[1, 2, 3, 1]

[2, 7, 9, 3, 1]

[2, 1, 1, 9]

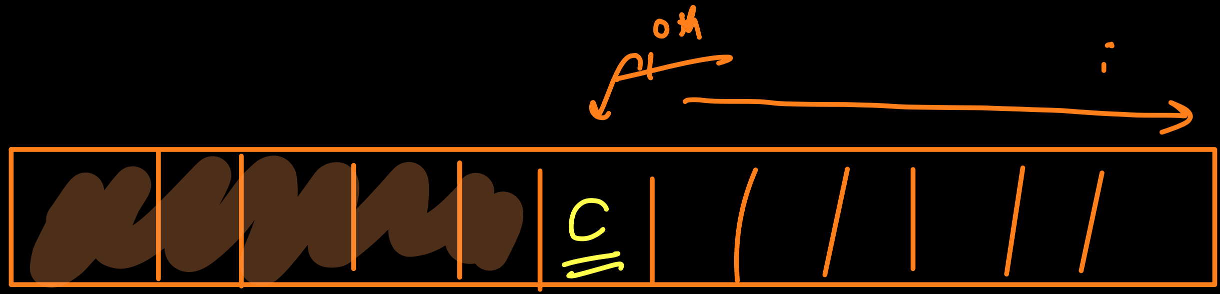
[1,	2,	3]	
✗	✗	✗	[{}]
✓	✗	✗	{1}
✗	✓	✗	{2}
✗	✗	✓	{3}
✓	✓	✗	[1, 2]
✓	✗	✓	{1, 3}
✗	✓	✓	{2, 3}
✓	✓	✓	{1, 2, 3}

→ we have a lot of ways to decide -

[1, 2, 3]

2^3

Brute force → try all possibilities of loot



loot

no loot

$c + (i+2 \to n-1)$

$0 + f(arr, i+1 \to n-1)$

added to profit

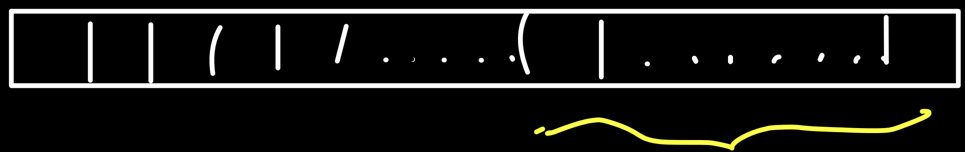
$$f(arr, i) = \max \begin{cases} arr[i] + f(arr, i+2) & \text{you decide to rob the ith house} \\ 0 + f(arr, i+1) & \text{you don't rob the ith house} \end{cases}$$

↓

this recursive func^n returns max profit by looting houses from index 'i' to 'n-1'. Such that no 2 adjacent houses are looted.

