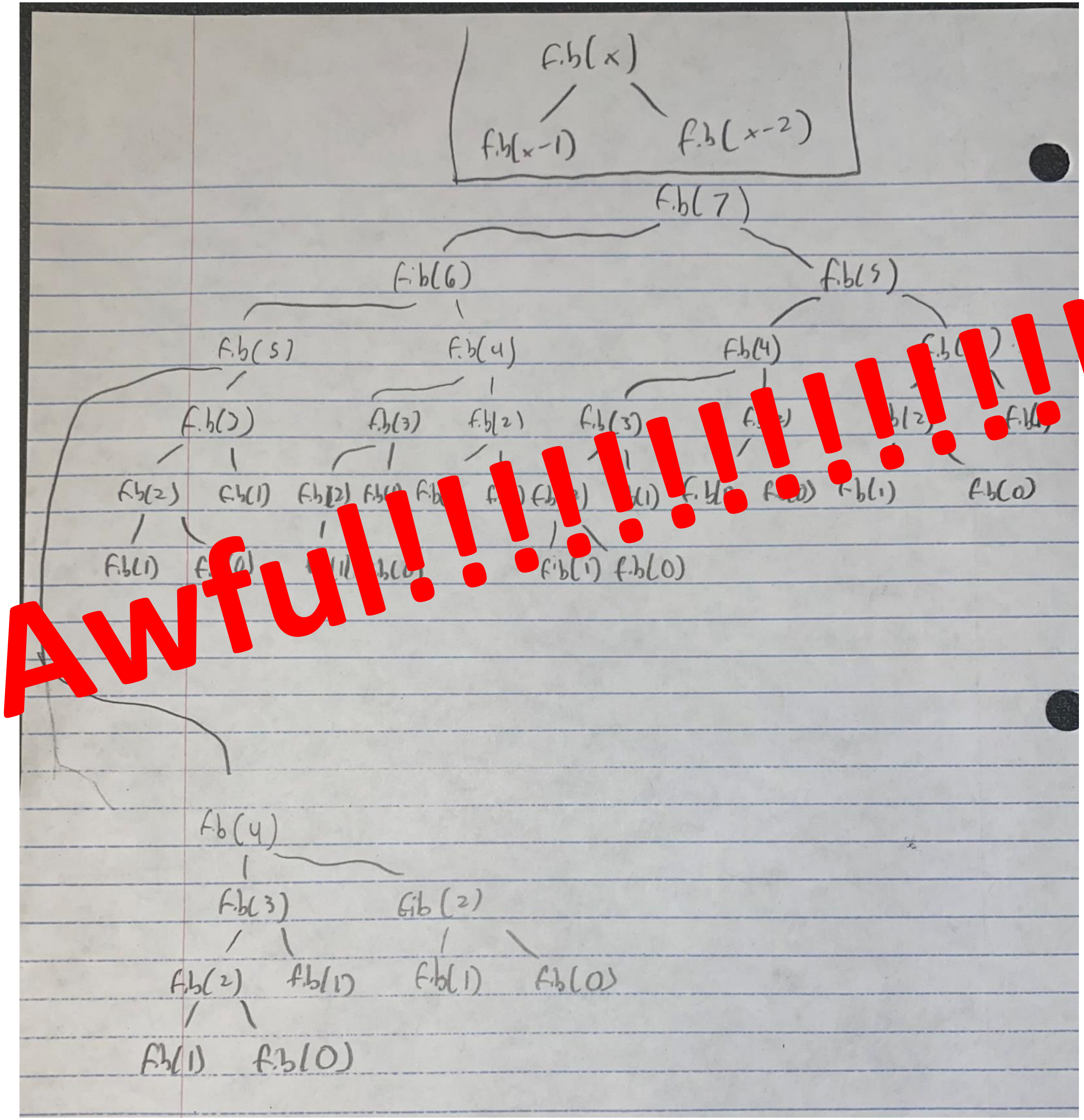
Dynamic Programming

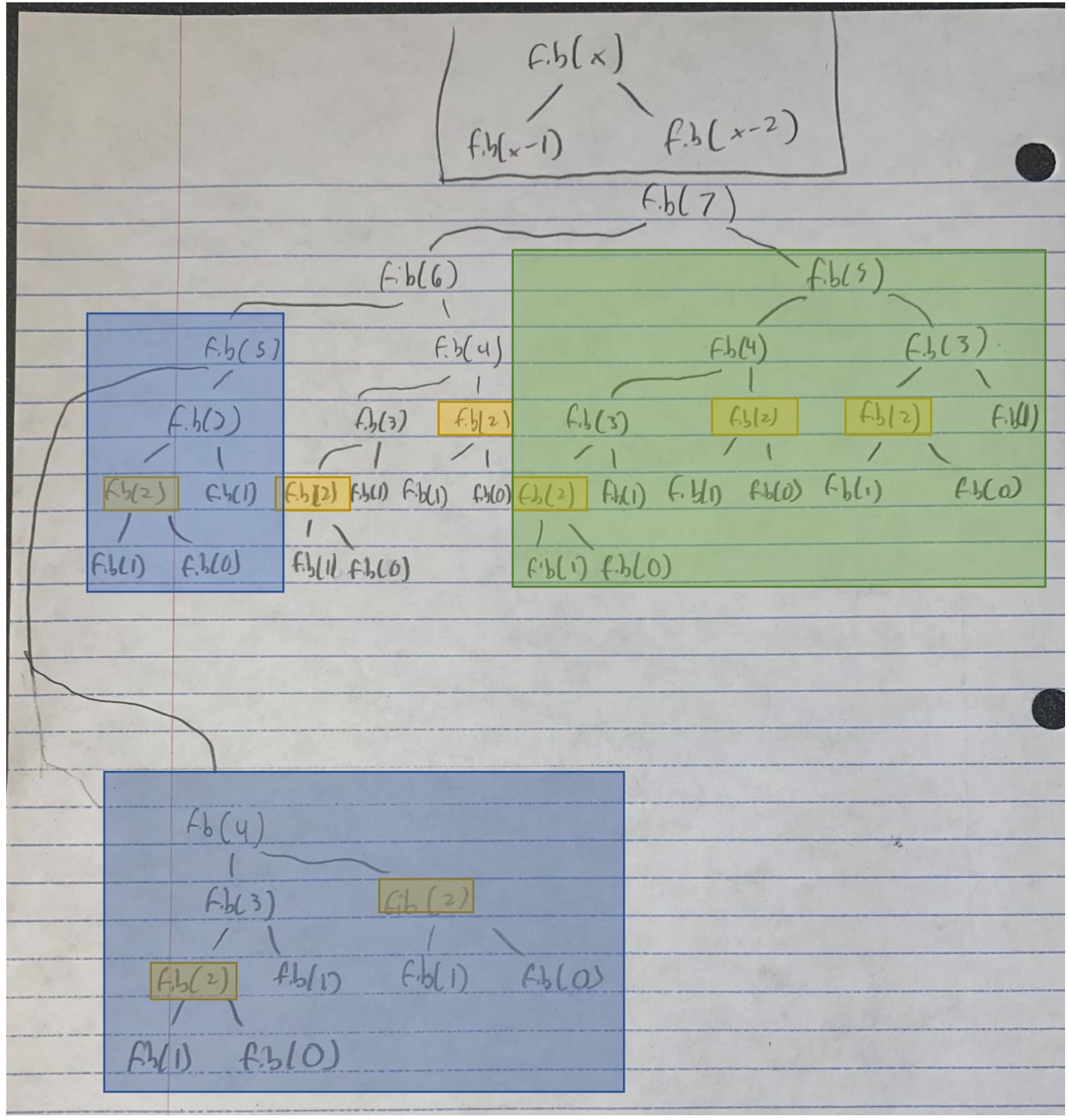
Why?

- Reduces redundant work & makes things more efficient
  - Sometimes, *it's the only way*
- Not taught formally here at WWU

