WIS(I, w): // AN ITERATIVE ALGORITHM FOR "FILLING UP" THE (MATRIX/TABLE) M

// $I = [(s_1, f_1), (s_2, f_2), \dots, (s_n, f_n)]$ WITH $f_1 \le f_2 \le \dots \le f_n$

// $w[i]$ IS THE WEIGHT OF INTERVAL i

EX: PROVE THAT P CAN BE FILLED UP IN $O(n)$ TIME

LET $P[i]$ BE THE LARGEST $j < i$ S.T. $f_j < s_i$, OR $P[i] = 0$ IF NO SUCH j EXISTS // $P[i] = p(i)$

```
M = [None] * (n+1)                           O(n)
FOR i = 0, 1, ..., n                         O(n)
    IF i == 0:
        M[i] = 0                             O(1)
    ELSE:
        M[i] = max(w[i] + M[P[i]], M[i-1])
RETURN M
```

DEF WIS_SOL(I, w):

LET $P[i]$ BE THE LARGEST $j < i$ S.T. $f_j < s_i$, OR $P[i] = 0$ IF NO SUCH j EXISTS // $P[i] = p(i)$

```
M = WIS(I, w)          // O(n)
O = []
i = n
WHILE i > 0:           // ≤n ITERATIONS (WE DECREASE i IN EACH ITER.)
    IF M[i] == w[i] + M[P[i]]:
        O.APPEND(i)                          O(1)
        i = P[i]
    ELSE:   // M[i] == M[i-1]
        i = i - 1
RETURN O
```

THM: WIS-SOL RETURNS AN OPTIMAL SOLUTION IN TIME $O(n)$ (PROVIDED THAT I IS SORTED).

THE HEART OF A DYNAMIC PROGRAMMING ALGORITHM IS ITS TABLE M.

AS WE SAID DYNAMIC PROGRAMS SOLVE PROBLEMS BY MEANS OF THEIR SUBPROBLEMS.

FOR A DP APPROACH TO GO THROUGH IT IS USEFUL:

1. THERE IS ONLY A POLYNOMIAL NUMBER OF SUBPROBLEMS TO CONSIDER;
2. THE (VALUE OF THE) SOLUTION TO A PROBLEM CAN BE COMPUTED EFFICIENTLY GIVEN (THE VALUES OF THE) SOLUTIONS TO ITS SUBPROBLEMS;
3. THE SUBPROBLEMS SHOULD HAVE SOME "ORDERING", SO THAT YOU CAN SOLVE THE RECURRENCE.

IN WIS WE HAD ONLY 2 SUBPROBLEMS $(i-1, p(i))$ TO CONSIDER PER SUBPROBLEM (i)

THE|YOUTH|EVENT

THE|YOU|THE|VENT

THEY|OUT|HE|VENT

THE|YOUNG

THEY|OUNG

THEX   THE X

$C_{ij} = 0$ IF $S[i:j+1]$ IS A WORD IN ENGLISH

$C_{ij} = j - i + 1$ O/W

SEGMENTATION

$i_1=1$ $i_2=4$ $i_3=9$

T,H,E|Y,O,U,N,G

$C[1..8][1..8]$

WE ARE GIVEN A SEQUENCE $p_1, \dots, p_n$ OF "TOKENS" (CHARACTERS, POINTS, ETC.) AND WE WANT TO SEGMENT IT — SELECT $1 = i_1 < i_2 < \dots < i_k = n+1$, FOR SOME $k \ge 1$ — THAT IS, TO CUT THE SEQUENCE INTO SEGMENTS $(p_{i_1}, \dots, p_{i_2 - 1}), (p_{i_2}, \dots, p_{i_3 - 1}), \dots, (p_{i_{k-1}}, \dots, p_{i_k - 1})$.

FOR $i \le j$, THE COST OF THE SEGMENT $(p_i, p_{i+1}, \dots, p_j)$ IS c_{ij}.

WHAT IS THE SEGMENTATION OF MINIMUM COST (WHOSE SUM OF COSTS OF SEGMENTS IS MINIMUM)?

LET US DEFINE OPT(j) TO BE THE COST OF THE MINIMUM COST SEGMENTATION ON THE INPUT $p_1, p_2, \dots, p_j$

L: $$OPT(j) = \min_{1 \le i \le j} \left(c_{ij} + OPT(i-1)\right), \quad OPT(0) = 0$$

1 2 3 … j-1 j

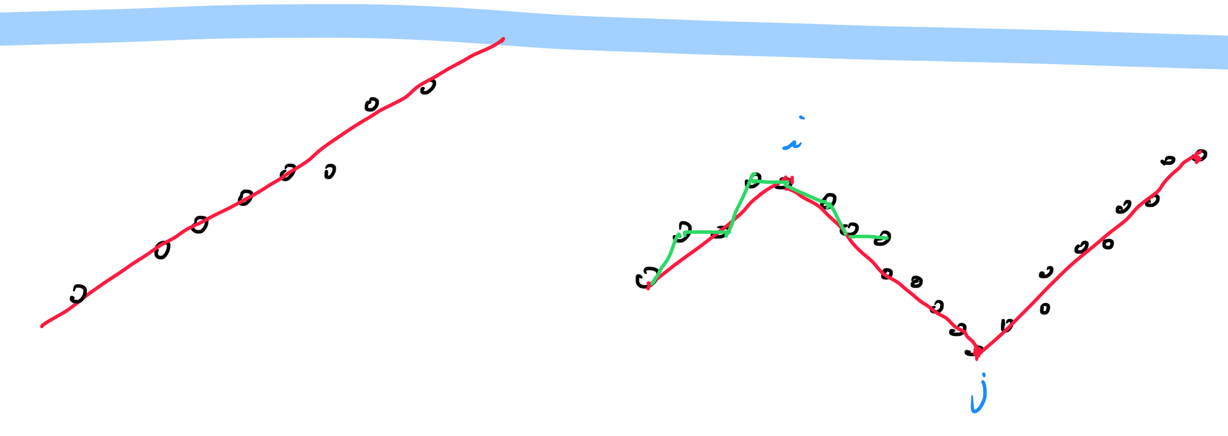
$$= \min\left(c_{1j} + OPT(0), c_{2j} + OPT(1), \dots, c_{jj} + OPT(j-1)\right).$$

DEF SEGMENTATION(c): // c IS AN $n \times n$ ARRAY

```
M = [None] * (n+1)
M[0] = 0

FOR j = 1, ..., n
    m = c_{1j}                 // c[1][j]
    FOR i = 2, ..., j:
        IF c_{ij} + M[i-1] < m:
            m = c_{ij} + M[i-1]

    M[j] = m
RETURN M          // M[n] CONTAINS THE MINIMUM COST OF THE INSTANCE.
```



$C_{ij} = 1$ + "THE COST OF APPRX POINTS $i, i+1, \dots, j$ WITH THE 'BEST' LINE"