i

n-1 th house
is last one



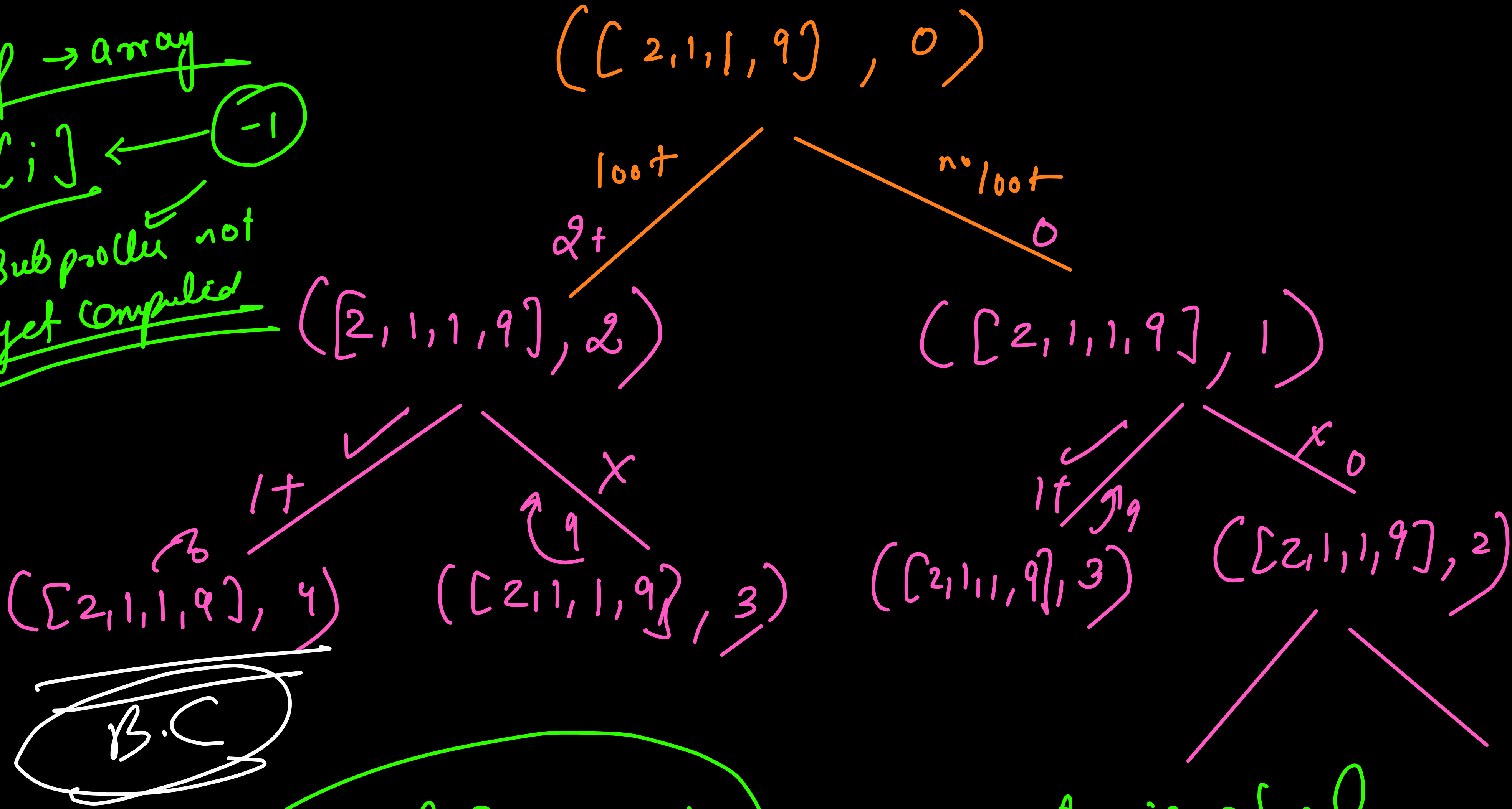
final ans → f(arr, 0) ← (call from main)

BC {
if (i == n-1)
→ single one
return arr[i];
if (i == n-2)
max(arr[i], arr[i+1]);
}

dp → array

dp[i] ← -1

subproblem not yet computed



dp[i] != -1 → ith state is already computed. Return it

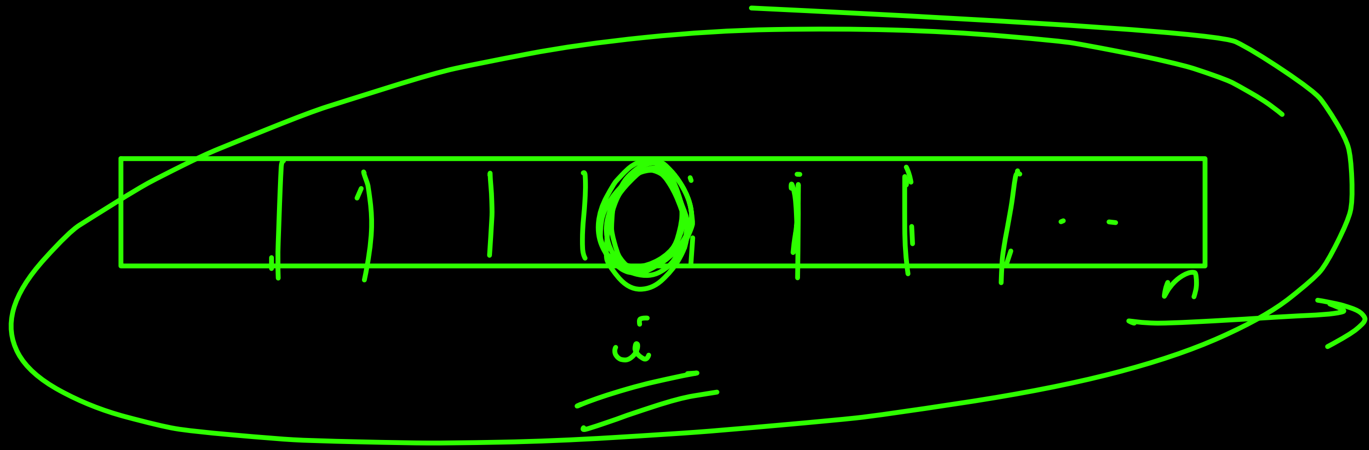
State of the dp → a set of all the parameters using which we can identify a subproblem uniquely.

how many unique subproblem will be there ??

