Steps for solving problems w/ Dynamic Programming

0. try something else! greedy, standard graph alg, ...?
1. Come up w/ a recurrence. ✱ ← hard part.
   (1a) if subproblems don't overlap, rejoice! → divide & conquer.
2. Memoize the recurrence — i.e. store outputs in a data structure to avoid recomputation.
3. Remember your choices to reconstruct optimal solution.

Example: hire low-stress jobs. Maximize total $$ if taking high-stress job requires taking the previous week off.

	1	2	3	4	5	6	7	8 ← week
L	2	2	1	20	5	40	50	63
H	1	5	10	100	23	70	100	42

Greedy?

Doesn't work?

	30	99
	50	100

Recurrence? Let $M(i)$ = max amount we can make if we only work weeks $1-i$.

base cases:

$$M(0) = 0$$
$$M(1) = \max(L[i], H[i])$$

general recursive case:

$$M(n) = \max \begin{cases} L[n] + M(n-1) \\ H[n] + M(n-2) \end{cases}$$

Overlapping subproblems, so memoize!

	1	2	3	4	5	6	7	8
L	2	2	1	20	5	40	50	63
H	1	5	10	100	23	70	100	42

M	0	2	5	12	105	110			
J		L	H	H	H	L	H	L	L

— $\Theta(n)$ time to fill in tables!

To reconstruct actual best schedule:

- Create another array, J, where

  $J[i]$ = which job should you take in week i to maximize earnings in weeks $1-i$.

  When taking max of $L[i] + M(i-1)$ and $H[i] + M(i-2)$, fill in $J[i]$ depending on which was bigger.

- Work backwards through J, following the same recurrence to reconstruct actual best schedule.