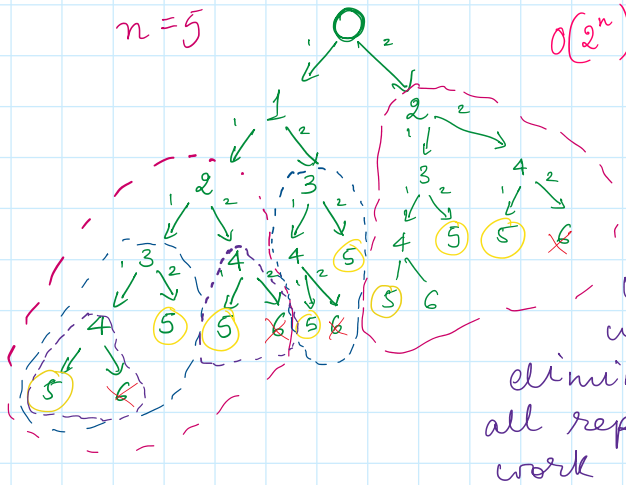
DP Climbing Stairs LeetCode 70

Monday, 11 September 2023 10:56 AM

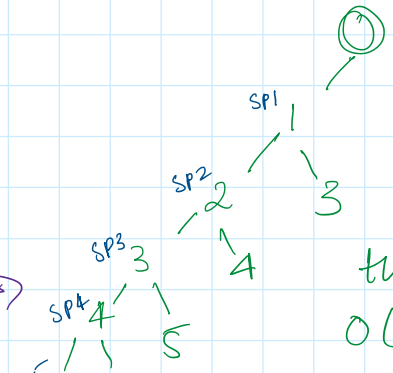
Visualize w/ tree. At every step there are 2 decisions to be made: 1 step or 2 step. Leads to 2 diff paths to get soln.

Aim: How many diff. ways we get to n?

SP: Subproblem



when we eliminate all repeated work our tree would look like



this is O(n) only solving each subproblem once.

Pink Subtrees: same calc. twice

Store left subtree in memory.
(caching, memoization)

We can also do bottom up Approach

Start at sub problem SP 5 (Base Case)

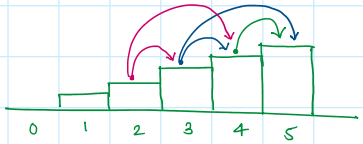
Store results in array.

DP_array[0:n] =

8	5	3	2	1	1
0	1	2	3	4	5

here n=5

Stores how many diff. ways from a node can we reach n(5)



5→5: 1 way to reach 5
4→5: 1 way to reach (+1)
3-5: 2 way to reach (1+1) = 2
2-5: 3 ways to reach (2+1) = 3
1-5: 5 ways (2+3) = 5
0-5: 8 ways (5+3)

2nd step would take us out of bounds

Add previous 2

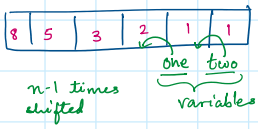
dependent on the 2 sub problems that [illegible] it is SP5 & SP4

0-5 : 8 ways (5+3)

dependent on the 2 sub problems that come after it i.e. SP5 & SP4
so ways to reach n in this case would be SP5 + SP4 = 1+1 = 2

fibonacci nos. starting at base case 1, 1.

Don't really need full array.
just need 2 values that come after it.



n-1 times shifted

variables

```
class Solution:
    def climbStairs(self, n: int) -> int:
        one, two = 1, 1

        for i in range(n-1):
            temp = one
            one = one + two
            two = temp
        return one
```