Given n friends, each one can remain single or can be paired up with some other friend. Each friend can be paired only once. Find out the total number of ways in which friends can remain single or can be paired up.

$n = 3$ A B C

1st way → (A) (B) (C)

2nd way → (A B) (C)

3rd way → (A) (B C)

4th way → (A C) (B)

Eg

```
Input  : n = 3
Output : 4
Explanation:
{1}, {2}, {3} : all single
{1}, {2, 3} : 2 and 3 paired but 1 is single.
{1, 2}, {3} : 1 and 2 are paired but 3 is single.
{1, 3}, {2} : 1 and 3 are paired but 2 is single.
Note that {1, 2} and {2, 1} are considered same.
```

$n = 4$

A B C D

(A) (B) (C) (D)

(AB) (C) (D)

(AC) (B) (D)

(BC) (A) (D)

(AD) (B) (C)

(AD) (BC)

(BC) (AD)

(B) (A) (DC)

(A) (C) (DB)

⋮

$\Rightarrow$ $n=1$ A $\rightarrow$ 1 way

$\Rightarrow$ $n=2$ A B $\rightarrow$ 2 ways

(A) (B)

(AB)

$\rightarrow$ $n=3$ A B C $\rightarrow$ 4 ways

(A) (B) (C)

(AB) (C)

(BC) (A)

(AC) (B)

$\rightarrow$ $n=4$ A B C D $\rightarrow$ 10 ways

A B C

let's take decision on a single friend & try to assume recursively for the remaining

1 choice will be to go single

2nd choice, to go in a pair

we also need to consider no. of ways to make a pair

Case 1 → You want to go Single.

A B C → single

↳ if C decides to go Single, then in all the possible ways A, B can go, we can attach C to it.

2 ways $\begin{cases} (A)(B)\ (C) \\ (A B)\ (C) \end{cases}$

$\underline{f(4)}$ A B C D $\rightarrow$ $\underline{\text{single}}$

4 ways

$$\left\{\begin{array}{llll} (A) & (B) & (C) & (D) \\ (AB) & (C) & (D) & \\ (AC) & (B) & (D) & \\ (BC) & (A) & (D) & \end{array}\right\} \underline{f(3)}$$

No. of ways in which n^{th} friend goes single is equal to no. of ways the remaining friends go to party.

Case 2

n^{th} friend decides to go in a **pair**

A B C D → pair

A B C

$f(2) \times 3 \Rightarrow 2 \times 3$
$\rightarrow 6$

Here D can make 3 diff pairs, so we need to have all possibilities → 6 ways

p_1 → (D A) (B) (C)
(D A) (B C)
2

p_2 → (D B) (A) (C)
(D B) (A C)
2

p_3 → (D C) (A B)
(D C) (A) (B)
2

after making a pair, D lock another friend & we are left with n-2 friends.

no. of ways in which n^{th} friend makes a pair

$$f(n-2) \times (\text{no. of ways to make a pair})$$

$$f(n-2) \times (n-1)$$

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

total no. of ways in which n friends go to party.

if n^{th} friends decide to go single, then we just need no. of ways for remaining friends.

if n^{th} friend makes a pair

Base → $n == 1 \rightarrow 1$

$n == 2 \rightarrow 2$

Problem Statement:

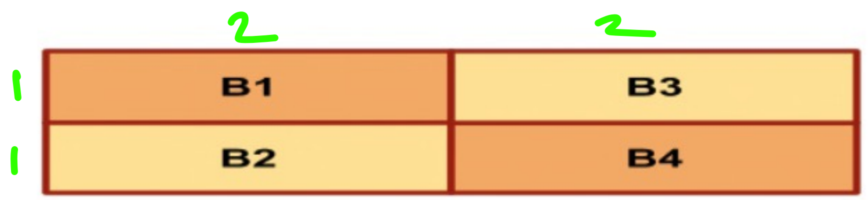
Consider a board of size ***2 * N*** and tiles of size "***2 * 1***". You have to count *the number of ways in which tiling of this board is possible*. You may place the tile vertically or horizontally, as per your choice.

1. ***N = 4***

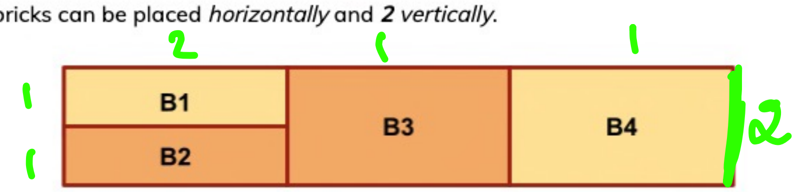
The tiling of this board is possible in ***3*** different ways. Let us see how:

- All ***4*** bricks can be placed *vertically.*

- All ***4*** bricks can be placed *horizontally.*



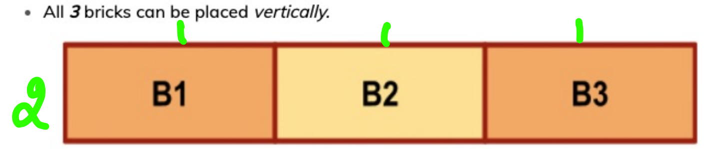
- ***2*** bricks can be placed *horizontally* and ***2*** *vertically.*



