O(N)

O(1) space

"abacbc" → whether every character is occ. Same no. of times

How about we calc the no. of occ. of each char.

Key value

↓ char

↳ frequency

temp = 2

{
'a' : X2
'b' : X2
'C' : X2
}

Because we have to keep track if occ, we can map the freq

Power function Recursive

Q → Given two integer values 'a' and 'b', calculate a^b recursively.

Ex → a = 2 b = 3

ans → 8

Iterative

```
ans=1
for(i=1; i<=b; i++){
    ans *= a;
}
```

$4^3 = \underbrace{4 \times 4 \times 4}$

$a^b = \underbrace{a \times a \times a \ldots\ldots a}_{b \text{ times}}$

Recursive

$a^b = a \times a \times a \times a \ldots\ldots a$ (b times)

$a^{b+1} = \underbrace{a \times a \times a \times a \ldots\ldots a}_{a^b} \times a$ (b+1 times)

$$a^{b+1} = a^b \times a$$

Similarly

$$a^b = a^{b-1} \times a$$

$$2^4 = 2 \times 2^3$$

my task

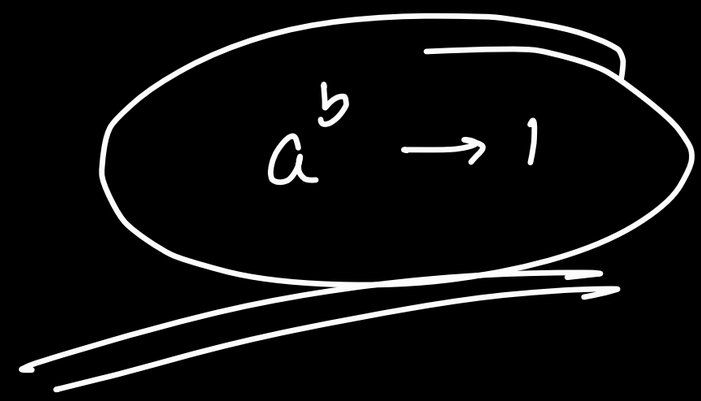
$$2^3 = 2 \times 2^2$$

$$2^3 = 2 \times 2^2$$

$$2^2 = 2 \times 2^1$$

$$2^1 = 2 \times 2^0$$

The most trivial (Base case) $\rightarrow$ $b \rightarrow 0$



$f(a, b) = a \times f(a, b-1)$ → self work

$f(a,b)$ ↓ returns a raise to power b

$f(a, b-1)$ → assumption

no. of ins → b x c → O(b)

