Time and Space Complexity of Recursive Problems

Special class

Love Babbar • Mar 31, 2023

T.C → T.T

S.C → S.T

3 way

→ Mathematical

→ Rec relation ✓

→ ○

Problems

→ T.C

→ S.C

Rec tree

→ intuition → T.C → no of operations

→ time taken by all calls

→ S.C → max. depth

Master's theorem ✗

Upper Bound $\leftarrow O(n)$

```
for (int i=0;  i<n; i++)
{
    cout << i;
}
```

$O(10)$

$O(1) \rightarrow$ cout

T.C

```
if (a>0)
```

$\rightarrow O(1) \rightarrow k, c, b$

```
cout << 5;
```

$\rightarrow O(1) \rightarrow k$

```
a = a+b;
```

$\rightarrow O(1) \rightarrow$

S.C

BAD

```
a = 5;
b = 7;
int arr[n]
```

8 bytes
b/w and

O(n)

Counting

factorial

Power

Max

Min

checkSorted

Recursion

```
print (n)
{
  //B.C
  cout << c
  //R.R
  print (n-1)
}
```

$f(n) \longrightarrow f(n-1)$

$T(n) = K_1 + K_2 + T(n-1)$

$$\boxed{T(n) = K + T(n-1)}$$

```
factorial (n)
{
  //B.C
  return n* fact(n-1)
}
```

$f(n) \longrightarrow f(n-1)$

$T(n) = K_1 + K_2 + T(n-1)$

$$\boxed{T(n) = K + T(n-1)}$$

2^n

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

$2 \times 2^{n-2}$

$2 \times 2^{n-3}$

2×2^{1}

2×2^{0}

1 call → K time

(n+1) calls → (n+1) * K

$O((n+1) \times K)$

$O(n+1)$

$\boxed{O(n)}$

$\underbrace{0, 1, 2, 3, \ldots, n}_{(n+1)\text{ calls}}$

Counting

n = 5

```
void printCounting(int n)
{
    // B.C
    if (n == 0)
        return;

    cout << n << endl;

    // R.R
    printCounting(n-1);
}
```

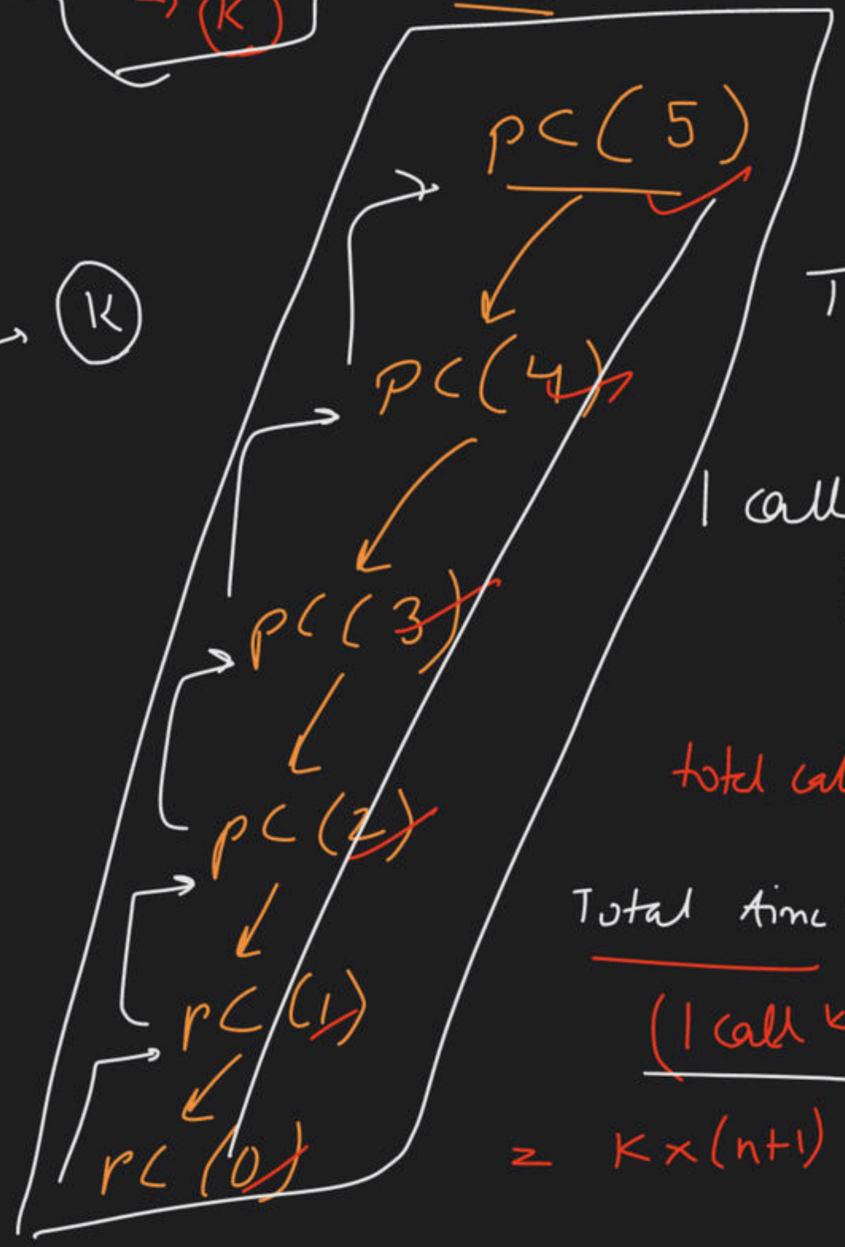
1 var → K

Time → K_1

1 call → $K_1 + K_2$ → K

T.C

n = 5



T.C =

1 call → time

total calls = (n+1)

Total time =

(1 call ka time) × (total calls)

= K × (n+1)

T.C $\longrightarrow$ $O\left((n+1)\frac{k}{\alpha}\right)$

$\longrightarrow$ $O\left(n+\frac{1}{\alpha}\right)$

T.C $\longrightarrow$ $\boxed{O(n)}$

Space Complexity

depth / ~~function call stack~~

max depth

Total calls → (n+1) call

1 call space → K

Total space → (1 call space) × (total calls)

= K × (n+1)

$$S.C \rightarrow O\left((n+1) * \frac{k}{\alpha}\right)$$

$$\rightarrow O\left(\frac{n+1}{\alpha}\right)$$

$$\boxed{S.C \rightarrow O(n)}$$

power

2^n

```
int power(int a, int b)
{
    // B.C
    if (b==0)
        return 1;

    return a * power(a, b-1);
}
```

