**Dynamic Programming**

(Like how you solve the knapsack problem) means storing results that we already calculated, and use those results for further calculations to save time.

Steps to generate Dynamic Programming solution:

1. Find Recursive solution
2. Store (Memoize) repeated computations
3. Bottom-up process

A picture containing text, font, screenshot, line

Description automatically generated**Example**: Fibonacci (**Recursive**) 🡪 **O(2n)**

NOTE: There are repeated calculations, why don't we **memoize?**

A picture containing text, font, screenshot, design

Description automatically generatedFibonacci (**Memoized**) 🡪 **O(n)**

NOTE: We memoized the results at index n in an array size (n + 1)

Why don't we just explicitly build this array from **Bottom-up** (left to right), instead of having to do it recursively?

Fibonacci (**Bottom-up**) 🡪 **O(n)**

This means we just simply build the array from the simplest values, and then get the result at nth element! (just like adding from 1, 1, 2, 3, … to get the result at nth element.

A picture containing text, font, line, number

Description automatically generated