#70%

```
if (b==0)
  return 1;
```
} Base Case

$a^b = a + a^{b-1}$

$= a + (a + a^{b-2})$

. . - - - - -

(No. of func^n calls) x (no. of ins in each call) ← No. of ins

1. Base Case → if (b == 0)
   return 1;

2. Assumption → assume $f(a, b-1)$ works correctly and a^{b-1} is recursively computed

3. Self work → multiply a with a^{b-1}

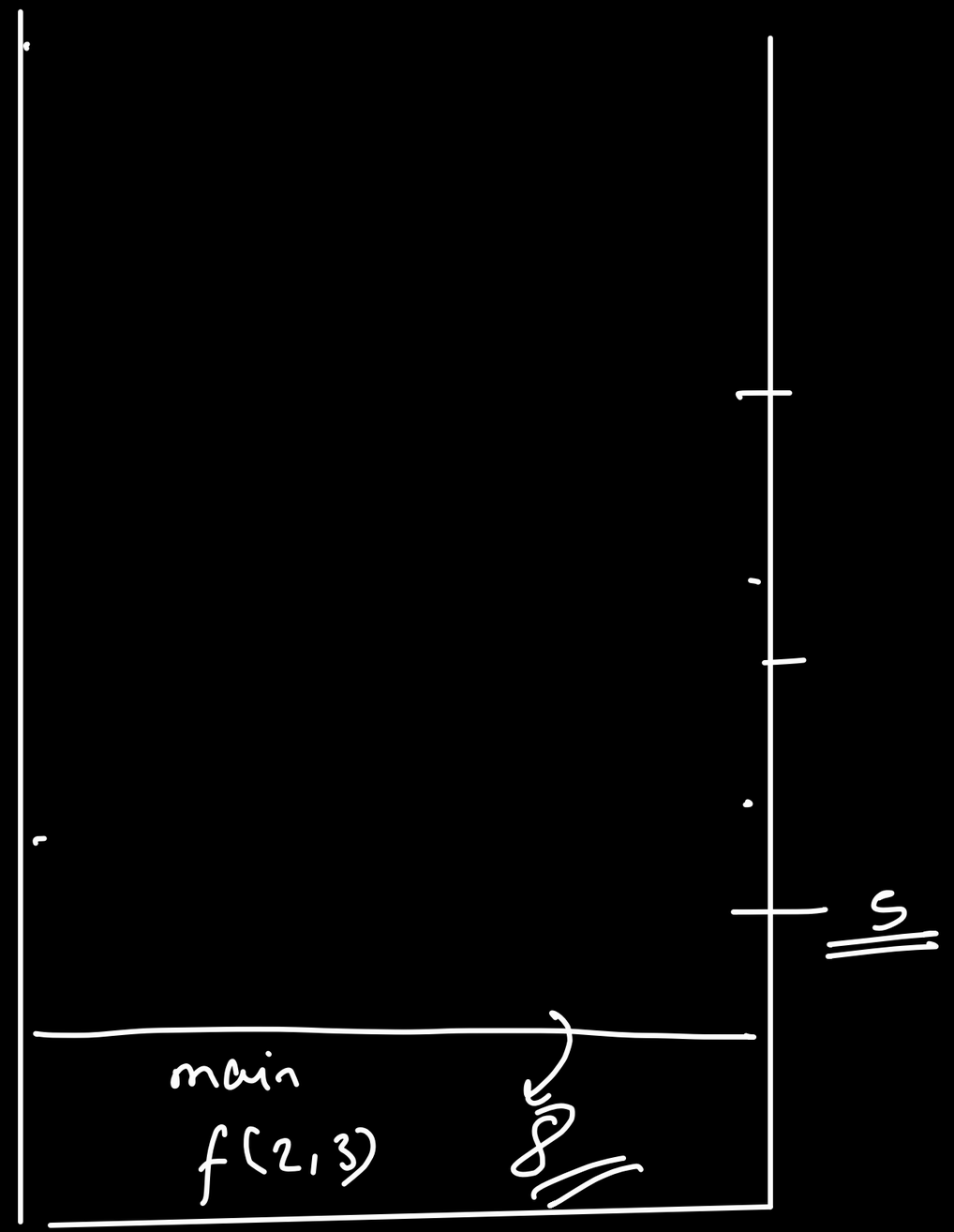
```c
int f(int a, int b) {
    if(b == 0) { // base case
        return 1;
    }
    return a * f(a, b-1);
}

```

main()

f(2,3)

Space → O(b)

Time → O(b)



$$2^8 = 2 \times 2^7$$

$$2^8 = 2^4 \times 2^4$$

$$2^9 = 2 \times \left(2^4 \times 2^4\right)$$

$$a^b = \begin{cases} a^{b/2} \times a^{b/2} & b \rightarrow \text{even} \\ a \times a^{b/2} \times a^{b/2} & b \rightarrow \text{odd} \end{cases}$$

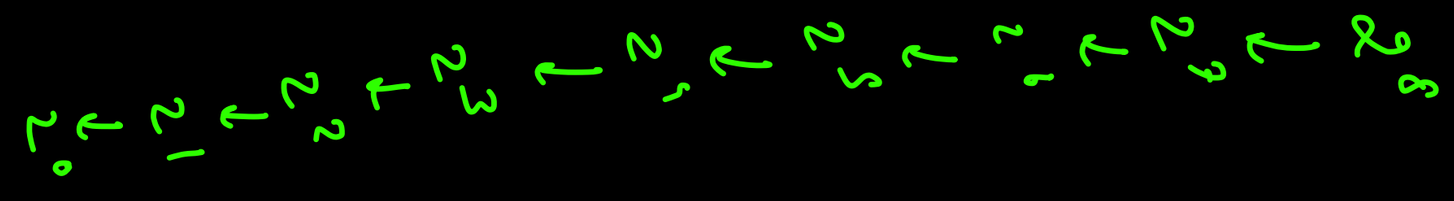
$$f(a,b) = \begin{cases} f(a,b/2) \times f(a,b/2) & \text{if } b \to \text{even} \\ f(a,b/2) \times f(a,b/2) \times a & \text{if } b \to \text{odd} \end{cases}$$