K_1

K_2

$(2, n)$

$2 * (2, n-1)$

$2 * (2, n-2)$

$2 * (2, n-3)$

$2 \times (2, 1)$

$2 \times (2, 0)$

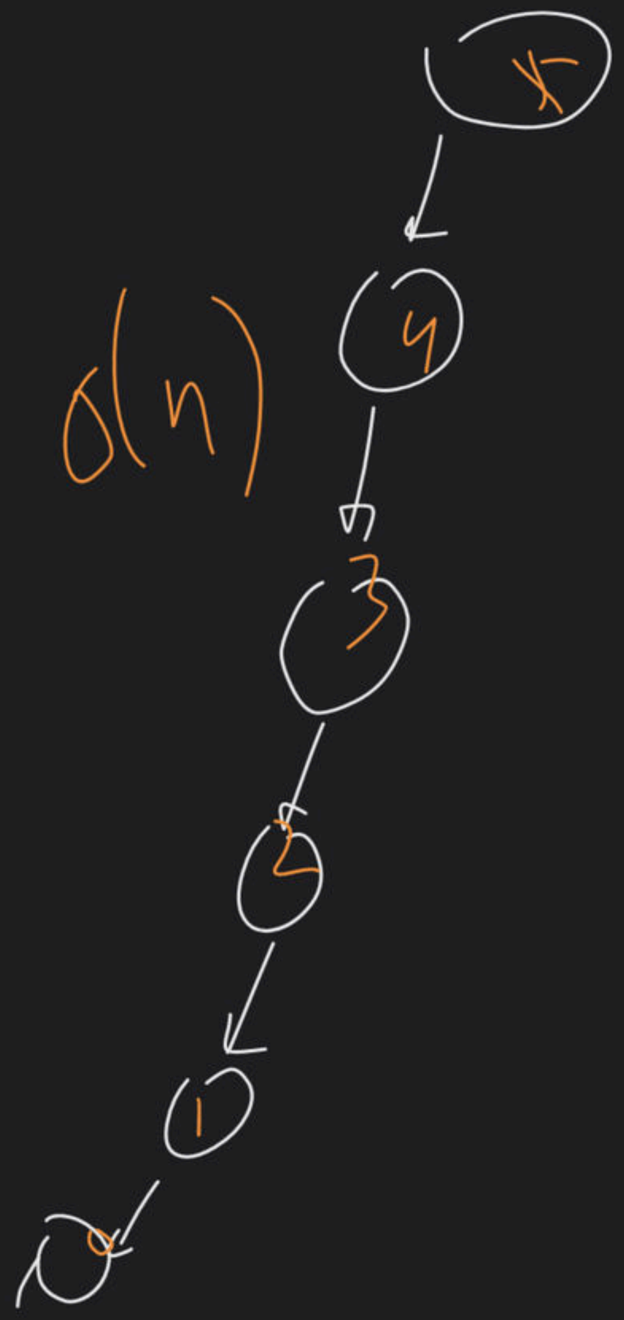
1 call time → K

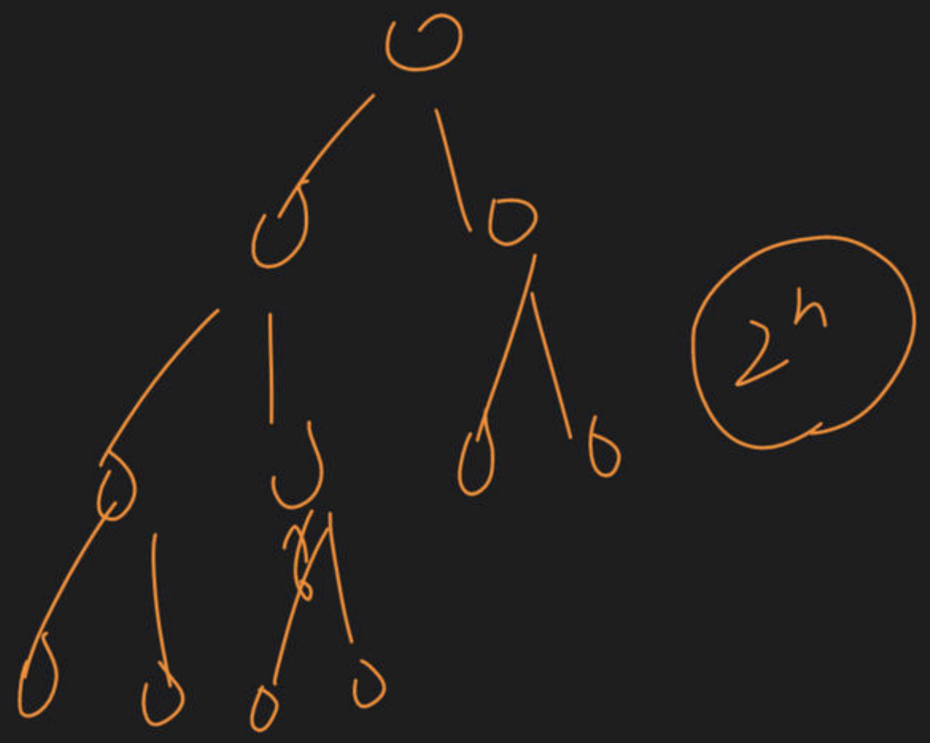
1 call space → K

T.C → $O((n+1) \times k) \rightarrow O(n)$

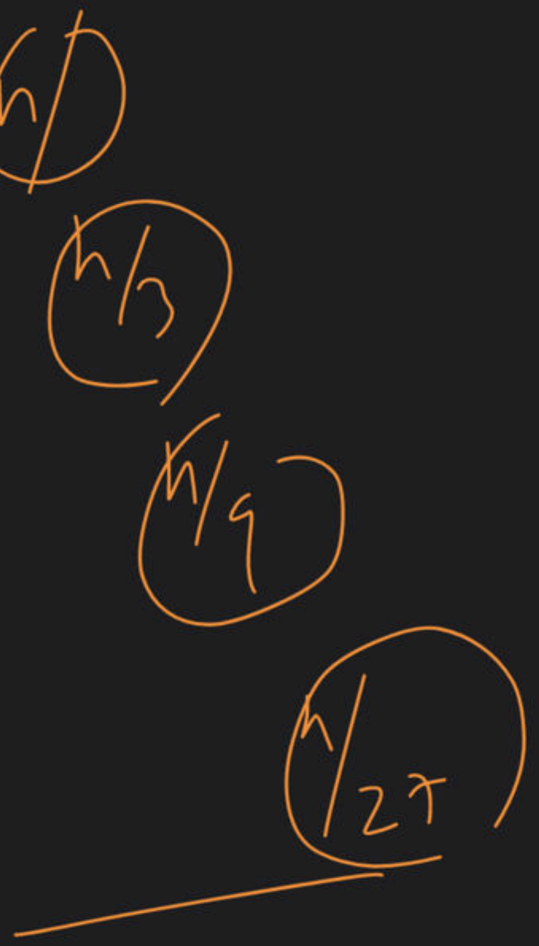
S.C → $O((n+1) * k) \rightarrow O(n)$

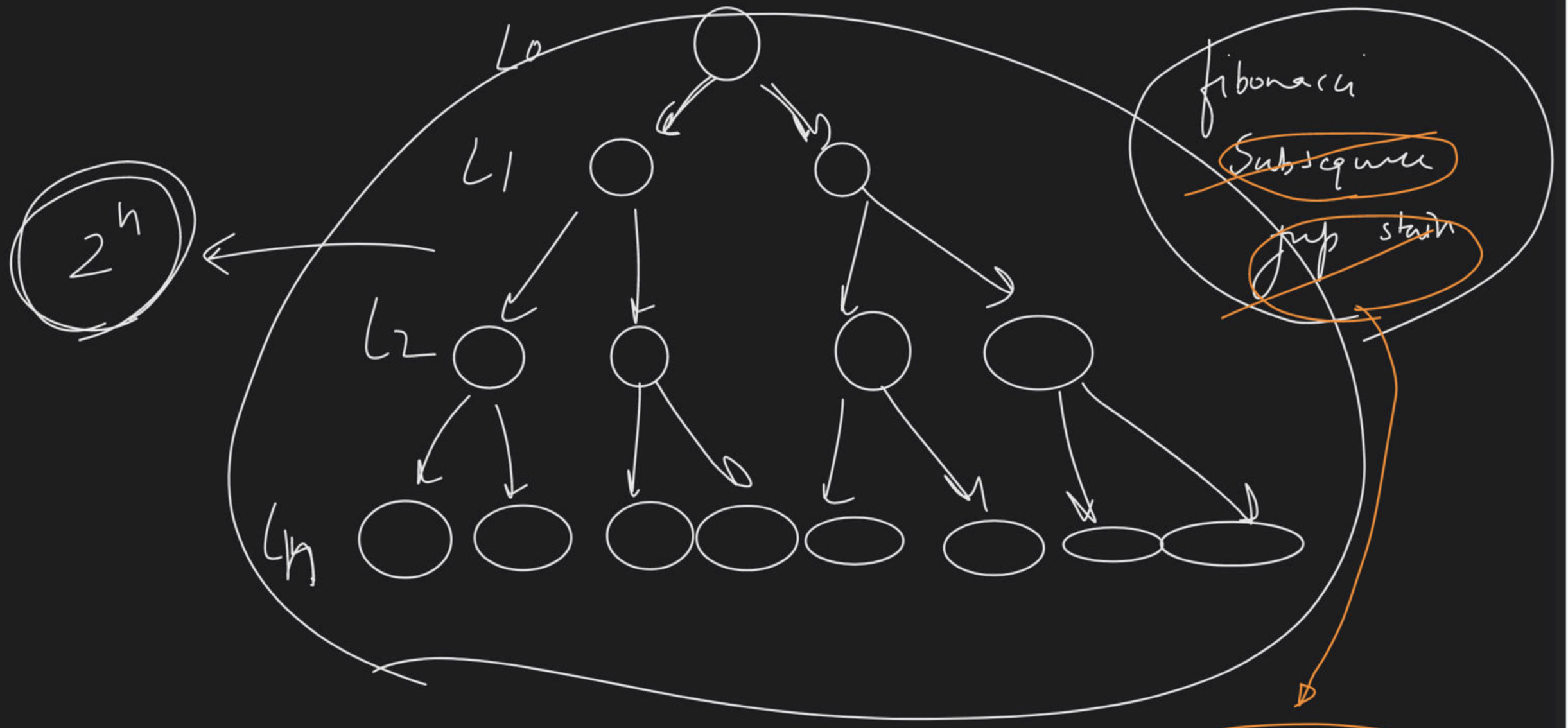
$(n+1)$ calls


O(n)
5
4
3
2
1
0


2^n

n
n/2
n/4
n/8


n/1
n/3
n/9
n/27



~~fib~~. $f(n) = f(n-1) + f(n-2)$

fibonacci series :-

R.R

$$\boxed{f(n) = f(n-1) + f(n-2)}$$

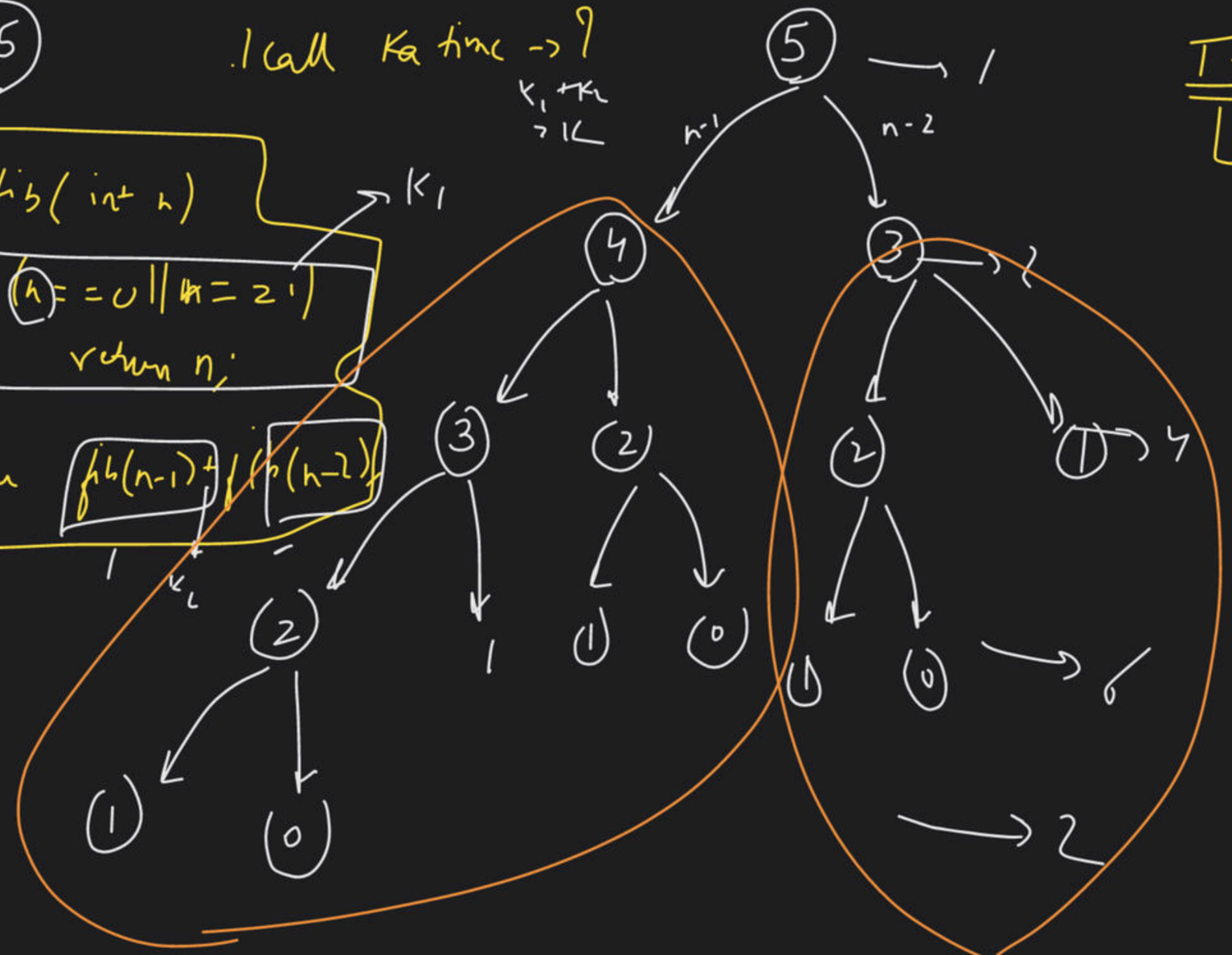
n=5

1 call ka time -> ?

$K_1 + K_2 \to K$

```
int fib(int n)
{
    if (n==0 || n==1)
        return n;

    return fib(n-1) + fib(n-2);
}
```



T.C

(1 call ka time) * (Total no of calls)