↳ no. of subproblems depend on i. [0, n-1]

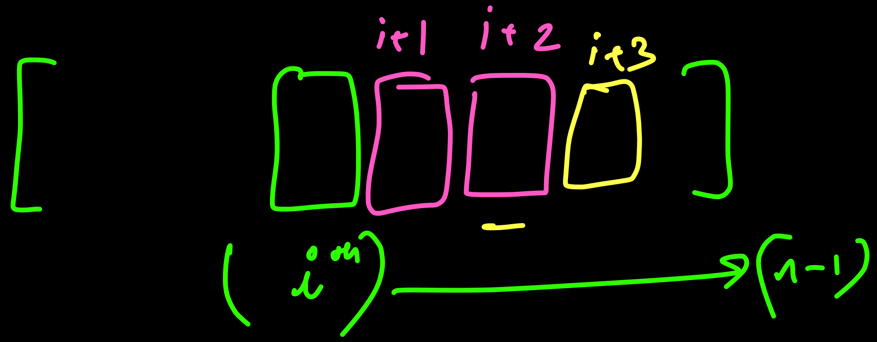
↳ Total n unique subproblems

1 variable → 1 Dimensional dp → 1d array


i
n

Bottom Up

iterative



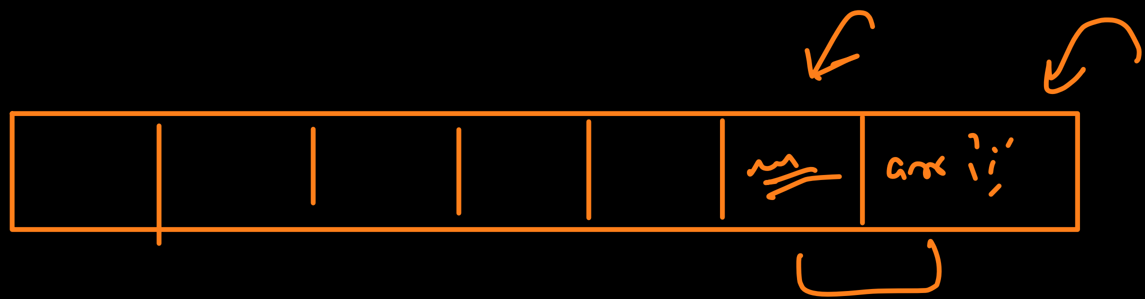
to calculate ans of any i^{th} state, we need to find ans of all the states $\geq i$

$$f(arr, i) = \max \begin{cases} f(arr, i+1) \\ arr[i] + f(arr, i+2) \end{cases}$$

$$\Downarrow$$

2

$$dp(i) = \max \begin{pmatrix} dp(i+1) \\ arr[i] + dp(i+2) \end{pmatrix}$$



[2, 7, 9, 3, 1]

$f(arr, i)$

$\downarrow$

profit for $[0, i]$

$n-2$ $\quad$ $n-1$

12	10	10	3	1

i

$$dp(i) = \max \begin{cases} dp(i+1), \\ dp(i+2) + arr[i] \end{cases}$$

$a = i+1$

$b = i+2$

$c = i$

$c = i-1$

final ans $\rightarrow$ $dp(0)$

$$f(arr, i) = \max \begin{cases} arr[i] + f(arr, i-2) \\ f(arr, i-1) \end{cases}$$

if $(i == 0)$ return $arr[0]$

if $(i == 1)$ $\max(\text{—})$

Q. Given a no. n, you can perform any of the following ops on it some no. of times.

1. → Reduce n to $n-1$
2. → if n is divisible by 2 to make it $n/2$
3. → if n is divisible by 3 make it $n/3$

Find out in how many minimum steps you can reduce n to 1.