$f(a,b)$ returns a^b

$(b == 0) \to 1$

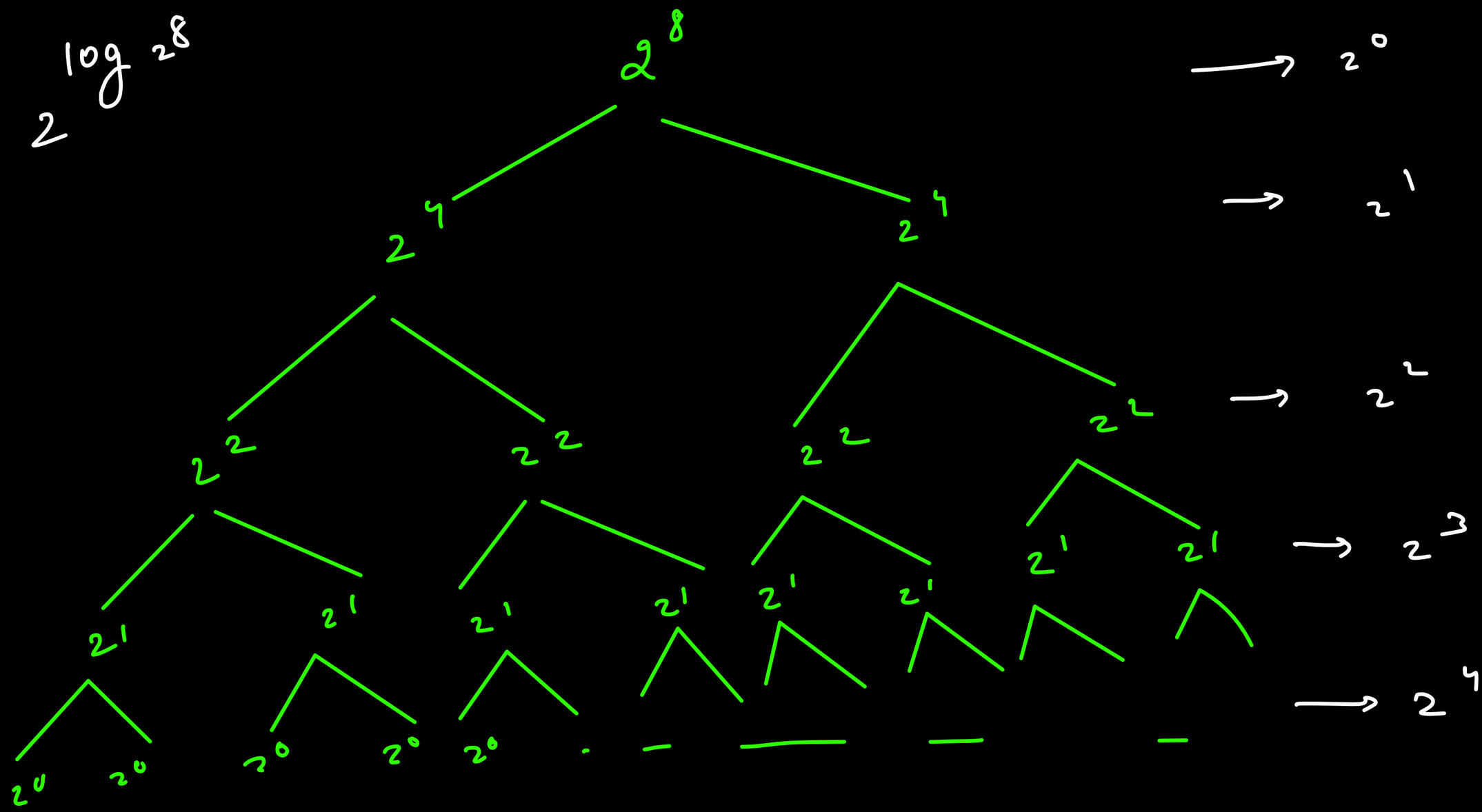
2

$O(b)$

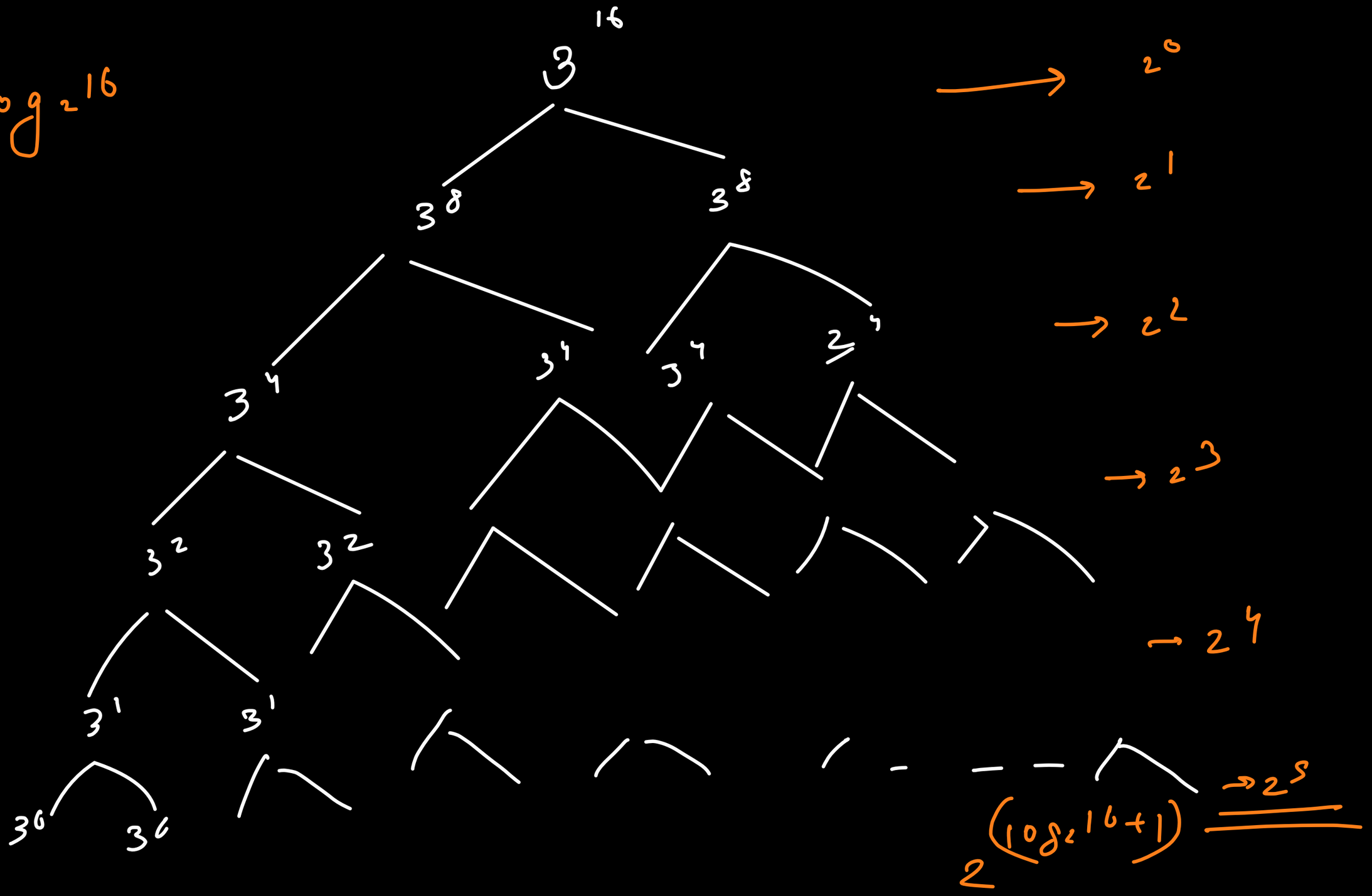


JOIN THE DARKSIDE

$2^{\log_2 8}$



$2^{\log_2 16}$



$f(a,b) \rightarrow 2^0 + 2^1 + 2^2 \cdots\cdots\cdots 2^{\log_2 b + 1}$

~~GP~~ $\rightarrow \underbrace{a \quad ar \quad ar^2 \quad ar^3 \cdots\cdots\cdots ar^{n-1}}$

