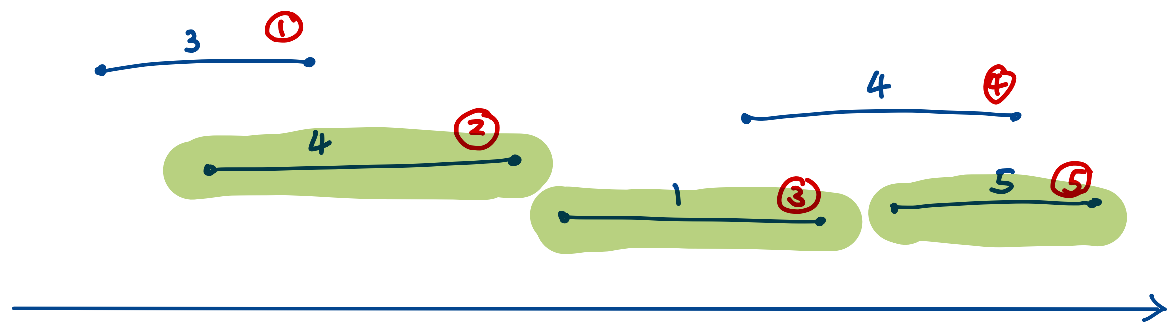
Dynamic Programming [chapter 6]

Weighted Interval Scheduling (WIS)



Input: n intervals labeled $[s_i, f_i]$ for interval i.
Also, weights $w_i > 0$ for interval i.

Output: A subset $S \subseteq \{1, 2, \dots n\}$ of mutually compatible / non-overlapping intervals with the maximum weights $\sum_{i \in S} w_i$

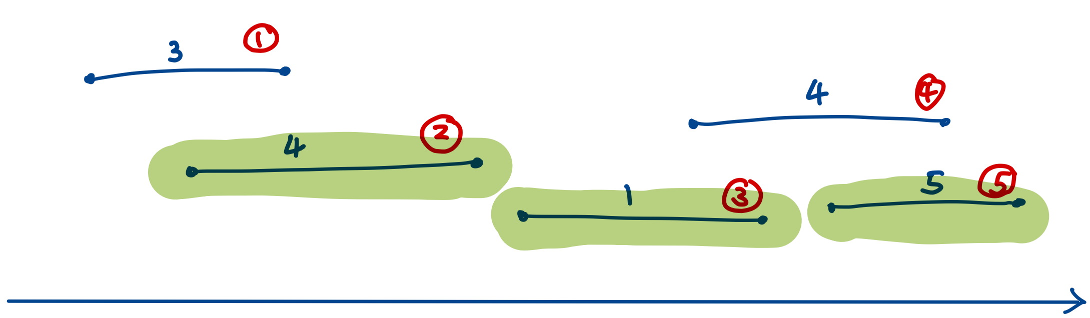
Brute Force: $O(2^n \cdot n)$

For every subset $S \subseteq \{1, 2, \dots n\}$ $O(2^n)$

1) Check compatible. $O(n)$

2) Check max weight. $O(1)$

There is no greedy algorithm known for WIS.



$P(5) = 3$
$P(4) = 2$
$P(3) = 2$
$P(2) = 0$
$P(1) = 0$

Assume that the intervals are labeled in order of their finish times. $f_1 \leq f_2 \leq \cdots \leq f_n$.

Define for each interval j,

$$P(j) = \max\{i < j : i \text{ and } j \text{ do not overlap}\}.$$

Ex: given the input, compute $P(j)$ for all j.

Take $O(n)$ if input is sorted.

or $O(n^2)$ naively with double for loop.

Dynamic Program:

1) Define subproblems appropriately.
2) Find the recurrence relation between solution to the subproblem.

For WIS:

1) $OPT(j)$ is the weight of the solution to WIS if only the intervals $\{1, \dots, j\}$, $[s_1, f_1] \dots [s_j, f_j]$, $w_1, \dots, w_j$, were given as input.

In other words, $OPT(j)$ is the best an algorithm can do if it's only allowed to choose from intervals $\{1, \dots, j\}$.

$OPT(1)$, $OPT(2)$, $\dots$ $OPT(n)$

$OPT(1) = w_1$

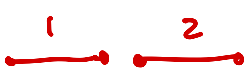
Our answer $= OPT(n)$.

2) We need a recurrence relation decribing $OPT(j)$ in terms of $OPT(j-1), OPT(j-2), \ldots, OPT(1)$.

$OPT(j)$
- it has interval j $= w_j + OPT(P(j))$
- it does not have interval j $= OPT(j-1)$

$$\boxed{\begin{array}{l} OPT(1) = w_1 \qquad OPT(0) = 0 \\ OPT(j) = \max\left(w_j + OPT(P(j)),\ OPT(j-1)\right) \end{array}}$$

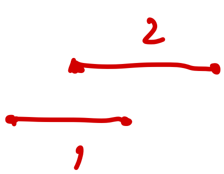
$OPT(2) = \max\left(w_2 + OPT(P(2)),\ OPT(1)\right)$



→ if $P(2) = 1$ ⇒ 1 and 2 don't overlap

$OPT(2) = w_2 + w_1$

→ if $P(2) = 0$ ⇒ 1 and 2 overlap



$OPT(2) = \max\left(w_2 + \underbrace{OPT(0)}_{=0},\ OPT(1)\right)$

$= \max\left(w_2, w_1\right)$

Algorithm: $\nearrow [1, 2, \ldots n]$

$n\log n \rightarrow$ Order (& relabel) the intervals by finishing time.

$\leq n^2 \rightarrow$ Compute $P(j)$ for all $j = 1$ to n.

$O(1) \rightarrow$ Initialized an array $A[0, \ldots, n]$, $A[0] = 0$, $A[1] = w_1$.

$O(n) \nearrow$ For $j = 2$ to n.

$$A[j] = \max\big(\underbrace{w_j + A[P(j)]}_{①}, \underbrace{A[j-1]}_{②}\big)$$

End for

Output $A[n]$

if ① > ②, add j to the solution set.

* $A[n]$ gives us the weight of the solution, not the intervals.

$O(n)$. $\searrow$

How do we get the intervals?

$O(n)$

$i = n$,
while ($n > 0$)
  if $A[i] > A[i-1]$
    Add i to solution set.
    $i = P(i)$
  else $i = i - 1$.