```
DEF SEGMENTATION (C) :        // C IS A n×n ARRAY
    M = [None] * (n+1)
    M[0] = 0
    FOR j = 1..n :
        m = C_1j
        FOR i = 2, ..., j :
            IF C_ij + M[i-1] < m :
                m = C_ij + M[i-1]
        M[j] = m
    S = [ ]
    j = n
    WHILE j > 0:
        b = 1
        m = C_1j
        FOR i = 2, ..., j
            IF C_ij + M[i-1] < m :
                m = C_ij + M[i-1]
                b = i
        j = b - 1
        S.APPEND(b)
    RETURN S
```

DYNAMIC PROGRAMMING

2 - SUBPROBLEMS (WEIGHTED INT. SCH.)

"n" - SUPROBLEMS (SEGMENTATION)

"ADD A VARIABLE" $\Leftrightarrow$ "ENLARGE THE SOLUTION SPACE"

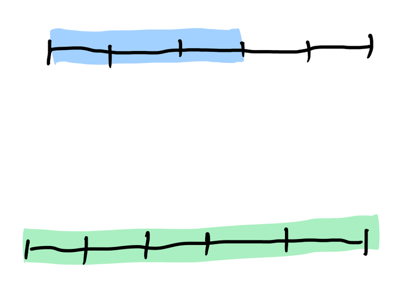
CONSIDER THIS PROBLEM

- THERE ARE n JOBS, THE iTH OF WHICH TAKES w_i SECONDS TO BE RUN. WE HAVE A TOTAL OF W SECONDS. (w_i, FOR $i = 1 \dots n$, 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 ENDS.

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

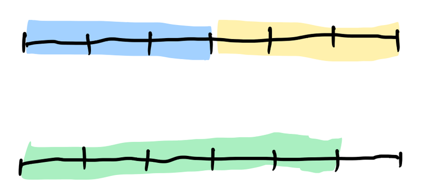
$w_1 = 3$ $\quad w_2 = 3$ $\quad w_3 = 5$ $\qquad W = 5$



GAIN = 3

GAIN = 5

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



GAIN = 6

GAIN = 5

HOW TO SOLVE THE PROBLEM?

IN DP WE AIM TO SOLVE PROBLEMS BY MEANS OF SOLUTIONS TO THEIR SUBPROBLEMS.

IN WIS / SEGM. WE USED PREFIXES.

SUPPOSE THAT OPT(i) IS THE OPTIMAL GAIN YOU CAN ACHIEVE USING THE FIRST i JOBS.

ACCORDING TO WHICH ORDERING?

IF O IS AN OPTIMAL SOLUTION:

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

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 ONLY $W - w_i$ SECONDS REMAIN FOR THE OTHER INTERVALS...

I HAVE TO KEEP TRACK OF TIME, AS WELL.

LET US CONSIDER THE FOLLOWING SUBPROBLEMS:

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

THEN,

$$OPT(i, V) = \max_{\substack{S \subseteq \{1,2,\dots,i\} \\ \sum_{j \in S} w_j \le V}} \sum_{j \in S} w_j .$$

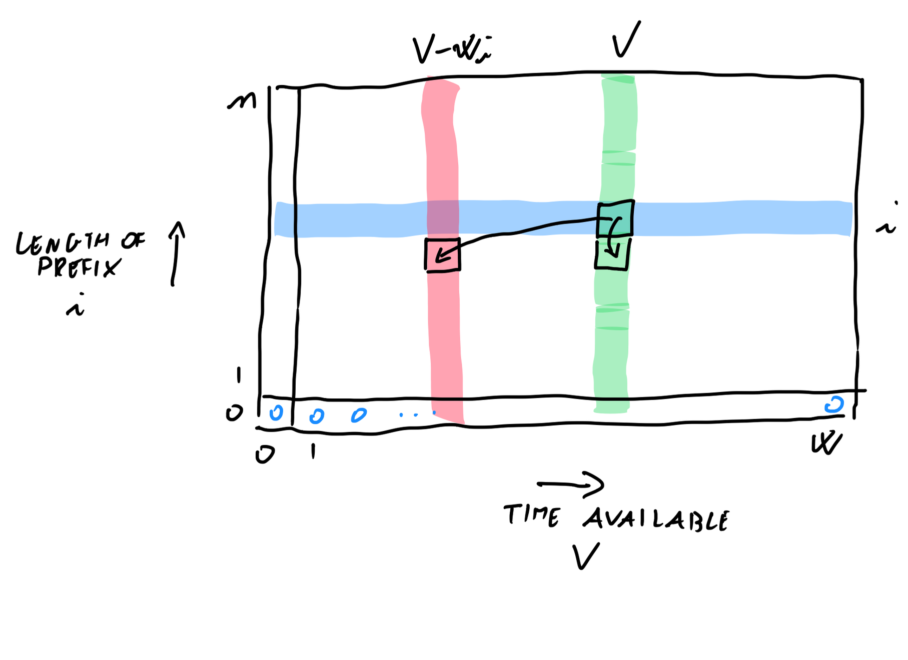
WITH THIS CLASS OF SUBPROBLEMS:

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

T: $OPT(0, V) = 0 \quad \forall V \ge 0$ (IF I HAVE NO JOBS, I CAN GET NO MONEY)

IF $i \ge 1$,

- IF $V < w_i$, THEN $OPT(i,V) = OPT(i-1, V)$
- IF $V \ge w_i$, THEN $OPT(i,V) = \max\left(OPT(i-1, V),\ w_i + OPT(i-1, V - w_i)\right)$



≤ 2 SUBPROBLEMS PER SUBPROBLEM

(BUT, OVERALL, $O(nW)$ SUB PROBLEMS)

```
SUBSET-SUM (n, W, [w_1, ..., w_n])
    INITIALIZE THE TABLE M[0...n][0...W]
    LET M[0][V] = 0   ∀ V ∈ {0, 1, ..., W}
    FOR i = 1, 2, ..., n                                  } O(nW)
        FOR V = 0, 1, ..., W
            USE THE RECURRENCE IN THM TO FILL M[i][V]
            IF V < w_i :                                  } O(1)
                M[i][V] = M[i-1][V]
            ELSE:
                M[i][V] = max(M[i-1][V], w_i + M[i-1][V-w_i])
    RETURN M[n][W]
```

EX: USE M TO FIND A SET OF INTERVALS OF MAXIMUM VALUE.

T: RUNTIME IS $O(nW)$.

TO WRITE W IN THE INPUT WE NEED ONLY $O(\lg W)$ BITS

SUPPOSE THAT $W = 2^n$; THEN WRITING W IN THE INPUT REQUIRES $n+1$ BITS.

THEREFORE "BEING POLYNOMIAL-TIME IN THE SIZE OF THE INPUT" MEANS RUNNING IN TIME n^c FOR A GIVEN $c > 0$.

IN FACT SUBSET SUM (OUR "SCHEDULING PROBLEM") IS VERY FAMOUS, AND IT IS NP-HARD. WE KNOW OF NO POLYTIME ALGORITHM FOR IT.