$r > 1$ $\quad \frac{a \times (r^n - 1)}{r - 1}$

$\hookrightarrow \frac{1 \times (2^{\log_2 b + 2} - 1)}{2 - 1} \Rightarrow 2^{\log_2 b + 2} - 1$

$\hookrightarrow \approx 2^{\log_2 b} \rightarrow b$

$(a^{\log_a b}) \rightarrow b$

$\log_2 b + 2 \rightarrow \log_2 b + \log_2 4$

$\log_2 4b$

$2^{\log_2 4b}$

$$f(a,b) = \begin{cases} f(a,b/2)^2 & \text{if } b \to even \\ a \times f(a,b/2)^2 & \text{if } b \to odd \end{cases}$$

$K \approx \log_2 b$

K		
2^8	$\rightarrow$	1
$\downarrow$		
2^4	$\rightarrow$	1
$\downarrow$		
2^2	$\rightarrow$	1
$\downarrow$		
2^1	$\rightarrow$	1
$\downarrow$		
2^0	$\rightarrow$	1

$(1+1+1-------1) \rightarrow (\log_2 b) \rightarrow O(\log b)$

$2^8 \rightarrow 0$

$\downarrow$

$2^4 \rightarrow 1$

$\downarrow$

$2^2 \rightarrow 2$

$\vdots$

$2^8 \rightarrow 2^4 \rightarrow 2^2 \rightarrow 2^1 \rightarrow 2^0$ (8)

(K steps)

$8 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 0$

$\downarrow$

$2^3 \rightarrow 2^2 \rightarrow 2^1 \rightarrow 2^0 \rightarrow 0$

(1C)

$2^0 \longrightarrow 2^1 \longrightarrow 2^2 \cdots\cdots 2^k$

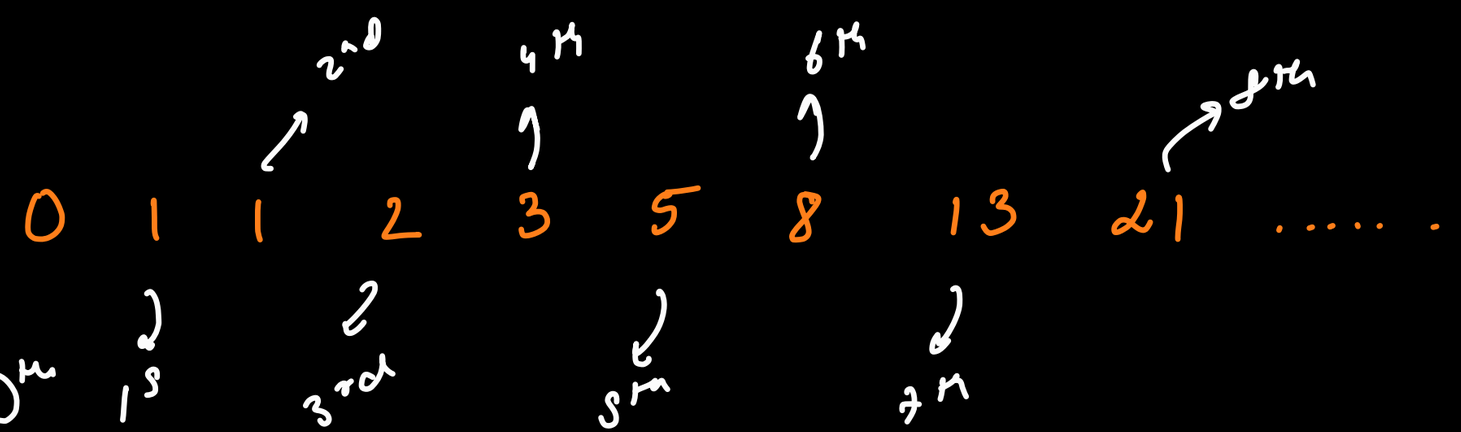
$2^k \approx b$

$\log_2 b \approx k$

Q→ Given a non-negative integer value n, Calculate the n^{th} fibonacci. (recursively)

Ex n = 5

ans → 5



n = 8

ans → 21

any i^{th} fib can be computed using the sum of $(i-1)^{th}$ & $(i-2)^{th}$ fib.

