Algorithms DATE: 3/5/22

DYNAMIC PROGRAMMING

2 - SUBPROBLEMS

Weighted Interval scheduling

n - SUBPROBLEMS

"Least square fit"
SEGMENTATION

SEGMENTATION

We are given a sequence $p_1, \dots, p_n$ "POINTS" and we aim to segment it (select $I = i_1 < i_2 < \dots < i_k = n+1$, for some $k \geq 1$; cut the sequence $(p_i, \dots, p_{i_2-1})$, $(p_{i_2} \dots p_{i_3-1}), \dots, (p_{i_{n-1}}, \dots p_{i_k-1})$).

FOR $i \leq j$, the cost of the segment $(p_i, p_{i+1}, \dots, p_j)$ is c_{ij}. (In least-square-fit, $c_{ij} = a_{ij} + c$)

What is the min-cost segmentation (the cost of a segmentation is the sum of the costs of its segments)?

Let us define $OPT(j)$ to be the value of the minimum cost segmentation, for the input $p_1, \dots, p_j$

L: $OPT(j) = \min_{1 \leq i \leq j} (c_{ij} + OPT(i-1))$; the segment $(p_i, \dots, p_j)$ is in an optimal solution for the input $p_1 \dots p_j$ IFF

$$OPT(j) = c_{ij} + OPT(i-1)$$

P: If the sequence is $p_1 \dots p_j$ the element p_j should be the right endpoint of a segment, of the last segment.

Where does the last segment begin? It will begin at some point p_i with $1 \leq i \leq j$

The cost of the sequence $p_1 \dots p_j$ if the last segment begins at i is equal to

$$c_{ij} + OPT(i-1),$$

Since i have to pay for the segment $(p_i, \dots, p_j)$, and for the best segmentation of the sequence $p_1, \dots, p_{i-1}$. Thus, the minimum cost for the $p_1 \dots p_j$ is:

$$OPT(j) = \min_{1 \le i \le j} \left(c_{ij} + OPT(i-1) \right)$$

And, the segment $(p_k, \dots, p_j)$ is part of an opt. solution for the sequence $p_1 \dots p_j$ IFF

$$c_{kj} + OPT(k-1) = \min_{1 \le i \le j} \left(c_{ij} + OPT(i-1) \right) \blacksquare$$

DYNAMIC PROGRAMMING

2 - SUBPROBLEMS

Weighted Interval scheduling

n - SUBPROBLEMS

"Least square fit" — SEGMENTATION

} look at prefixes

"ADDING A VARIABLE"

Consider the following "scheduling" problem:

- there are "n" jobs, the ith of which takes time w_i seconds (of our CPU)

Our CPU is available for W seconds.

If we schedule the generic job we are paid 1€ per second it runs.

Which subset S of jobs should we schedule to maximize our gain?

The set S of jobs is "feasible" (is a valid sol) IFF $\sum_{i \in S} w_i \le W$

Thus, we are asking: what is the set of jobs S satisfying $\sum_{i \in S} w_i \le W$, that maximizes $\sum_{i \in S} w_i$?

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

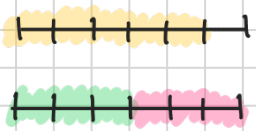
INVALID

VALID (value = 3)

OPTIMAL AND VALID (VALUE = 5)

HOW TO SOLVE IT?

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



In dynamic programming, we want to split a problem into some number of subproblems... which subproblems to use here?

For WIS/segm. we considered prefixes of the instance as subproblems.

Let us try to do the same.

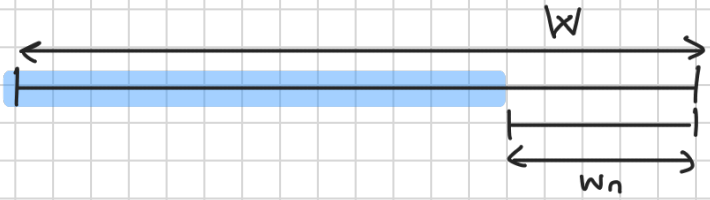
Let $OPT(i)$ be the optimal gain you can achieve by using only jobs $1, 2, \ldots, i$. Let O be an optimal solution.