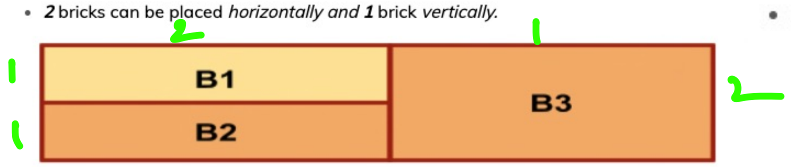
2. ***N = 3***

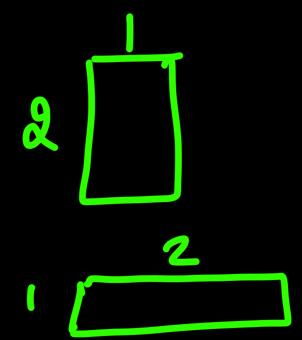
The tiling of this board is possible in ***2*** different ways. Let us see how:

- All ***3*** bricks can be placed *vertically.*



- ***2*** bricks can be placed *horizontally and* ***1*** brick *vertically.*





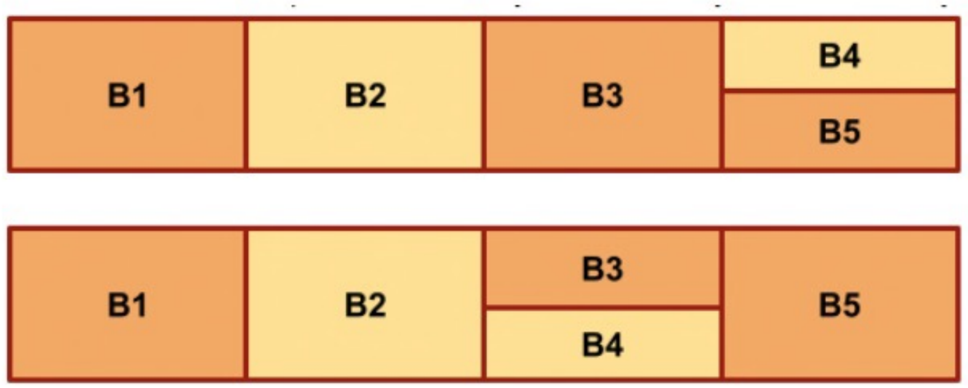
3. ***N = 5***

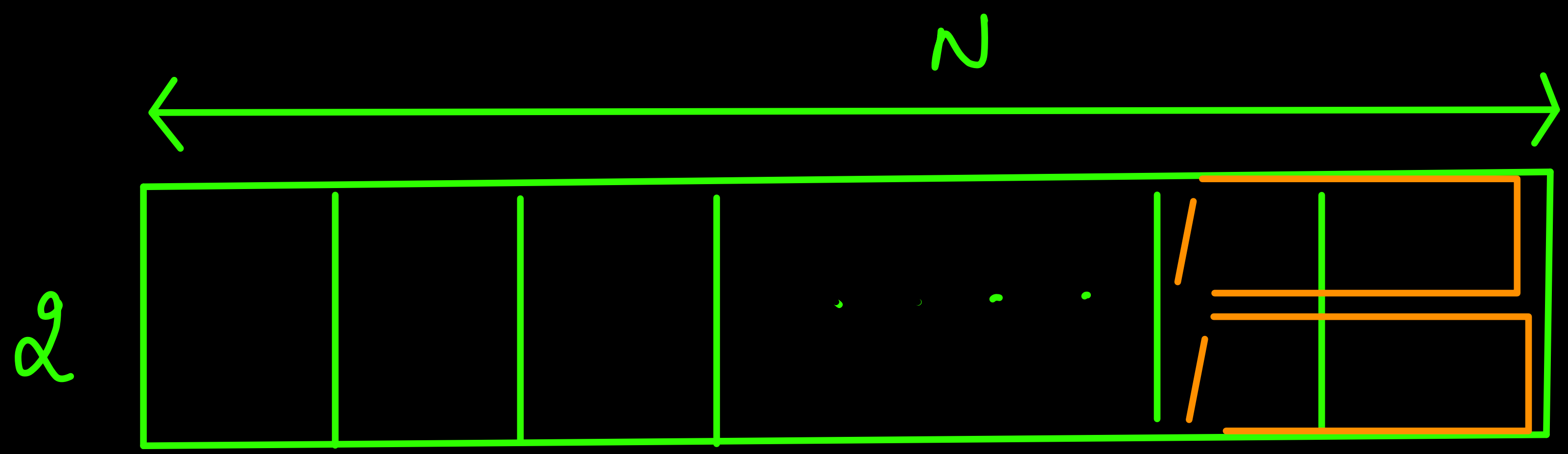
The tiling of this board is possible in ***3*** different ways. Let us see how:

