Dynamic Programming

Steps for solving problems w/ DP:

0. Try greedy, well-known graph algorithm, etc.
1. Come up w/ recurrence/recursive solution (reason by induction)
   - 1a. if the subproblems don't overlap, rejoice! → divide & conquer.
2. Make a table of intermediate results (memoization)
   - ↳ saving outputs of function to avoid recomputation.
3. Remember your choices to reconstruct optimal solution.

Example

- 2 jobs per week: hi-stress, lo-stress.
- to do hi-stress, must take previous week off.
- every job has different payment.
- Goal: maximize earnings.

eg.

week	1	2	3	4	5	6	7	8	9	10
L	2	2	1	(7)	5	26				
H	1	5	10	100	23	3				

Greedy! Doesn't work, as too complex.

Let $M[i]$ = max we can earn if we only work weeks $1-i$.

Base case(s):

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

Recursive case:

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

forgets which choice is better

Same rec. pattern as Fibonacci #'s — need to memoize!

eg.

week	1	2	3	4	5	6	7	8	9	10
L	2	2	1	(7)	5	26	20	3	19	
H	1	5	10	100	23	3	20	5	20	

M	0	2	5	12	105	110	136	156			
J		L	H	H	H	L	L	L			

$\Theta(n) - \Theta(1)$ to fill in each entry in the table.

Let J[k] = which job we should take @ week k
to optimize earnings for weeks 1..k.

Then start @ end and work backwards, using
recursive structure of M, to reconstruct optimal schedule.