App Academy

Dynamic Programming (1-D)

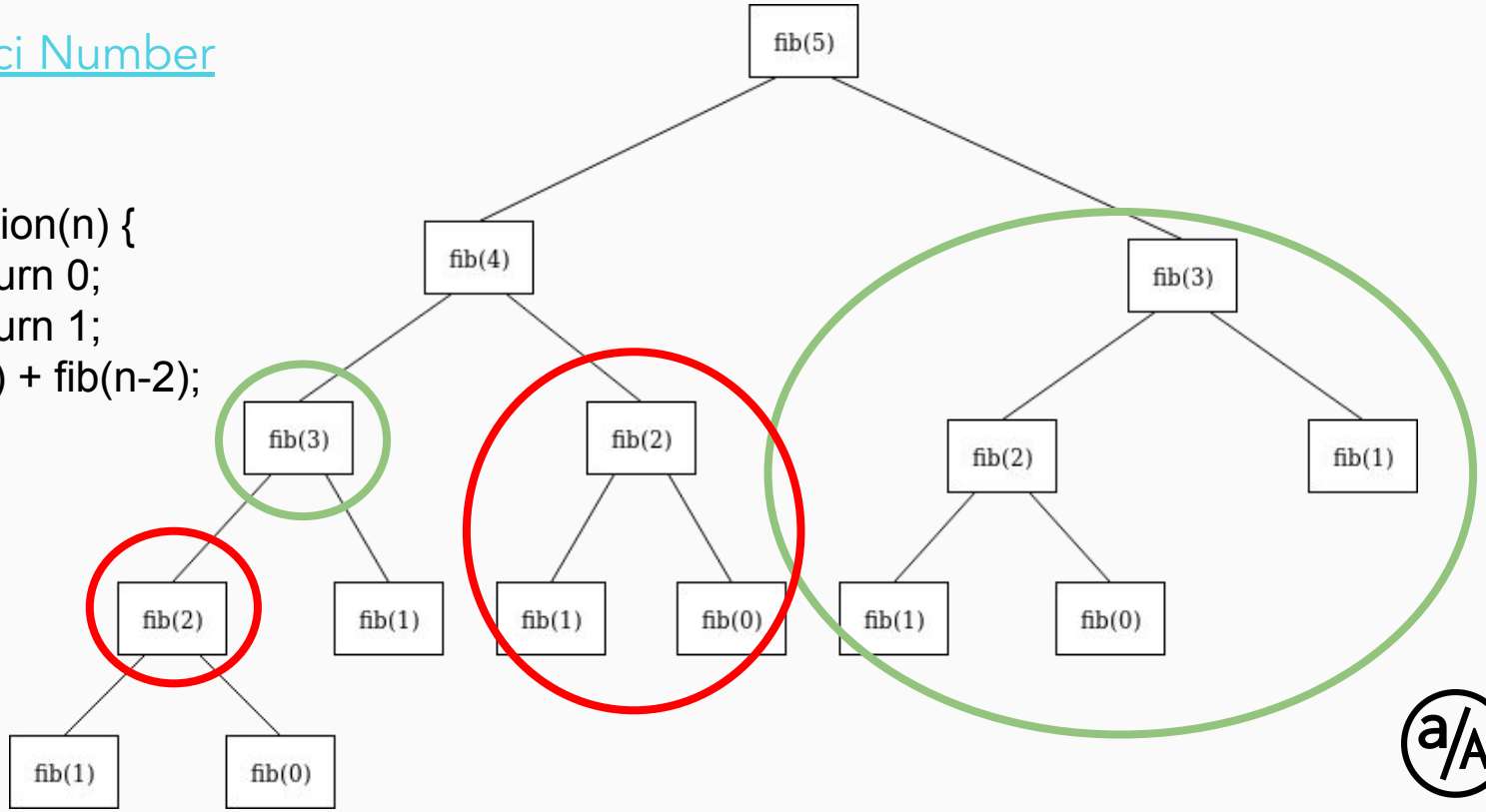
Intro to Dynamic Programming (DP)