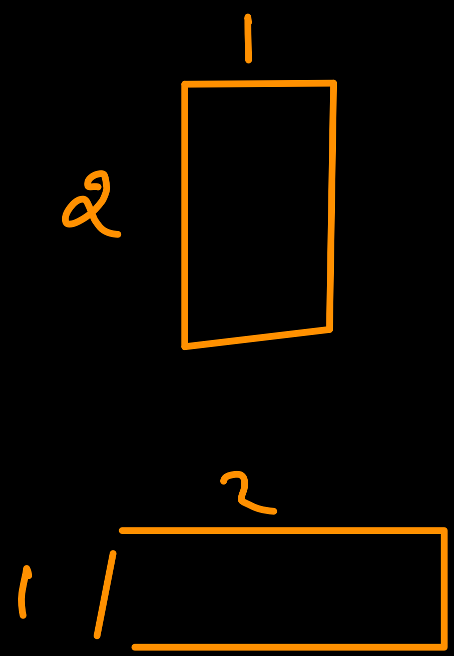
- All ***5*** bricks can be placed *vertically.*

- ***4*** bricks can be placed *horizontally* and ***1*** *vertically in* **2** *different ways.*

- ***2*** bricks can be placed *horizontally* and ***3*** *vertically in* **2** *different ways.*



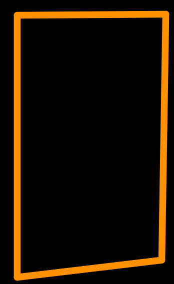




$n \to 0 \quad \to 0$

$n \to 1 \quad 1$

$n \to 2 \quad 2$



$$f(N) = f(N-1) + f(N-2) \to \text{fib}$$

ACODE

Alice and Bob need to send secret messages to each other and are discussing ways to encode their messages:

> Alice: "Let's just use a very simple code: We'll assign 'A' the code word 1, 'B' will be 2, and so on down to 'Z' being assigned 26."
>
> Bob: "That's a stupid code, Alice. Suppose I send you the word 'BEAN' encoded as 25114. You could decode that in many different ways!"
>
> B E A N
>
> Alice: "Sure you could, but what words would you get? Other than 'BEAN', you'd get 'BEAAD', 'YAAD', 'YAN', 'YKD' and 'BEKD'. I think you would be able to figure out the correct decoding. And why would you send me the word 'BEAN' anyway?"
>
> Bob: "OK, maybe that's a bad example, but I bet you that if you got a string of length 5000 there would be tons of different decodings and with that many you would find at least two different ones that would make sense."
>
> Alice: "How many different decodings?"
>
> Bob: "Jillions!"

For some reason, Alice is still unconvinced by Bob's argument, so she requires a program that will determine how many decodings there can be for a given string using her code.

For each input set, print all possible decodings for the input string.

Ex → "123"

