Q => Given a number n, print the first n natural numbers in increasing order recursively.

Ex → $n = 6$

Output → 1
2
3
4
5
6

$f(n) =$ ① $f(n-1)$
② print (n)

can print first n, natural numbers recursively

if ($n<1$)
return;

Base Case
↳ if you've $n<1$
↳ we don't need to proceed

Assumption → let's assume function f works correctly for $n-1$. i.e. $f(n-1)$ correctly prints the first $n-1$ natural numbers for us.

Selfwork → print n.

Q ⇒ Given a +ve integer value (> 0) n. Count the no. of Binary Strings (Strings which only got 0 or 1) of length n, such that there are no consecutive ones.

Ex → $n = 3$

↳ ans → 5 → (000, 001, 010, 100, 101)

$n = 1 \longrightarrow 2$

$n = 2 \longrightarrow 3$

$n = 3 \longrightarrow 5$

$n = 4 \longrightarrow 8$

$n = 5 \longrightarrow 13$

fibonacci

(0, 1)

(00, 01, 10)

(000, 001, 010, 100, 101)

(0000, 0001, 0010, 0100, 1000, 1001, 1010, 0101,

$$f(n) = f(n-1) + f(n-2)$$

base case

if (n == 1) return 2;

if (n == 2) return 3;

change

There are N stones, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), the height of Stone i is h_i.

There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

- If the frog is currently on Stone i, jump to Stone $i + 1$ or Stone $i + 2$. Here, a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

Ex → [10, 30, 40, 20]

ans → 30

frog

10	30	40	20
1	2	3	4

→ 4 stones

Ex → [10, 10]

ans → 0

$$\left.\begin{array}{l} \text{cost} \to |10-30| \to 20 \\ \text{cost} \to |30-20| \to 10 \end{array}\right\} \to 30$$

Ex → [30, 10, 60, 10, 60, 50]

ans → 40

$\boxed{f(i,n)}$

min cost

Frog

h_1	h_2	h_3	h_4	h_5	...	...	h_N	
1	2	3	4	5	...	...	N	N

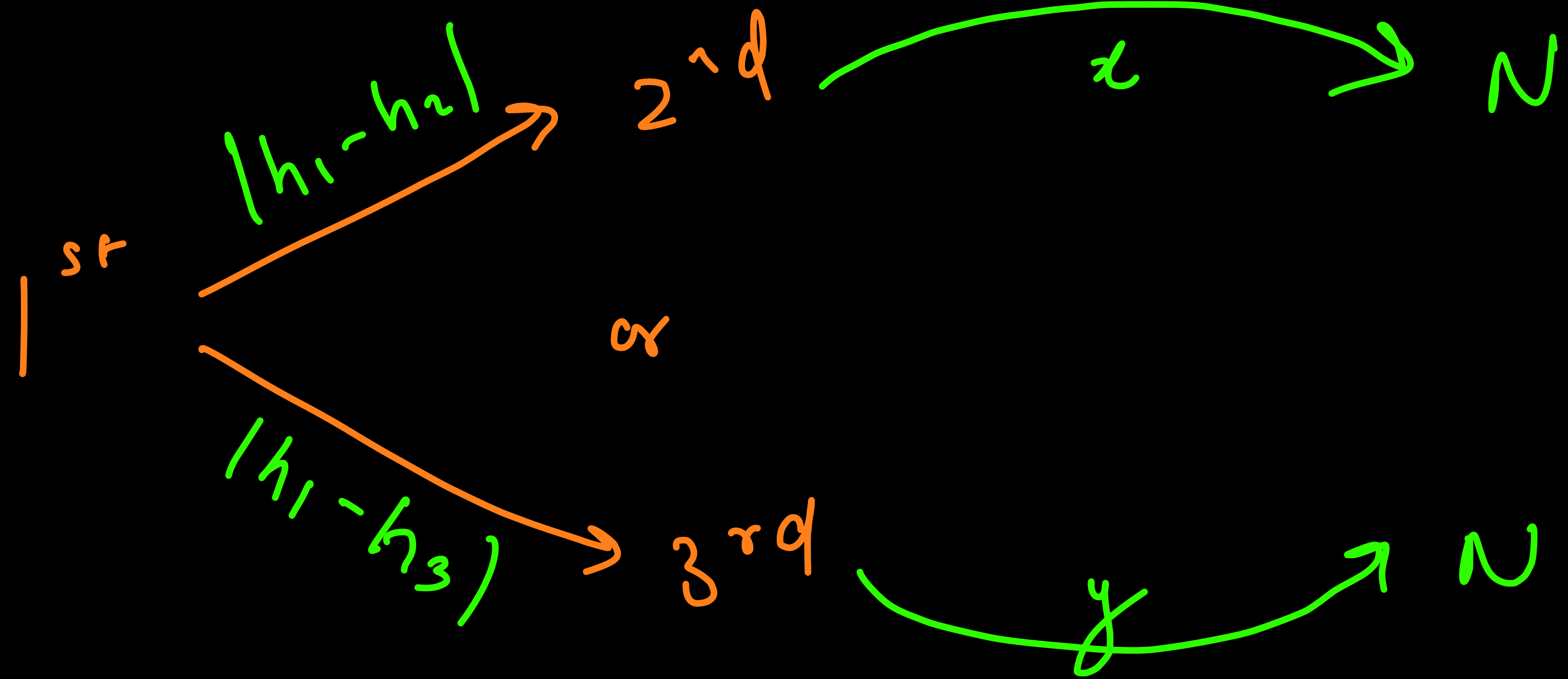
we will try to explore all possibilities

from 1^{st} stone, frog can jump either to the 2^{nd} or 3^{rd} stone.

$$1^{st} \to 2^{nd} \to |h_1 - h_2|$$

$$1^{st} \to 3^{rd} \to |h_1 - h_3|$$

if we somehow get the minimum cost to reach N^{th} stone from 2^{nd} stone (x) & the min cost to reach N^{th} stone from 3^{rd} stone (y).



$$\min\left(|h_1-h_2|+x,\ |h_1-h_3|+y\right) \rightarrow \text{ans}$$

$$f(i, n) = \min\Big(|h_i - h_{i+1}| + f(i+1, n),\ |h_i - h_{i+2}| + f(i+2, n)\Big)$$

$f(i, n)$ $\downarrow$ returns the min cost to reach n^{th} stone from i^{th} stone