- DP is not a standalone algorithm or technique.
- Rather, DP is a technique used to optimize less efficient solutions.
- We can identify when we can use DP by looking for sub-problems.
- When you have large complexities, it may be useful to think about whether DP could be used or not.
- 1D DP refers to the solution space of the problem.
- Types of DP
  - Memoization (top-down DP)
  - Tabulation (bottom-up DP)

Demo

- Fibonacci Number
- $O(2^n)$

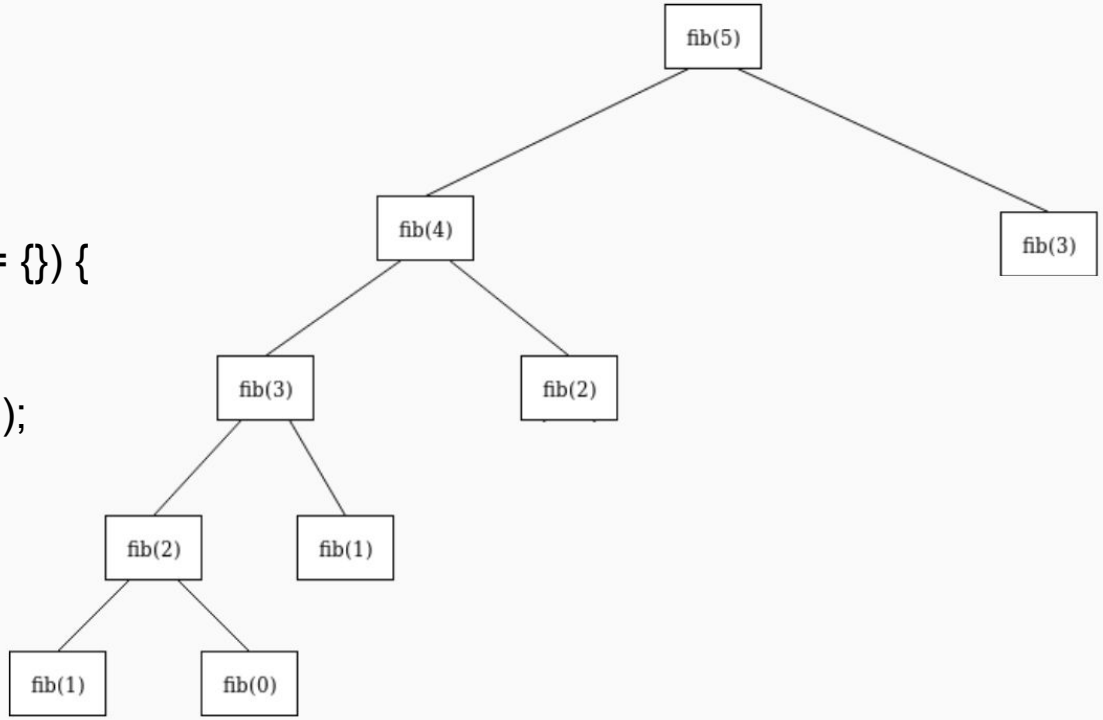
```
const fib = function(n) {
    if (n===0) return 0;
    if (n===1) return 1;
    return fib(n-1) + fib(n-2);
};
```



Memoization (top-down)

- Fibonacci Number
- O(2n) => O(n)

```
const fib = function(n, memo = {}) {
    if (n===0) return 0;
    if (n===1) return 1;
    memo[n] = fib(n-1) + fib(n-2);
    return memo[n];
};
```



Tabulation (bottom-up)

```
const fib = function(n) {
    if (n < 2) return n;
    let dp = [0,1];
    let i = 2;

    while (i <= n) {
        let tmp = dp[1];
        dp[1] = dp[0] + dp[1];
        dp[0] = tmp;
        i++;
    }
    return dp[1];
};
```

solution space is 1D

0	1	2	3	4	5

- The idea with bottom-up DP is to start from the bottom of our tree (i.e. our base case) and then work our way up towards the root (i.e. our original input).
- This solution is usually more difficult and requires some pre-planning to come up with rather than just modifying an existing recursive solution.

Questions?

Let’s practice!

- Review
  - Climbing Stairs
  - Coin Change
- Bonus
  - House Robber
  - Palindromic Substrings