Solving problems w/ dynamic programming

0. Try something else (greedy, reduce to standard graph algorithm, ...)
1. Come up w/ a recurrence. ★
   - opt: prove correct via induction.
2. Check: do subproblems overlap?
   - No: regime → divide & conquer.
   - Yes: proceed w/ rest of steps
3. Memoize: store outputs in a data structure to avoid recalculating.
4. To reconstruct optimal solution, remember choices made + reconstruct in reverse.

Example.

week	1	2	3	4	...	n	
low stress job	3	10	2	19		46	L_i
high stress job	6	2	50	22	...	31	H_i

- take one job per week
- to do high-stress job, must take previous week off.
- maximize total $$

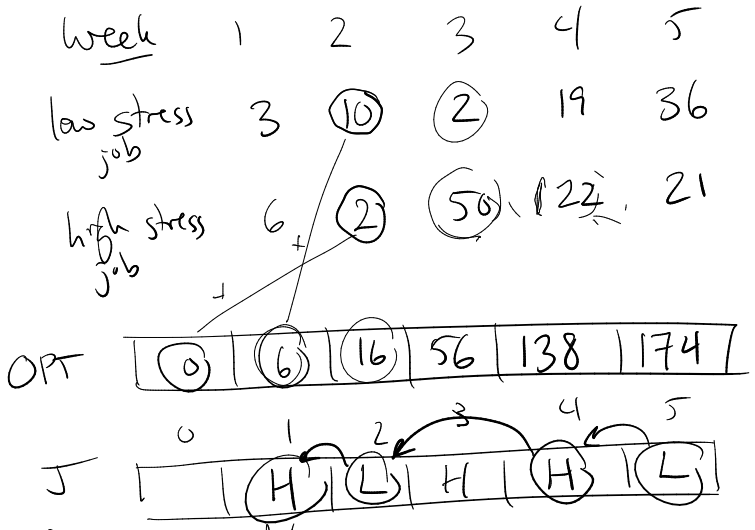
Greedy algorithm? Doesn't seem to work.

Recurrence? Let OPT(k) = max $$ we can make working only weeks 1 .. k.

- OPT(0) = 0
- OPT(1) = max(L_1, H_1)
- $OPT(k) = \max \begin{cases} L_k + OPT(k-1) \\ H_k + OPT(k-2) \end{cases}$

we might as well work week k, so 2 choices

Subproblems overlap — use memoization!



$\Theta(n)$ to fill in table.

To remember optimal choices, store another table J where $J[k]$ = which choice in week k leads to optimal value for working only weeks $1 \ldots k$.

Then start @ end and work backwards according to the recurrence to reconstruct optimal schedule.