ans → A B C

A W

L C

Y A A D

2 5 1 1 4

B E A A D

$f(str, i, out)$ = $f(str, i+1, out + char(str[i]))$

prints all possible decodings from i^{th} index to the last index

$f(str, i+2, out + char(str[i]str[i+1]))$

if $str[i]str[i+1] \le 26$

B

2 5 1 1 4

Y

every digit can be considered individually or with adjacent pair

≤ 26

E

5 2 1 1 4

X

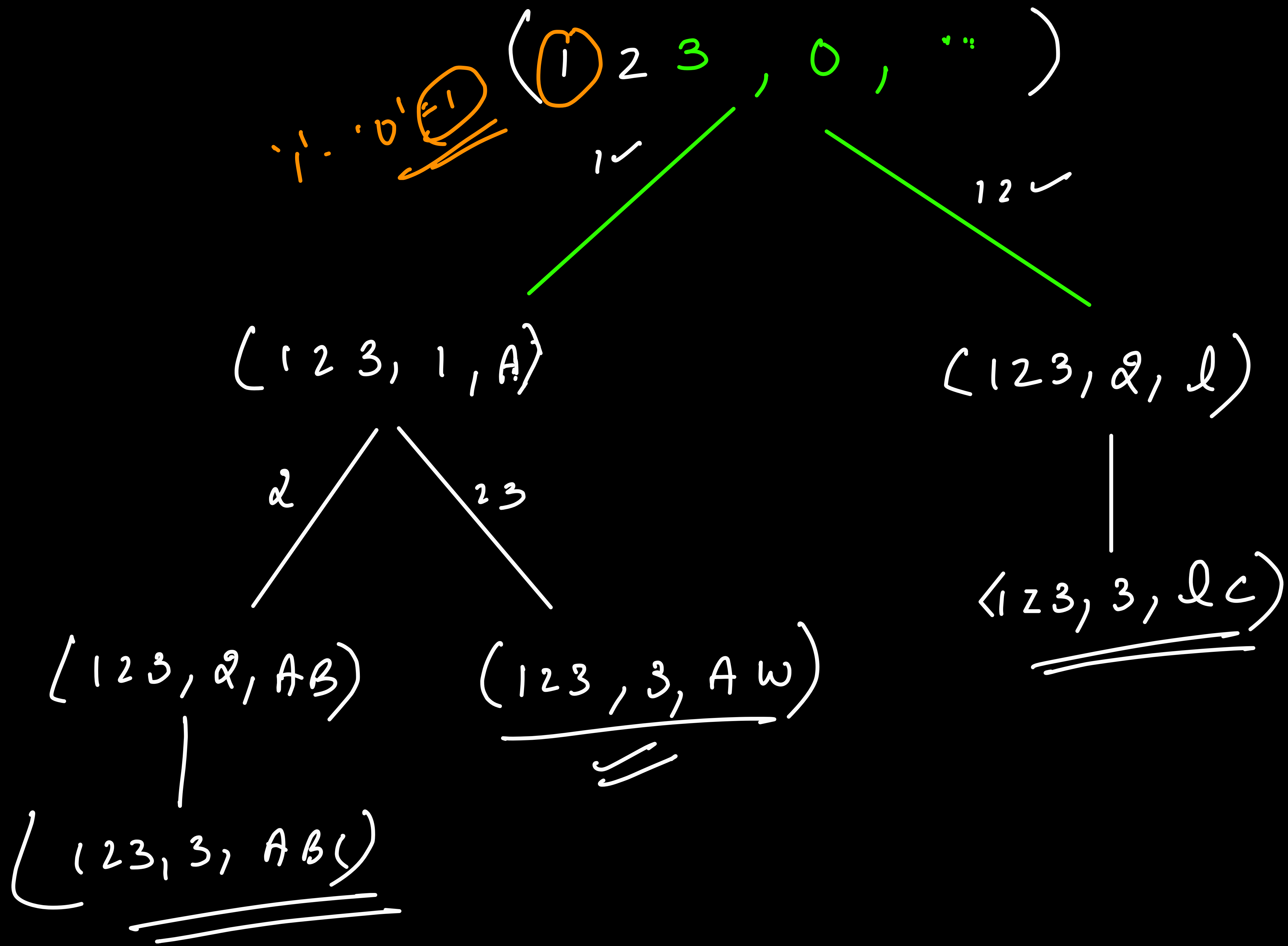
$2 \ 5 \ 1 \ 1 \ 4$

$2 \rightarrow B$

≤ 26

$25 \ 114$

$25 \rightarrow Y$


(123, 0, "")
'1' - '0' = 1
1
12
(123, 1, A)
(123, 2, l)
2
23
(123, 2, AB)
(123, 3, AW)
(123, 3, lc)
(123, 3, ABC)

$A \to 1$

$B \to 2$

$C \to 3$

$\vdots$

I

$64 + 1 \to 65$

$64 + 2 \to 66$

$64 + 3 \to 67$

$64 + 9 \to$

Rajesh Kuthrapali has a weird family structure. Every male member gives birth to a male child first and then a female child whereas every female member gives birth to a female child first and then to a male child. Rajesh analyses this pattern and wants to know what will be the Kth child in his Nth generation. Help him.

Note:

1. Every member has exactly 2 children.
2. The generation starts with a male member (Rajesh).
3. In the figure given below:

```
                 M ------------ 1st generation
               /   \
              /     \
             /       \
            M         F ------- 2nd generation
          /   \     /   \
         M     F   F     M
                   |
                 3rd child of 3rd generation
```

Input

First line specifies T, the number of test cases.

Next T lines each gives 2 numbers, N and K.