Then,

L: IF $n \notin O$, then $OPT(n) = OPT(n-1)$

Ok but what if $n \in O$?

IF $n \in O$, then the remaining jobs will only have $W - w_n$ seconds available.



The amount of time available strictly depends on the length of the job we schedule.

In particular, we cannot obtain $OPT(n)$ in terms of just $OPT(1)$, $OPT(2)$, .., $OPT(n-1)$.

We need to solve the following subproblems?

- Given V and i, what is the optimal value if I have time V and the jobs $1, 2, \ldots, i$.

("Adding a variable"
↳ we come up with the idea of using V)

Assume that W is an integer and, also, that $w_1, \dots, w_n$ are also integers.

Then,

$$OPT(i, W) = \max_{S \subseteq \{1,\dots,i\}} \left(\sum_{j \in S} w_j \right) \quad \left(\sum_{j \in S} w_j \leq W \right)$$

Using these "enlarged" set of subproblems, we can solve the original problem:

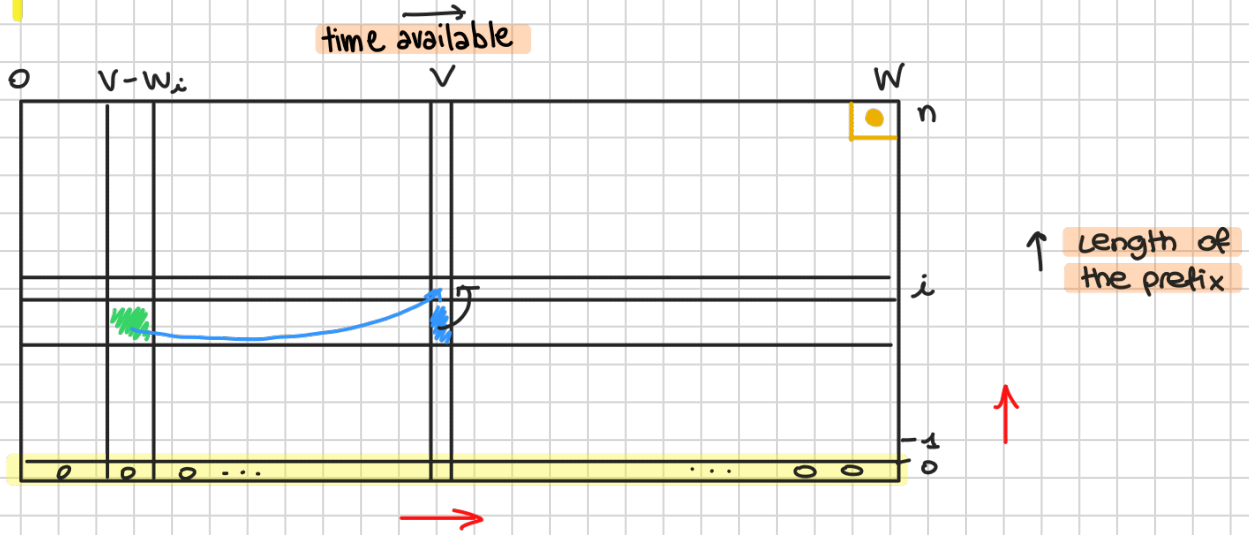
- If $n \notin O$, then $OPT(n, W) = OPT(n-1, W)$
- If $n \in O$, then $OPT(n, W) = w_n + OPT(n-1, W-w_n)$,

(IF $w_n > W$, then $OPT(n, W) = OPT(n-1, W)$)

T1: If $W < w_i$ then $OPT(i, W) = OPT(i-1, W)$

Otherwise;

$$OPT(i, W) = \max(OPT(i-1, W), w_i + OPT(i-1, W-w_i)).$$



SUBSET-SUM (n, W):

INITIALIZE THE 2D ARRAY $M[0 \dots n][0 \dots W]$ $O(n \cdot W)$

LET $M[0][V] = 0$ $\forall$ $V = 0, 1, \dots, W$ $O(W)$

$O(nW)$ {

FOR $i = 1 \dots n$

FOR $w = 0 \dots W$

Use the recurrence of T1 to set the value of $M[i][w]$

}

RETURN $M[n, W]$

$$W = \overbrace{100\ldots0}^{\text{BINARY } n} = 2^{n-1}$$