DP

https://www.youtube.com/watch?v=1BAsAgdx7Ac&t=31s

What? 查找有很多重叠子问题的情况的最优解。

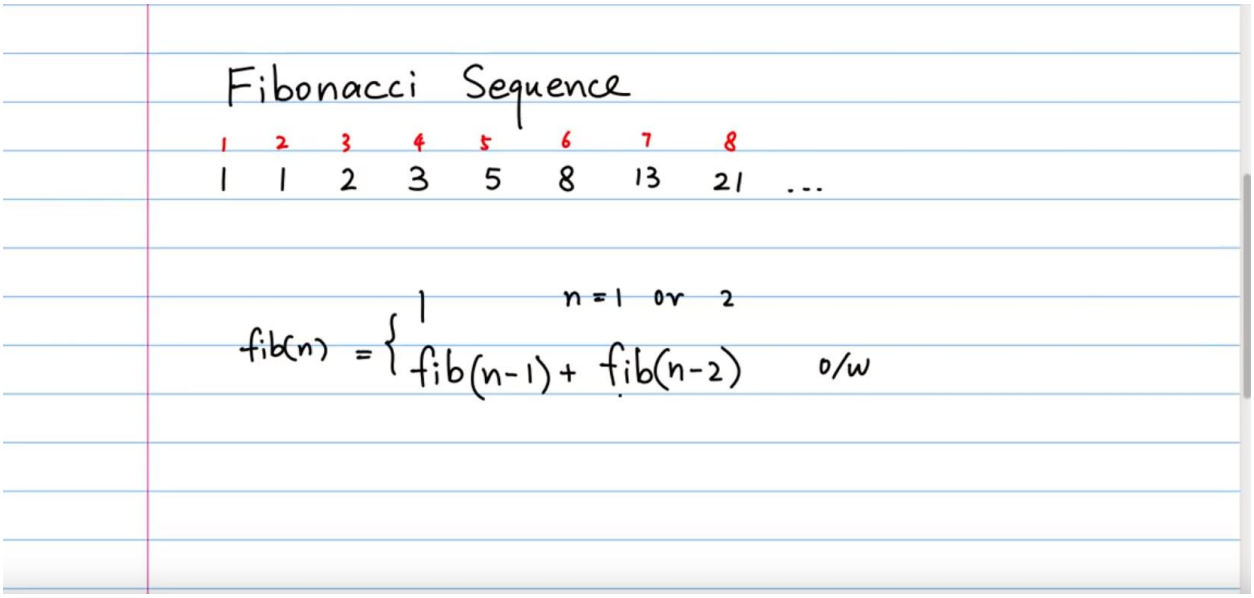
Why? 相对于递归节省大量时间

How? 为了避免多次解决这些子问题，它们的结果都逐渐被计算并被保存，从简单的问题直到整个问题都被解决。因此，动态规划保存递归时的结果，因而不会在解决同样的问题时花费时间。