$f(i+1, n)$ $\downarrow$ min cost to reach n^{th} stone from $(i+1)$

final ans $\rightarrow \underline{\Big(f(1, n)\Big)}$

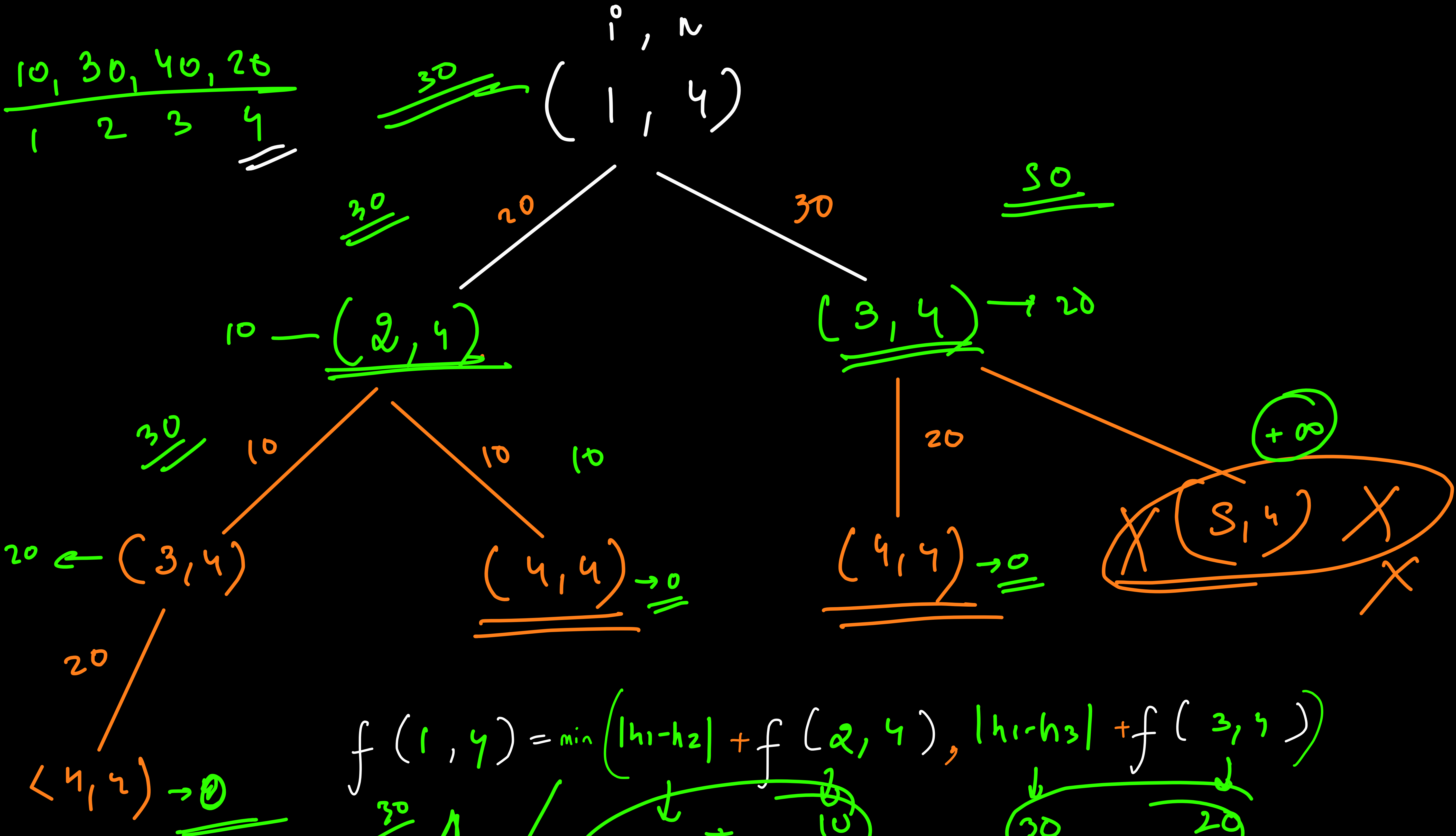
$$i^{th} \longrightarrow (i+1)^{th} \longrightarrow |h_i - h_{i+1}|$$

$$i^{th} \longrightarrow (i+2)^{th} \longrightarrow |h_i - h_{i+2}|$$

assumption → assume that function f works correctly for i+1 and i+2 i.e. func^n f correctly gives you min cost to reach N^{th} stone from $(i+1)^{th}$ stone & $(i+2)^{th}$ stone.

Selfwork → from i^{th} stone consider all possibilities and calculate min of it

10, 30, 40, 20
1 2 3 4



$$f(1,4) = \min\left(\underbrace{|h_1 - h_2| + f(2,4)}_{20 + 10 = 30},\ \underbrace{|h_1 - h_3| + f(3,4)}_{30 + 20 = 50}\right)$$

30

```
f(i, n) {
    if (i == n) return 0;
    if (i > n) return ∞;
    minCostViaiplus1 = |h[i] - h[i+1]| + f(i+1, n);
    minCostViaiplus2 = |h[i] - h[i+2]| + f(i+2, n);
    return min(minCostViaiplus1, minCostViaiplus);
}
```

There are N stones, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), the height of Stone i is h_i.

There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

- If the frog is currently on Stone i, jump to one of the following: Stone $i+1, i+2, \ldots, i+K$. Here, a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

Ex → $n = 5$, $k = 3$

[10, 30, 40, 50, 20]

ans → 30

$i \to (i+1)$

$i \to (i+2)$

Ex → $n = 3$ $k = 1$

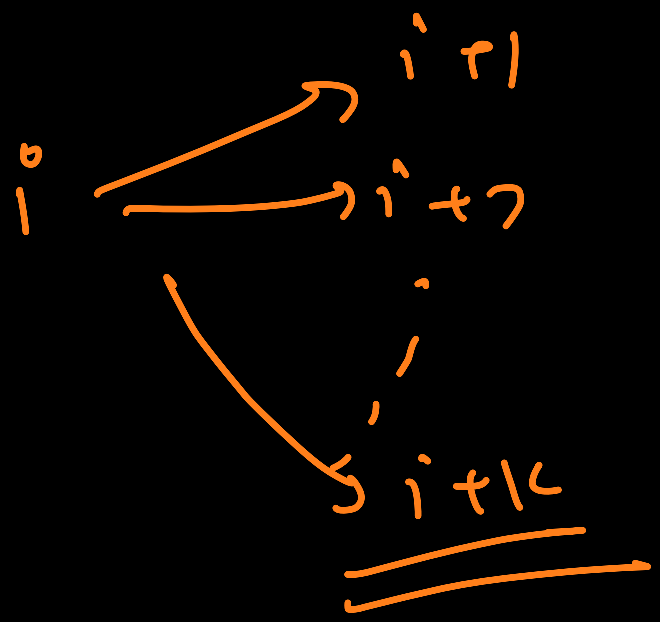
[10, 20, 10]

ans → 20

$$i \to i+1,\quad i \to i+2,\quad i \to i+3$$

$$i \to i+1,\quad i \to i+2,\quad i \to i+3,\quad i \to i+4$$

$$f(i,n) = \min\Big(|h_i - h_{i+1}|,\ f(i+1,n),\ |h_i - h_{i+2}|,\ f(i+2,n),\ |h_i - h_{i+3}|,\ f(i+3,n) \Big)$$