Fibonacci demo



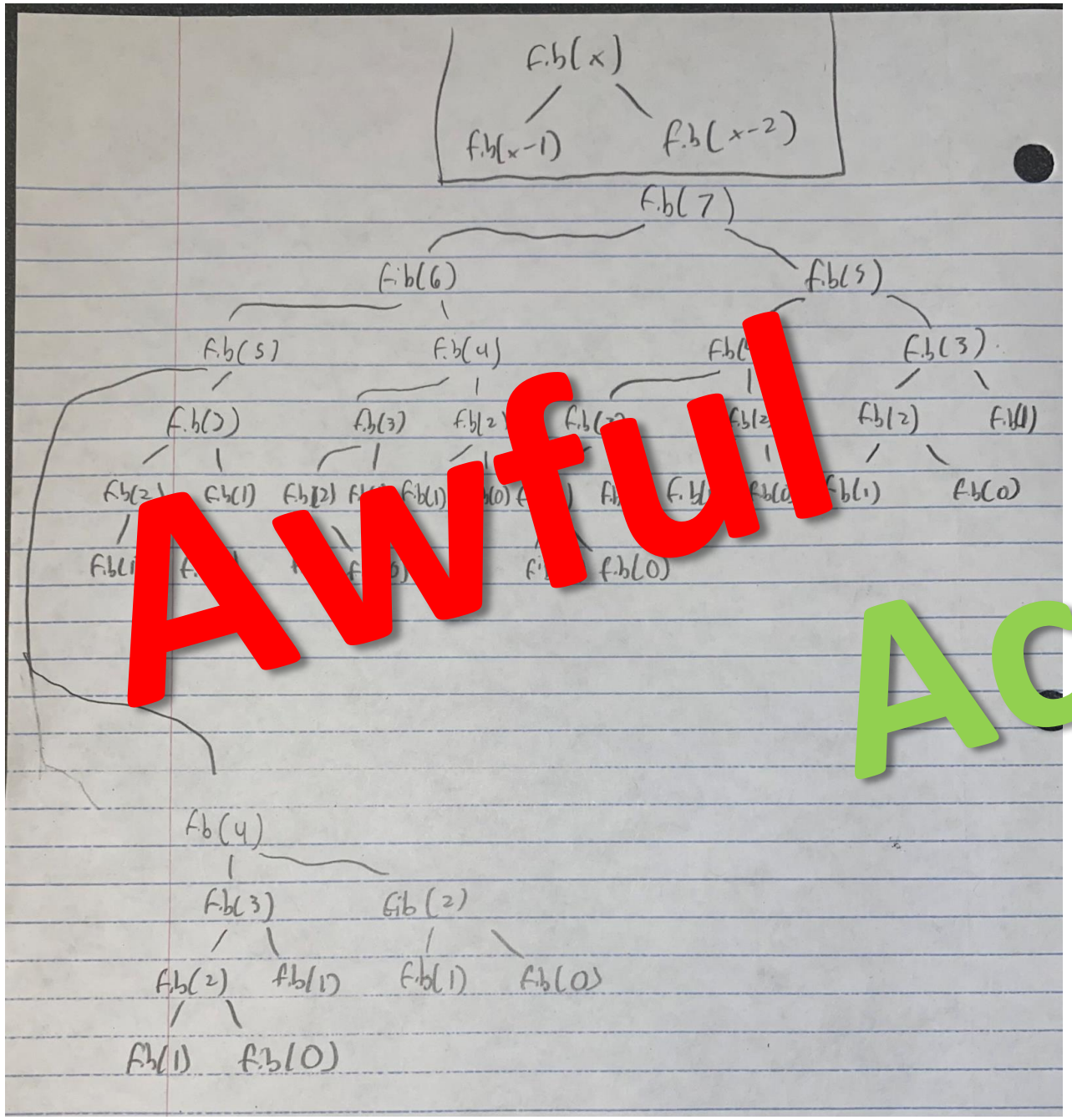
Awful!!!!!!!!!!!!!!!!!!!!!!