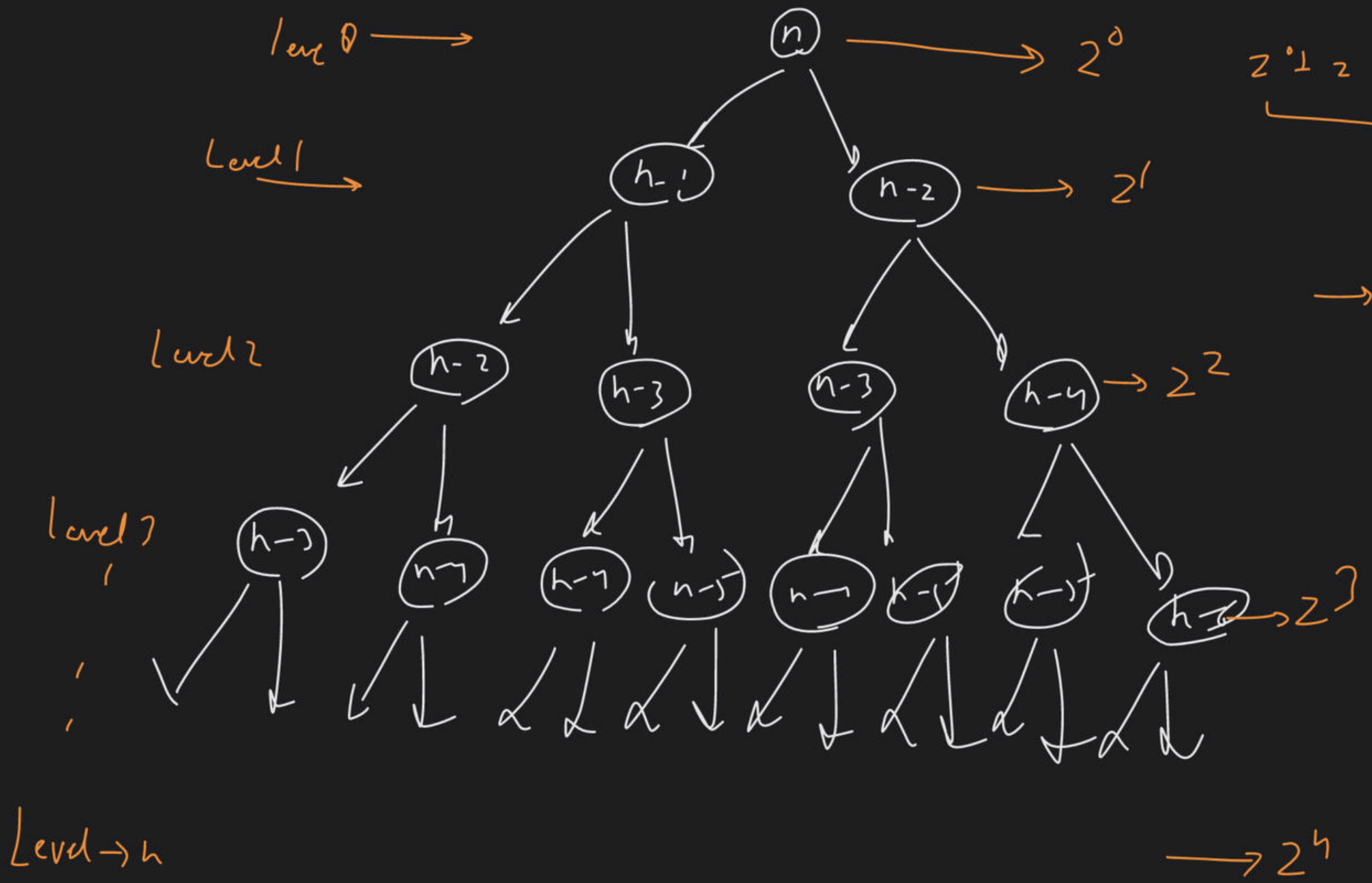
$K * (2^{n+1} - 1)$

T.C $O(K * (2^{n+1} - 1))$

$O(2^{n+1} - 1)$

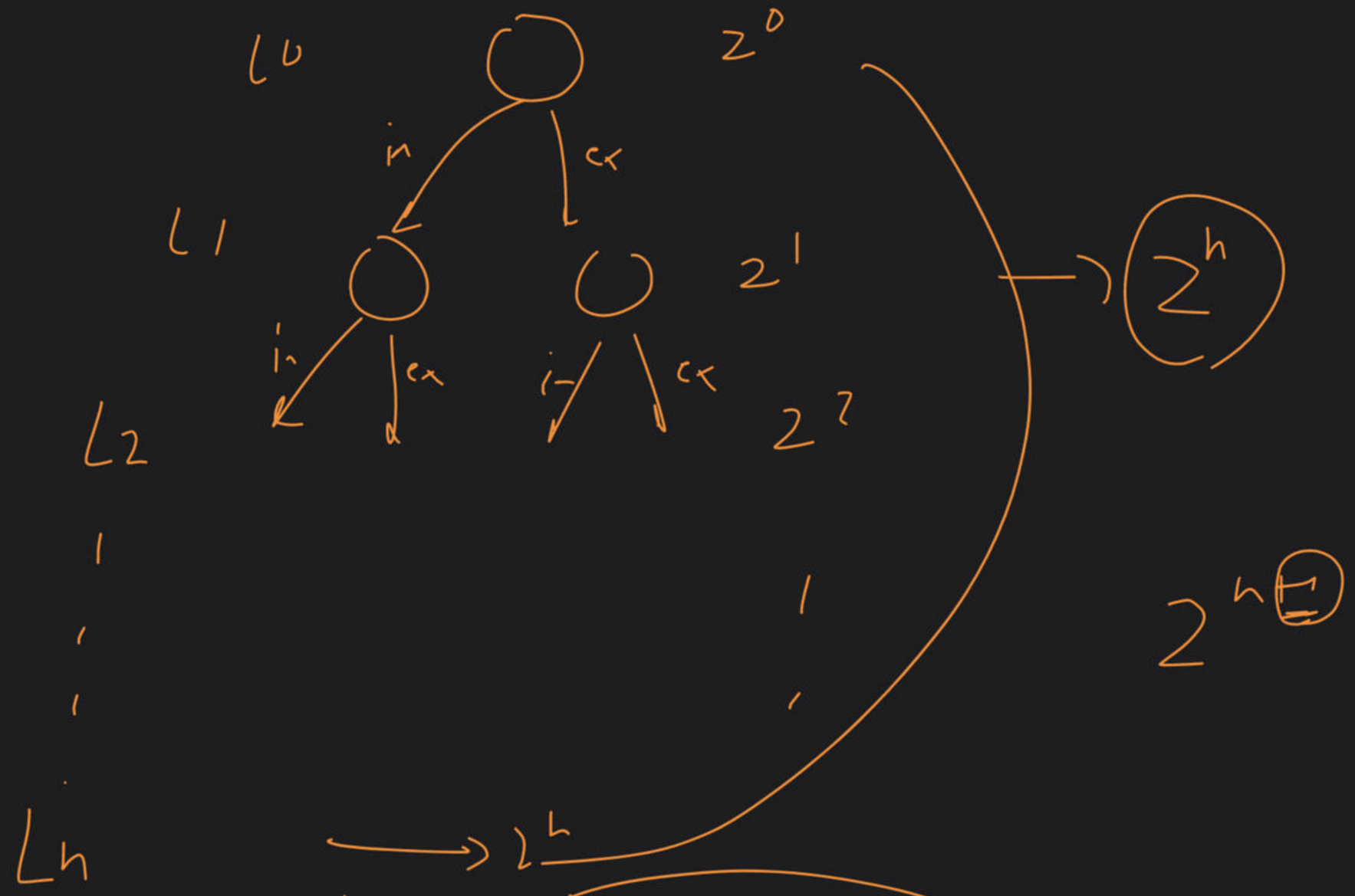
$O(2^{n+1})$

$O(2^n \times 2) \to O(2^n)$ exponential

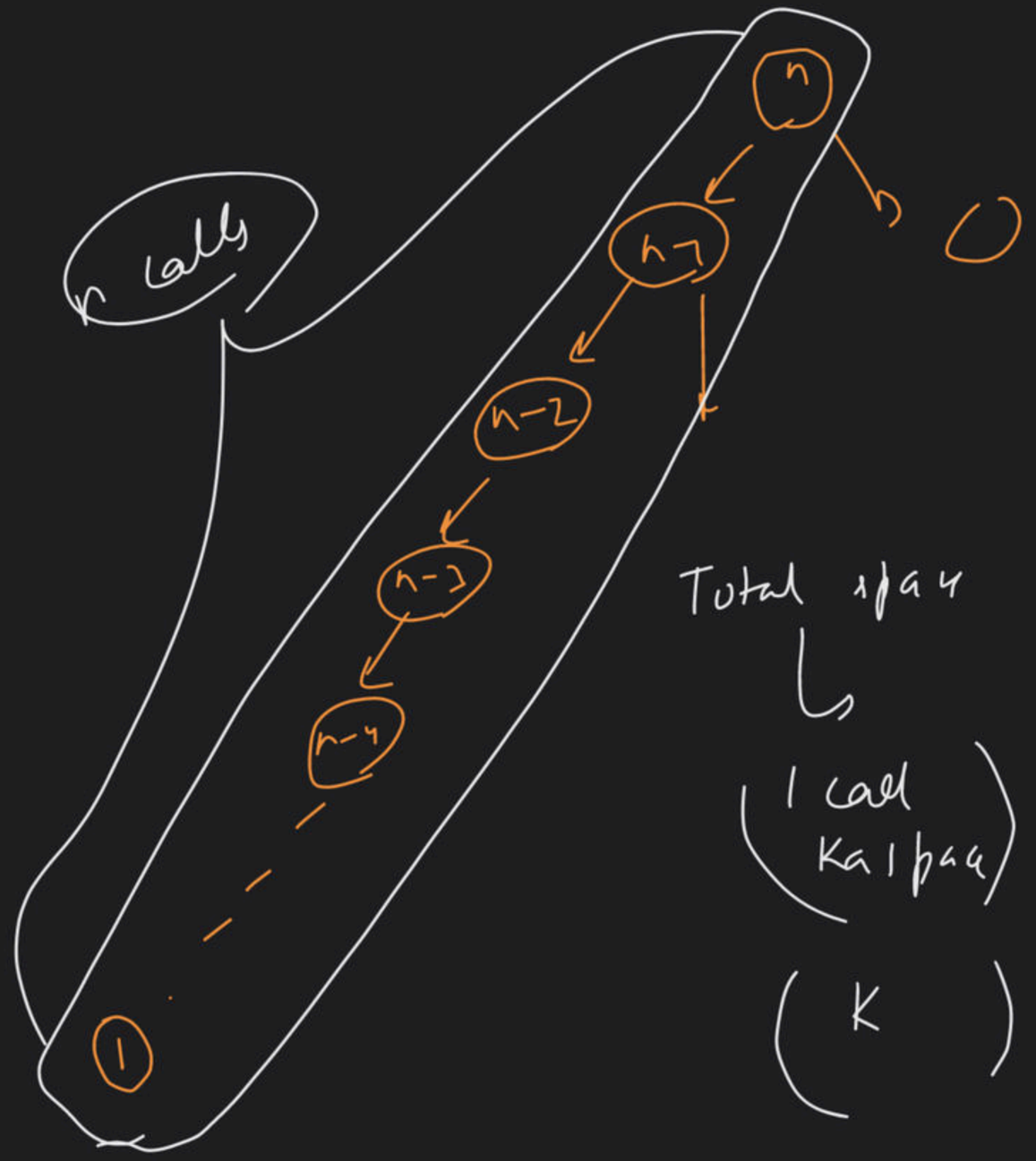


$$\underbrace{2^0 + 2^1 + 2^2 + 2^3 + \dots 2^n}_{n \cdot P}$$

$$\rightarrow 2^{n+1} - 1$$



$n \rightarrow$ length of string



S.C → n

Max Depth

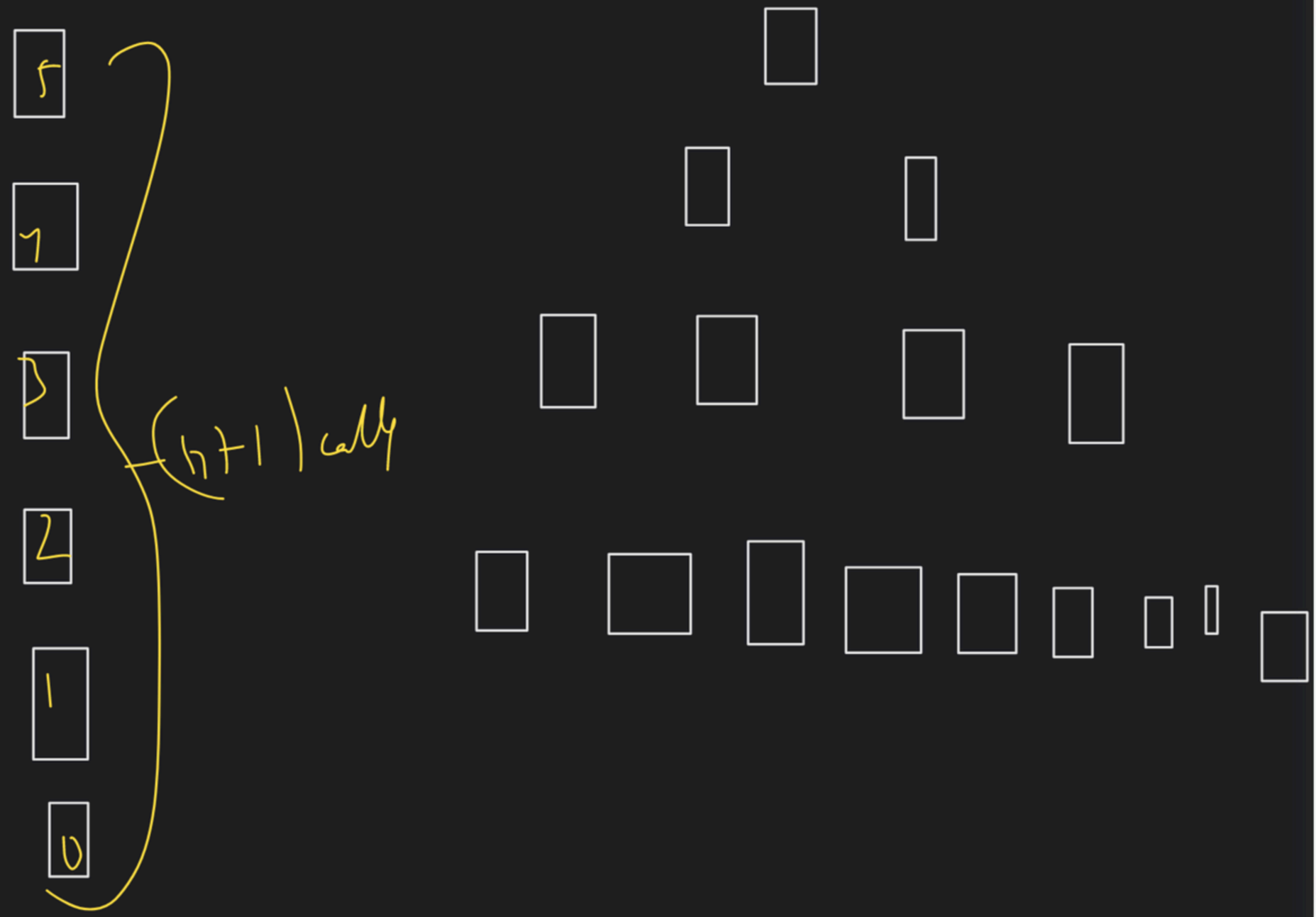
Total space

(1 call Kaipaa) * (Total calls)

(K) * n

O(n*K) → O(n) → S.C

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


5
4
3
2
1
0
(n+1) calls

Binary Search

→ a level

$a \geq \log n$

Total time

$= \left(\begin{matrix}\text{1 call kq} \\ \text{time}\end{matrix}\right) \times \left(\begin{matrix}\text{total} \\ \text{call}\end{matrix}\right)$

$= (k) * \log n$

$= k \log n$

T.C

$O(k * \log n)$

$O(\log n)$

~~total~~

total calls

$= a$

$= \log n$

a times

$n \rightarrow \frac{n}{2^0}$

$n/2 \rightarrow \frac{n}{2^1}$

$n/4 \rightarrow \frac{n}{2^2}$

$n/8 \rightarrow \frac{n}{2^3}$

$\vdots$

$1 \rightarrow \boxed{\frac{n}{2^a}}$

Total Space

$\left(\begin{matrix}\text{1 call ka} \\ \text{space}\end{matrix}\right) \times (\text{total call})$

(k)