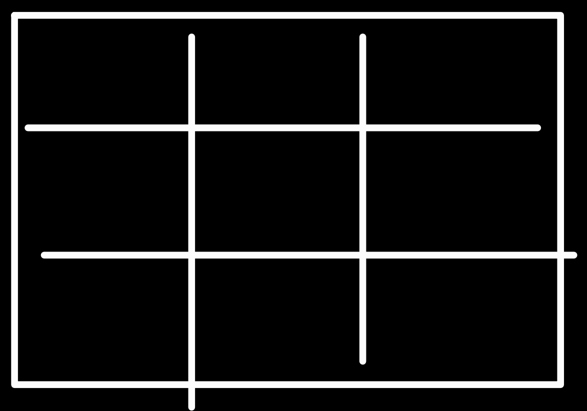
Base

$$f(i, n) = \min\left(|h_i - h_{i+j}| + f(i+j, n)\right)$$

$$\forall j \in [1, k]$$

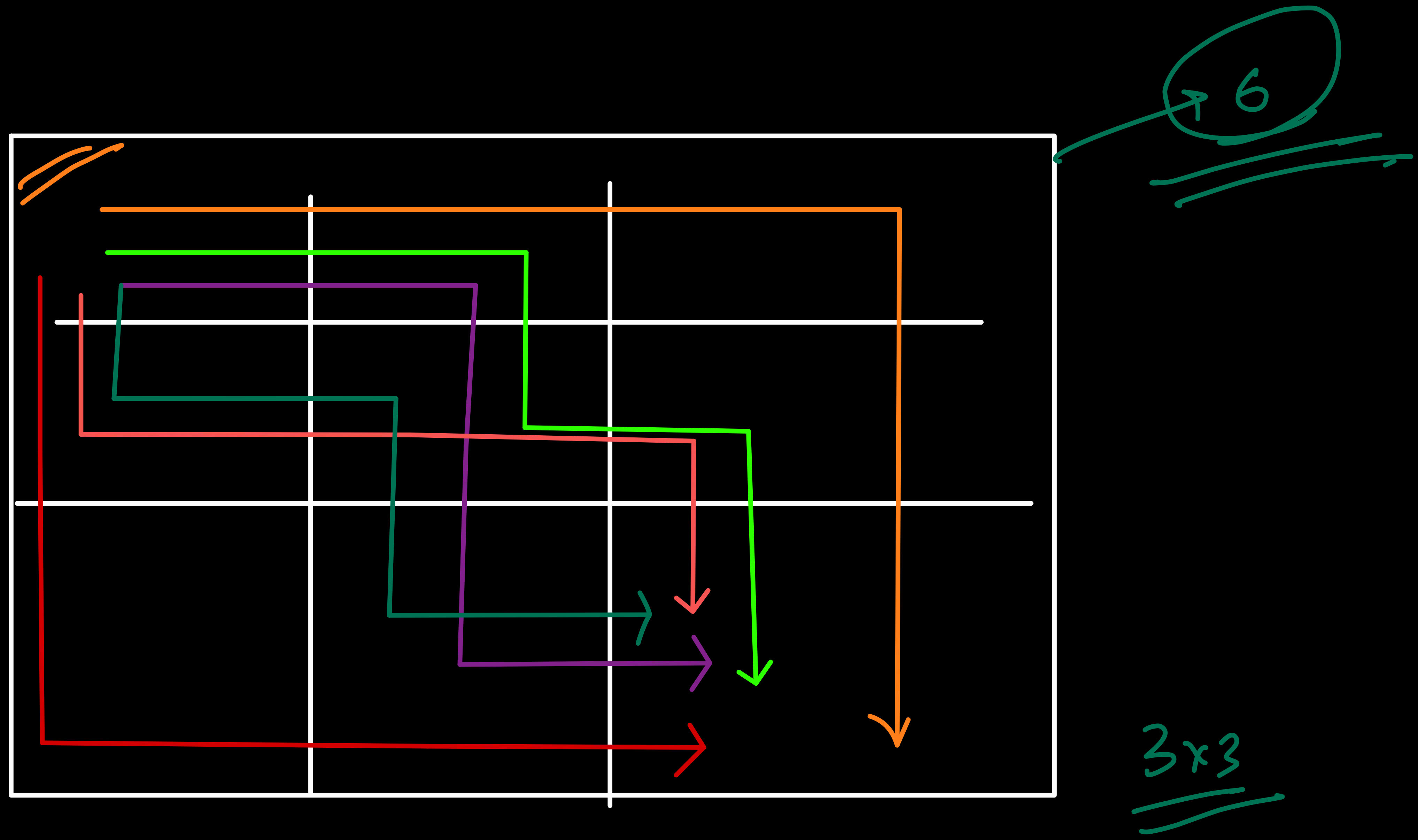
```
let result = Infinity
for (j = 1; j <= k; j++) {
    result = min( result, f(i+j, n) + |h[i] - h[i+j]|)
```

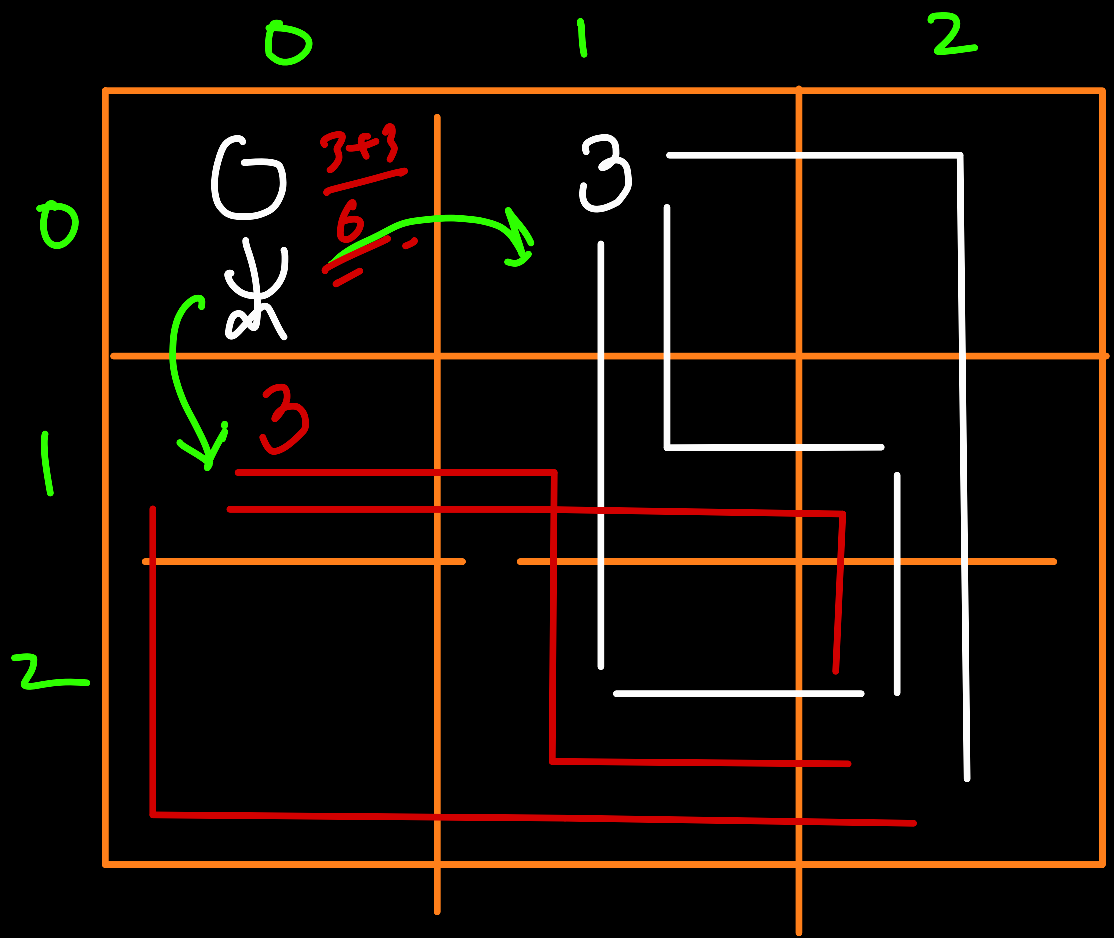
Q ⇒ Let's say, you are standing on the top left corner of a grid having dimension $n \times m$. find the total no. of ways in which you can reach the bottom right given the fact that from any cell of the grid you can move one step down or one step **right**.

