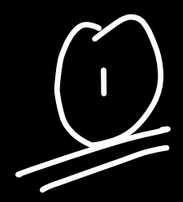
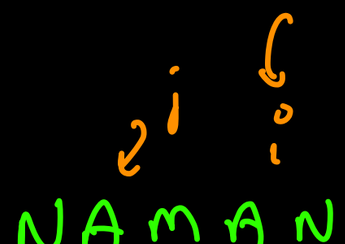
Q:1 Given a string of length n, calculate if the string is a palindrome or not recursively ??

Ex → NAMAN

ans → Yes

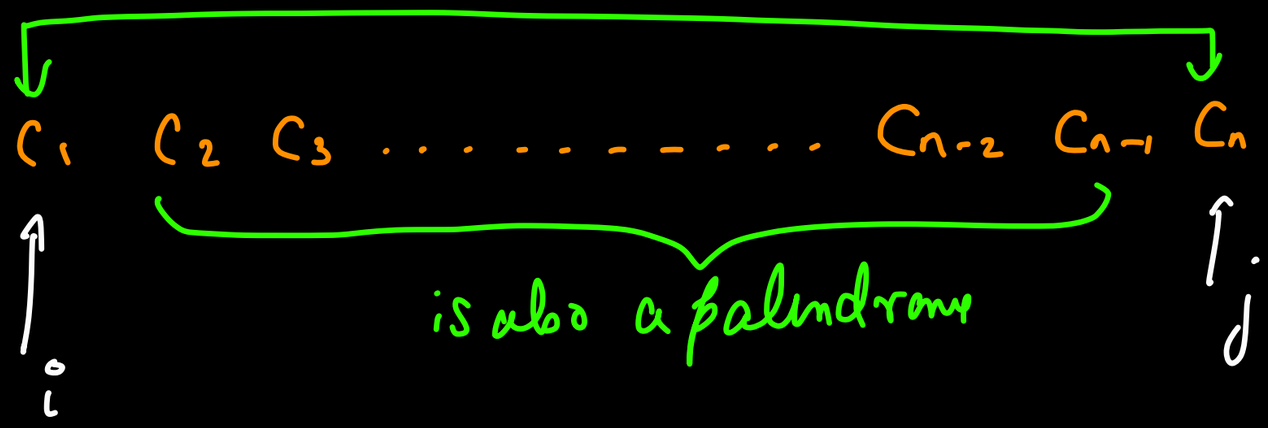
Iterative



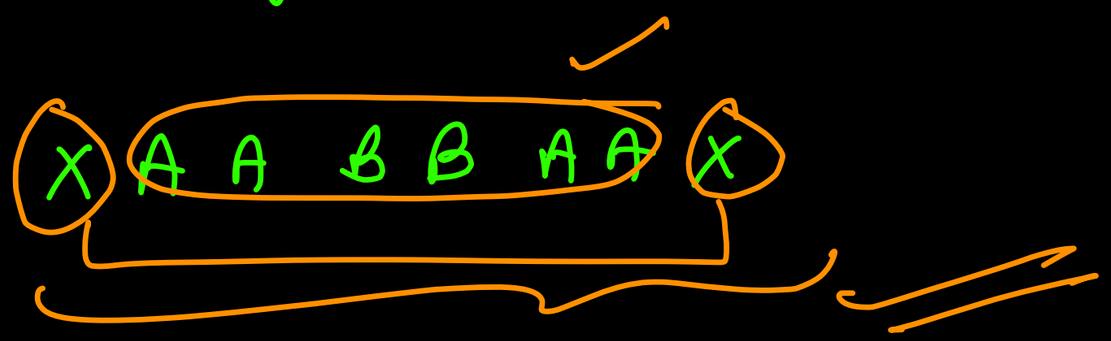



```
while (i<=j) {
    if (s[i] != s[j]) return false;
    i++;
    j--;
}
```

palindrome



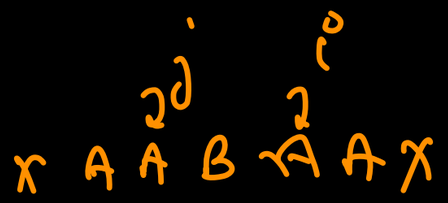
($C_1 == C_n$) and remaing string is also a palindrome then we can say the whole string is a palindrome.