Input→

4

1 1

2 1

2 2

4 5

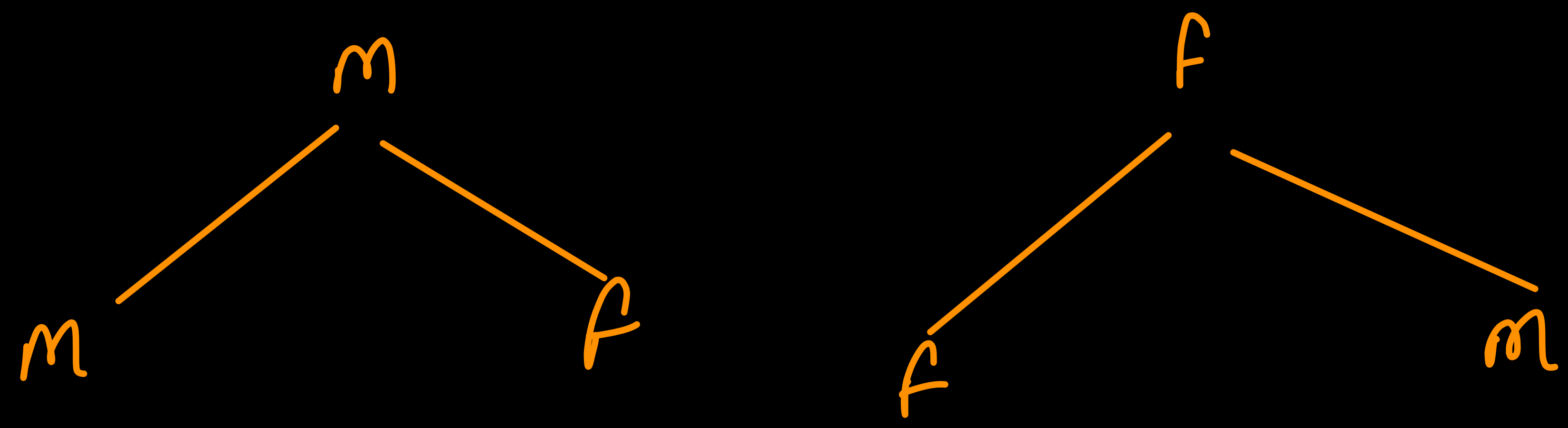
Output

Male

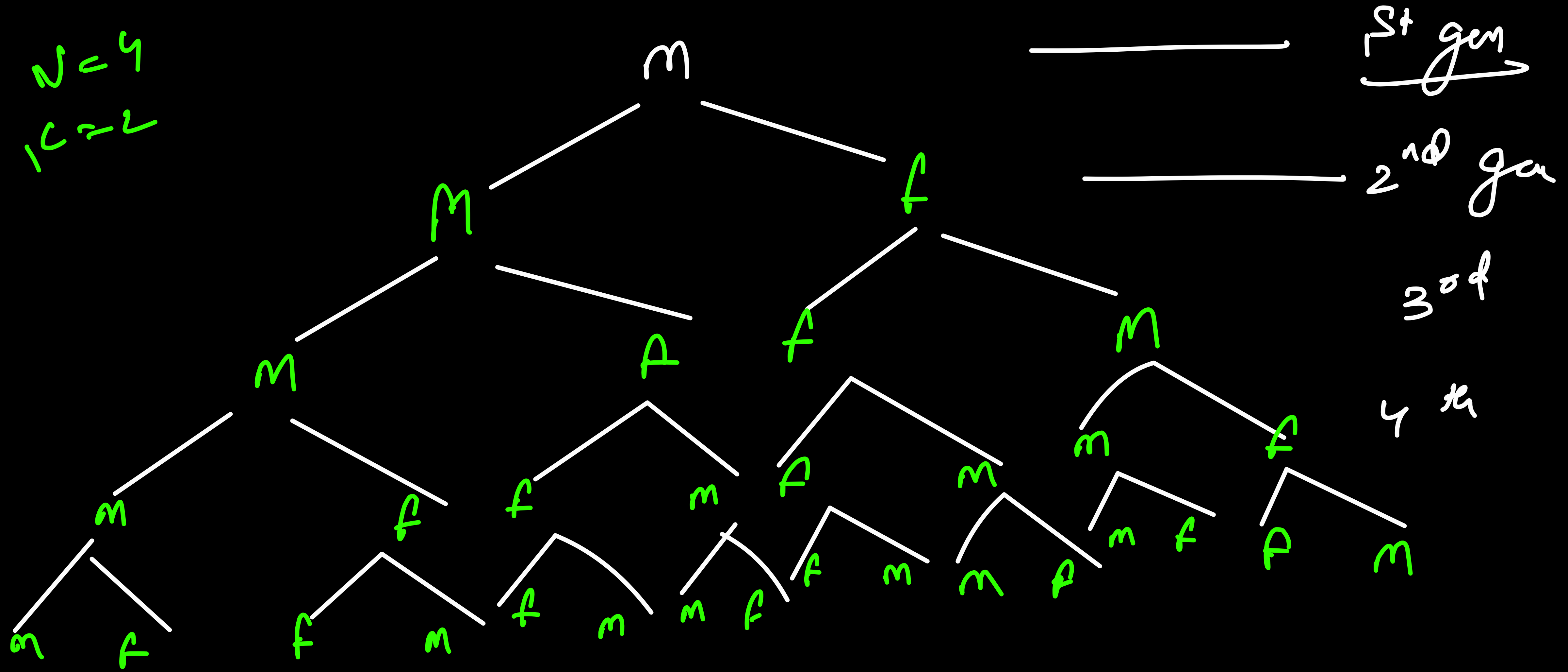