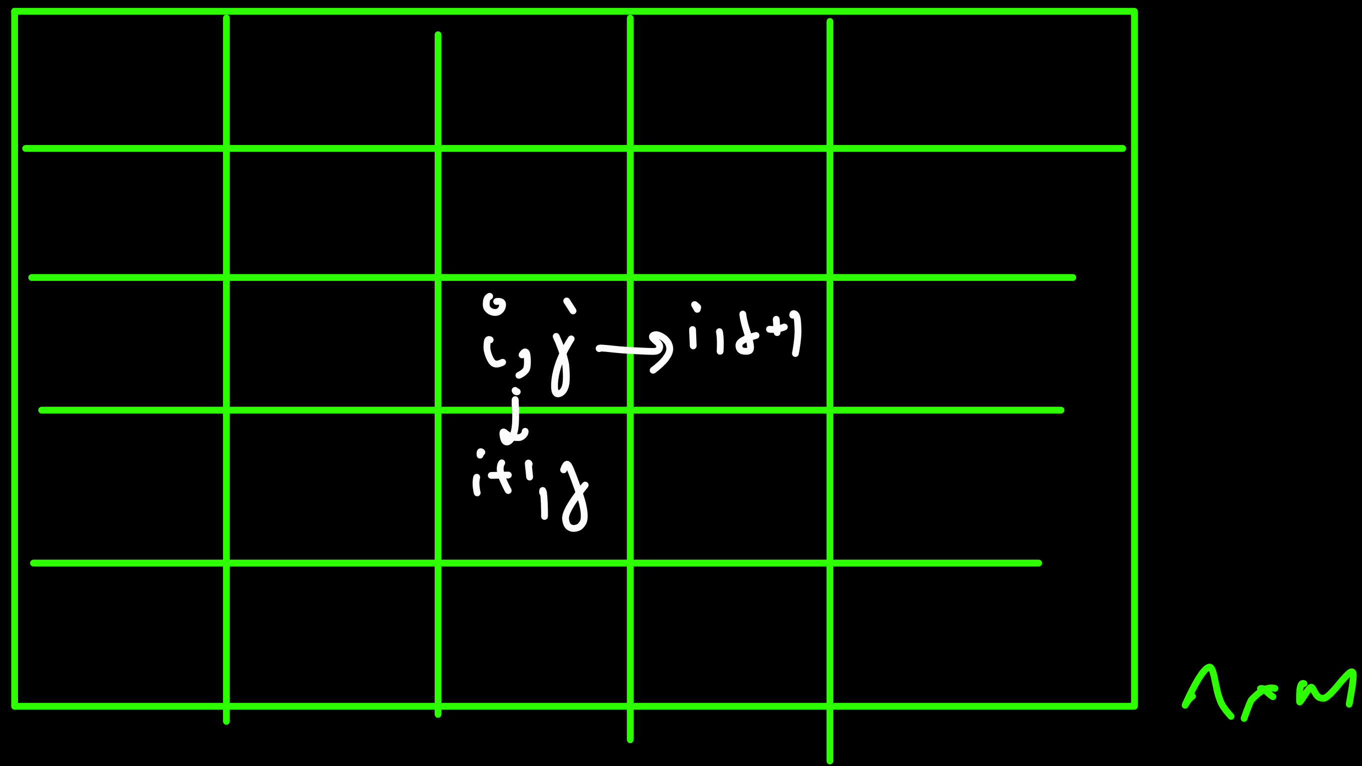


3×3

$\longrightarrow \underline{6}$


6
3x3


0
1
2
0
1
2
6
3+3
6
3
3


i, j
i, j+1
i+1, j
n×m

$f(i, j, n, m) = f(i, j+1, n, m) + f(i+1, j, n, m)$

self work

no. of ways to reach n-1, m-1 from any cell i, j

assumption

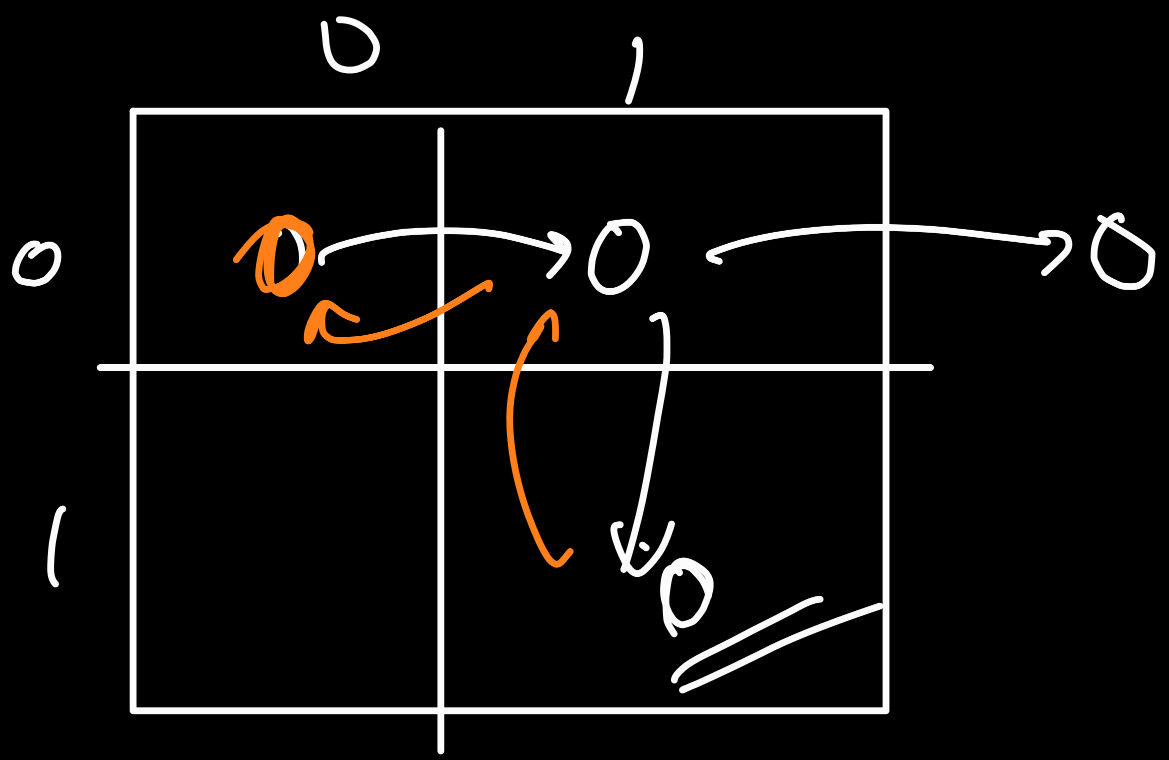
if (i == n-1 and j == m-1)
return 1;
if (i >= n or j >= m)
return 0;