$f(str, i, j)$ = $(str[i] == str[j])$ and $f(str, i+1, j-1)$

↳ if $(i <= j)$

whether str from index i to j is a palindrome or not ??



How can we solve the prev prob, without using the Str.Size() function

↳ let's keep the i^{th} index global and push the j^{th} index to the last ??

$f(str, j) \rightarrow f(str, j+1)$

if (str[i] == str[j])

global
i = 0 1 2 3

f(NAMAN, 5)

f(NAMAN, 4)

f(NAMAN, 3)

f(NAMAN, 2)

f(NAMAN, 1)

f(NAMAN, 0)

Somehow we will delete it is the base case

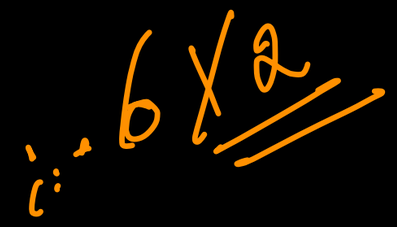
null char Loop

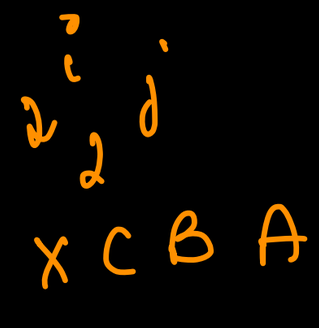
true

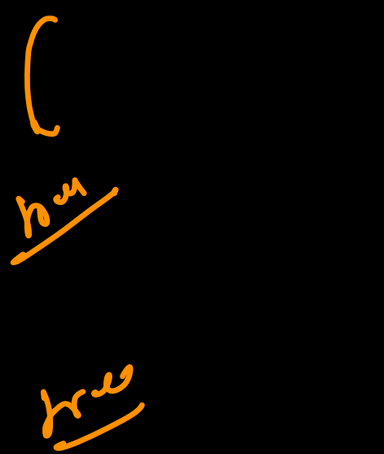
true

true

true







Q⇒ for an input of n, print the following pattern recursively.

n = 5

1 ★
2 ★ ★
3 ★ ★ ★
4 ★ ★ ★ ★
5 ★ ★ ★ ★ ★

```
for( row=1; row<=n; row++) {
    for( col=1; col<=row; col++){
        print (*)
    }
    print (\n)
}
```

for any ith row → col ∈ [1, i]

$f(n, r)$ ⟹

for(col=1; col<=r; col++) C

print(*)

}

print("n")

$f(n, r+1)$

Base ⇒ if (r>n) return;

$f(n, 1)$ my self work will be to print stars for the r^{th} row.

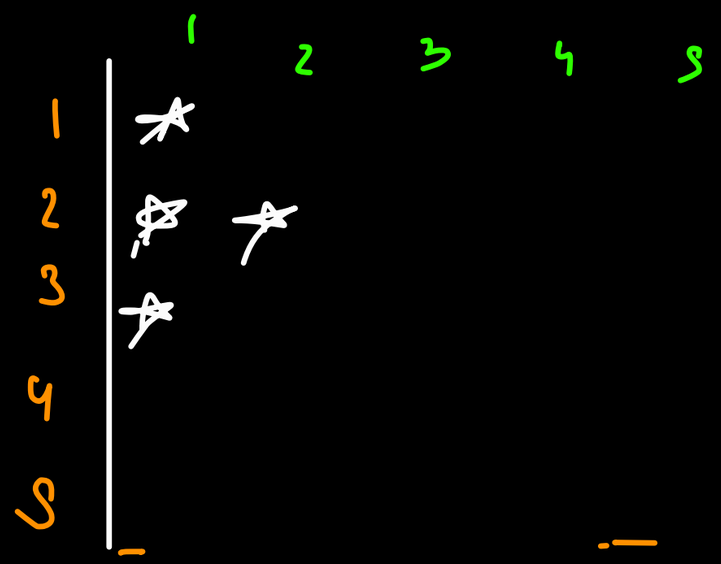
this func" prints the pattern from r^{th} row to the n^{th} row.