$\log n$

$\frac{n}{2^a} = 1$

$n = 2^a$

$\log n = a$

$a = \log n$

S.C

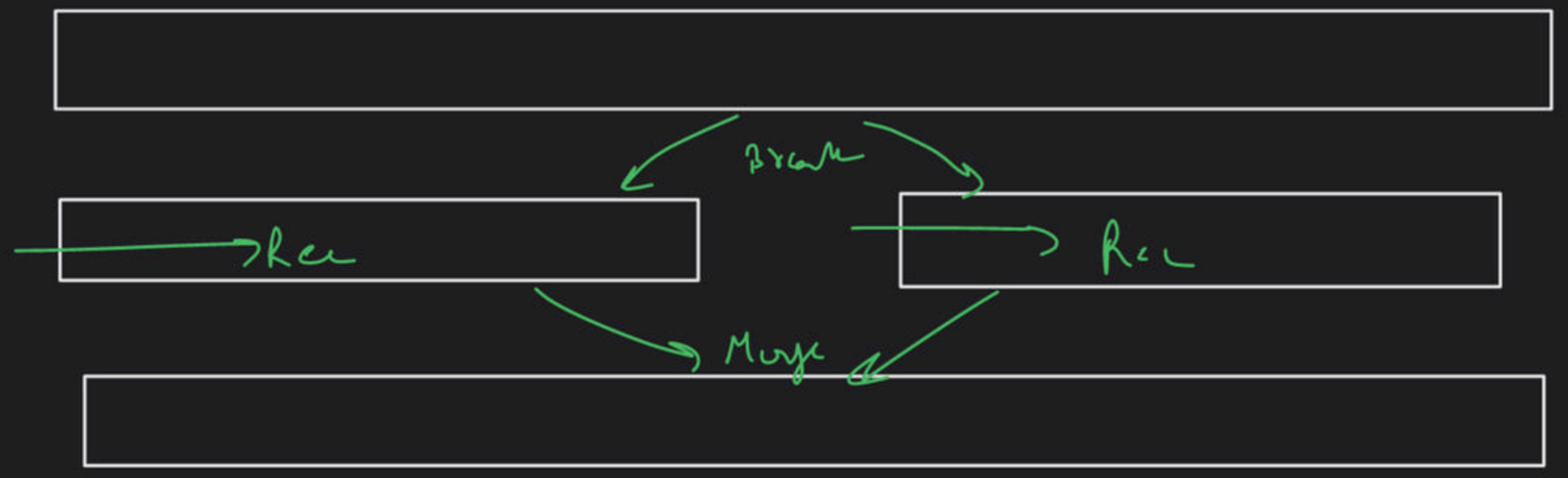
$O(k * \log n)$

$O(\log n)$

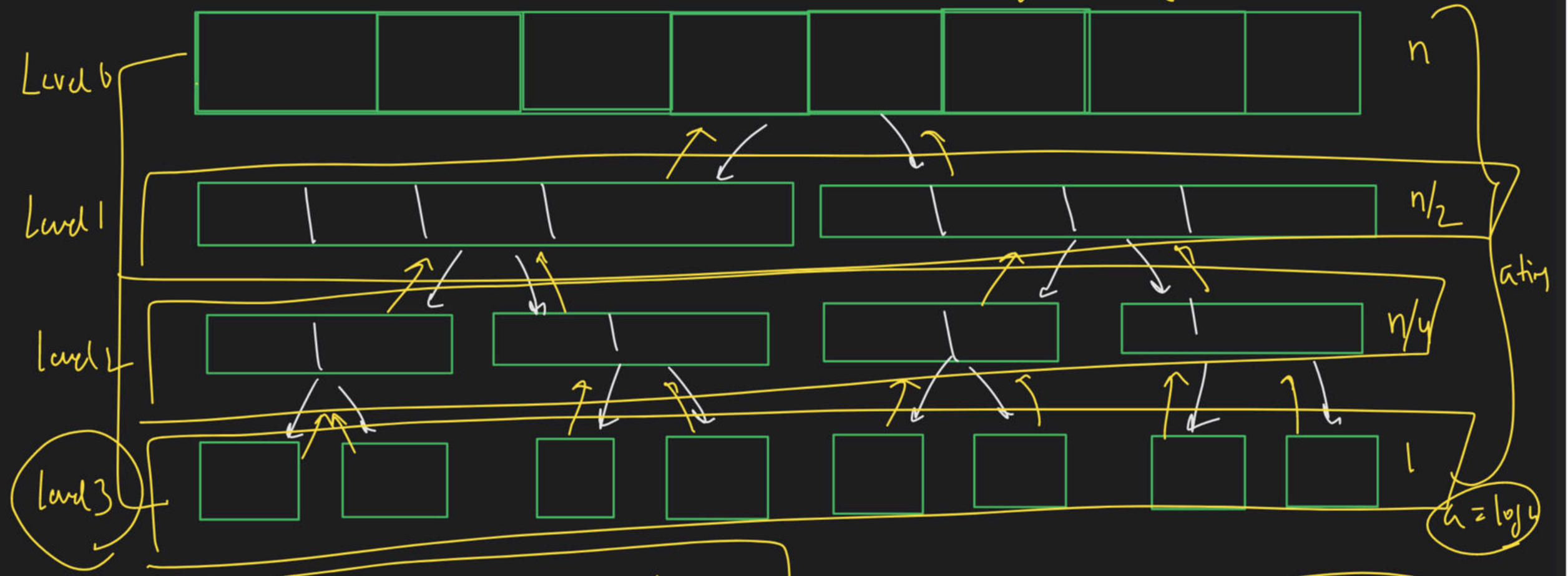
```
bool binarySearch (arr, s, e)
{
    // B.C                                  → K1
    // mid calculation                      → K2
    if ( — , — )                            → K3
    {
        // left call
    }
    else
    {
        // right call
    }
}
```

$$\frac{K_1 + K_2 + K_3}{K}$$

Merge Sort:-



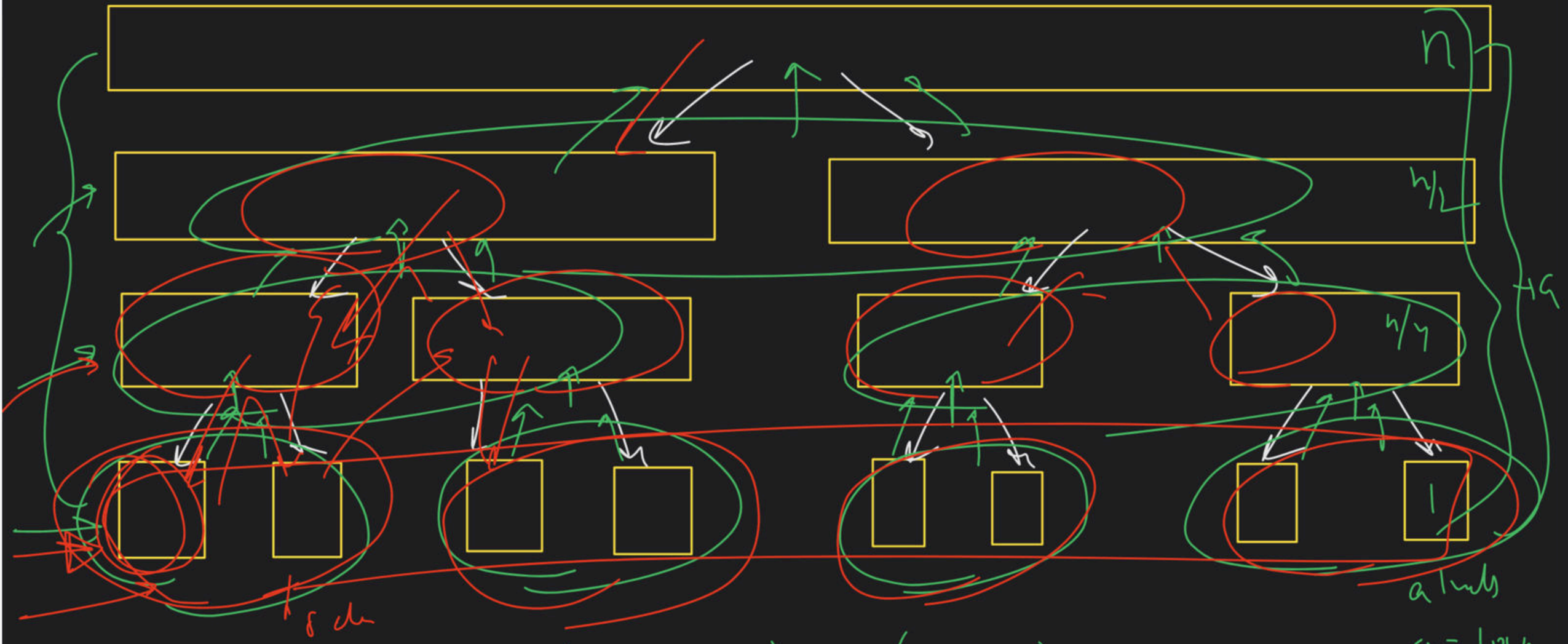
1 Level → merge → $O(n)$



total no of level = Θ = $\log n$

total time → (1 level ka time) × (total level) = $n \times \log n$
= $O(n \log n)$

2 Kaam
↳ break
↳ merge



$a = \log n$

Total time = (1 level merge time) * (total levels)

$= n * \log n \rightarrow O(n \log n)$

Space Complexity

$$\left(\frac{n}{2}+\frac{n}{2}\right)+\left(\frac{n}{4}+\frac{n}{4}\right)+\left(\frac{n}{8}+\frac{n}{8}\right)\ \cdots\ (1+1)$$

$$\left[\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\ \cdots\ 1\right]$$

$$\cancel{n\left[\frac{n}{2}+\right.}$$

$$n\left[\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\ \cdots\ 1\right]$$

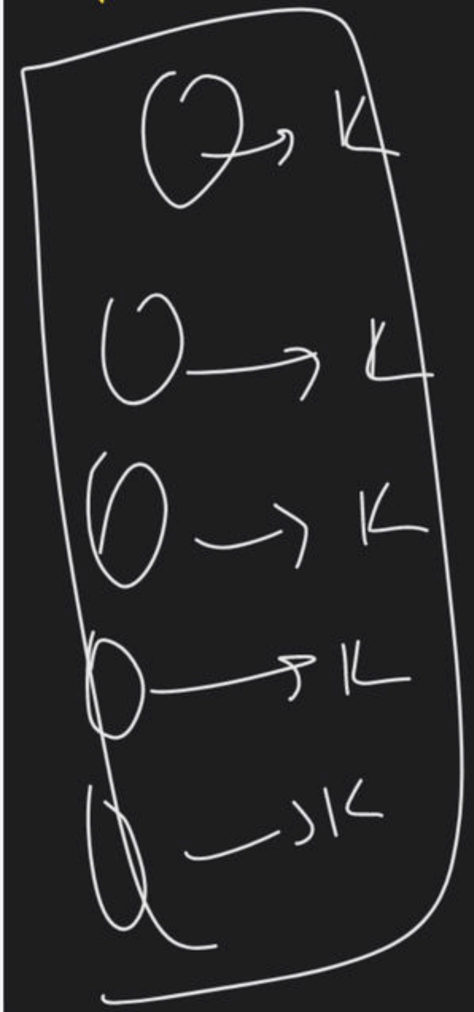
$$2 = 2n \longrightarrow O(n)$$

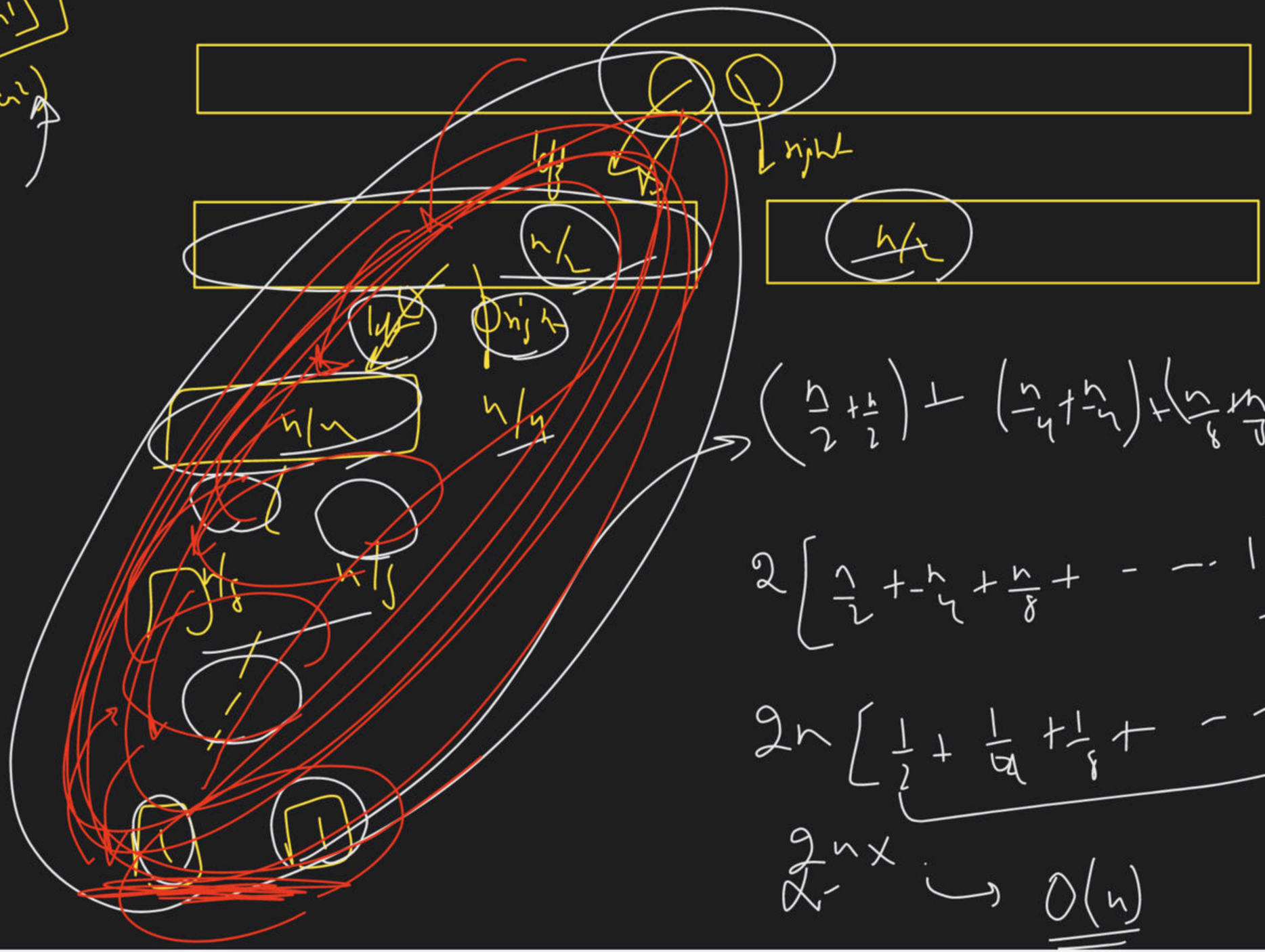
S.C

len = mid - l + 1
len2 = r - mid

int* left = new int [len1]

int* right = new int [len2]





$$\left(\frac{n}{2}+\frac{n}{2}\right) + \left(\frac{n}{4}+\frac{n}{4}\right) + \left(\frac{n}{8}+\frac{n}{8}\right) = (1+1)$$