Ex → $n = 7$

ans → 3

$$7 \xrightarrow{-1} 6 \xrightarrow{/2} 3 \xrightarrow{/3} 1$$

greedy will give a wrong ans.

n = 10

10
↓ /2
5
↓ -1
4
↓ /2
2
↓ /2
1

greedy → 4 steps

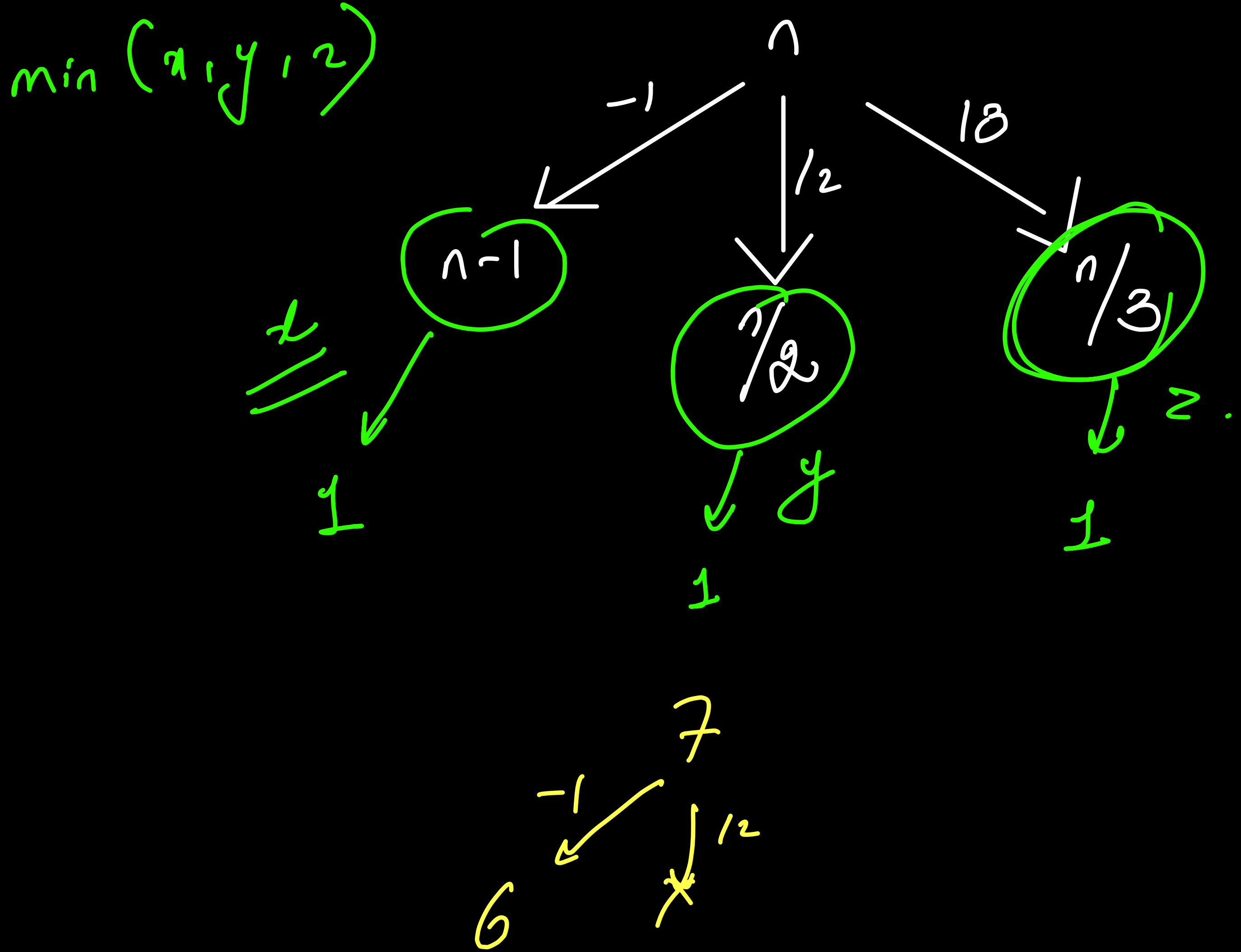
10
↓ -1
9
↓ /3
3
↓ /3
1

3 steps

↳ dp → **optimization technique**

dp

