0	1	2	3	4	5	6	7
0	1	1	2	3	5	8	13

Fibonacci Series

Using Recursion

$n = 0$ ——— 0

$n = 1$ ——— 1

$n = 2$ ——— $1 + 0 = 1$

$fib(6) = fib(5) + fib(4)$

$= 5 + 3$

$= 8$

$n \geq 0$

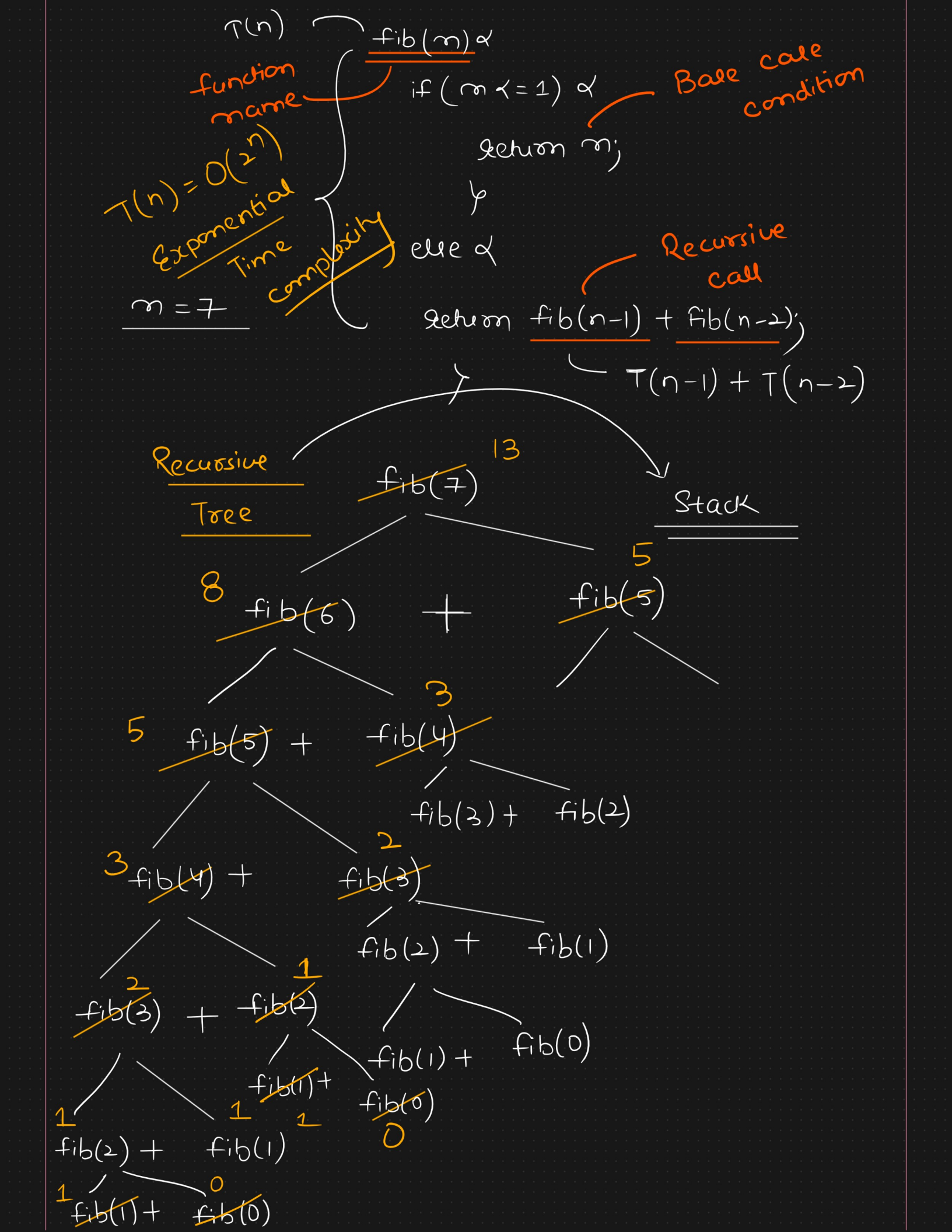
Recursive code

Base case condition

return n; $n \leq 1$

Recursive calls

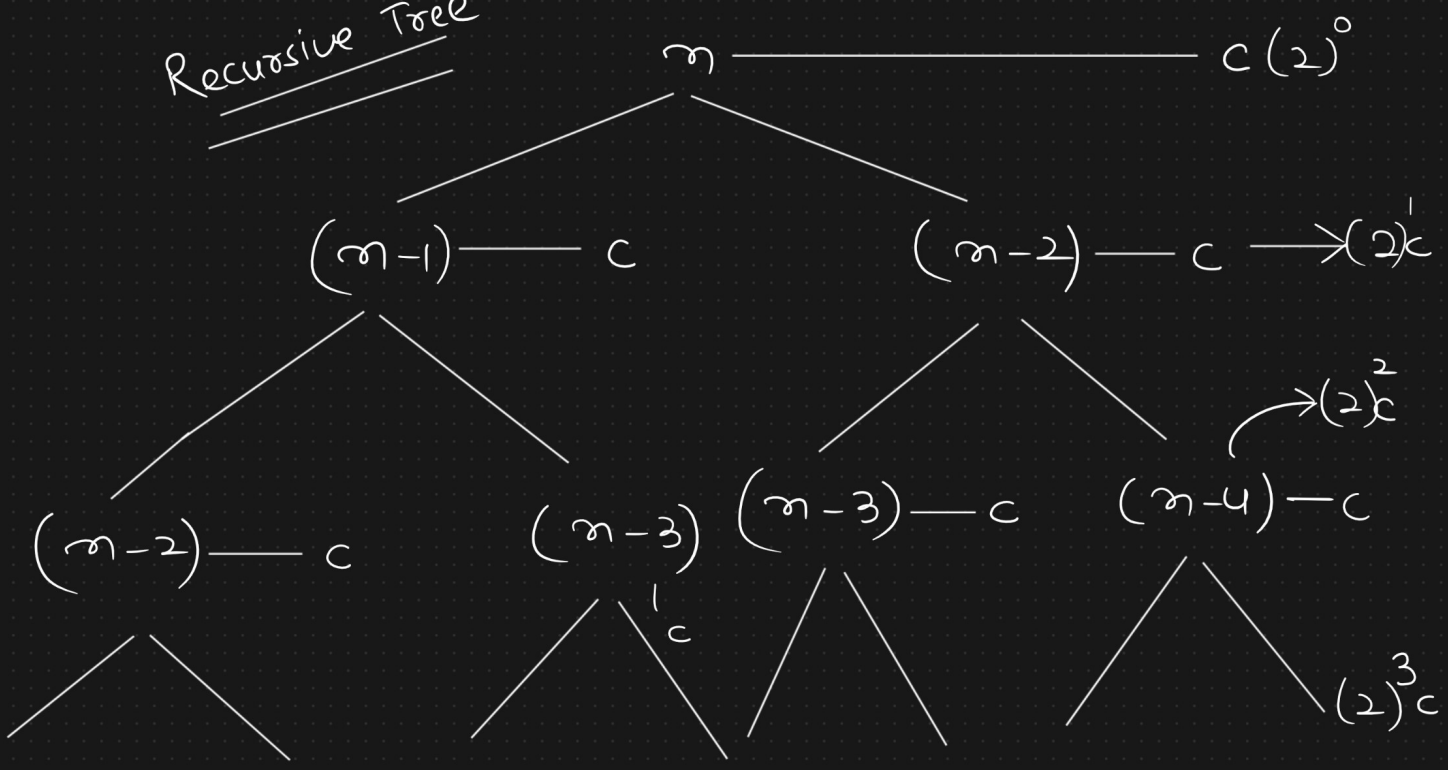
$fib(n-1) + fib(n-2)$; $n > 1$


T(n)
fib(n) {
function name
if (n <= 1) {
Base case condition
return n;
}
else {
return fib(n-1) + fib(n-2);
Recursive call
T(n-1) + T(n-2)
T(n) = O(2^n)
Exponential Time complexity
n = 7
Recursive Tree
Stack
fib(7)
13
fib(6)
8
+
fib(5)
5
fib(5)
5
+
fib(4)
3
fib(4)
3
+
fib(3)
2
fib(3)