Male

Female

Female



$N = 4$
$K = 2$



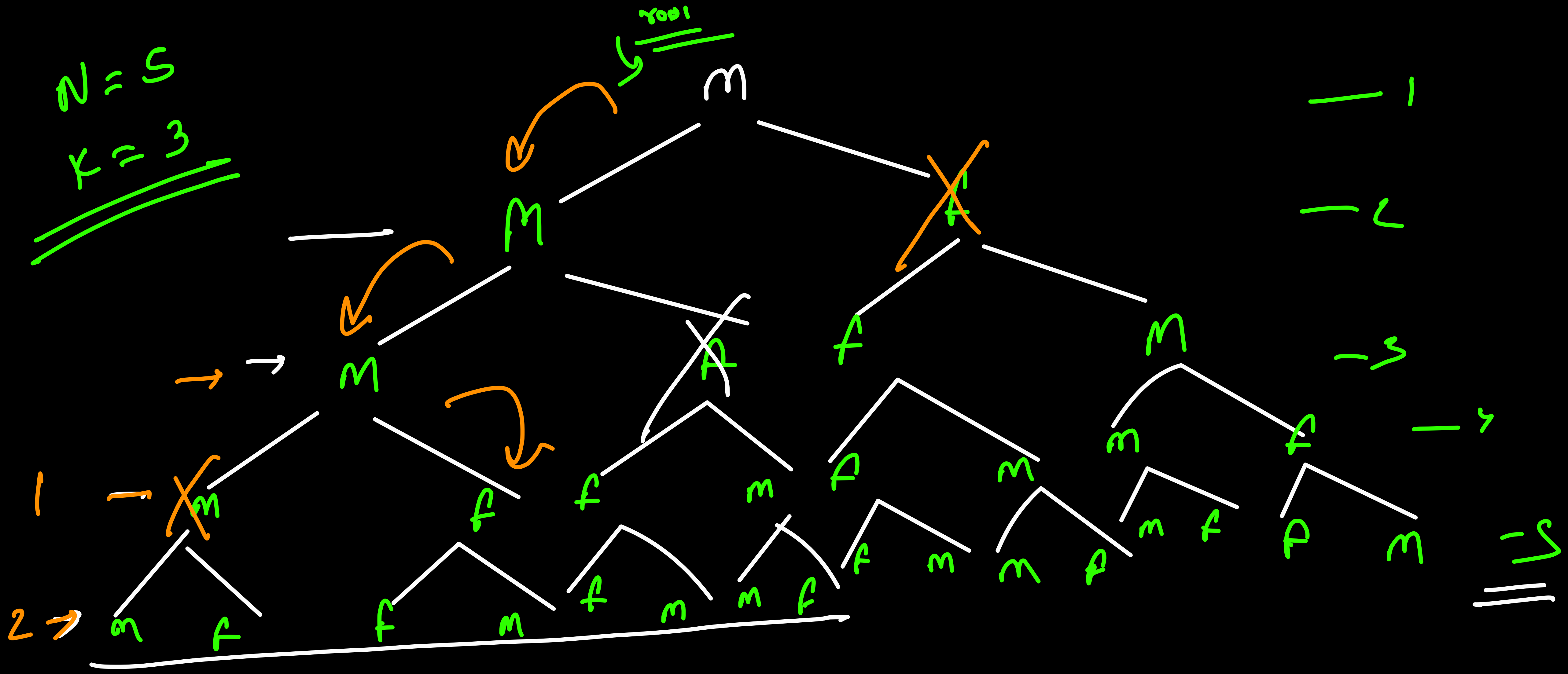
$N, K \rightarrow$ depends on the legacy