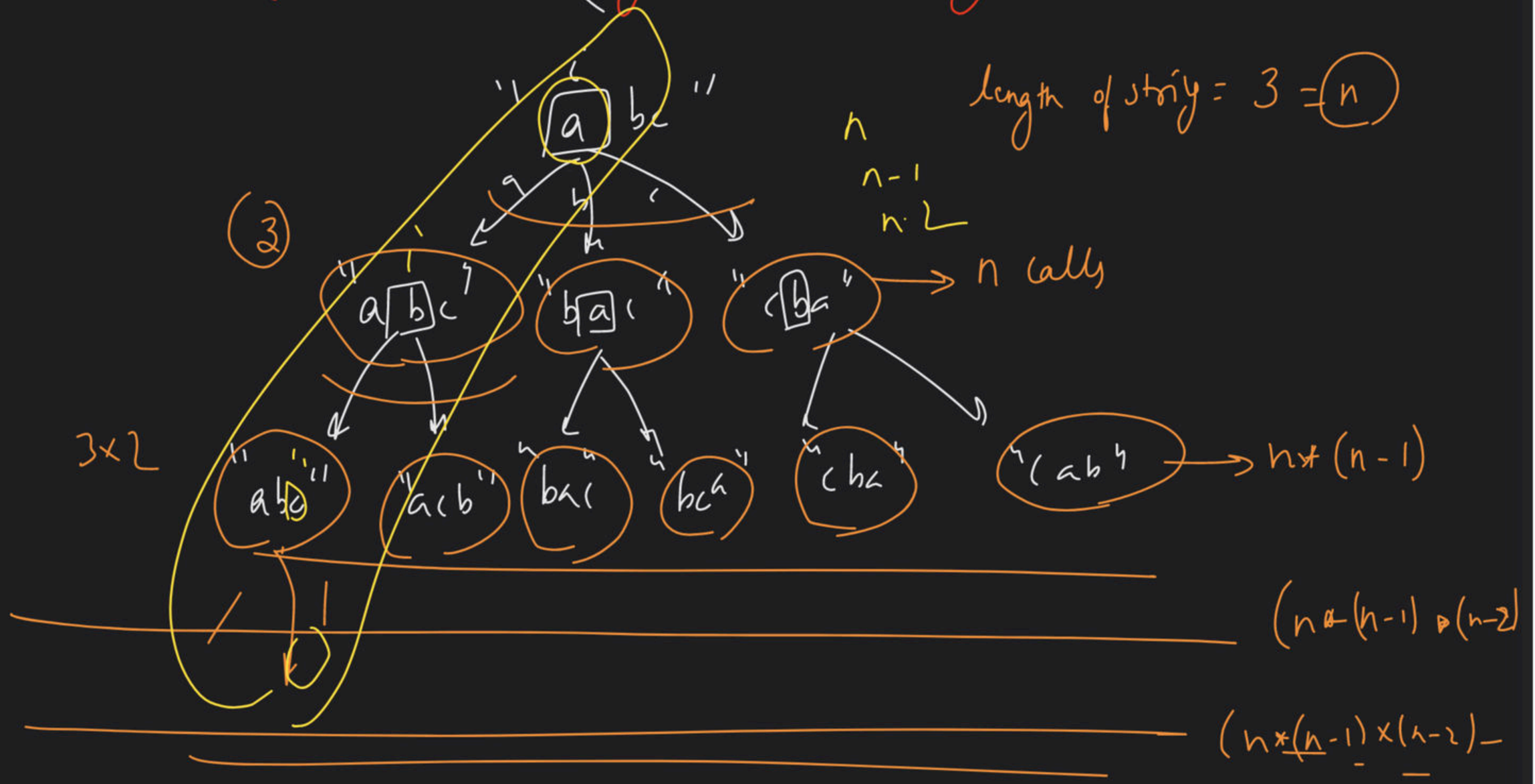
$$2\left[\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\cdots 1\right]$$

$$2n\left[\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots 1\right]$$

$$2^{n}x \rightarrow O(n)$$

Permutations of string (using Backtracking) → H/W

length of string = 3 = n



12 char

n choices

12 calls

1 call

(n-1) choices

12 × (n-1) choices → 12 * 11

12 × 11 → calls

1 call

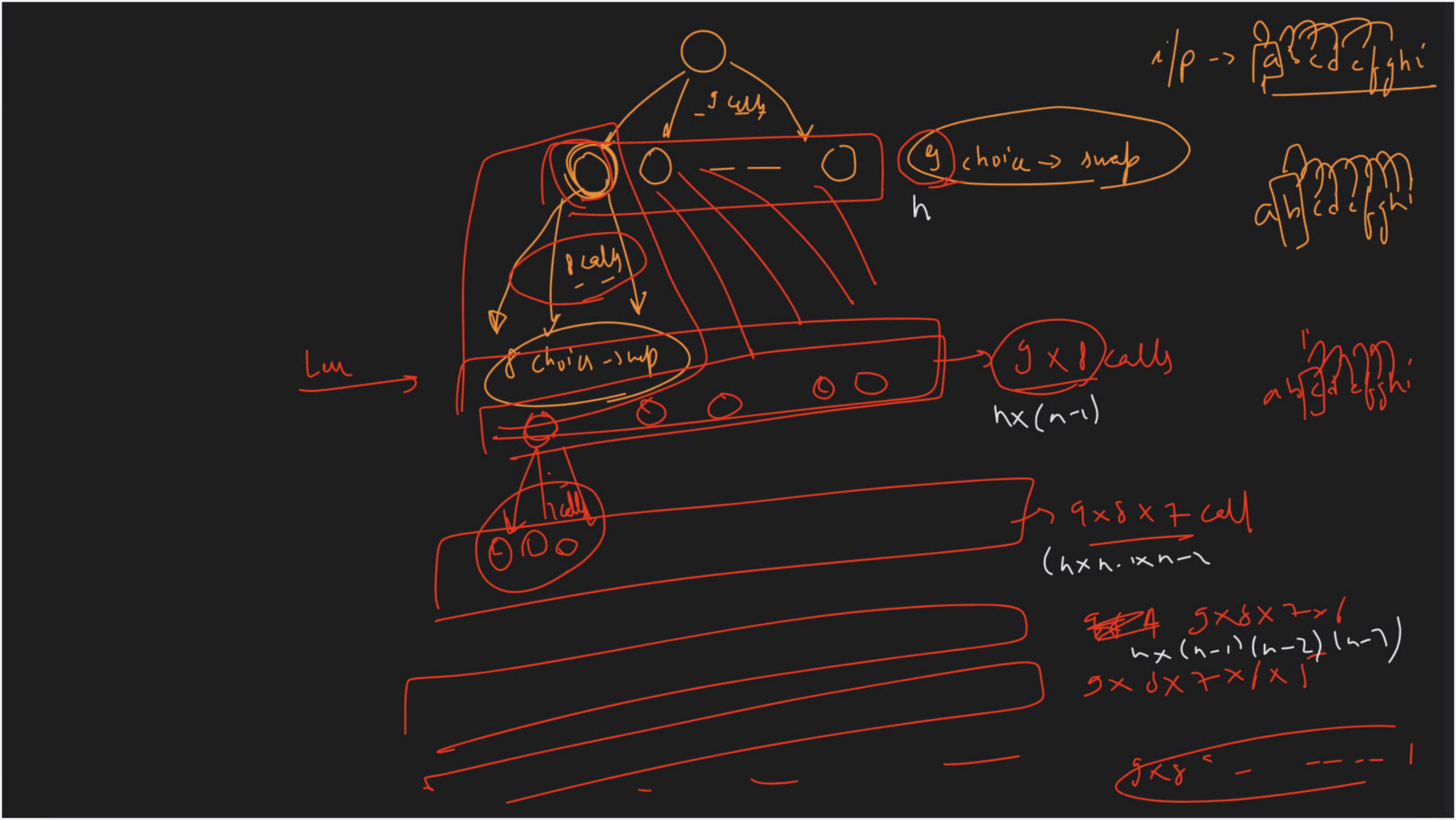
n-2 choices

1 step

12 * 11 × 10

12 × 11 × 10 ... 1

n^n


i/p -> abcdefghi
9 choice -> swap
h
9 calls
8 calls
8 choice -> swap
Lm
9 x 8 calls
n x (n-1)
7 calls
9 x 8 x 7 call
(n x n-1 x n-2
9 x 8 x 7 x 6
n x (n-1) (n-2) (n-3)
9 x 8 x 7 x 1 x 1
9 x 8 ...

$\cancel{n^3} + n^5 = n^5$

$n!$

Total Call

$\hookrightarrow (n) + (n \times (n-1)) + (n * (n-1) * (n-2) + \cdots (n \times (n-1) \times \cdots 1)$

n^n → Now →

$= n \left[1 + (n-1) + (n-2)(n-2) + \cdots ((n-1)(n-2)(n-3) \times \ldots) \right]$

$= n! \left[\frac{1}{(n-1)(n-2)\cdots 1} * 1 - \cdots + 1 \right]$

$= n! * n = O(\cancel{n} * n!) \rightarrow O(n!)$

~~Total~~

Total time :-

(1 call ka time) × (total call)

K ×

$P_{cm} \rightarrow O(N!)$

$5 \cdot C \rightarrow ?$

$O(n+1)$

$O(n)$

N^n $\quad$ $N!$

$N = 10$

10^{10} $\quad$ 101

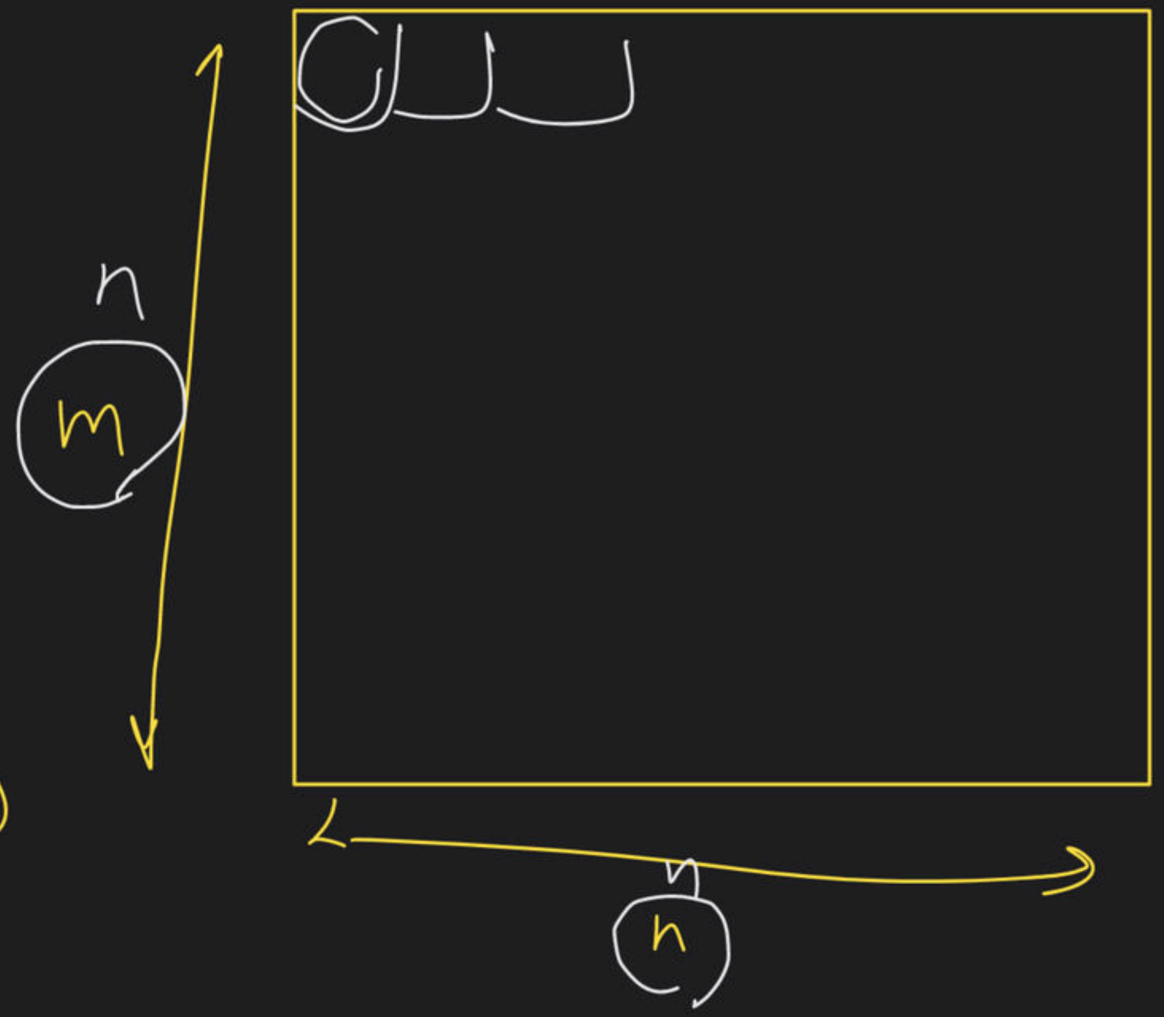
Rat in a maze:-

rows → m

cols → n

total cells

→ m * n



solve ()

