$:

if $n == 1$: return w_1.
if $n == 0$: return 0.
Construct P_n by doing a linear scan

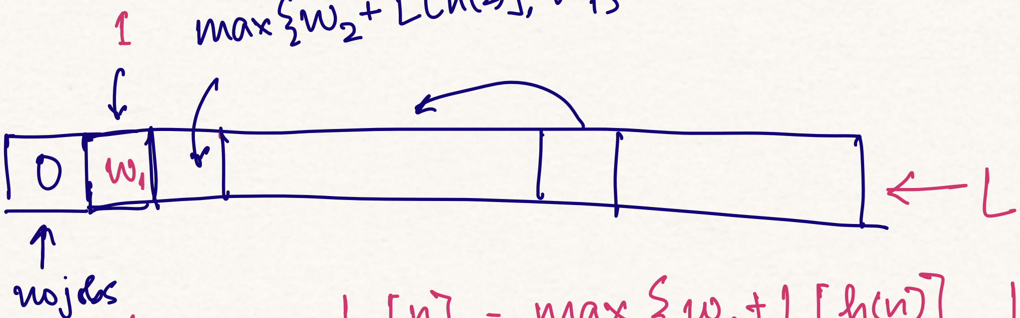
Let $h(n)$ be the index of the last non-overlapping job with J_n.
$P_n = \{J_1, \ldots, J_{h(n)}\}$.

$\hat{w}$ = w_n + OptimalSubsetWt(P_n)
$\tilde{w}$ = OptimalSubsetWt$((J_1, \ldots, J_{n-1}))$

Return $\max\{\hat{w}, \tilde{w}\}$.

Sub problems
OptimalSubsetWt$(J_1, \ldots, J_k)$
for some $k \in \{1, \ldots, n\}$.
set of

$\max\{w_2 + L[h(2)], w_1\}$.



$L[n] = \max\{w_n + L[h(n)], L[n-1]\}$.

$\max\{w_1 + 0, 0\}$

0-1 Knapsack.

Items $I_1, I_2, \dots, I_n$
$w_1, w_2, \dots, w_n$ ← integral wts.

Knapsack of max wt W.

Obj: $\max\limits_{S \subseteq [n]} \sum\limits_{i \in S} w_i$
subj to $\sum\limits_{i \in S} w_i \leq W$

Sort it
→ Pick wts as long as it does not exceed W.

1, 2, 3, 10
↑ decr. order.

10, 4, 4, 4 | 12.

Case 1: w_n (I_n) is chosen.
↳ $\max\limits_{S \subseteq [n-1]} \sum\limits_{i \in S} w_i$
subj to $\sum\limits_{i \in S} w_i \leq W - w_n$.

Case 2: w_n is not chosen
↳ $\max\limits_{S \subseteq [n-1]} \sum\limits_{i \in S} w_i$
subj to $\sum\limits_{i \in S} w_i \leq W$

max over these two cases.

OptimalSubsetWt(P, W)

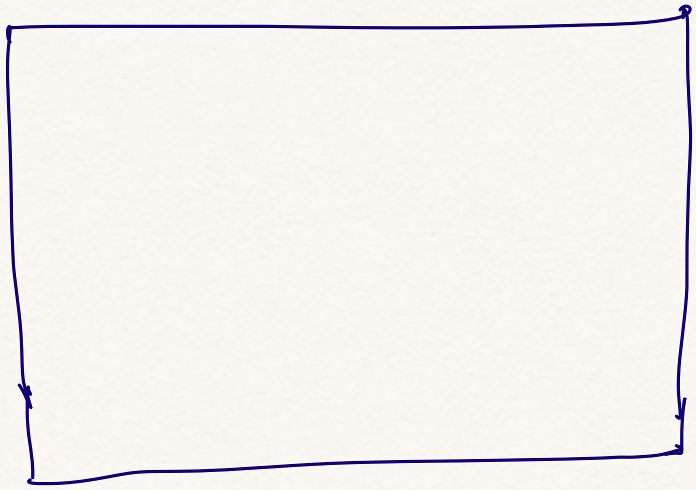
If no. of items == 1:
  if wt of item < W: return wt
  else 0.

$\hat{w}$ = w_n + OptimalSubsetWt(P \ {item n}, $W - w_n$)
$\tilde{w}$ = OptimalSubsetWt(P \ {item n}, W).

Return $\max\{\hat{w}, \tilde{w}\}$.

$\hookrightarrow$ Sub problem structure

Optimal Subset Wt (Subset of item, a weight bound)



$\leftarrow$ # of rows $\leq n+1$

# of cols $\leq \underline{\underline{W}}$

$w_1, \ldots, w_n$

$\longleftarrow$

poly(n, W)

$\left.\begin{array}{l}\text{unary}\\ \text{binary}\end{array}\right\}$.

$N \longleftarrow$ binary

$\log N = t$

$\sqrt{N} \sim 2^{\frac{\log N}{2}} \sim 2^{t/2}$