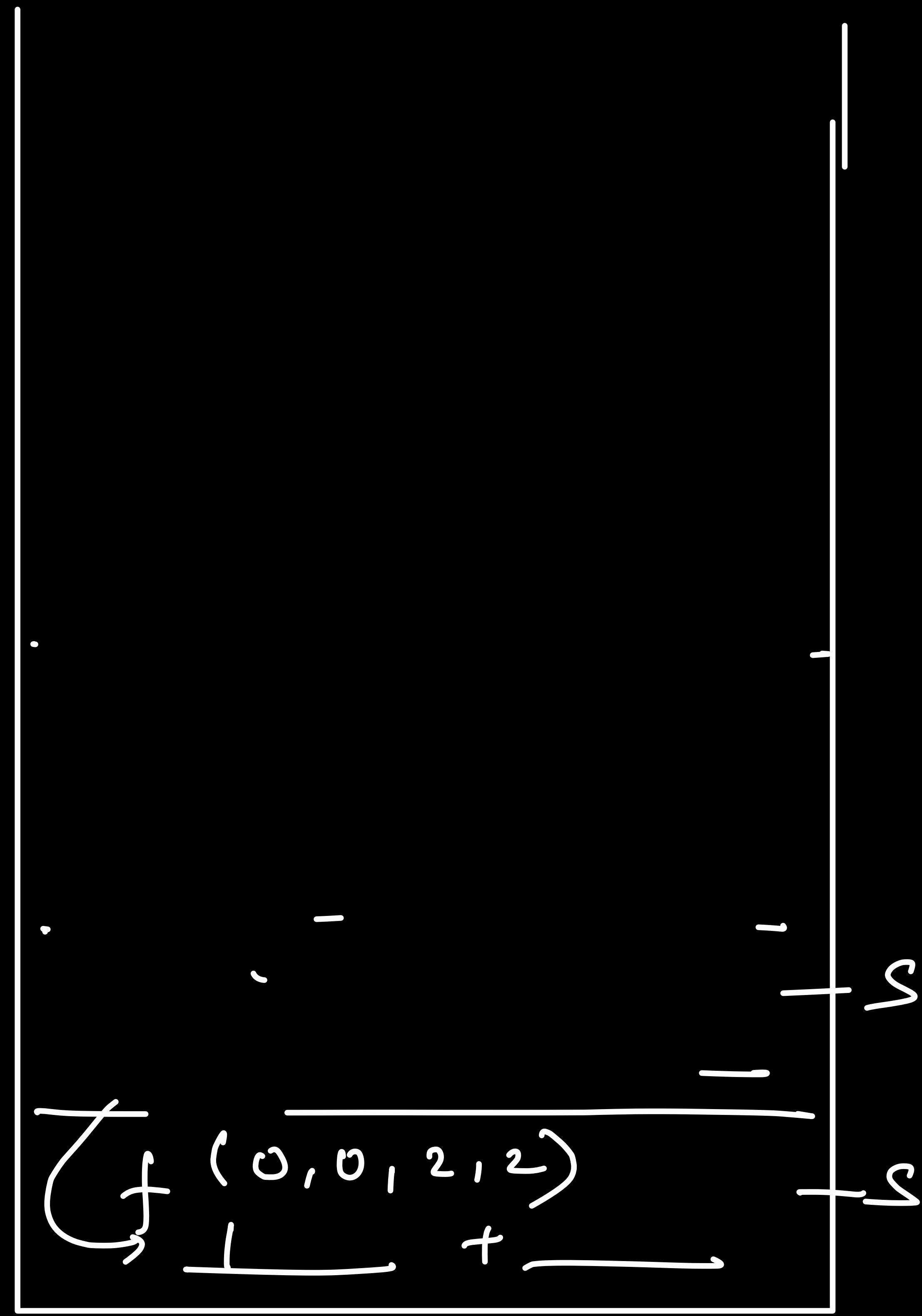
actual ans → $f(0, 0, n, m)$

```
function f(i, j, n, m) {
    if(i == n-1 && j == m-1) return 1;
    if(i >= n || j >= m) return 0;

    return f(i, j+1, n, m) + f(i+1, j, n, m);
}

console.log(f(0, 0, 2,2));
```





Q ⇒ Given a positive non zero number n, calculate the min no. of steps to reduce n to 1. For this reduction we can do the following operations on n,

1. if n is divisible by 3, divide by 3
2. if n is divisible by 2, divide by 2
3. Subtract 1 from n.