2
+
fib(2)
1
fib(3)
2
+
fib(2)
1
fib(2)
+
fib(1)
1
1
fib(1)
+
fib(0)
0
fib(2)
+
fib(1)
1
1
fib(1)
+
fib(0)
1
0
fib(2) +
fib(1)
fib(0)
fib(3) + fib(2)
fib(1) + fib(0)

Recurrence Relation — Recursive Tree Method

$$T(n) = \begin{cases} c & n = 1 \\ T(n-1) + T(n-2) + c & n > 1 \end{cases}$$

Recursive Tree



Left side

$$(n - k) = 1$$

$$(n - 1) = k$$

Right side

$$k = \left(\frac{n-1}{2}\right)$$

$$(n - 2k) = 1$$

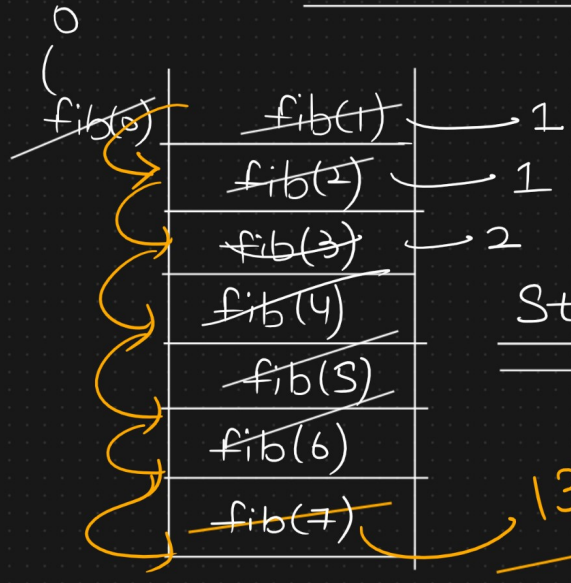
$$c\left(\underbrace{2^0 + 2^1 + 2^2 + \cdots + k \text{ times}}_{\text{\# Levels}}\right)$$

$$c * 2^n = O(2^n)$$

↓

Exponential Time complexity

Space complexity



$\Rightarrow O(n)$

Stack — LIFO