{

// D C → K

// D isSafe → 3 cases

// L

// R

// U

}

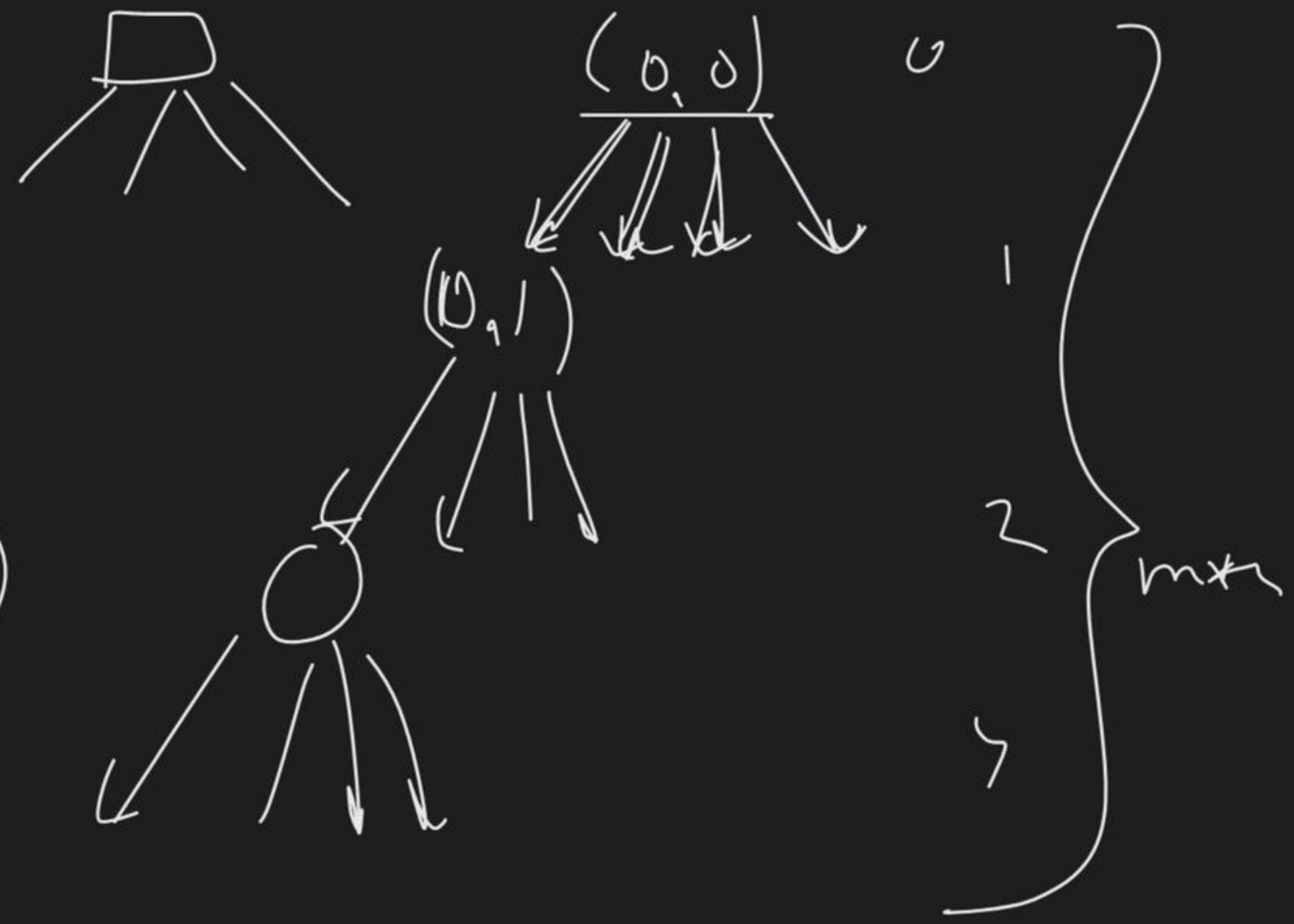
1 Last → K

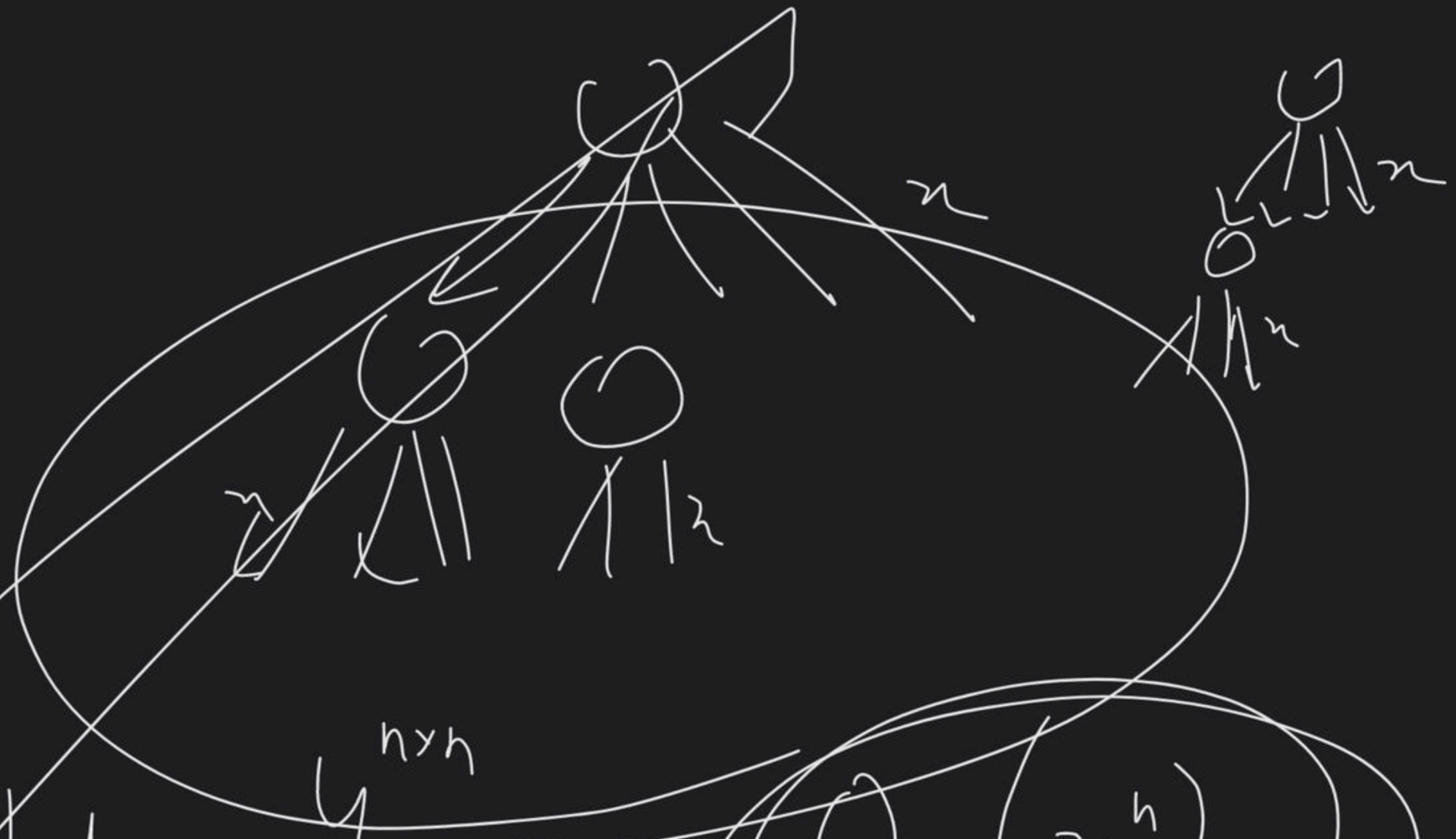
Bhul gao

total options to explore on a single cell → 4

total cell → m*n

Total time $O(4^{m*n})$





$y^{n \times n}$

$= y^{hL}$

$O(n^h)$

hlwd

score cell

$m*n$

R.M

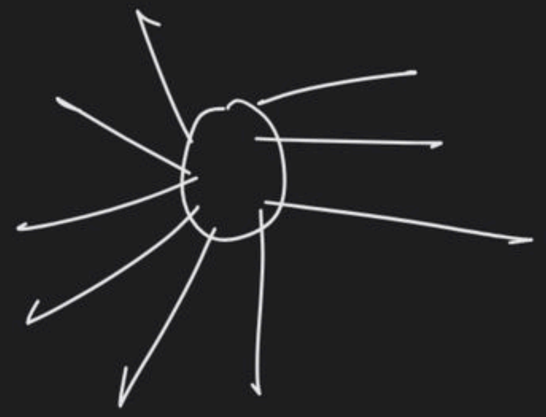
S.C → ?

$O(m*n) + O(m*n)$

$O(2mn)$

$O(mn)$

Sudoku

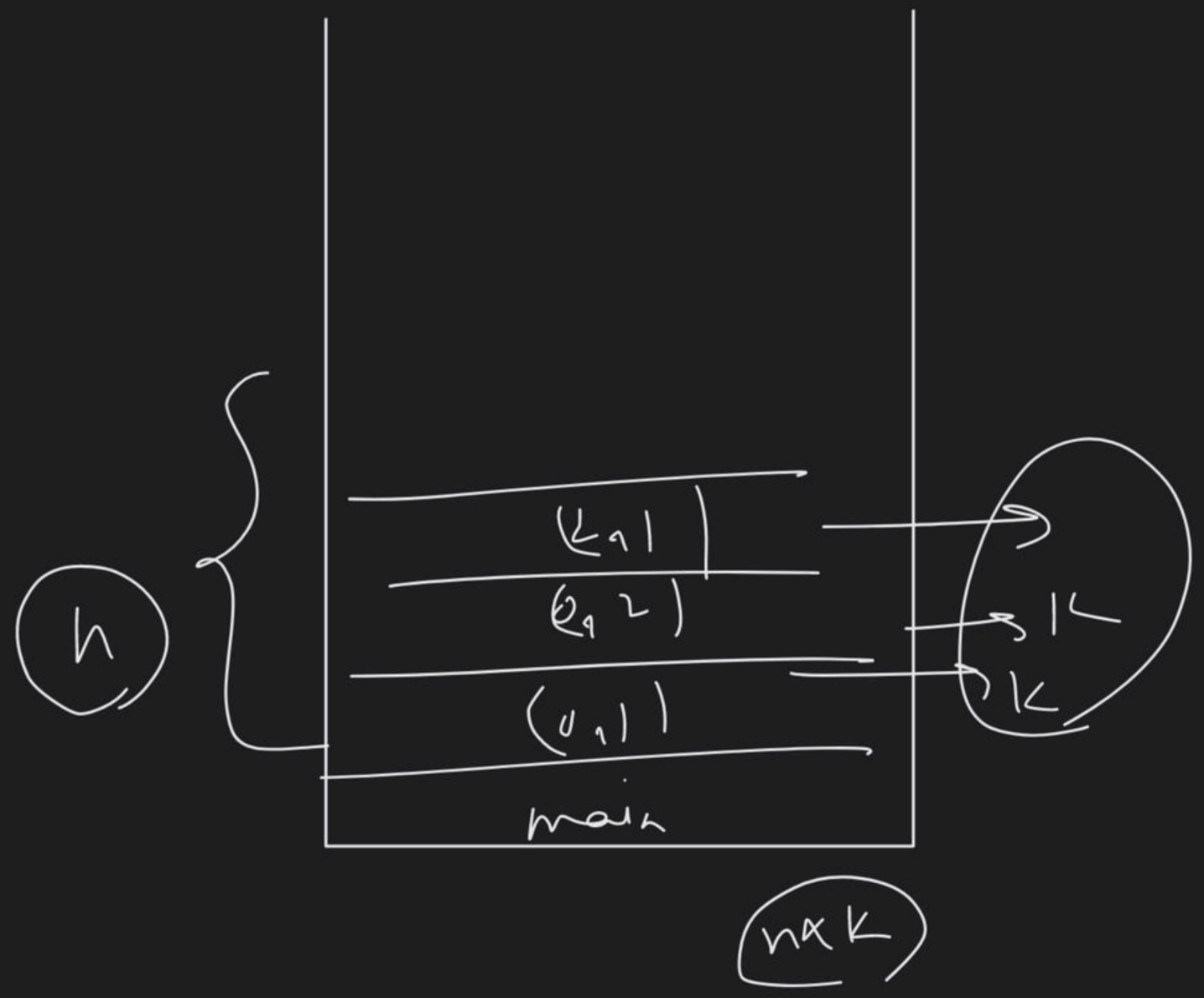


9 choices → 1 cell

total cell → m*n

S.C → O(m*n)

T.C → O(9^(m*n))


