Updating the Problem

$OPT(i, j, f)$ is the maximum amount of eggs Baby Yoda can collect on a legal path from (i, j) to $(0,0)$ using the force f times to knock over rocks.

For simplicity, assume there are no rocks at the starting location (r-1,c-1)

Here was the old rule without the force – how do we update?

$$OPT(i, j, f) = \begin{cases} -\infty & \text{if } i < 0 \text{ or } j < 0 \text{ or } f < 0 \\ eggs(0,0) & \text{if } i = 0 \text{ and } j = 0 \text{ and } f \geq 0 \\ \max\{OPT(i-1, j, f - rocks(i-1, j)), OPT(i, j-1, f - rocks(i, j-1))\} + eggs(i, j) & \text{otherwise} \end{cases}$$

`rocks(i,j)` doesn't guarantee $-\infty$ anymore. Only if you were out of force uses before trying to jump onto that location.

Dynamic Programming Process

1. Define the object you're looking for

2. Write a recurrence to say how to find it

3. Design a memoization structure

4. Write an iterative algorithm

Trying to Recurse

0	1	2	3	4	5	6	7
5	-6	3	4	-5	2	2	4

$OPT(3)$ would give i =2, $j = 3$

$OPT(4)$ would give $i = 2, j = 3$ too

$OPT(7)$ would give $i = 2, j = 7$ – we need to suddenly backfill with a bunch of elements that weren't optimal…

How do we make a decision on index 7? What information do we need?

Two Values

Pollev.com/robbie

Need two recursive values:

$INCLUDE(i)$: sum of the maximum sum subarray among elements from 0 to i **that includes index i** in the sum

$OPT(i)$: sum of the maximum sum subarray among elements 0 to i (that might or might not include i)

How can you calculate these values? Try to write recurrence(s), then think about memoization and running time.