Recursion

1) Base case condition

2) Recursive call — function calls itself

Fibonacci Series

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

$$fib(n) \Rightarrow \begin{cases} n & n <= 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

Method name → fib(n) {

① Base case condition

```
if (n<=1) {
    return n;
}
```

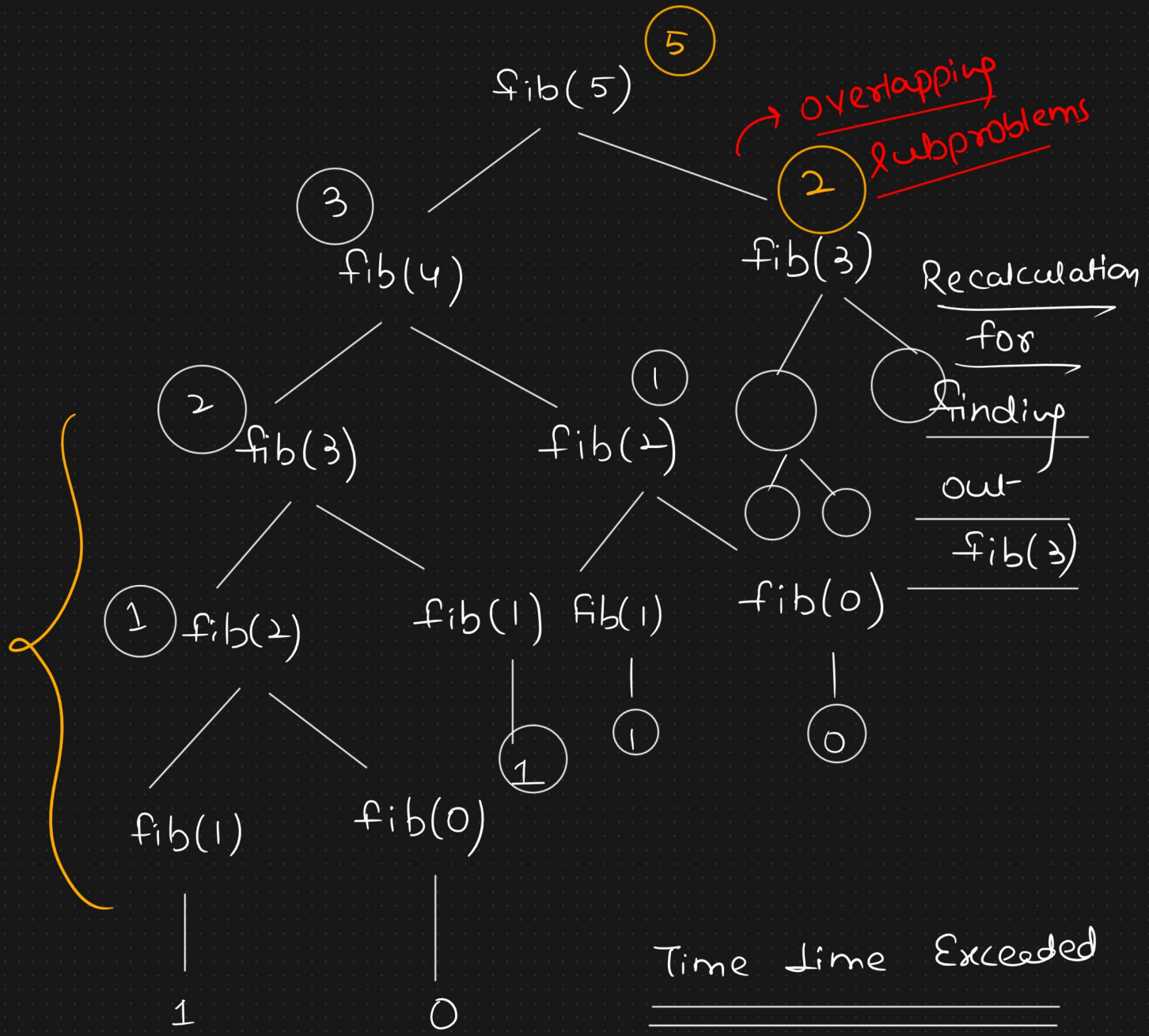
else {

different set of parameters

