[[0, 30]
[5, 10]
[15, 20]]
start ↗ ↖ end

[[15, 20]
→[5, 10]
[0, 30]]

$\underbrace{[0,30]}_{1}$ $\underbrace{[15,20] \; [5,10]}_{2}$ ⇒ 2 meeting rooms.

$\underbrace{[15,20]}_{1}$ $\underbrace{\cancel{[5,10]}}_{2}$ $\underbrace{[0,30]}_{3}$

[1, 15]
[15, 20]

→ [1, 18]
→ [2, 11]
→ [4, 15]
→ [5, 13]
→ [15, 29]
→ [18, 23]

[15, 29] [18, 23]
$\underbrace{(1,18)}_{1}$ $\underbrace{(2,11)}_{2}$ $\underbrace{(4,15)}_{3}$ $\underbrace{(5,13)}_{4}$

TC: $O(n^2)$.

Total no. of meeting rooms = Max no. of concurrent meetings taking place.

s	s	s	s	e	e	e	s	e	s	e	e
1	2	4	5	11	13	15	15	18	18	23	29
1	2	3	$\boxed{4}$	3	2	1	2	1	2	1	0

<int, char> ← 's'/'e'

TC: $O(n \log n)$

→ (4, 20)
(1, 10)
→ (1, 40)
(1, 30)
D ↗ ↖ P

$\underset{(0-1)}{40}$ $\underset{(1-2)}{20}$ ⇒ 60.

(1, 20)
→ (2, 40)
(2, 10)

$\underset{(0-1)}{20}$ $\underset{(1-2)}{40}$ ⇒ 60.

TC: $O(n \log n + n * \min(n, \max Deadline))$

deadline = 4

$\underset{(0-1)}{0}$ $\underset{(1-2)}{1}$ $\underset{(2-3)}{2}$ $\underset{(3-4)}{3}$ $\underset{(4-5)}{4}$ 5 6 7 8 9 10

$\underset{(0-1)}{0}$ $\underset{(1-2)}{1}$

1 → (1, 100)
2 → (2, 100)

$\underset{0-1}{1}$ $\underset{1-2}{2}$

→ (1, 2, 100)
→ (2, 1, 19)
→ (3, 2, 27)
→ (4, 1, 25)
→ (5, 1, 15)

(3, 2, 27)
~~(4, 1, 25)~~
(1, 2, 100)

→ jobsDone = ~~1~~ 2
profit = ~~25~~ 125

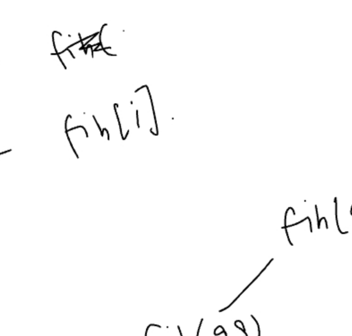
$O(n \log n + n \log n)$. ⇒ $O(n \log n)$.

---

fib(10)
fib(9) fib(8)
fib(8) fib(7) fib(7) fib(6)
fib(7) fib(6)

\# 2 necessary conditions for a problem to be solved using DP.

1) Overlapping subproblems
2) Optimal soln. to the subproblems should contribute to the optimal soln. of the overall problem.



Longest Path (A→B) = 3 (A→C→B)
LP (B→C) = 3 (B→A→C).

$\underline{LP(A \to C)} \neq \underline{LP(A \to B) + LP(B \to C)}$

memo[n+1] = ~~fib~~
↳ memo[i] = fib[i].

fib(100)
fib(99) fib(98)
fib(98) fib(97)

n

0	1	2	3	4	5	6	n
0	1	1	2	3	5	8 ...	

LCS

ABCDGM → $2^6 = 64$. 2^n

skip i inc
i+1 i+1

2^n 2^n

$(2 * 2^n + n * 2^n) = O(n \times 2^n)$

S1 = ABCDGM
S2 = AEDFHR

⇒ ADM (3).

A

ABCMGD
AEDFHR ⇒ 2

LCS(ABCDGM, AEDFHR) = 1 + LCS(BCDGM, EDFHR)
max(
LCS(CDGM, EDFHR),
LCS(BCDGM, DFHR))

LCS(i, j) = 1 + LCS(i+1, j+1) ⇒ if s1[i] = s2[j]
= max(LCS(i, j+1), LCS(i+1, j)) ⇒ otherwise

n m TC = $2^{\max(n,m)}$

(i, j)
(i+1, j) (i, j+1)
(i+2, j) (i+1, j+1) (i+1, j+1) (i, j+2)

Step-1: Recognize the state of rec / the changing params in rec.

Step-2: Create a memo table using the state variables.

Step-3: Memoize.

TC = No. of unique rec state × amt. of work done in each rec state

		0	1	2	3	4	5	6
		""	A	E	D	F	M	R
0	""	0	0	0	0	0	0	0
1	A	0	1	1	1	1	1	1
2	B	0	1	1	1	1	1	1
3	C	0	1	1	1	1	1	1
4	D	0	1	1	2	2	2	2
5	G	0	1	1	2	2	2	2
6	M	0	1	1	2	2	3	$\boxed{3}$

---

val: 50 70 100
wt: 10 10 50

W = 50

100

val: 10 10 100
wt: 20 30 50

W = 50