- **combinatorics**
  - **counting**
- **optimization**
  - **max**
  - **min**


min(x, y, z)
n
-1
/2
/3
n-1
n/2
n/3
x
y
z
1
1
1
7
-1
/2
6

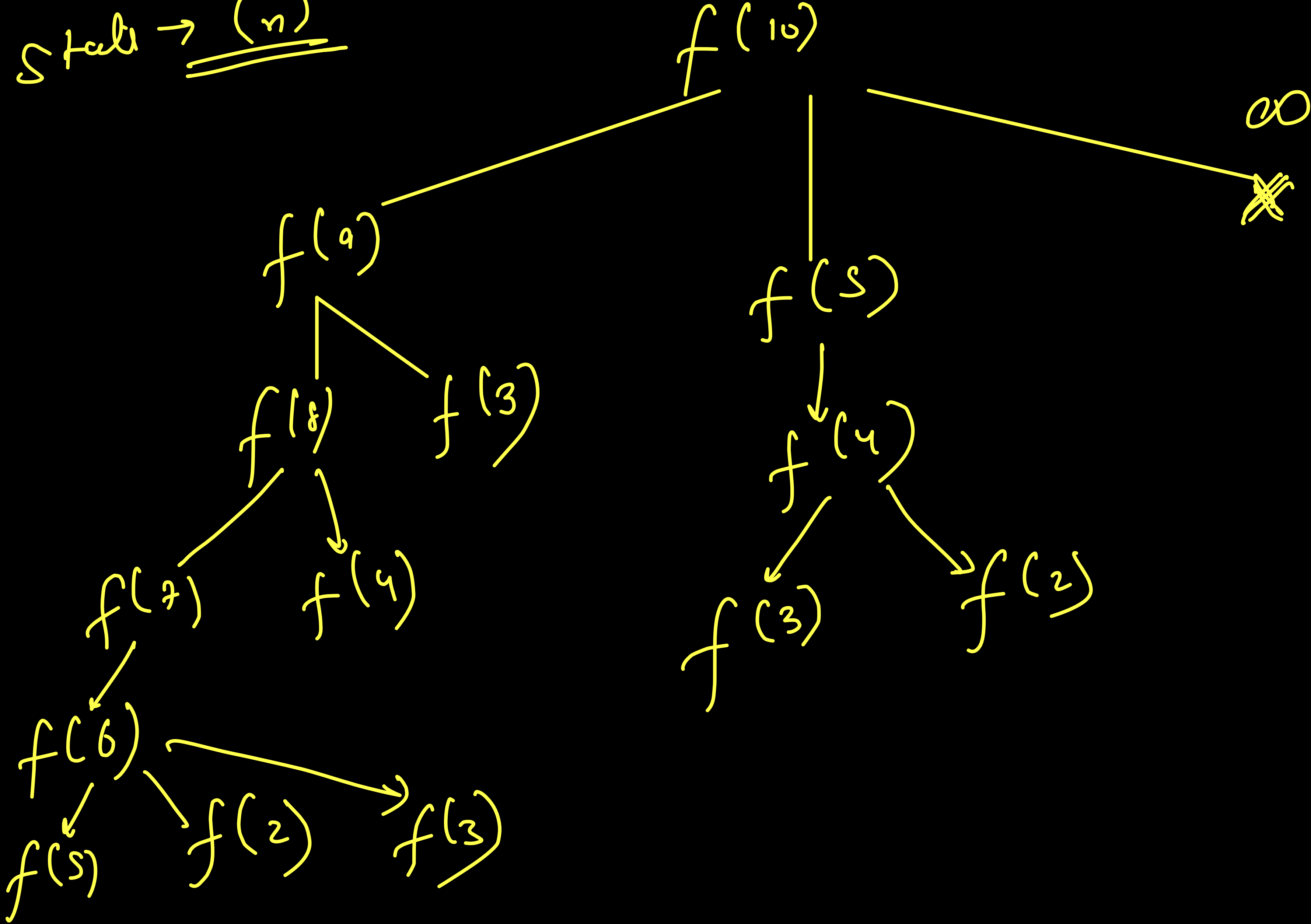
$$f(n) = 1 + \min \begin{cases} f(n-1) \\ (n\%2 == 0)\ ?\ f(n/2) : \infty \\ (n\%3 == 0)\ ?\ f(n/3) : \infty \end{cases}$$