h
(k, 1)
(k, 2)
(v, 1)
main
k
k
max k

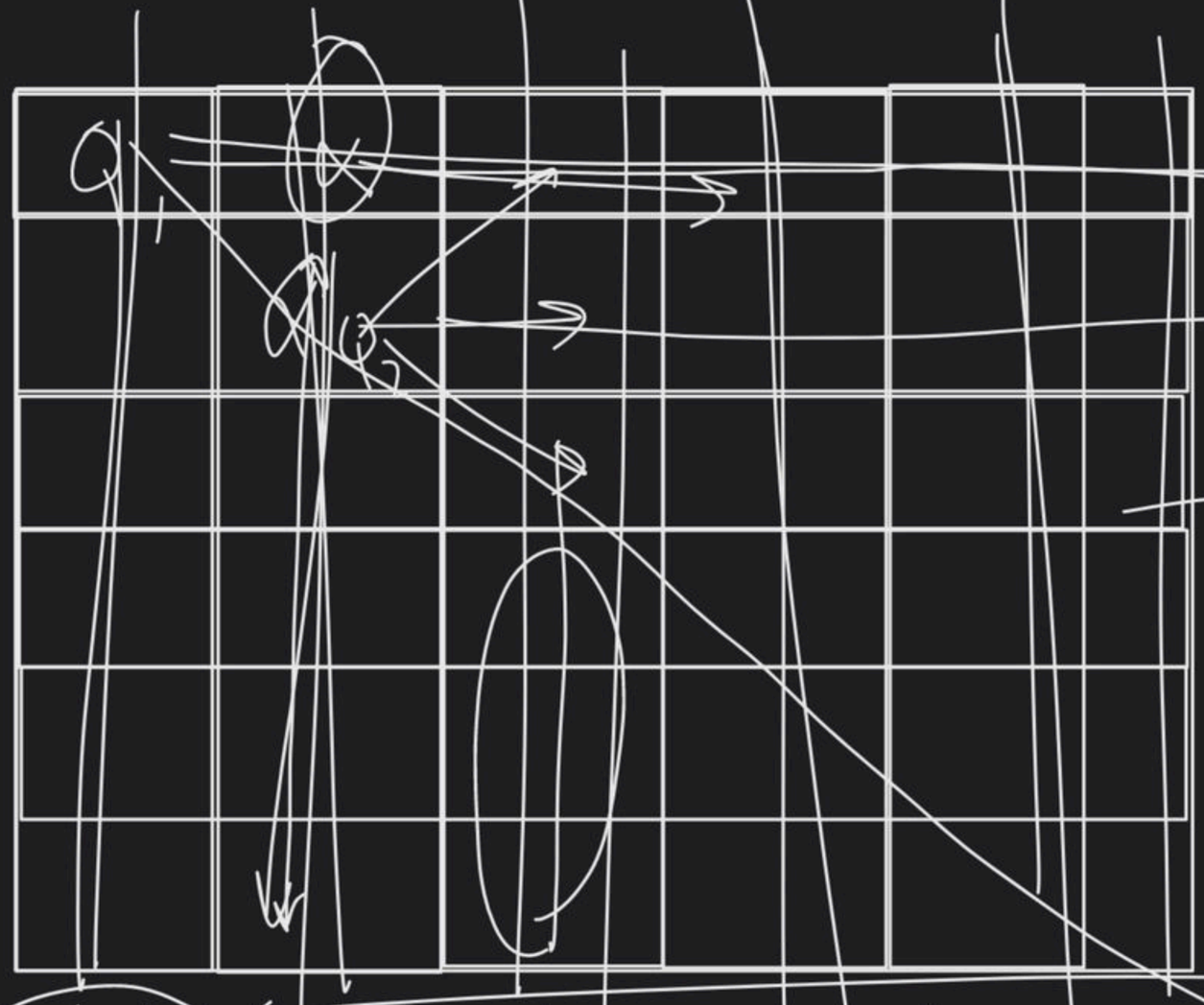
N - Queen

$n = 6$

Worst Case

N^n

Avg → $n!$

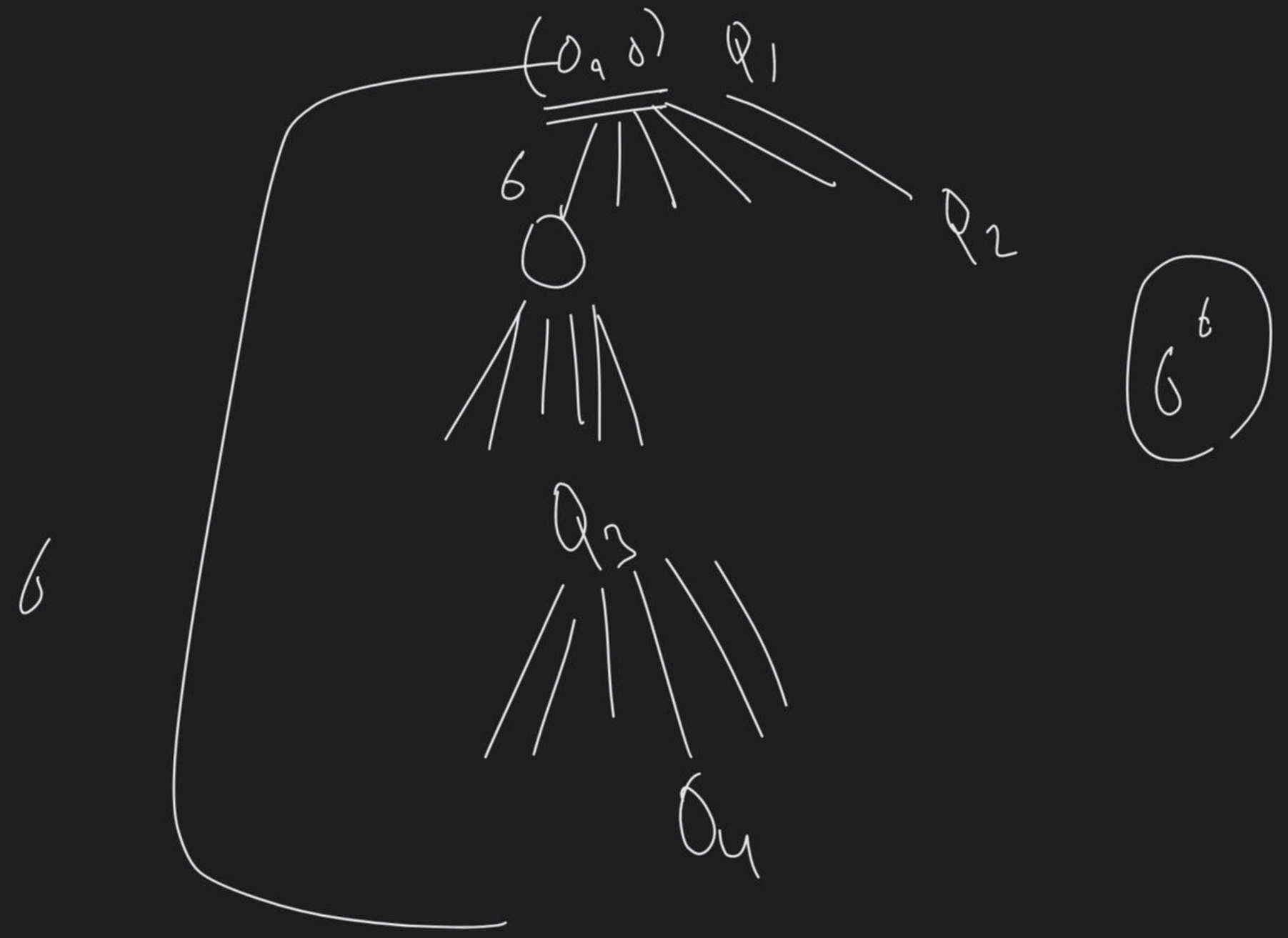


$T.C \rightarrow O(n^n)$

choices $(n-1) \times (n-2) \times (n-3) \times (n-4) \ldots 1$

$n-3$

$T.C \rightarrow n!$


(Q_0, δ)
Q_1
σ
Q_2
σ^t
Q_3
σ
Q_4

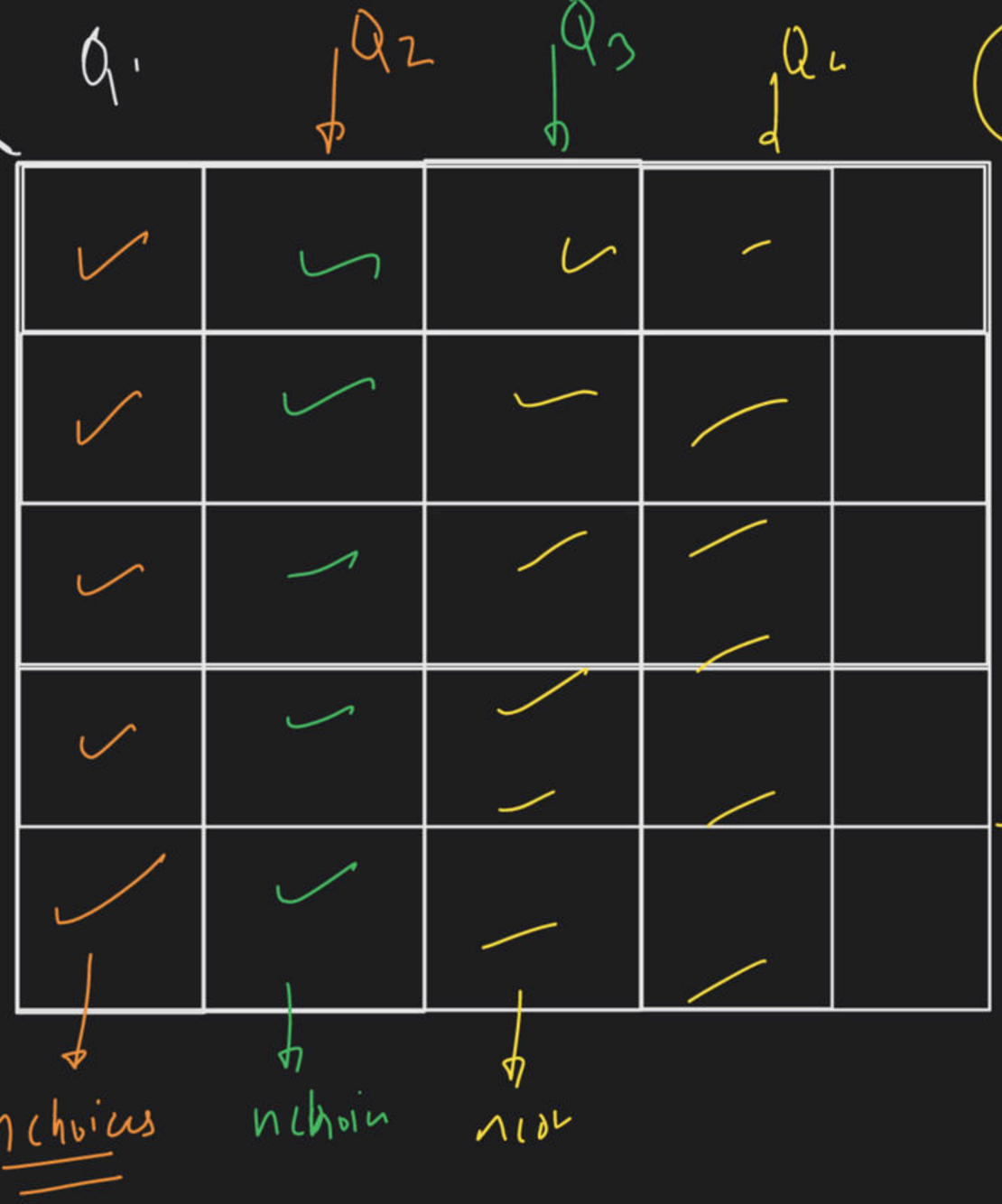
$n = 5$

5 → rows → 5

5 → cols

5 – Queen

Space



$O(n^n)$

Space

board → $n * n$

ans → $n * n$

assume single solution

variable → K

3 maps → n

$O(n * n)$

solve (board, col, n, ans)