$f(n, r, c)$ =

prints the pattern

from r^{th} row, c^{th} col

to the n^{th} row & n^{th} col

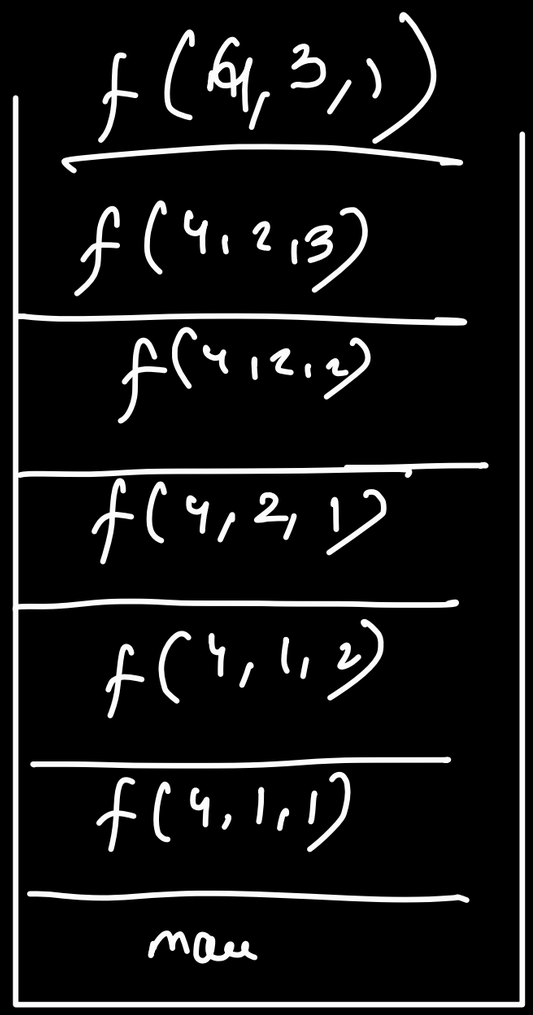



```
if (c > r) {
    print ('\n')
    f(n, r+1, 1)
    return;
}
print (*)
f(n, r, c+1)
```

