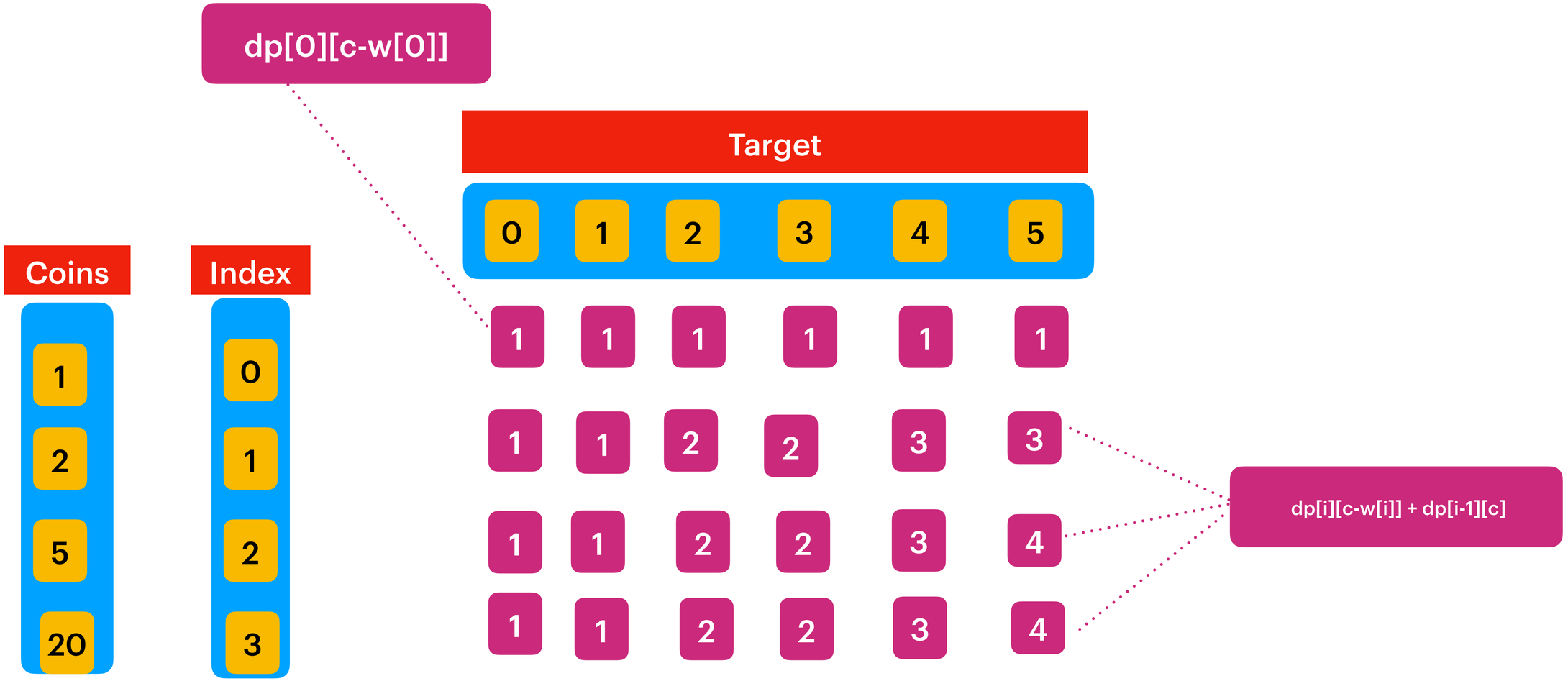
Coin Change

Given an infinite supply of ‘n’ coin denominations and a total money amount, we are asked to find the total number of distinct ways to make up that amount.

Denominations: {1,2,5,20}
Total amount: 5
Output: 4
Explanation: There are 4 ways to make the change for '5', here are those ways:
1. {1,1,1,1,1}
2. {1,1,1,2}
3. {1,2,2}
4. {5}


dp[0][c-w[0]]
Target
0
1
2
3
4
5
Coins
1
2
5
20
Index
0
1
2
3
1 1 1 1 1 1
1 1 2 2 3 3
1 1 2 2 3 4
1 1 2 2 3 4
dp[i][c-w[i]] + dp[i-1][c]

Fibonacci Pattern

Staircase:

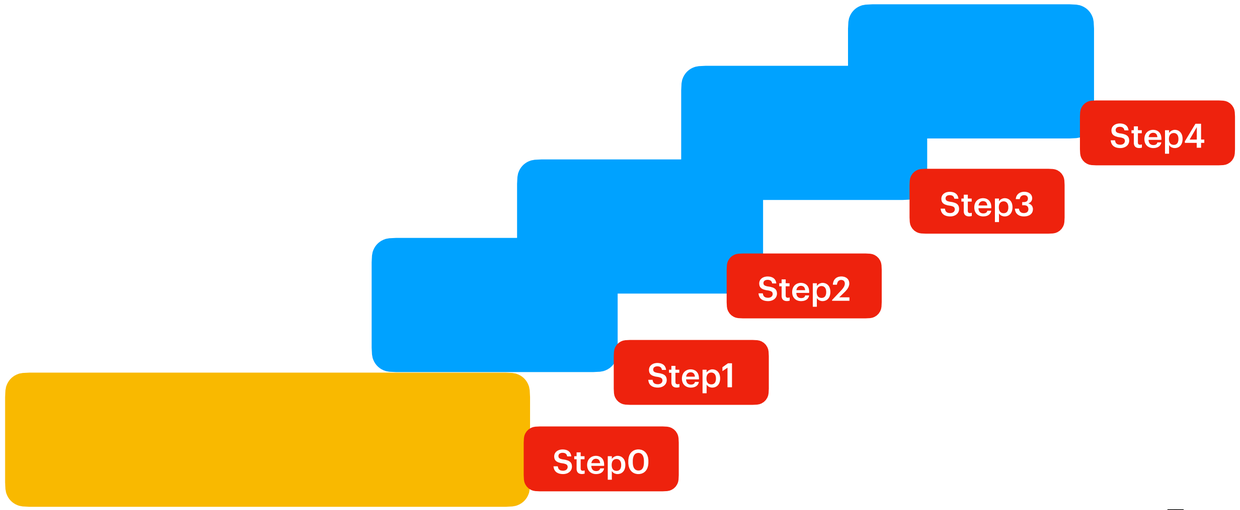
Given a stair with ‘n’ steps, implement a method to count how many possible ways are there to reach the top of the staircase, given that, at every step you can either take 1 step, 2 steps, or 3 steps.

Number of stairs (n) : 3
Number of ways = 4
Explanation: Following are the four ways we can climb : {1,1,1}, {1,2}, {2,1}, {3}

Number of stairs (n) : 4
Number of ways = 7
Explanation: Following are the seven ways we can climb : {1,1,1,1}, {1,1,2}, {1,2,1}, {2,1,1}, {2,2}, {1,3}, {3,1}

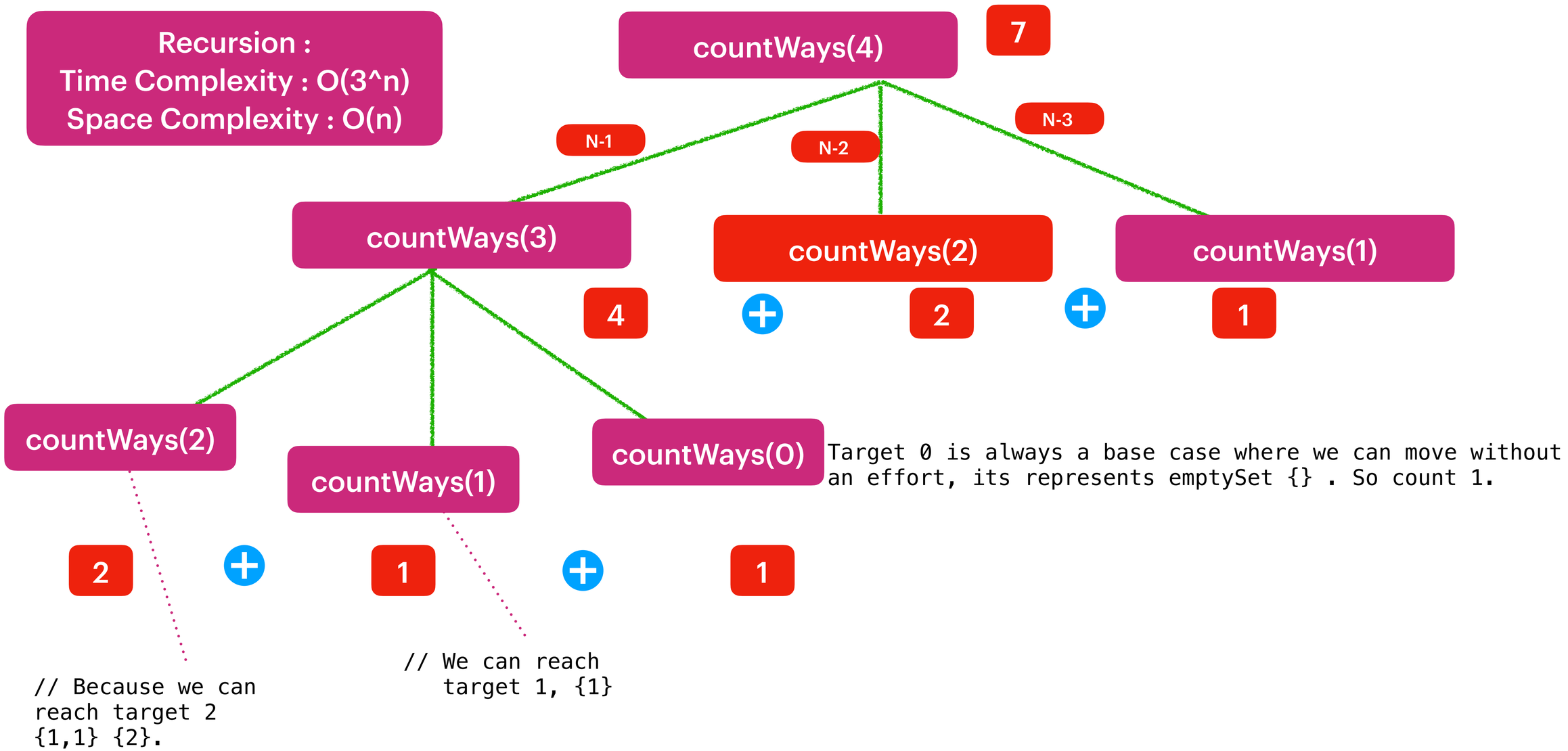
Target is 3 : either you can choose step1 or 2 or 3