col → K

ans → $n \times n$

$n \times n$

α

map → n

n

n

α

|| B.C

for {

|| ———

|| R.R

|| BACKTRACK

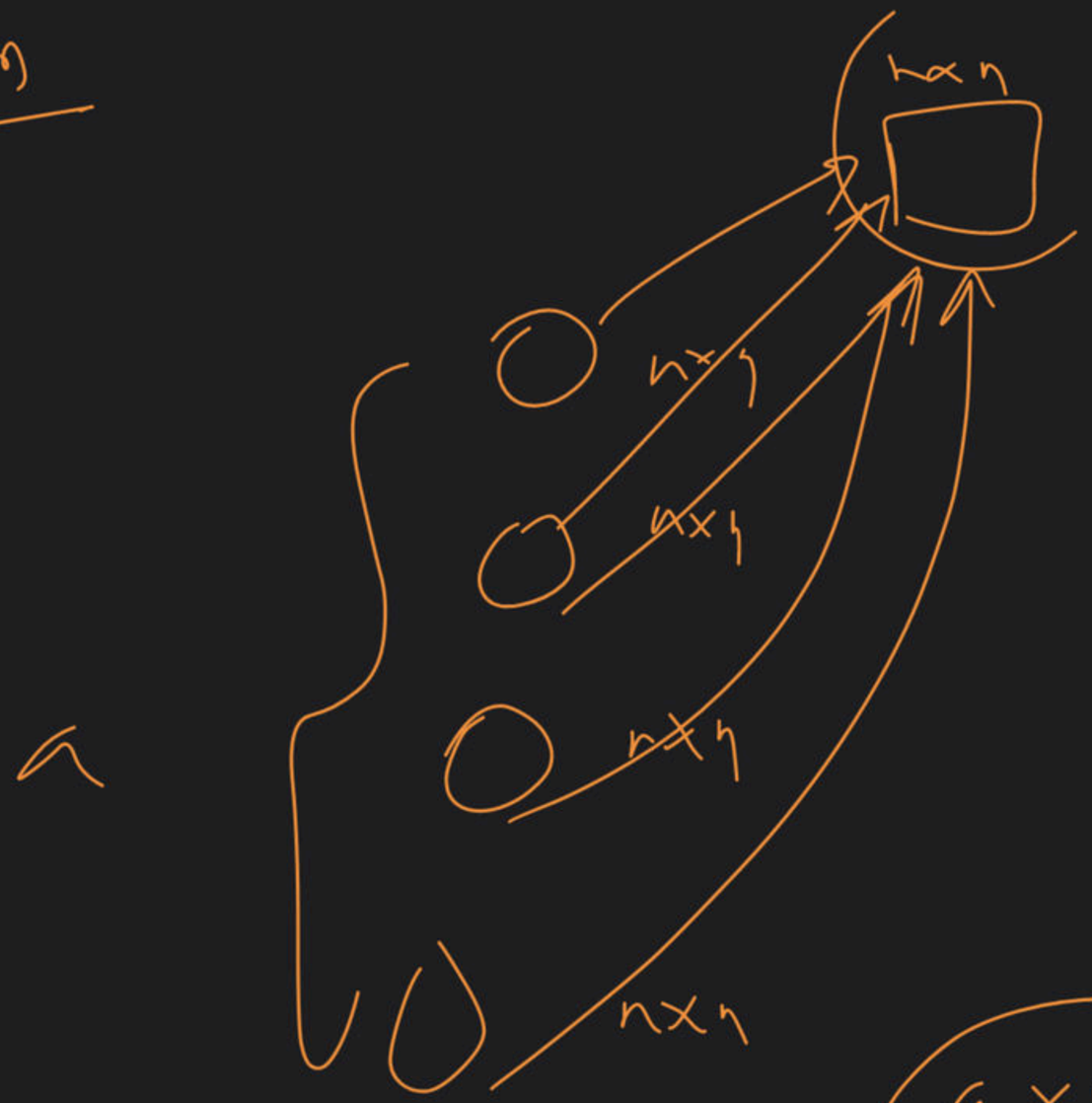
$O(n \times n)$

S.C

Single solution

3 3

Phone Keypad Problem

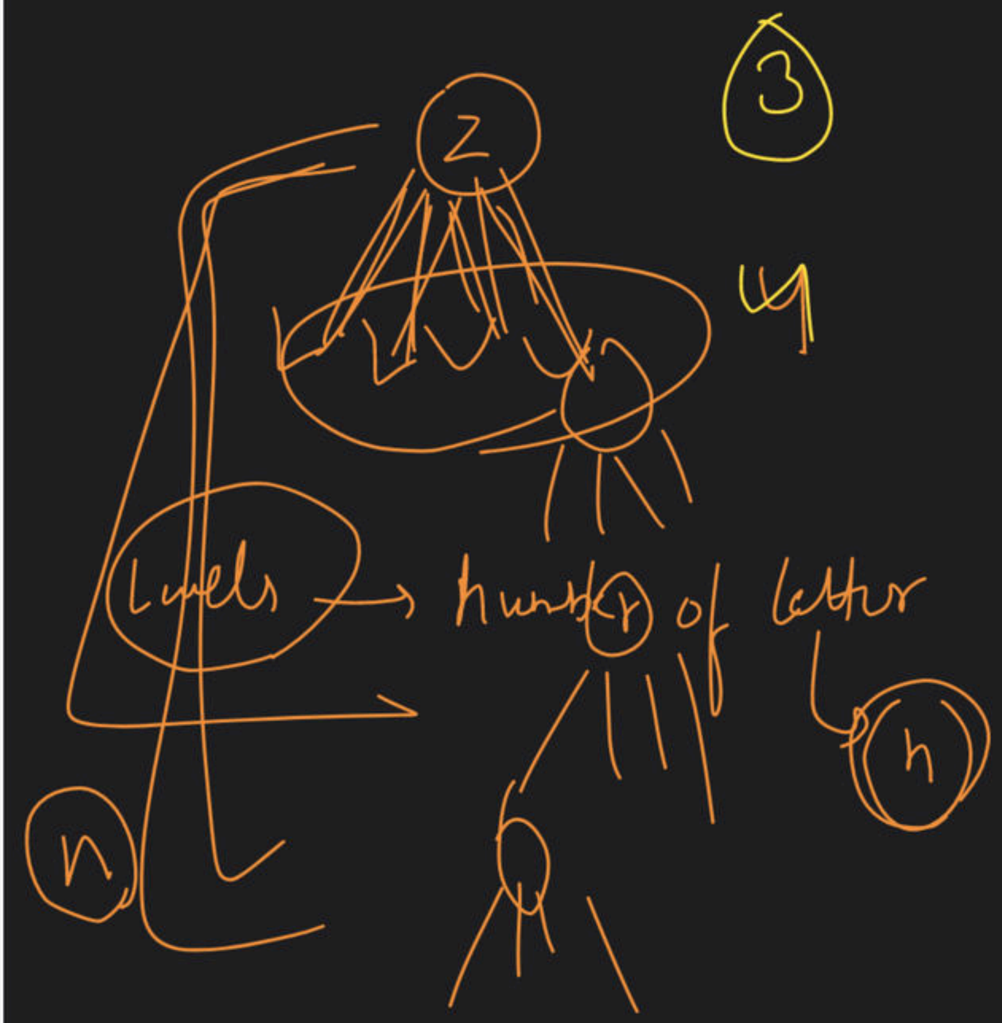


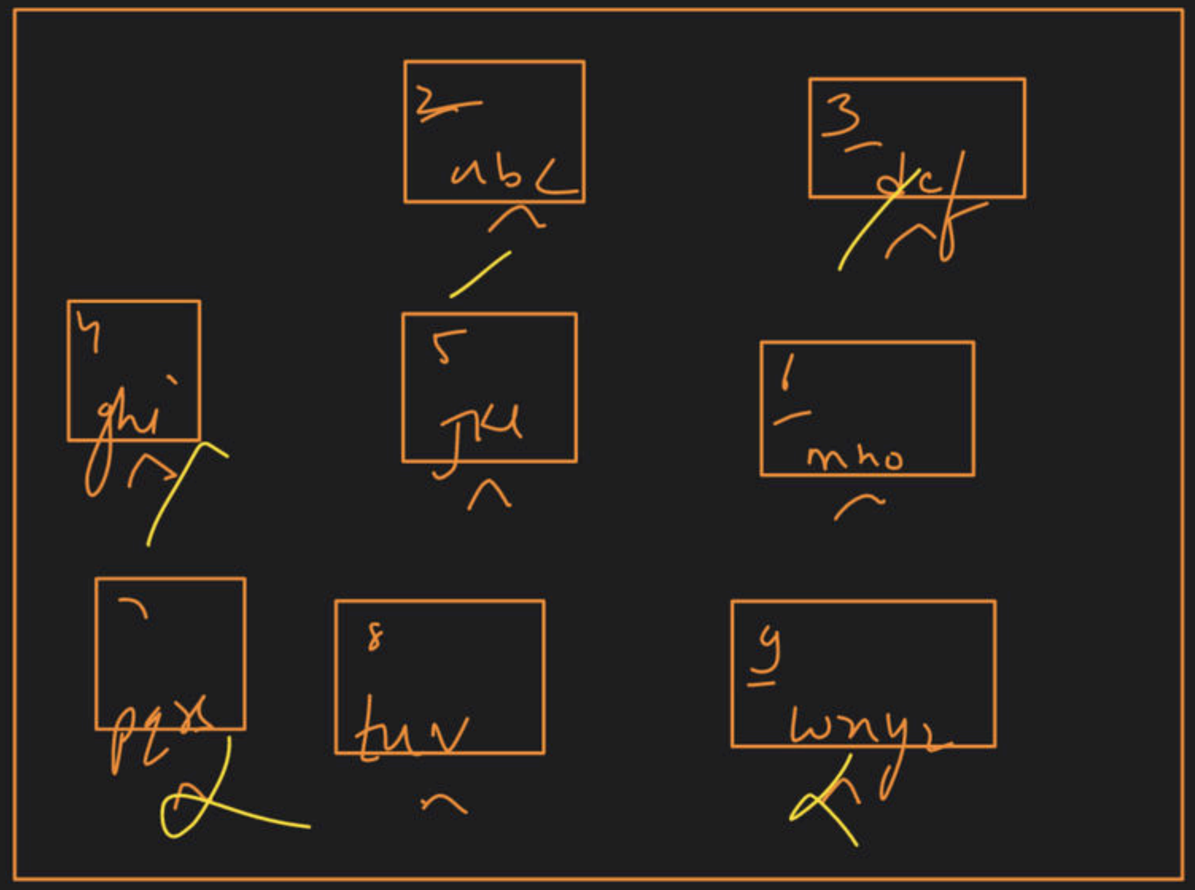
$a \times n \times n$

n
4^n

Phone Keypad

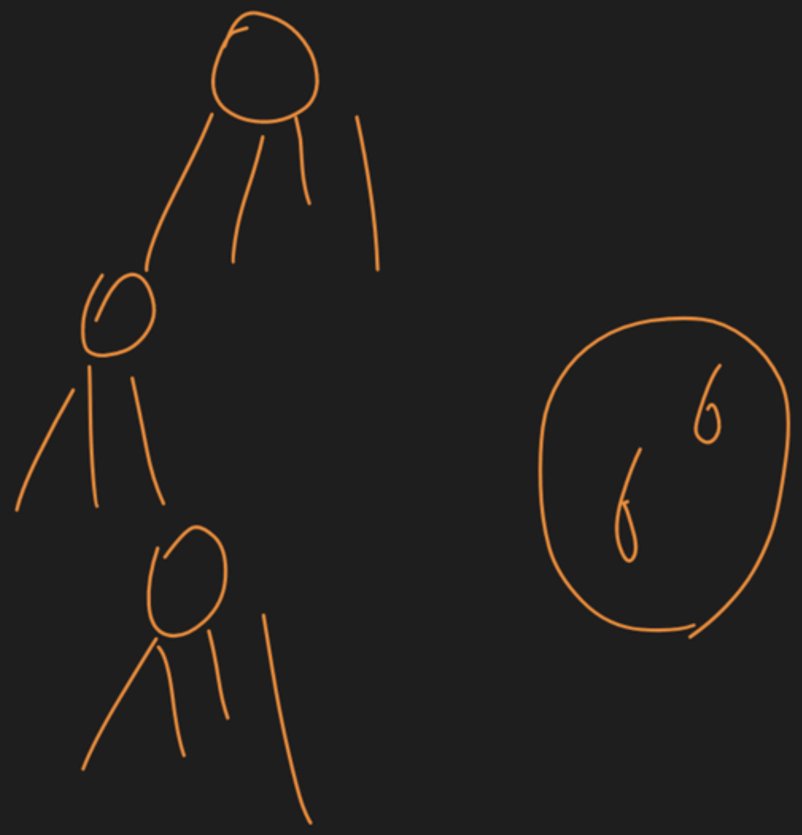
2345

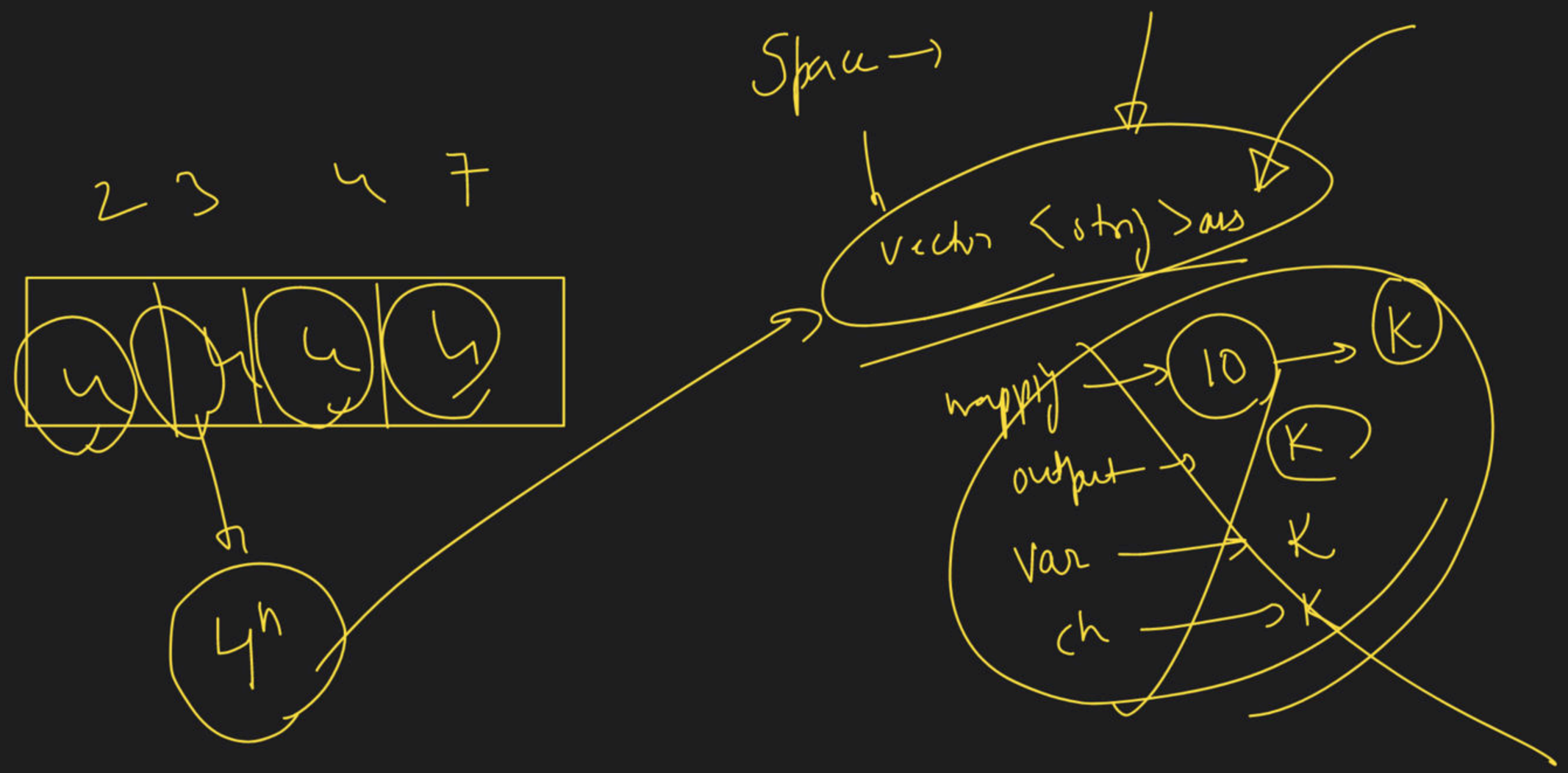




S.C

→ T.C → $O(4^n)$




2 3 4 7
4
4
4
4
4^n
Space →
vector <string> ans
mapping → 10 → K
output → K
var → K
ch → K

n

7	9

7 → pqrs

9 → wxyz

pw	rw
px	ry
py	rx
pz	rz
qw	sw
qx	sy
qy	sx
qz	sz

4^n

4	4	4	4	4	4

→ 16

→ 4 × 4 → 16

16 strings

$n = 2$ case

→ 4^n → total answers

7	9

7 → pqrs

9 → wxyz

pw	qw	rw	sw
px	qx	rx	sx
py	qy	ry	sy
pz	qz	rz	sz

16 → ?

How

4	4

→ $4^2 = 16$

4	4	4	4	4

$\underbrace{\quad\quad\quad\quad\quad}_{n}$ → 4^n

answers

vector <string> ans

↳ 4^n, string

galat ?

$O(4^n)$

1 → 2 → 3 → 4 → 5

n/n

n/n-1

n/n-2

n/n-3

n choices

(n-1) choices

(n-2) choices

n!

level K addition
per complexity