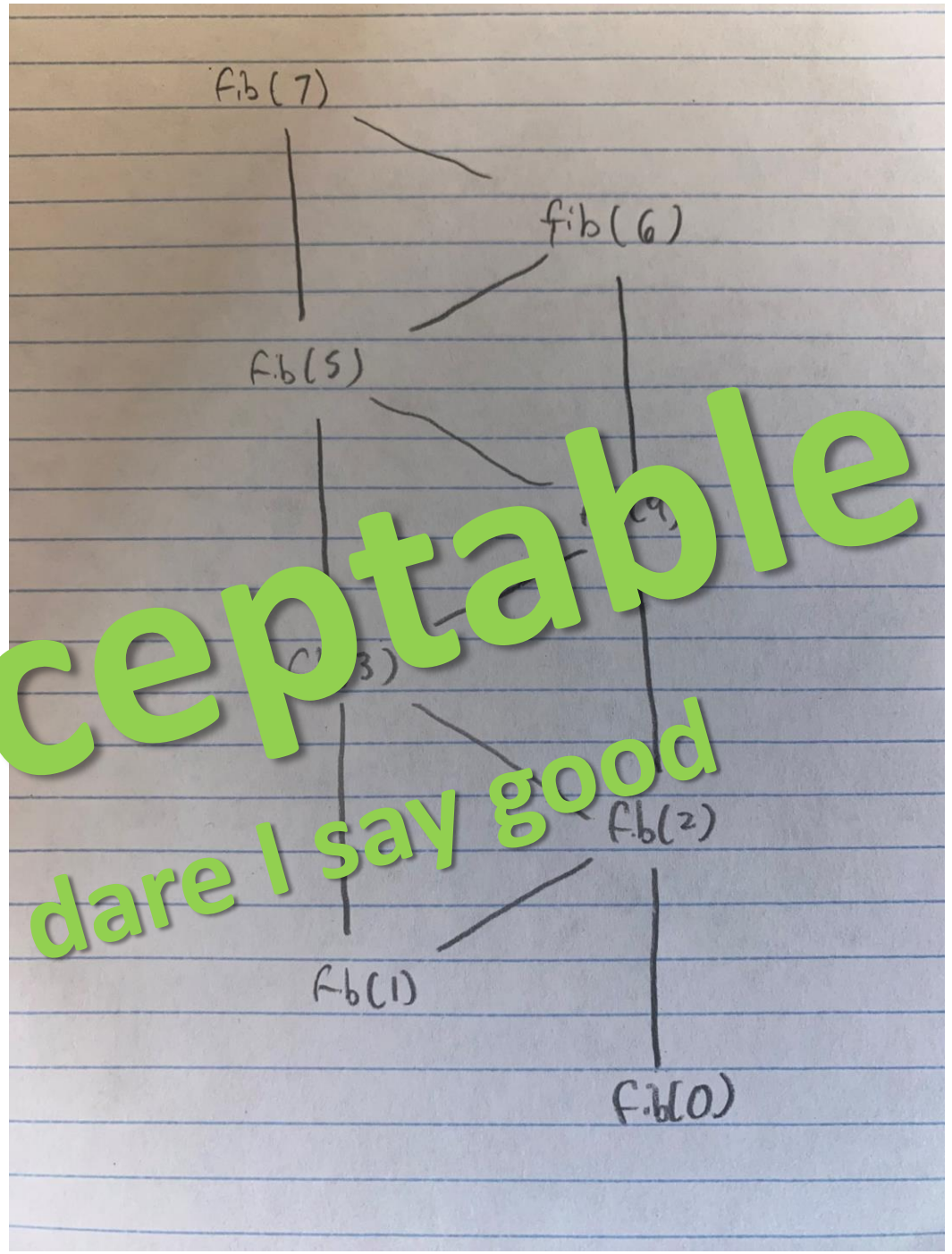

fib(x)
fib(x-1)
fib(x-2)
fib(7)
fib(6)
fib(5)
fib(5)
fib(4)
fib(4)
fib(3)
fib(3)
fib(2)
fib(3)
fib(2)
fib(2)
fib(1)
fib(2)
fib(1)
fib(2)
fib(1)
fib(1)
fib(0)
fib(2)
fib(1)
fib(1)
fib(0)
fib(1)
fib(0)
fib(1)
fib(0)
fib(1)
fib(0)
fib(1)
fib(0)
fib(4)
fib(3)
fib(2)
fib(2)
fib(1)
fib(1)
fib(0)
fib(1)
fib(0)

How do we stop doing stuff we already did?

How do we stop doing stuff we already did?

- If you've done it, **save it**
- Before you do it, **check to see if you've done it.**


fib(x)
fib(x-1)
fib(x-2)
fib(7)
fib(6)
fib(5)
fib(4)
fib(3)
fib(2)
fib(1)
fib(0)
Awful


fib(7)
fib(6)
fib(5)
fib(2)
fib(1)
fib(0)
Acceptable
dare I say good

Types of Dynamic Programming

Memoization

- Recursive

Tabulation

- Iterative

Fibonacci Numbers using Tabulation

Tabulation:

- Do most of the work of each step before you get to it
- Build the pyramid layer by layer, so that you're always on solid ground
- "Bottom-up"