$f(n) = f(n-1) + f(n-2)$ → schework

$f(n)$ → n^{th} fibonacci

$f(n-1)$ → assume f returns $(n-1)^{th}$ fib correctly

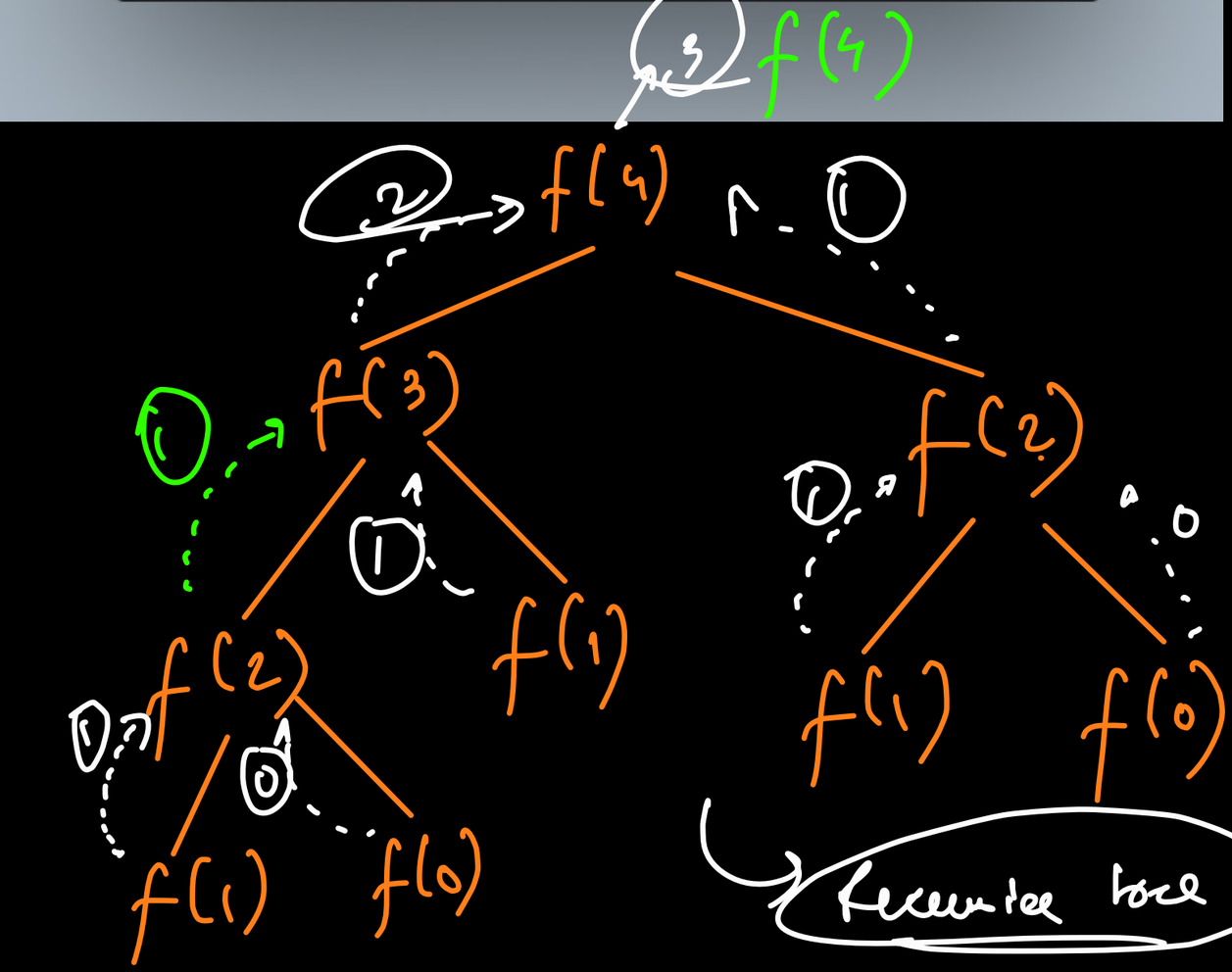
$f(n-2)$ → assume f returns $(n-2)^{th}$ fib correctly