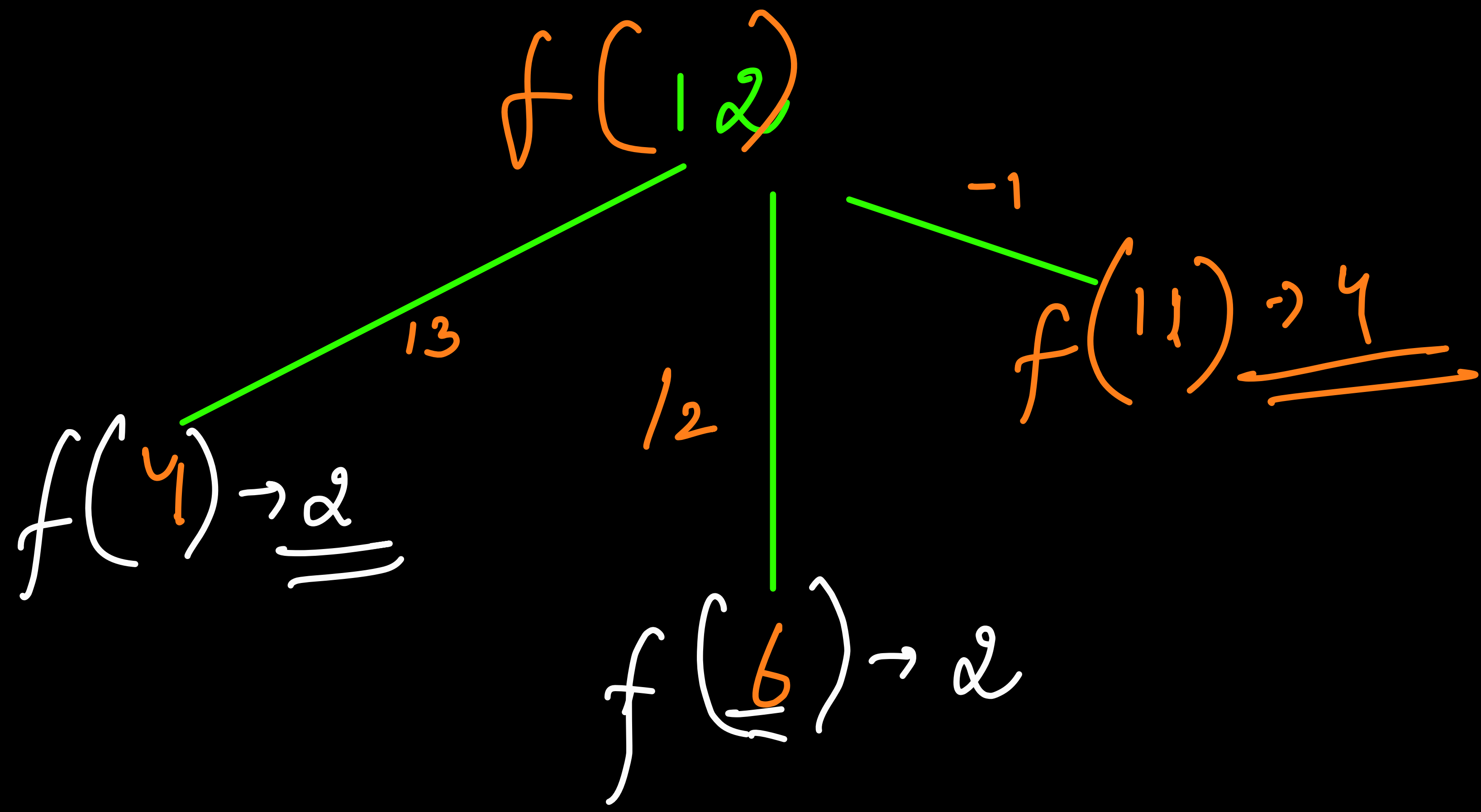
Ex → n = 10 ⟶ ans → 3

$10 \xrightarrow{/2} 5 \xrightarrow{-1} 4 \xrightarrow{/2} 2 \xrightarrow{/2} 1$ ⟶ 4 steps

$10 \xrightarrow{-1} 9 \xrightarrow{/3} 3 \xrightarrow{/3} 1$ ⟶ 3 steps

We try all possibilities

$\min(2, 2, 4) \rightarrow 2$



11
↓
10
↓
9
↓
3
↓
1

$4 \xrightarrow{/2} 2 \xrightarrow{/2} 1$

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

f(4-

$$f(n) = 1 + \min \begin{cases} f(n/3) \rightarrow \text{if } (n \% 3 == 0) \\ f(n/2) \rightarrow \text{if } (n \% 2 == 0) \\ f(n-1) \end{cases}$$

$f(n)$ → min no. of steps to reduce n to 1

if $(n == 1) \rightarrow 0$

if $(n < 1) \rightarrow \infty$

$n \rightarrow 22$

$\underbrace{1, 2, 3, 4, 5, 6 \dots\dots\dots 22}$

[B, C, D, E, F, G, H, I, J, BA, BB, BC,

BD, BE ......]