```
return fib(n-1) + fib(n-2);
}
}
```

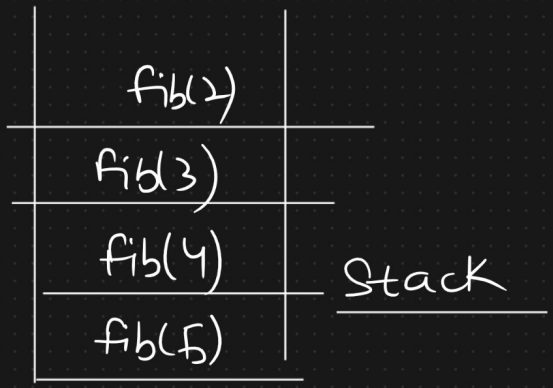
$O(2^n)$

② Recursion



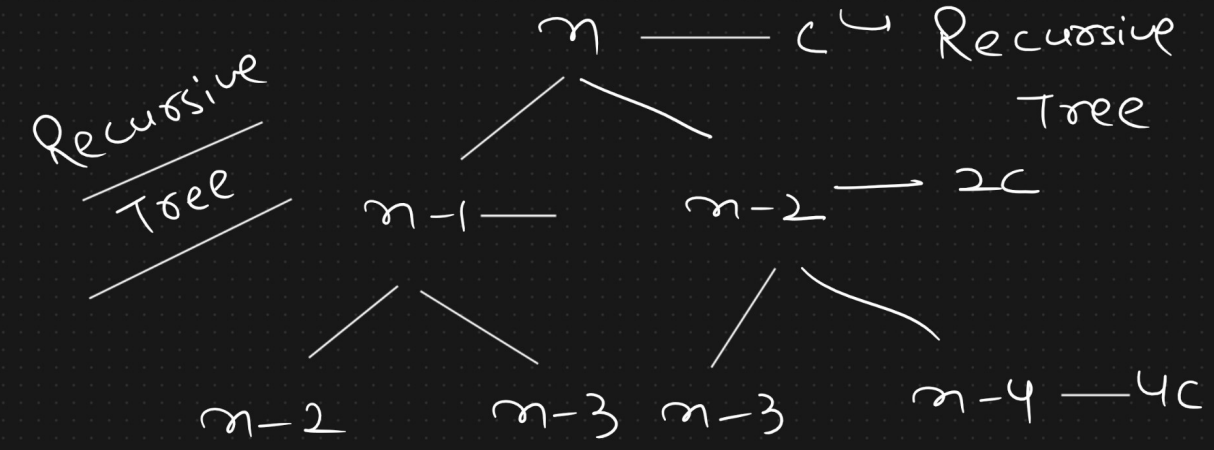
Time time Exceeded

Dynamic Programming

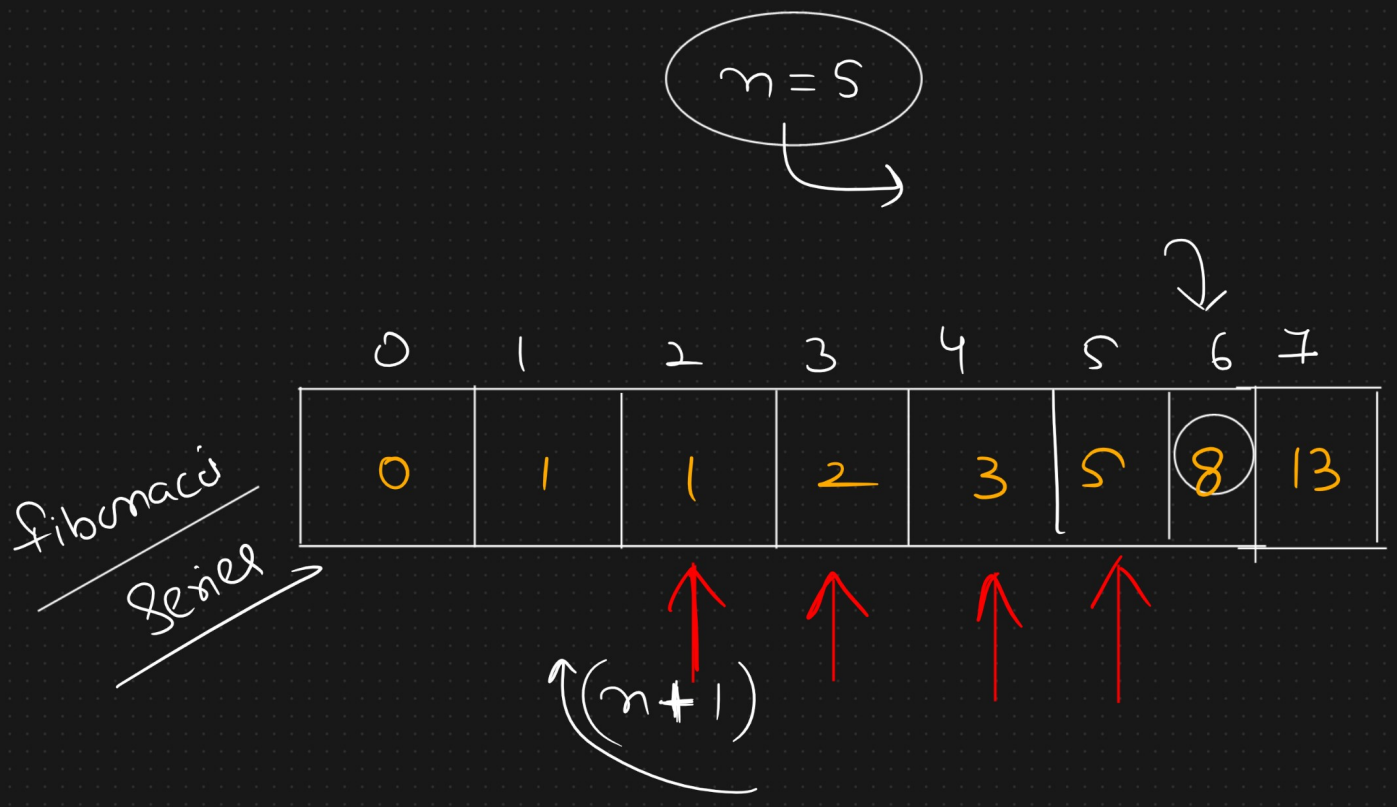


$$T(n) = T(n-1) + T(n-2) + c$$

Recursive Tree



$$O(2^n)$$



Climbing a Stairs —— (n)

n = 1

↳ 1 way — 1 step

n = 2

↳ 2 ways — 1 step, 1 step

2 steps

n = 3

1 step, 1 step, 1 step

