DYNAMIC PROGRAMMING

2- SUBPROBLEMS — WEIGHTED INTERVAL SCHEDULING

"n"-SUBPROBLEMS — SEGMENTATION

"ADDING A VARIABLE" ≡ "ENLARGING THE SPACE OF SOLUTIONS"

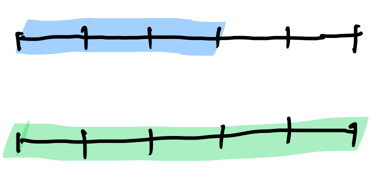
CONSIDER THE FOLLOWING PROBLEM:

- WE HAVE n JOBS, THE iTH OF WHICH TAKES w_i SECONDS ON OUR CPU. THE CPU IS AVAILABLE FOR W SECONDS. (w_i, $i=1,\dots,n$, IS A POSITIVE INTEGER; W IS ALSO A POSITIVE INTEGER).

  IF WE SCHEDULE JOB i WE GET PAID $w_i \cdot 0.01$ € (0.01€ PER SECOND), PROVIDED THAT JOB i FINISHES.

  WHICH SUBSET OF JOBS SHOULD I SCHEDULE TO MAXIMIZE MY GAIN?

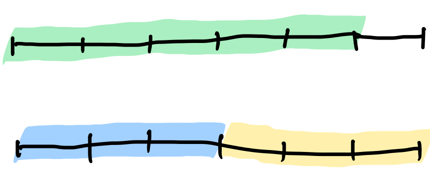
$w_1 = 3$ $\quad w_2 = 3$ $\quad w_3 = 5$ $\quad W = 5$



0.03€

0.05€ ✓

$w_1 = 3$ $\quad w_2 = 3$ $\quad w_3 = 5$ $\quad W = 6$



0.05€

0.06€

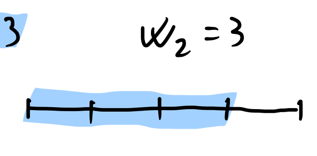
"HARD CONSTRAINT"

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

"OBJECTIVE"

$$\max \sum_{i \in S} w_i$$

$w_1 = 3$ $\quad w_2 = 3$ $\quad w_3 = 5$ $\quad W = 4$



0.03€

HOW CAN WE SOLVE IT?

IN DP WE WANT TO "SPLIT" A PROBLEM INTO SUBPROBLEMS... WHICH SUBPROBLEMS SHOULD WE USE?

IN WIS/SEGMENTATION WE CONSIDERED PREFIXES OF THE INPUT.

LET US TRY TO APPLY THE SAME APPROACH HERE.

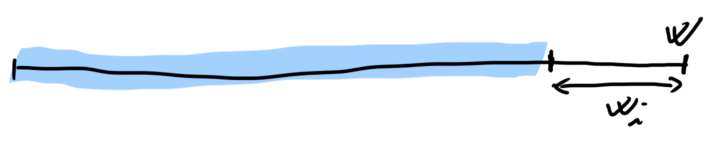
LET $OPT(i)$ BE THE OPTIMAL GAIN YOU CAN ACHIEVE USING THE FIRST i JOBS (SORTED HOWEVER YOU LIKE).

IF O IS AN OPTIMAL SOLUTION,

L: IF $i \notin O$, THEN $OPT(i) = OPT(i-1)$

BUT, WHAT HAPPENS IF $i \in O$? WE DO NOT KNOW HOW TO EXPRESS $OPT(i)$ IN TERMS OF $OPT(1), OPT(2), \dots, OPT(i-1)$.

IF WE SCHEDULE INTERVAL i, THEN WE ONLY HAVE $W - w_i$ TO SCHEDULE THE REMAINING JOBS (I HAVE TO KEEP TRACK OF THE REMAINING TIME AVAILABLE).



LET US THEN TRY WITH THE FOLLOWING CLASS OF SUB PROBLEMS:

- GIVEN V AND i, WHAT IS THE OPTIMAL VALUE I CAN ACHIEVE WITH V SECONDS AND JOBS $1, 2, \dots, i$?

THEN,

$$OPT(i, V) = \max_{\substack{S \subseteq \{1,\dots,i\} \\ (\sum_{j \in S} w_j \leq V)}} \sum_{j \in S} w_j .$$

WITH THIS CLASS OF SUB PROBLEMS, WE CAN INDEED SOLVE THE ORIGINAL PROBLEM:

- IF $i \notin O$, THEN $OPT(i, V) = OPT(i-1, V)$; AND
- IF $i \in O$, THEN $OPT(i, V) = w_i + OPT(i-1, V - w_i)$.