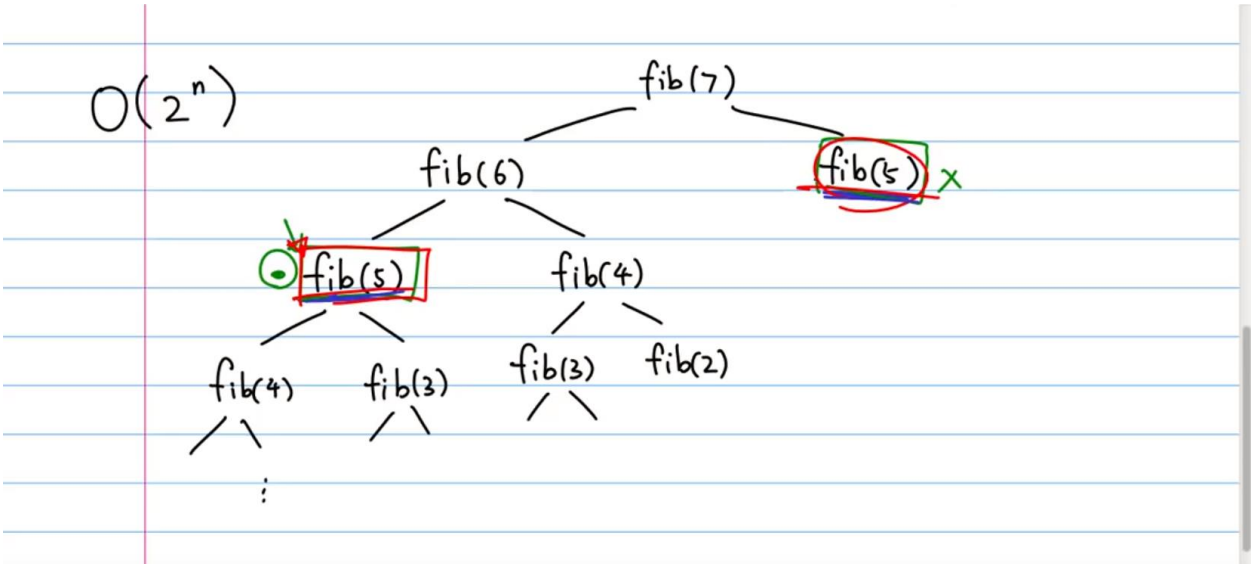
递归：从后往前算。

DP：从前往后算

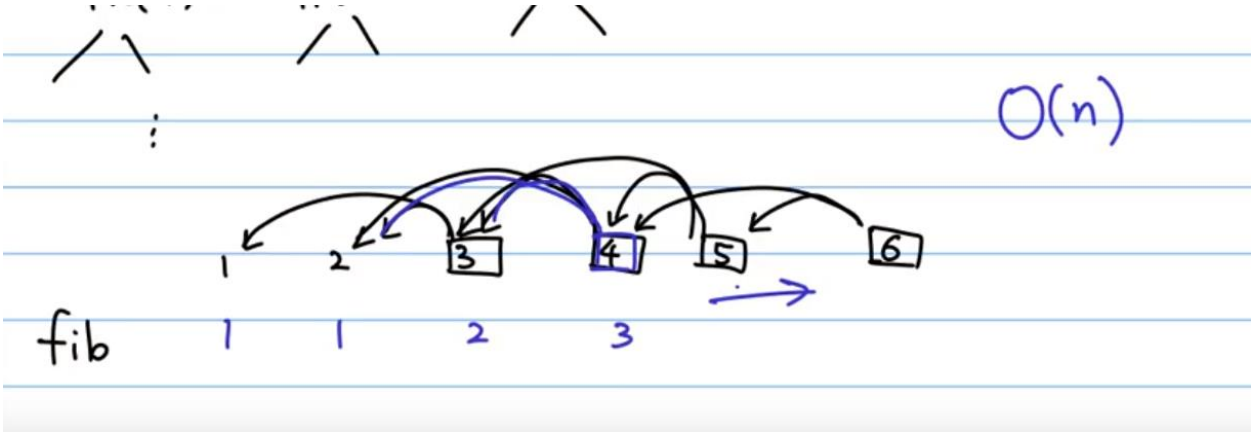
例 1：Fibonacci Sequence:



假设直接带入递归公式，会产生重叠子问题如 fib(5)和右边的 fib(5),dp 做的是把左边的 fib(5)保存起来，这样右边的 fib(5)就不用再展开计算。



Dp 的整个递归过程如下：




```python
arr = [1, 2, 4, 1, 7, 8, 3]

def rec_opt(arr, i):
    if i == 0:
        return arr[0]
    elif i == 1:
        return max(arr[0], arr[1])
    else:
        A = rec_opt(arr, i - 2) + arr[i]
        B = rec_opt(arr, i - 1)
        return max(A, B)
```

```python
def dp_opt(arr):
    opt = np.zeros(len(arr))
    opt[0] = arr[0]
    opt[1] = max(arr[0], arr[1])
    for i in range(2, len(arr)):
        A = opt[i-2] + arr[i]
        B = opt[i-1]
        opt[i] = max(A, B)
    return opt[ len(arr) - 1 ]

dp_opt(arr)
```