↳ 3 ways — 1 step, 2 step

2 step, 1 step

n = 4

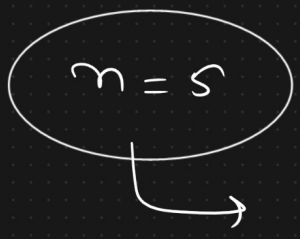
1 step, 1 step, 1 step, 1 step

↳ 5 ways — 1 step, 2 step, 1 step

2 step, 1 step, 1 step

2 step, 1 step, 1 step, 2 step

2 step, 2 step



8 ways

1 1 1 1 1

1 2 2

2 1 2

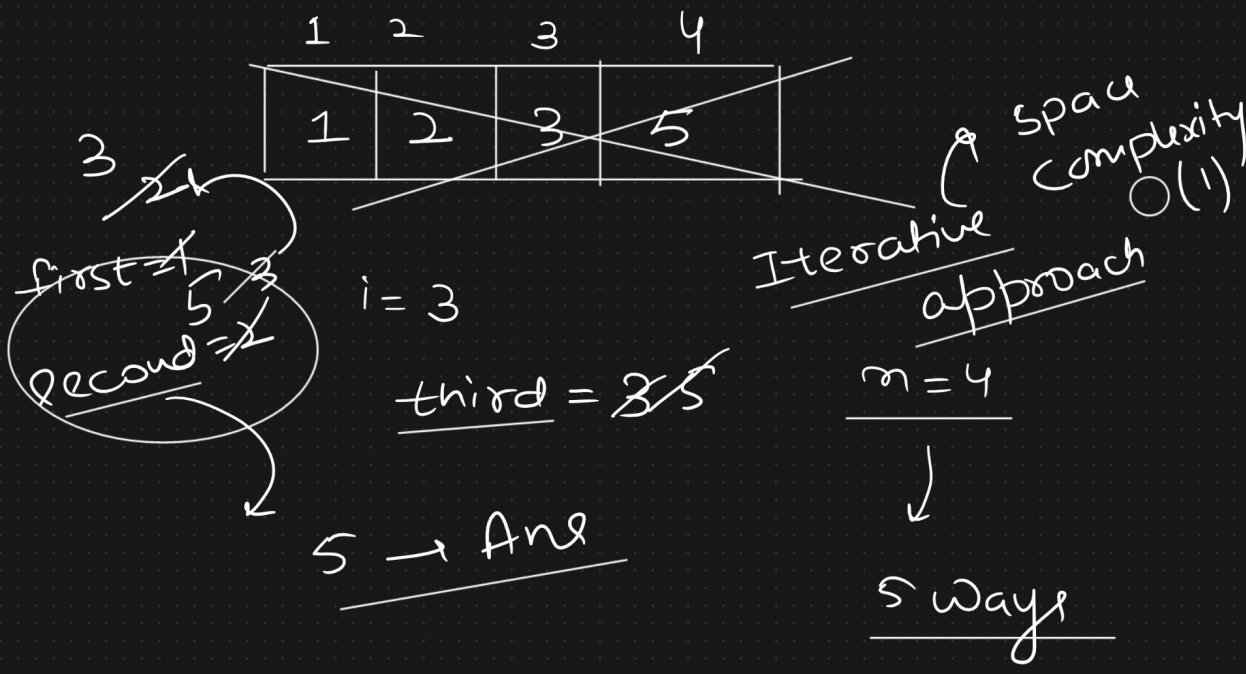
2 2 1

2 1 1 1

1 2 1 1

1 1 2 1

1 1 1 2



{ Time complexity → O(n)

Space complexity → O(1)