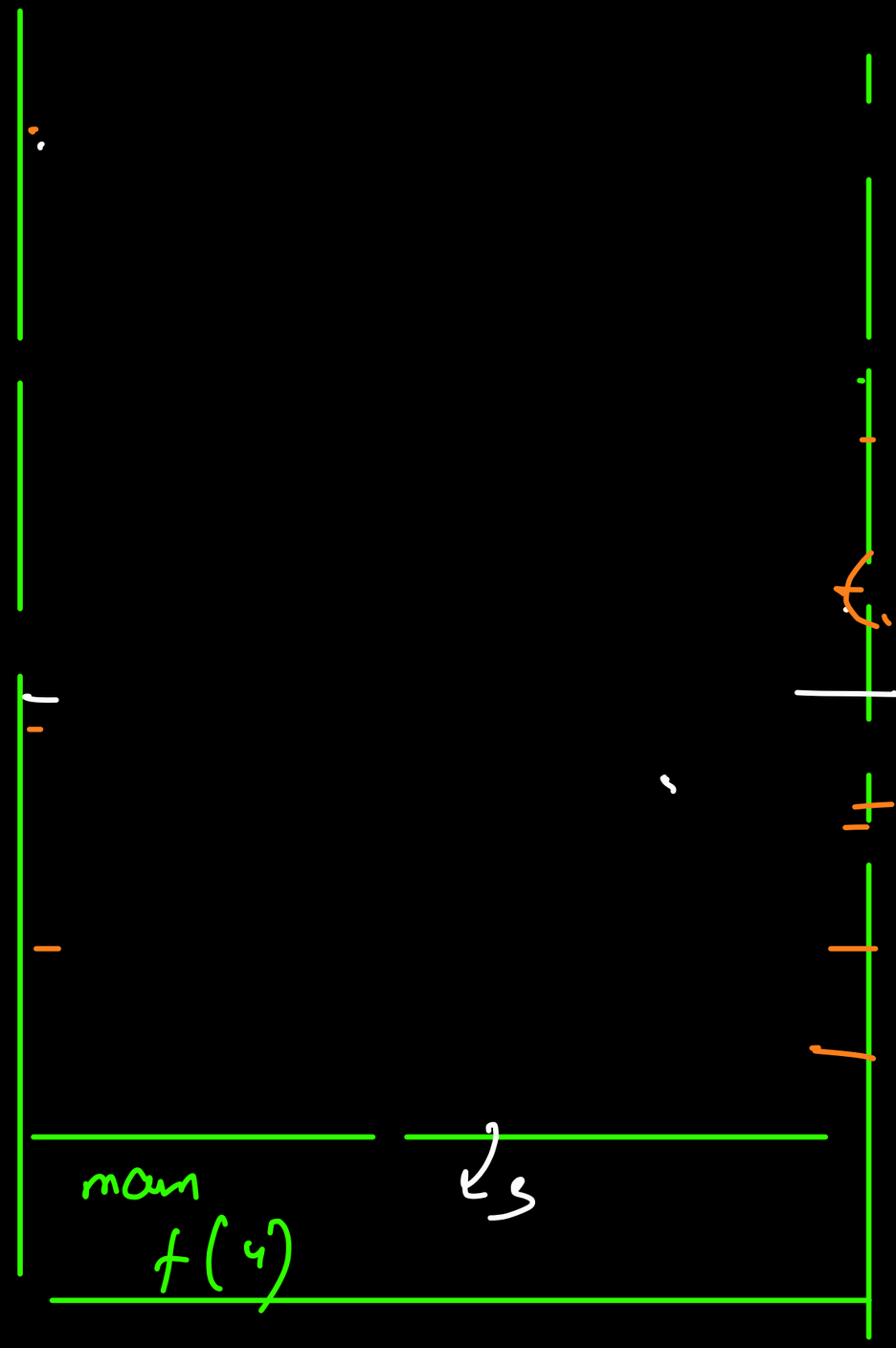
Base →

$f(0) \rightarrow 0$

$f(1) \rightarrow 1$

```
1  int f(int n) {
2      if(n == 0 or n == 1) return n;
3      return f(n-1) + f(n-2);
4  }
```





Q» Given a number n, print the first n natural no. in increasing order recursively.

Ex → n = 4

ans → 1
2
3
4

f(n) → f(n-1); print(n);

assume f works correctly for n-1 and print first n-1 natural no.

f(n) → print first n natural no. recursively

if (n → 0) ↓ don't do anything