0 – A

1 – B

2 – C

3 – D

4 – E

5 – F

6 – G

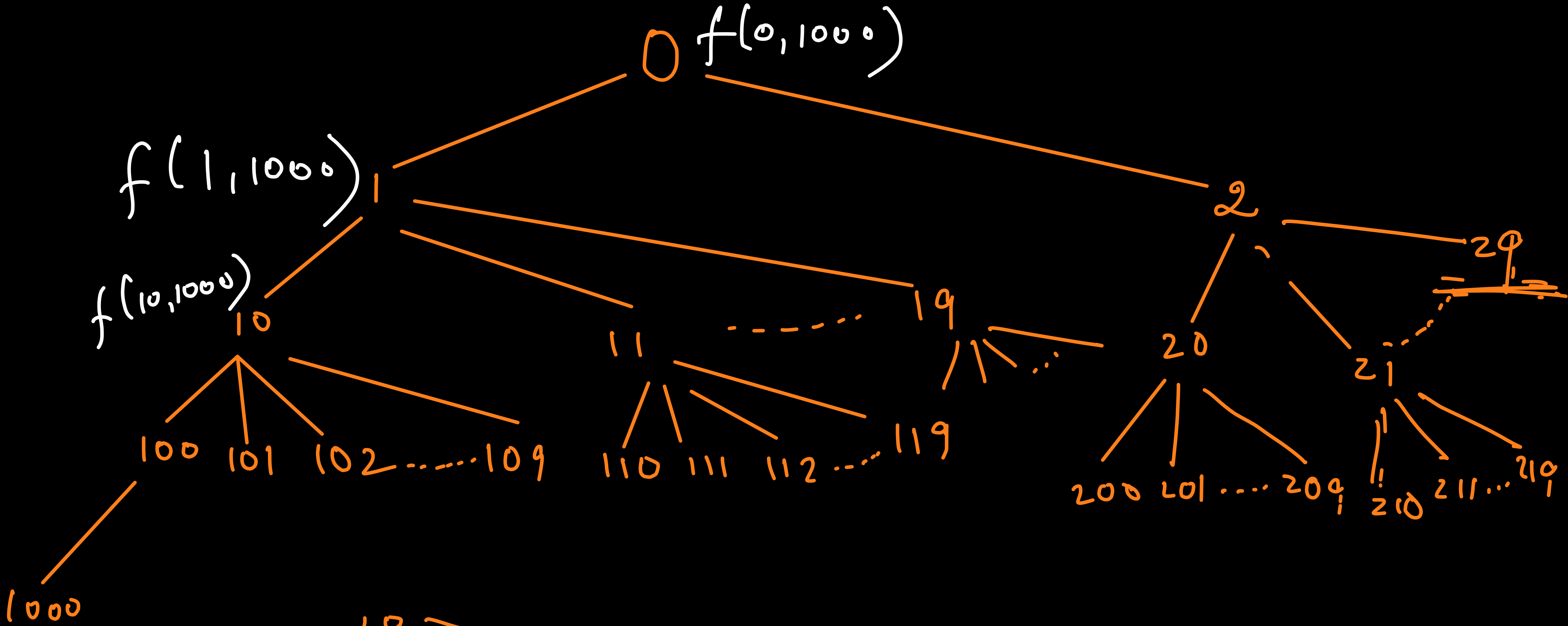
7 – H

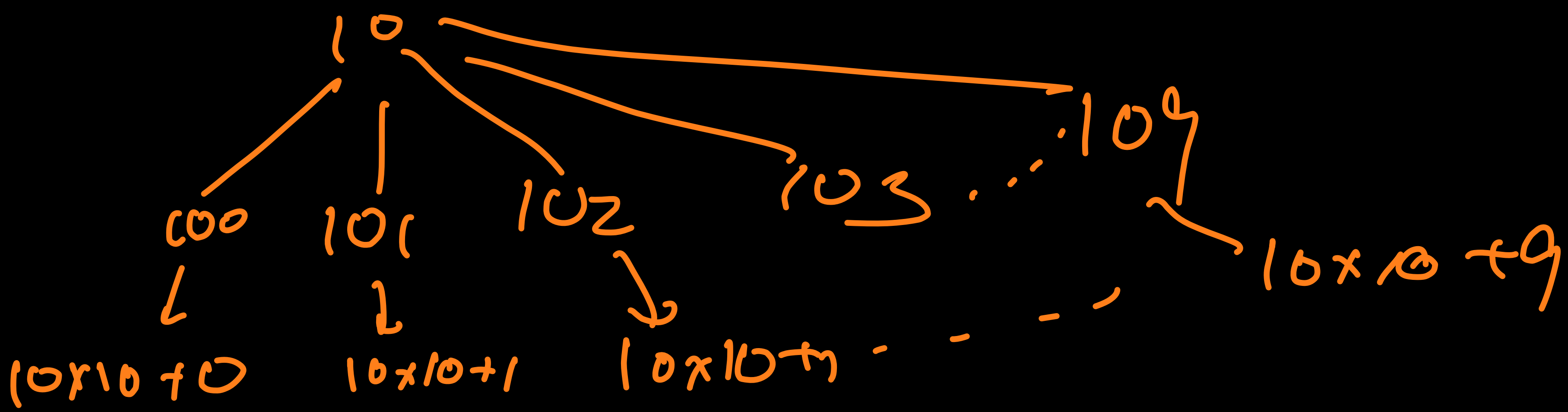
8 – I

9 – J

$n = 1000$

1
10
100
1000
101
102
103
⋮
109
11
110
111
112
⋮


f(0,1000)
0
f(1,1000)
1
2
f(10,1000)
10
11
19
20
21
29
100
101
102
109
110
111
112
119
200
201
209
210
211
219
1000


10
100
101
102
103
109
10x10+0
10x10+1
10x10+m
10x10+9

$f(i, n) \Rightarrow$ print(i) $\rightarrow$ selfcall

$f(10 * i + j, n)$

It prints all the no. in the range $[i, n]$ in lexico order $\rightarrow$ the no starts with i