{1,1,1} ,{1,2}, {2,1} {3} => 4Ways

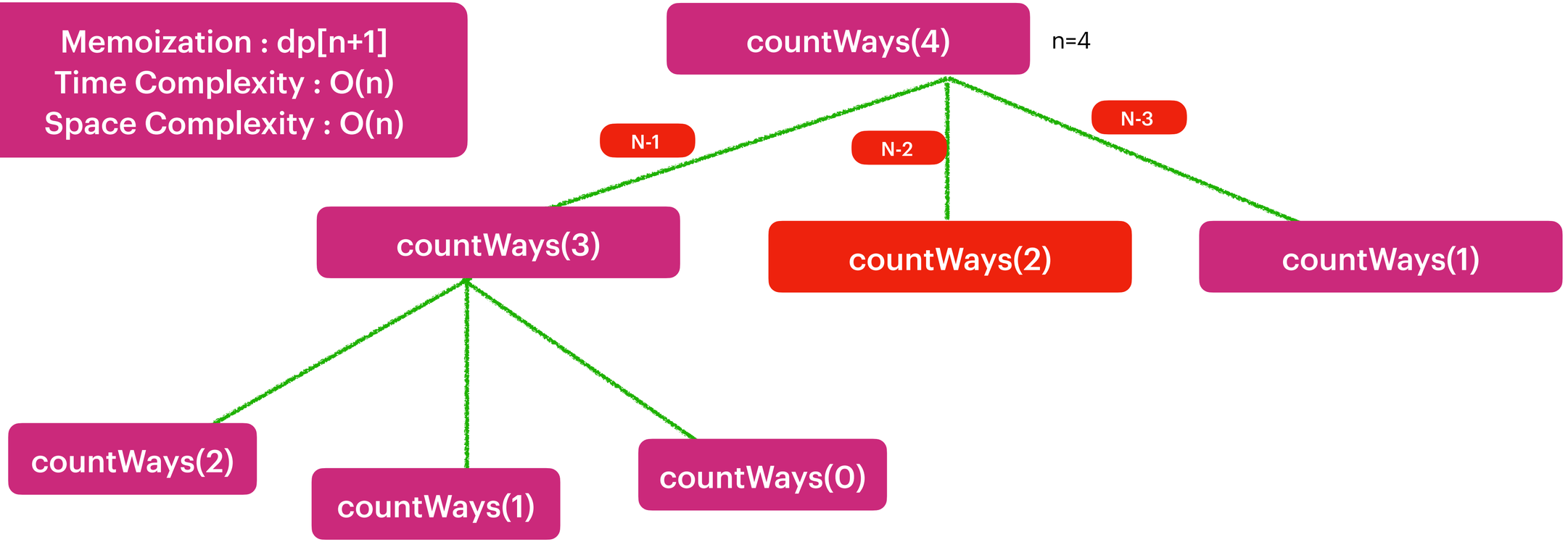


Target Stair 4

{1,1,1,1} , {1,1,2} , {1,2,1} , {1,3},{2,1,1},{2,2},{3,1} =7


Recursion :
Time Complexity : O(3^n)
Space Complexity : O(n)
countWays(4)
7
N-1
N-2
N-3
countWays(3)
countWays(2)
countWays(1)
4
2
1
countWays(2)
countWays(1)
countWays(0)
2
1
1
Target 0 is always a base case where we can move without an effort, its represents emptySet {} . So count 1.
// Because we can reach target 2 {1,1} {2}.
// We can reach target 1, {1}

Memoization : dp[n+1]
Time Complexity : O(n)
Space Complexity : O(n)



Tabulation : dp[n+1]
Time Complexity : O(n)
SpaceComplexity : O(n)

dp[0] = 1; // reach target 0 , {}
dp[1] = 1; // reach target 1, {1}
dp[2] = 2; // reach target 2 , {1,1} {2}

dp[3] = dp[2] + dp[1] + dp[0] = 2+1+1 = 4
dp[n] = dp[n-1] + dp[n-2] + dp[n-3];