go to a new row

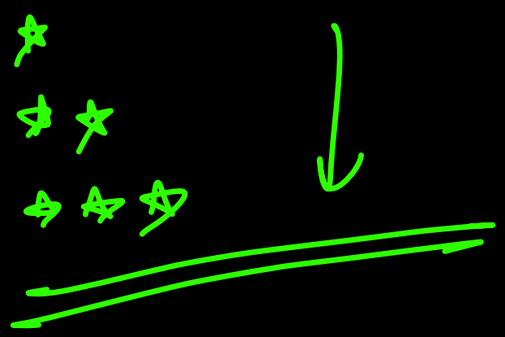
Base Case

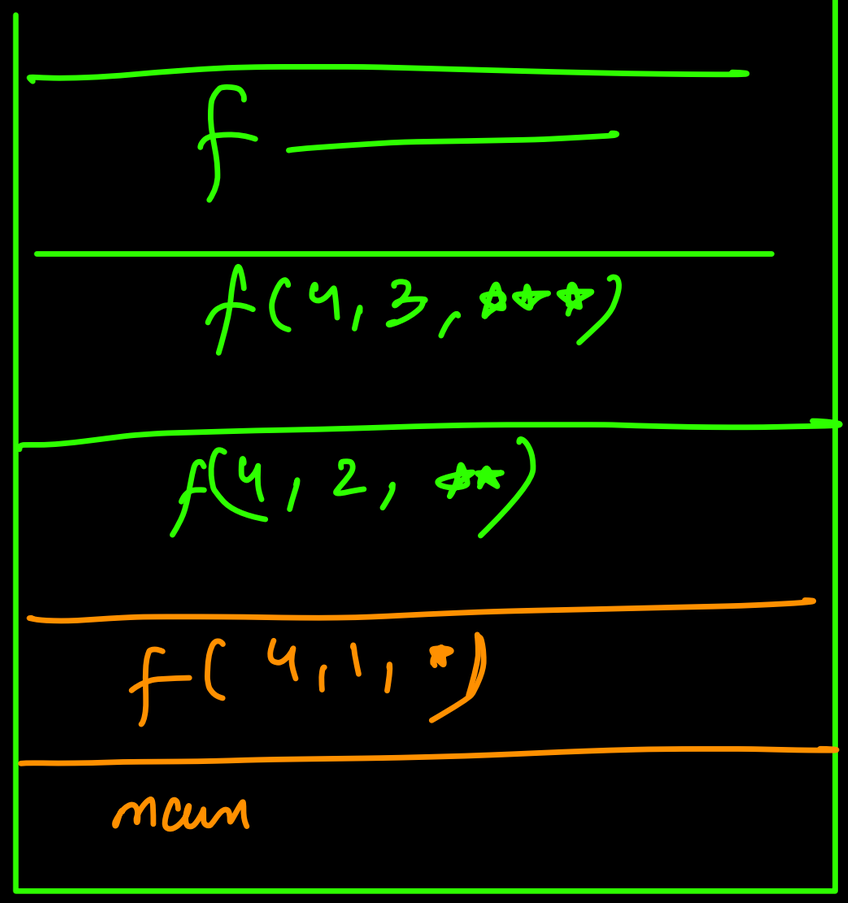
```
if (r > n)
    return;
```



```
f(n, r, output) = print(output)
                  f(n, r-1, output + "*")
```

f(n, 1, "*")





X — (N) ← Nth power of natural no.

1, 2, 3, 4, ... - - - - - - - - -

The natural no.'s that can sum upto x, will be in the range [1, x]