$\forall j \in [0, 9]$

if $i == 0$ then $\forall j \in [1, 9]$

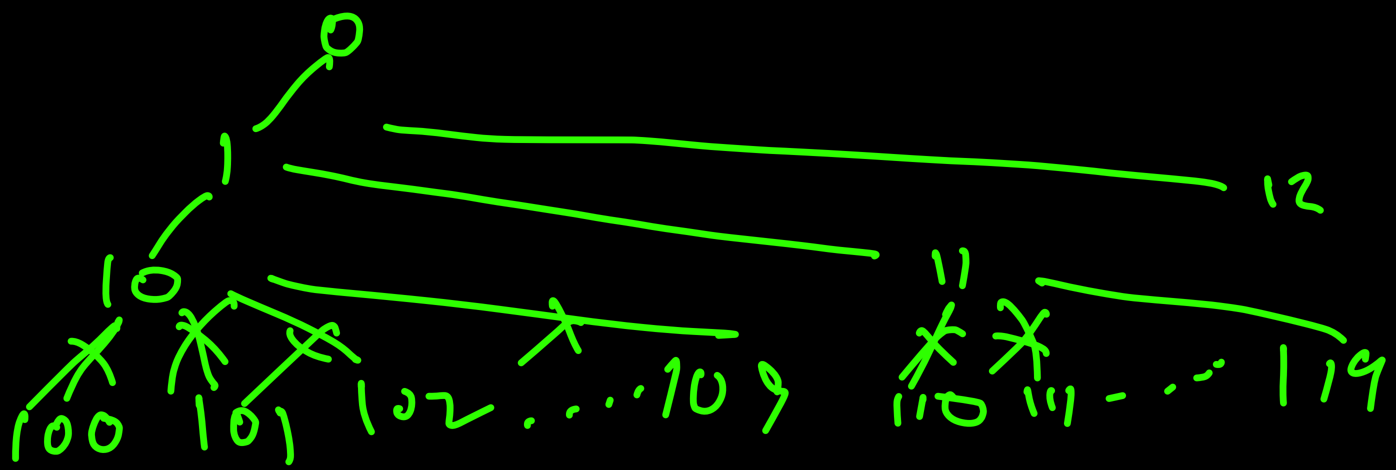
ans $\rightarrow$ $f(0, n)$

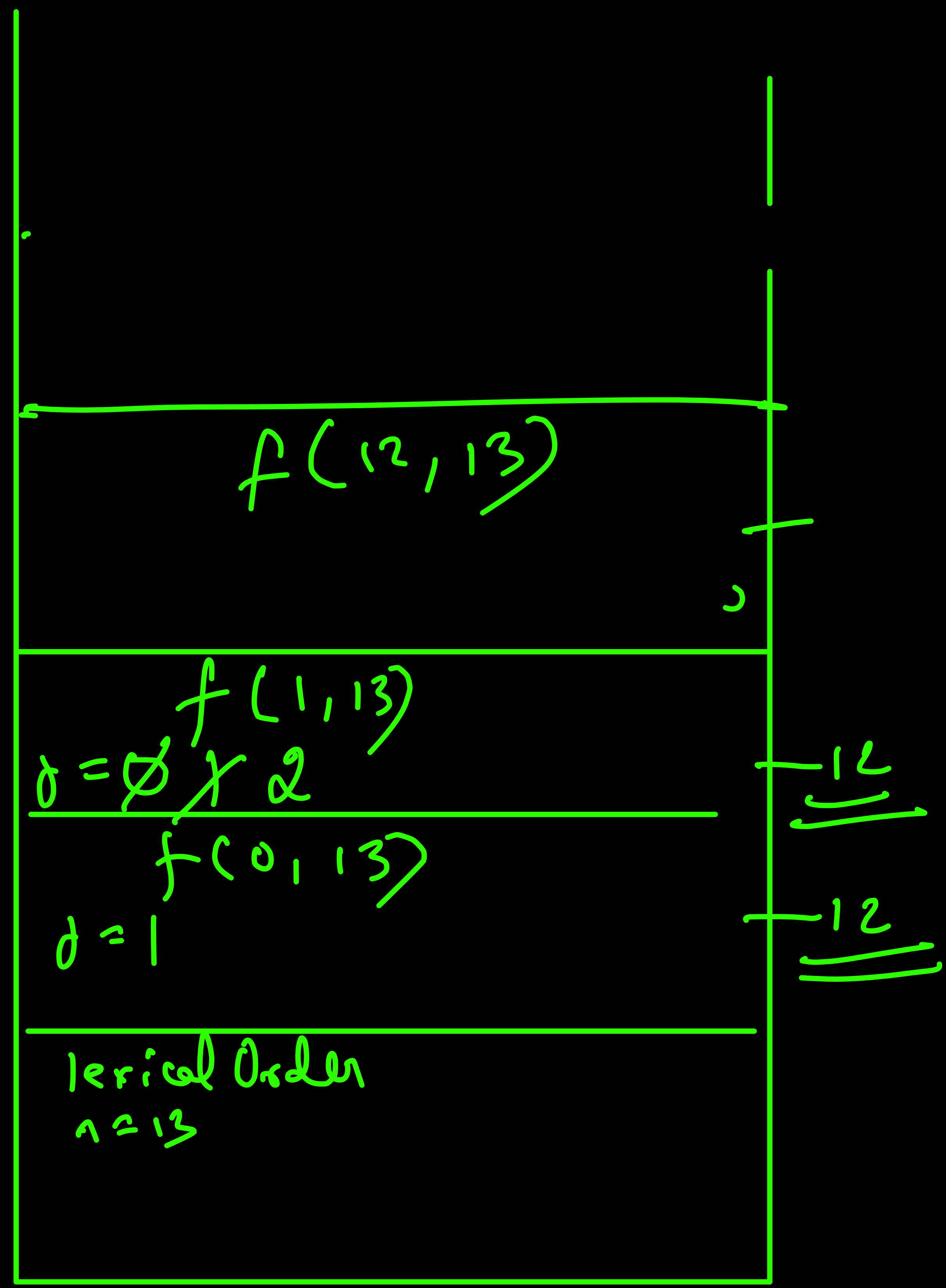
```
let arr;
function f(i, n) {
    if(i > n) return;
    if(i != 0) {
        arr.push(i);
    }
    for(let j = ((i == 0) ? 1 : 0); j <= 9; j++) {
        f(10*i + j, n);
    }
}

var lexicalOrder = function(n) {
    arr = [];
    f(0, n);
    return arr;
};
```

n = 13

arr = [1, 10, 11, 12 ...]





Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money x using the available coins in such a way that the number of coins is minimal.

For example, if the coins are $\{1, 5, 7\}$ and the desired sum is 11, an optimal solution is $5 + 5 + 1$ which requires 3 coins.

Input

The first input line has two integers n and x: the number of coins and the desired sum of money.

The second line has n distinct integers $c_1, c_2, \dots, c_n$: the value of each coin.

Output

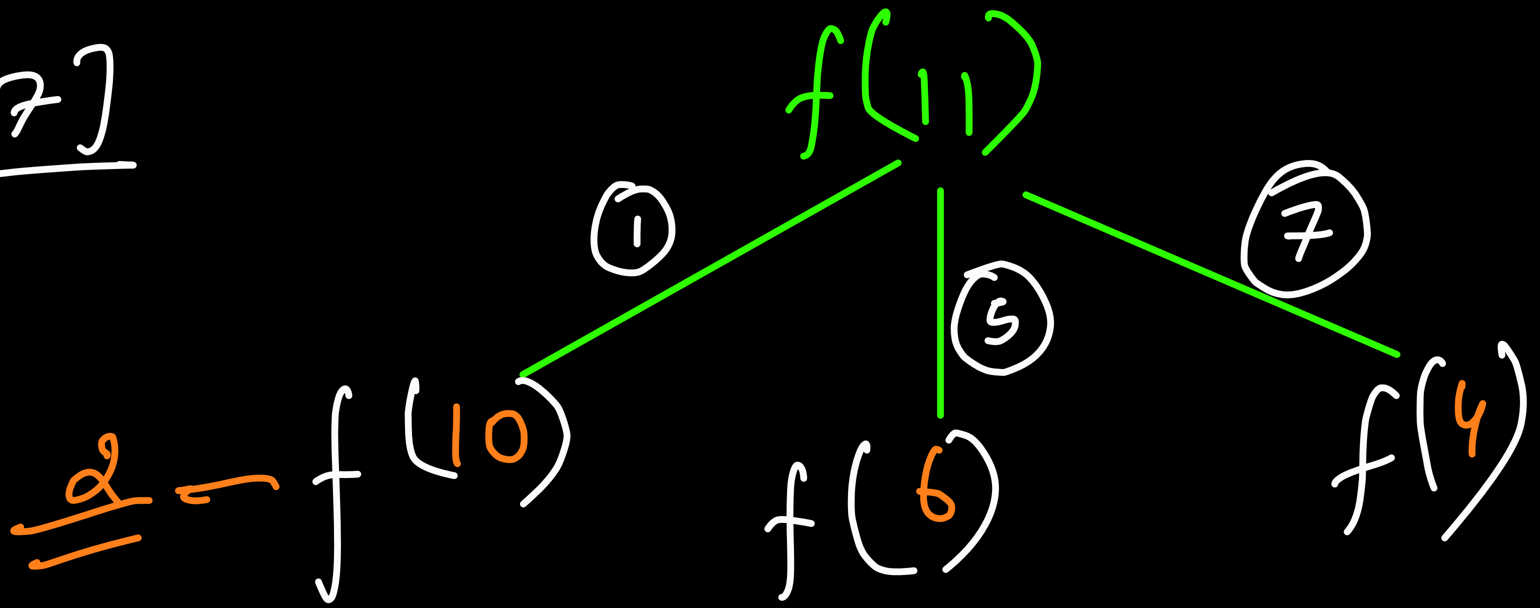
Print one integer: the minimum number of coins. If it is not possible to produce the desired sum, print -1.

Ex → $n = 3$ $x = 11$

$[1, 5, 7]$

ans → $\underline{\underline{3}}$

[1, 5, 7]



$$f(11) \to 1 + \min\left(f(10),\ f(6),\ f(4)\right)$$

$$f(coins, x) = 1 + \min\left(f(coins, x - coins[i])\right)$$

min coins req'd to get x.

$\forall i \in [0, coins.length - 1]$

$x = 11$, $[1, 5, 7]$

$$f([1,5,7], 11) = 1 + \min\Big(f([1,5,7], 11-1), f([1,5,7], 11-5),$$
$$f([1,5,7], 11-7)\Big)$$

$f([1,5,2],6)$