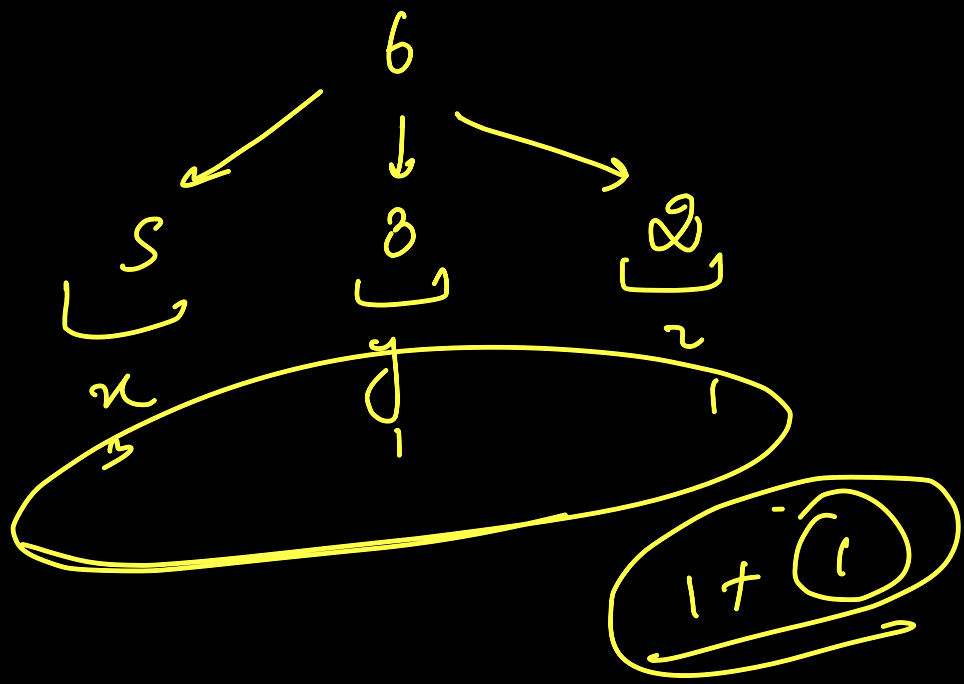
min steps to reduce n to 1.

Base Case →

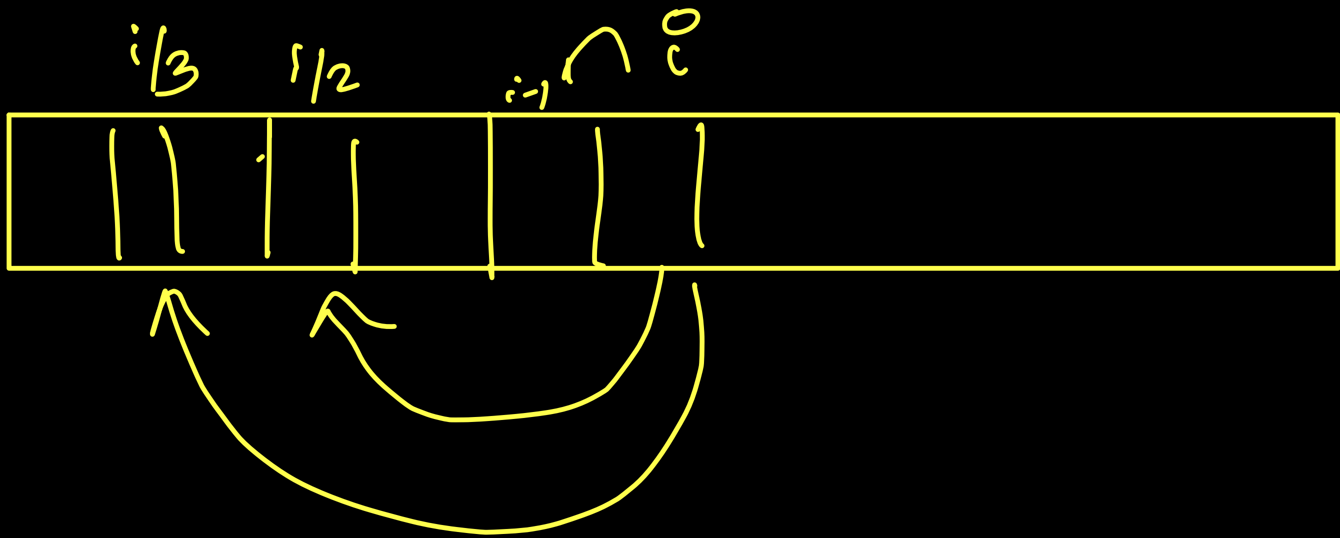
$$\left.\begin{array}{l} n == 1 \rightarrow 0 \\ n == 2 \rightarrow 1 \\ n == 3 \rightarrow 1 \end{array}\right\}$$

State → (n)




6
5
8
2
x
y
z
3
1
1
1 + 1

$$dp[i] = \min\left(dp(i-1),\ (i\%2==0)\ ?\ dp(i/2) : \infty,\ (i\%3==0)\ ?\ dp(i/3) : \infty\right)$$



0	1	2	3	4	5	6	7	8	9	10
X	0	1	1	2	3	2	3	3		

4

10

n+1