x
x = 100

x = 10

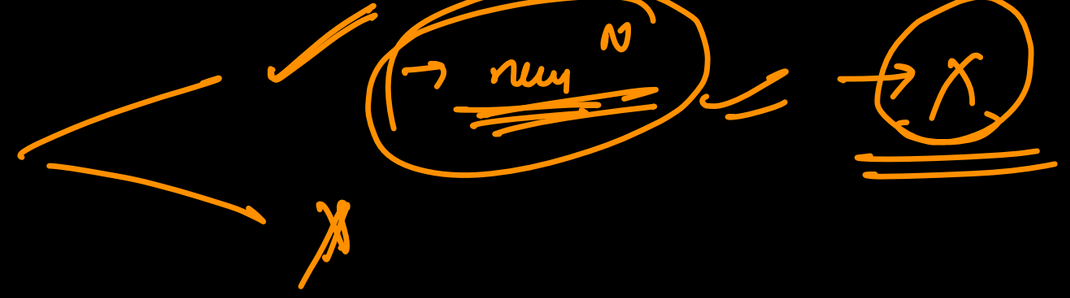
(2) ≤ N ≤ 10

(12) x [1, √x] ←

10 <

12

$[1, \sqrt{x}]$

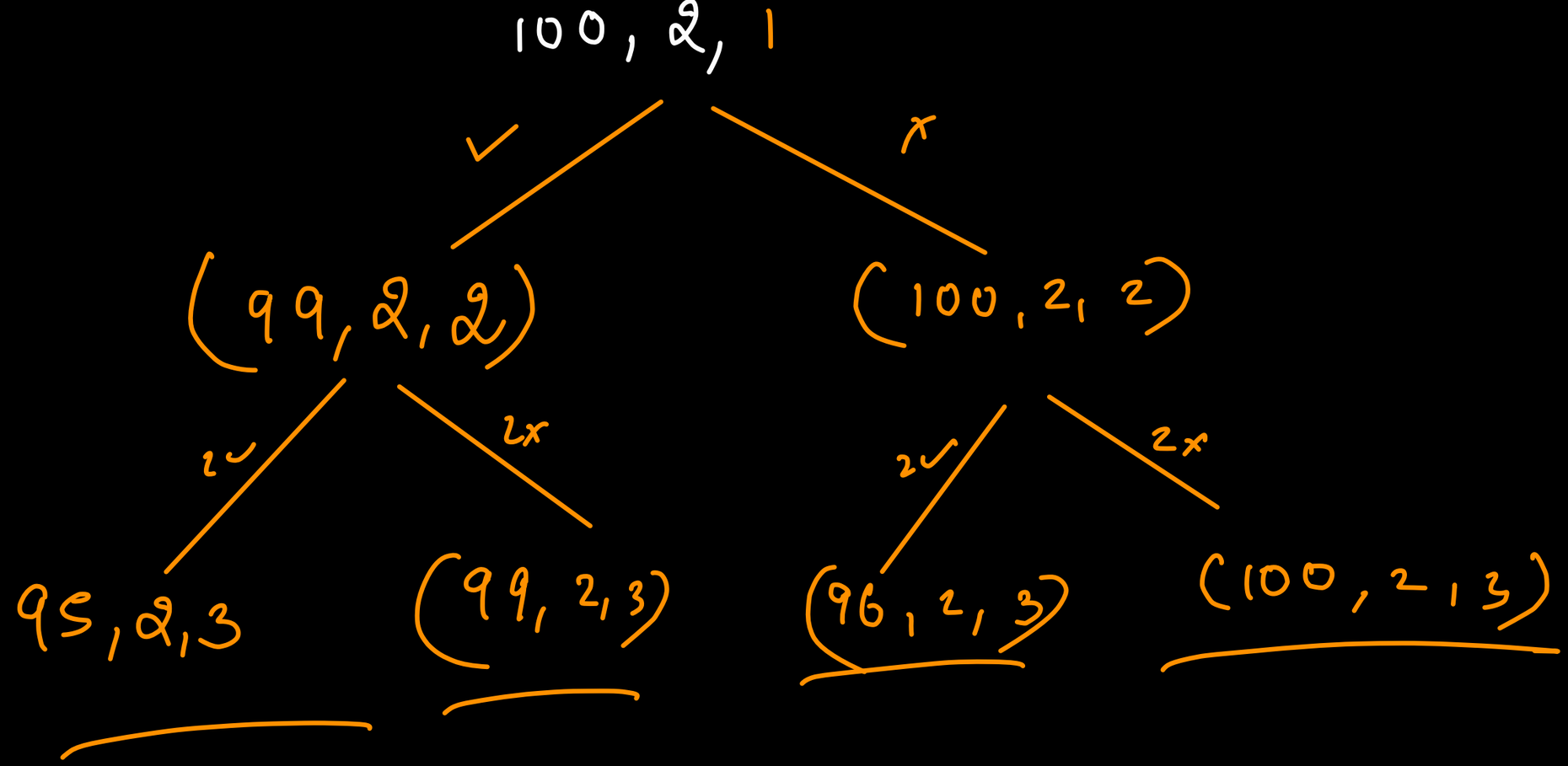


↓

How many combinations of no.'s we can have such that sum of N^{th} powers of that no. is equal to x.