in any N^{th} gen $\rightarrow$ 2^{N-1} persons

$K \leq 2^{N-1}$

$N = 5$

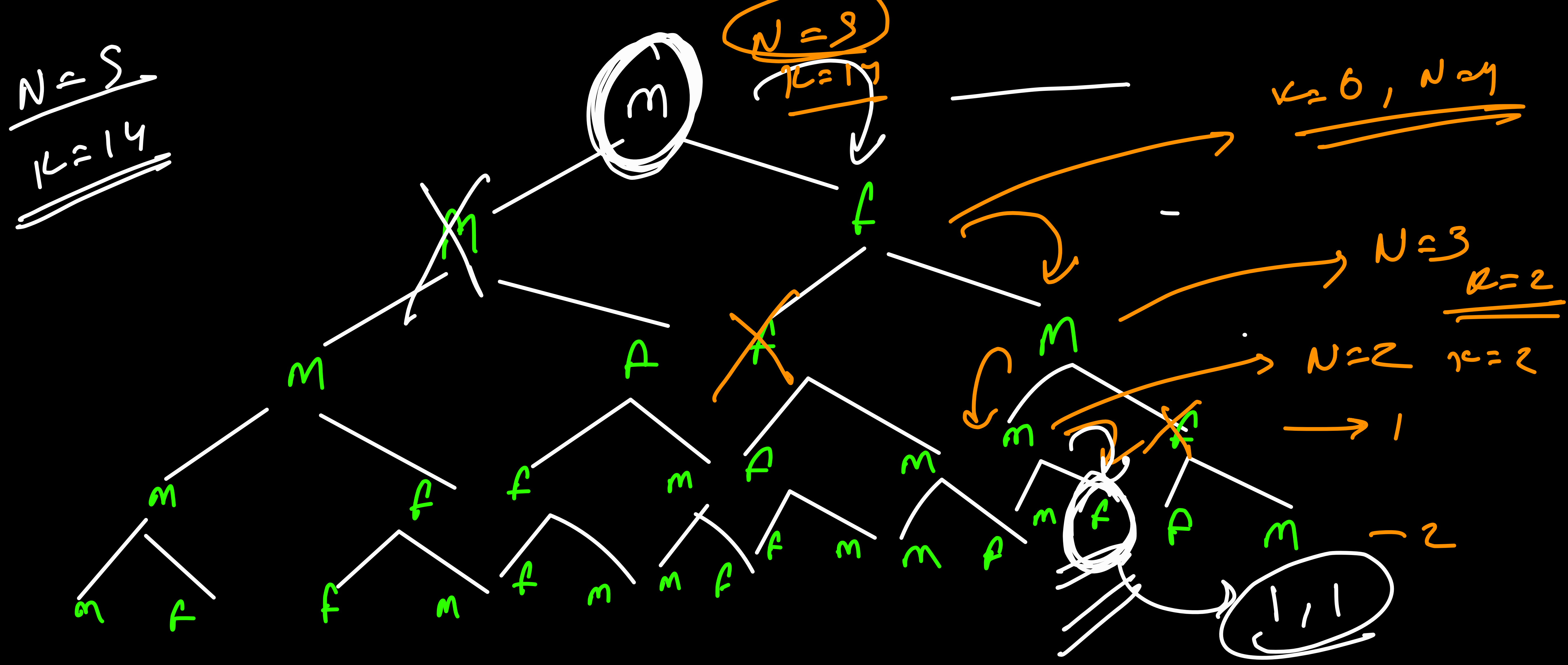
$K = 3$



$$2^{5-1} \rightarrow 2^4 \rightarrow 16$$

$$3 \le \frac{16}{2}$$

$$K \le \frac{2^{n-1}}{2}$$



$$14 - \left(\frac{2^{5-1}}{2}\right) \to 14 - \frac{16}{2} \to 14 - 8 \to \underline{6}$$

$$K = 6 - \left(\frac{2^{4-1}}{2}\right) \to 6 - 4 \to \underline{2}$$

$$f(\text{root}, n, K) = \begin{cases} f(\text{root}, n-1, K) & K \le \frac{2^{n-1}}{2} \to \text{jum } \underline{\underline{\text{left}}} \\ f(\text{new\_root}, n-1, K - \frac{2^{n-1}}{2}) & \underline{\underline{\text{else}}} \end{cases}$$