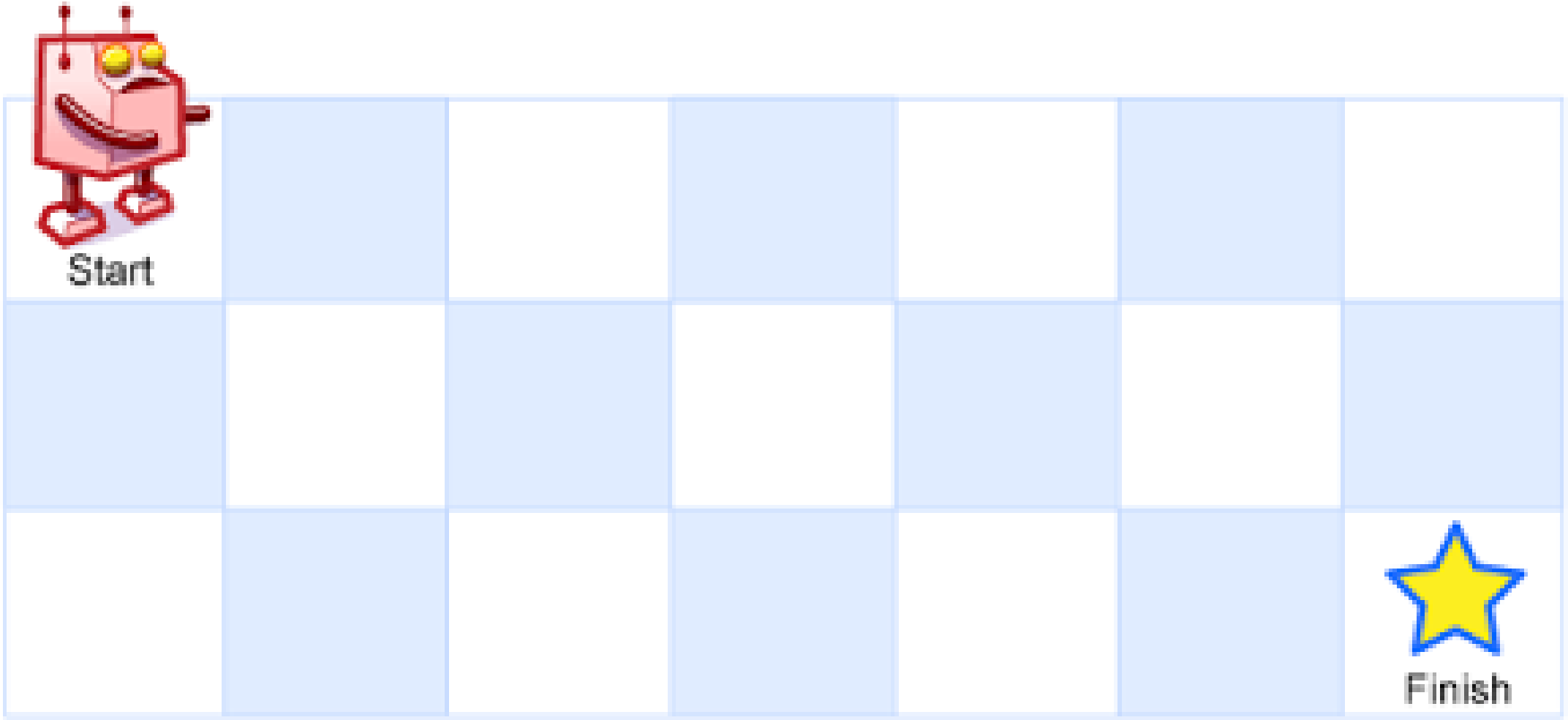
Memoization:

- Check to see if you've done this entire step before
- Build the pyramid under you at the point that you need it
- "Top-down"

Most importantly:
ONLY BUILD ONE PYRAMID!

One Last Sample Question

Hi pdf viewer! This problem is “Unique Paths” on Leetcode.



When do I use it?

shoutout to Sam for making this slide possible

- Recursive
- When you need to deal with smaller problems to find solutions to bigger ones
- When it’s easy to figure out what smaller problems need to be answered to find each solution
  - Metaphorically, for tabulation: when you can make sure that the bricks you need are already in your pyramid

Pros & Cons

Tabulation:

- Less memory usage
- Large recursive stack can cause exceptions

Memoization:

- Often more intuitive
- Works when subproblems overlap