$100 \rightarrow [1, \sqrt{100}] \rightarrow [1, 10]$

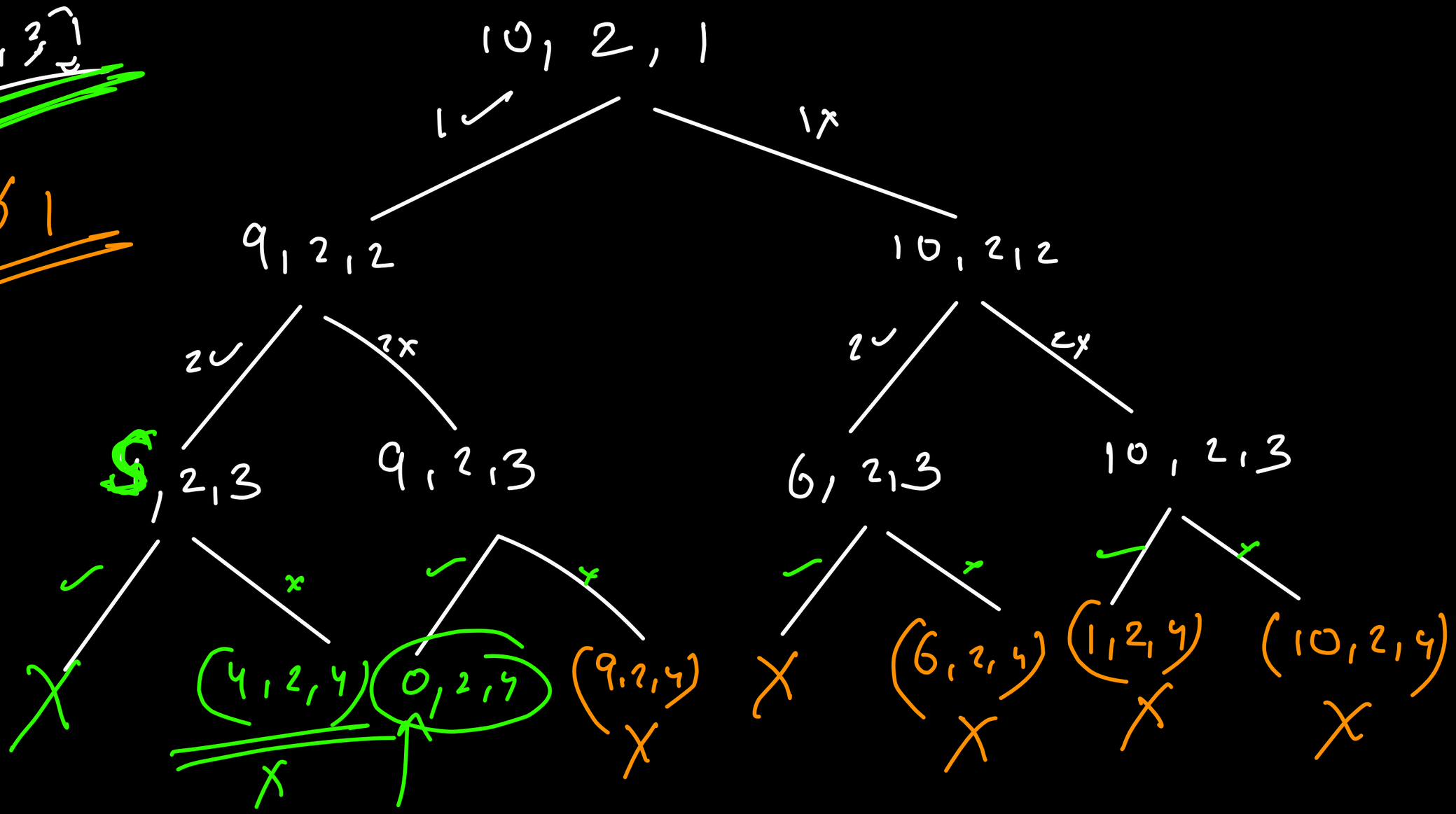
$N = 2$



Try to reduce the problem such that it becomes similar to subsequences

[1,3]