K^{th} child of n^{th} gen

will give root

$\underline{\underline{O(N)}}$

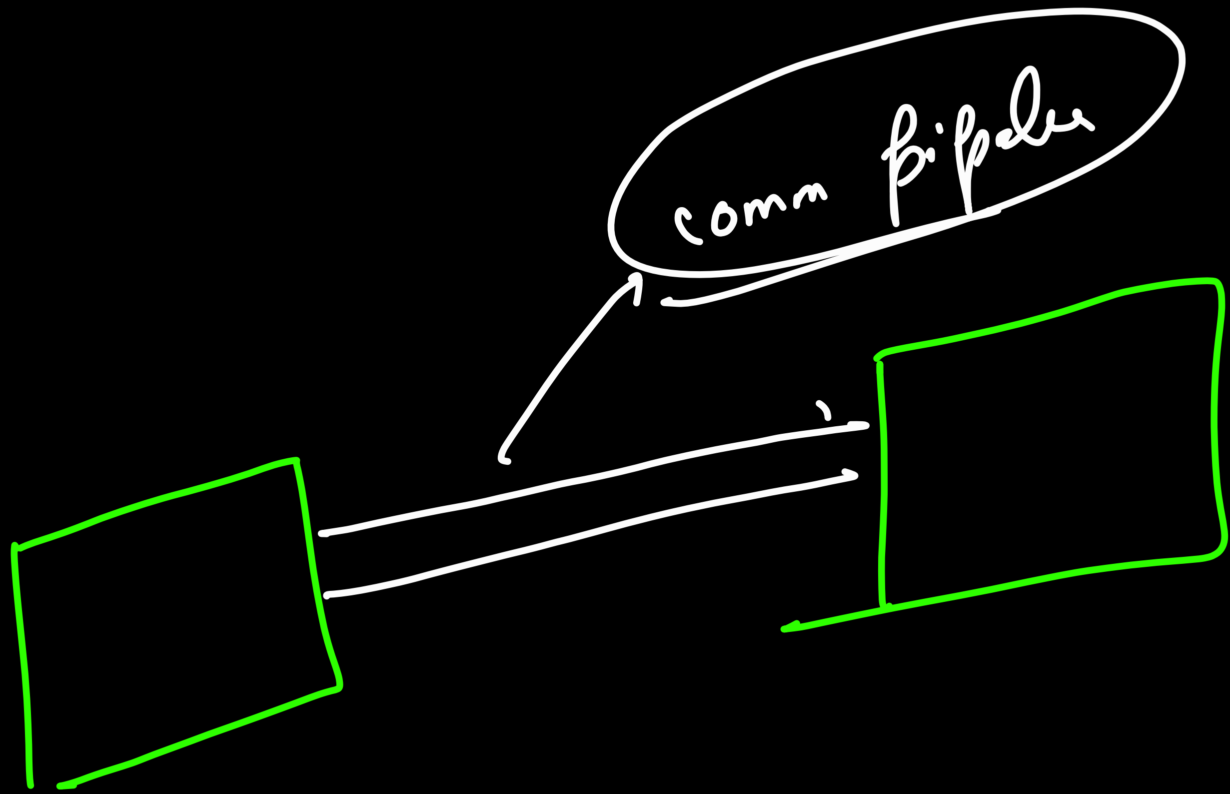
$\underline{\underline{1 << (n-1)}}$

$\underline{\underline{2^{n-1}}}$

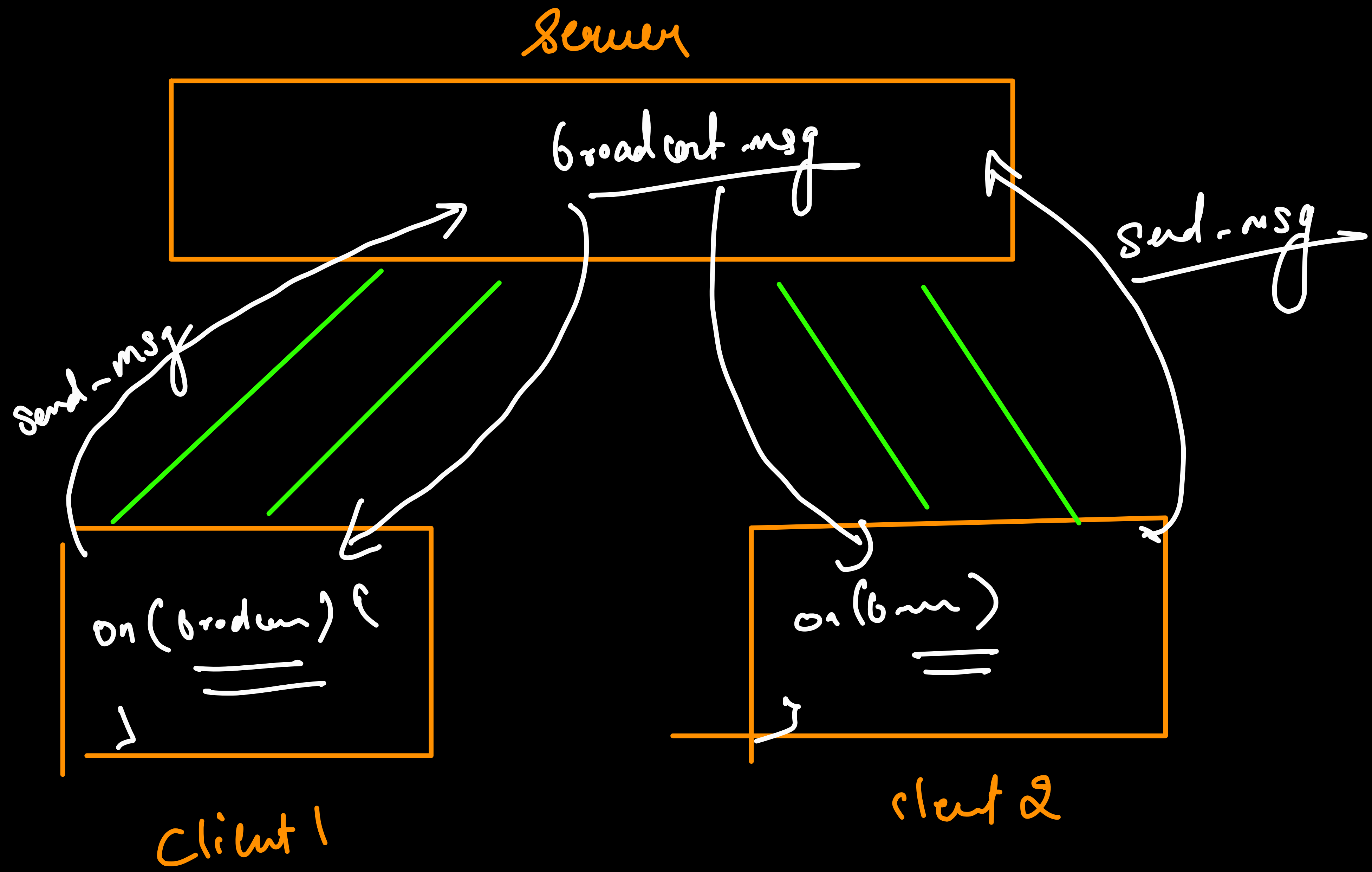
$1 << 3 \to 2^3$

$(00001) << 3$

$(01000) \to 8 \to 2^3$



Full duplex


Server
broadcast_msg
send_msg
send_msg
on(brodcast) {
}
on(b...)
Client 1
client 2