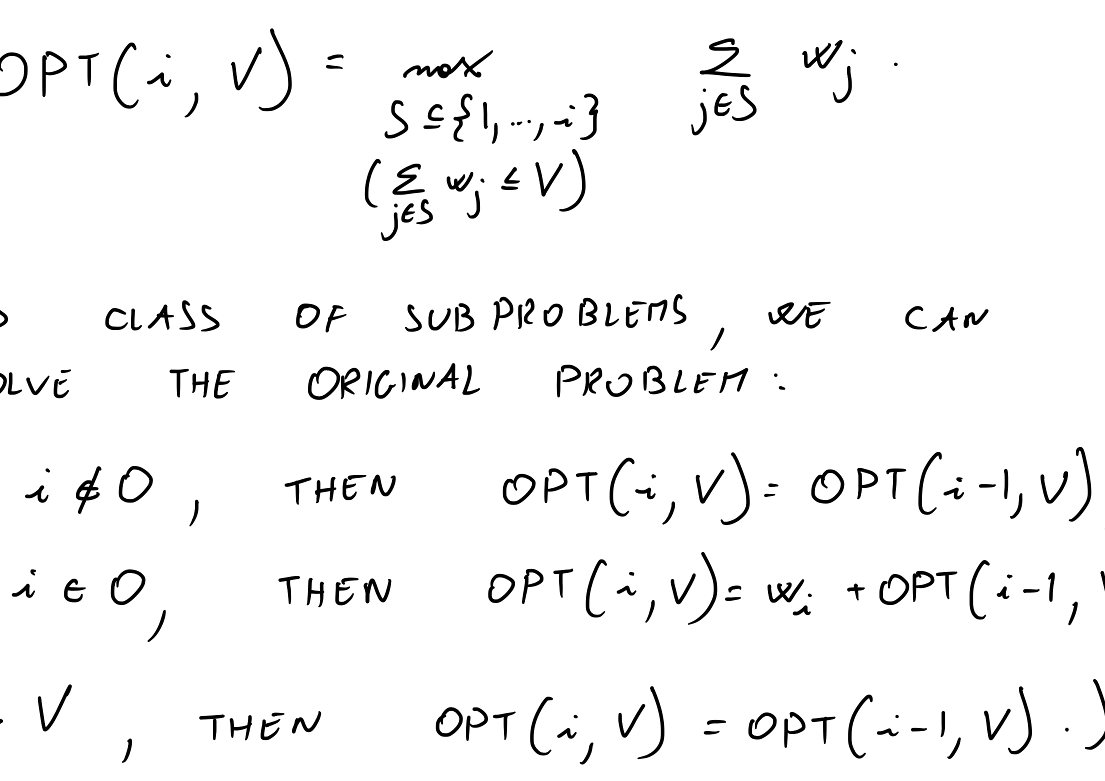
(IF $w_i > V$, THEN $OPT(i, V) = OPT(i-1, V)$.)

T: $OPT(0, V) = 0 \quad \forall V \geq 0$. (IF I HAVE NO JOB, I GET 0€).

IF $i \geq 1$, THEN

- IF $V < w_i$ THEN $OPT(i, V) = OPT(i-1, V)$;
- IF $V \geq w_i$ THEN

$$OPT(i, V) = \max\left(OPT(i-1, V),\ w_i + OPT(i-1, V - w_i)\right).$$



SUBSET-SUM$(n, W, [w_1, \dots, w_n])$

$O(nW)$

INITIALIZE THE TABLE $M[0 \dots n][0 \dots W]$

LET $M[0][V] = 0 \quad \forall V = 0, 1, \dots, W$

$O(nW)$ {

FOR $i = 1, \dots n$

  FOR $V = 0, \dots, W$

    USE THE RECURRENCE OF THE THM TO FILL $M[i][V]$

RETURN $M[n][W]$ // VALUE OF THE OPTIMAL SOL.

EX: USE M TO COMPUTE AN OPTIMAL SOLUTION.

INPUT: $\overbrace{w_1, w_2, \dots, w_n}^{n \text{ INTEGERS}}, W$

$W = 2^n = 1\overbrace{00\dots0}^{n \text{ TIMES}}{}_b$

THIS IS, THEN, A PSEUDO-POLYNOMIAL TIME ALGORITHM.

(IT CAN TAKE EXPONENTIAL TIME IF SOME OF THE WEIGHTS HAS EXPONENTIAL VALUE, BUT IT TAKES POLYNOMIAL TIME IF ALL THE WEIGHTS HAVE POLYNOMIAL VALUE)