ans: 1



$$f(x, N, i) = f(x - i^N,\ N,\ i+1)$$

$$f(x, N, i+1)$$

if (x == 0) counter

if ($i^N > x$) x

$i * i > x$ x

Base

$$f(x, n, i, out) = f(x - i^n, n, i+1, out + i^n)$$

$$f(x, n, i+1, out)$$

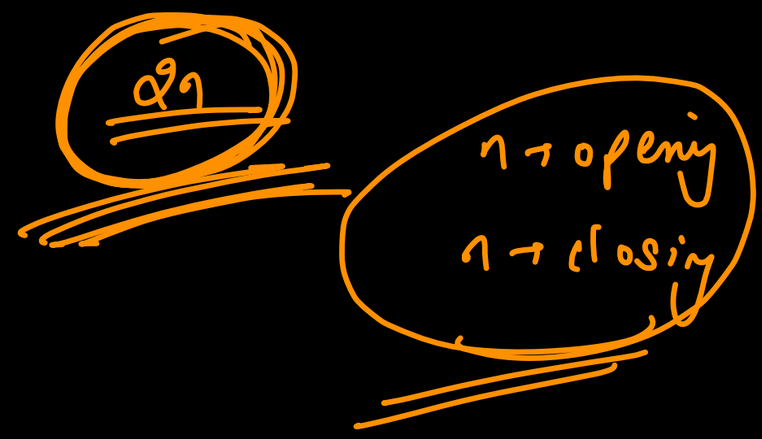
pow(i, n)

n=1 → ( )

n=2 ( )( ) (( ))

n=3 ( )( )( ) (( ))( ) ( )(( ))

((( ))) (( )( ))



Balanced

for every index we can either add an (or)

```
f(n, o, c, output) = if (o > c) {
                         f(n, o, c+1, output + ")")
                     }
                     3
                     if(o < n) {
                         f(n, o+1, c, output + "(")
                     }
```

3

Base case

```
if (c == n)
    print(output)
    return;
```

if (o > c)
↳ add)

3, 0, 0, ""

3, 1, 0, "c"

c ✓) ✓

(3, 2, 0, "cc")

(3, 1, 1, "()")

c) (

(3, 3, 0, "ccc")

(3, 2, 1, "c)")

(3, 2, 1, "()(")

(3, 3, 1, "ccc)")

(3, 3, 1, "c)c)")

(3, 2, 2, "c))")

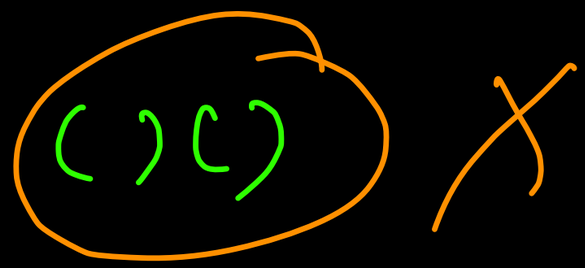
(3, 3, 2, "c)()")

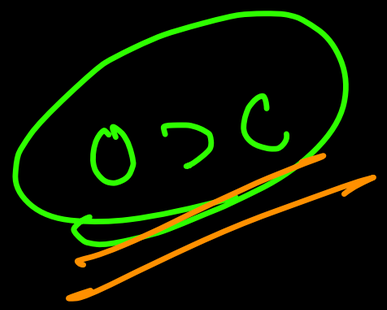
(3, 3, 2, "()c)")

(3, 3, 3, "c)()")

(3, 3, 3, "()()")

(3, 3, 3, "())()")





( ) ( ) (( ))