# Time & Space Complexity

→ Linear Search :-

```
bool isPresent (int *arr, int n, int value)
{
    for (int i=0; i<n; i++)
    {
        if (arr[i] == value)
            return true;
    }
    return false;
}
```

T.C → O(n) (n → arr size)

S.C → O(1)

→ Recursion:- Fac / Power / Counting

Factorial → T.C → time required as function of input

$f(n)$

```
int fact (int n)
{
    // Base Calc
    if (n==0)
        return 1;

    // Rec. Call
    return n * fact(n-1);
}
```

$K_1$

$n * fact(n-1)$ → $K_1$ , $T(n-1)$

→ T.C → ?

① Try -> Eq -> Recurrence

$$f(n) = n * f(n-1)$$

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

$$T(n) = K + T(n-1)$$

$$T(n-1) = K + T(n-2)$$

$$T(n-2) = K + T(n-3)$$

n times

$$T(1) = K + T(0)$$

$$T(0) = K_1$$

$$T(n) = n * K + K_1$$

$$T(n) = Kn + K_1$$

$$= \alpha$$

$$T(n) = Kn$$

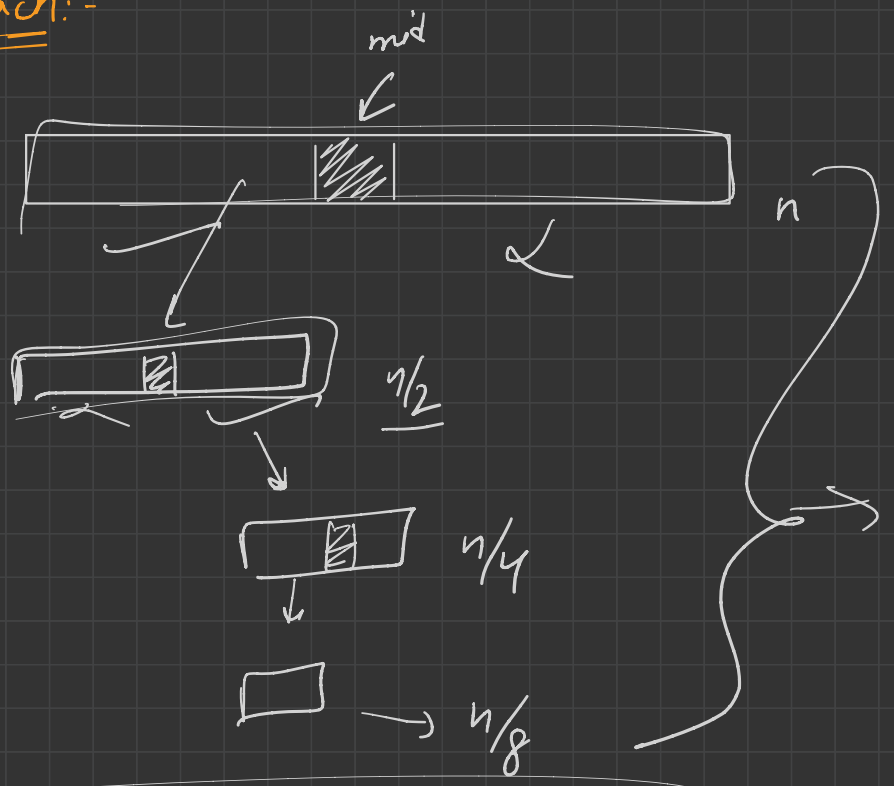
$T(n) = n$

$O(n)$

Factorial

T.C → $O(n)$

# Binary Search:-




$$f(n) = \underbrace{xyz}_{} + f(n/2)$$

$$T(n) = K_1 + K_2 + T(n/2)$$

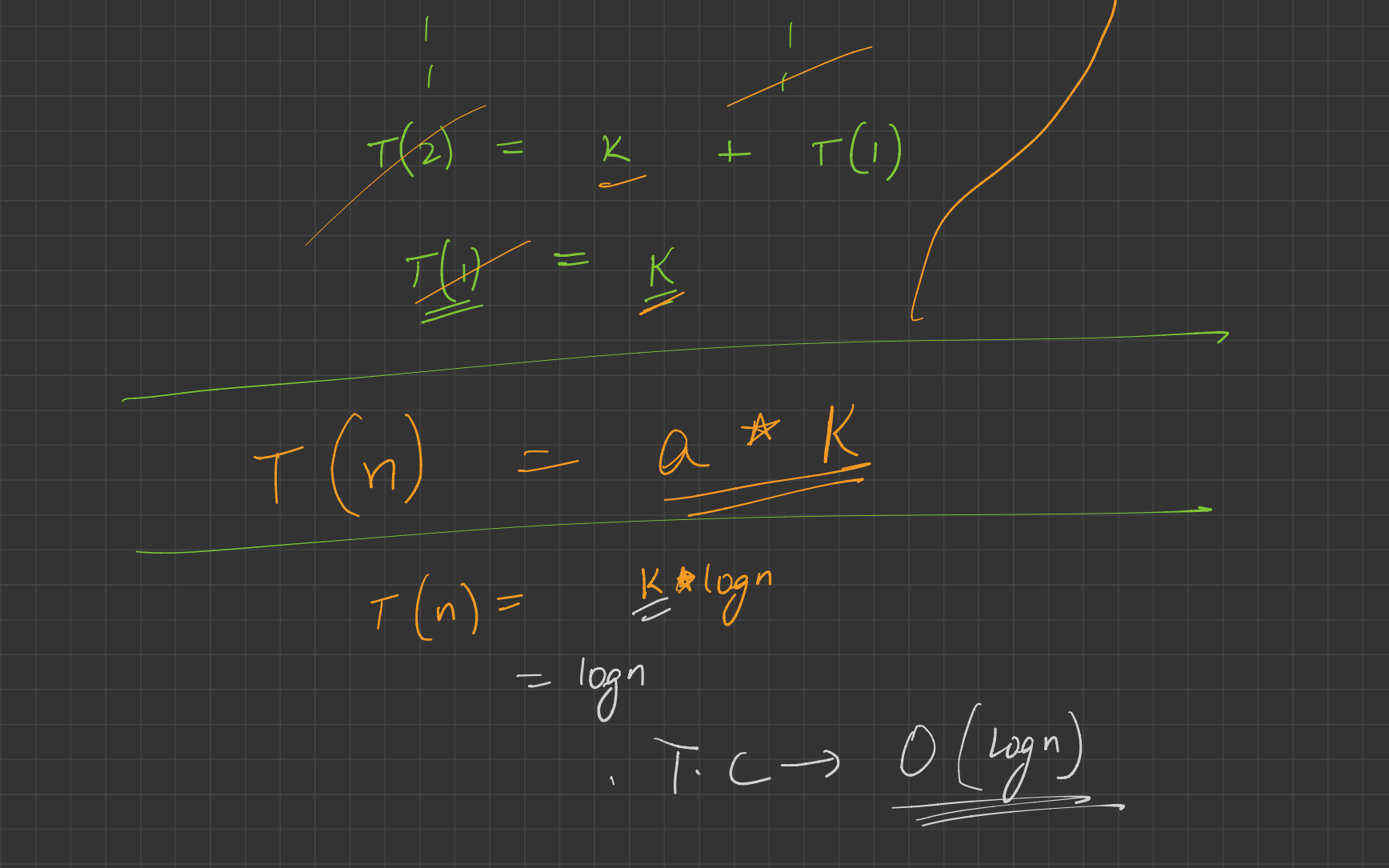
$$K$$

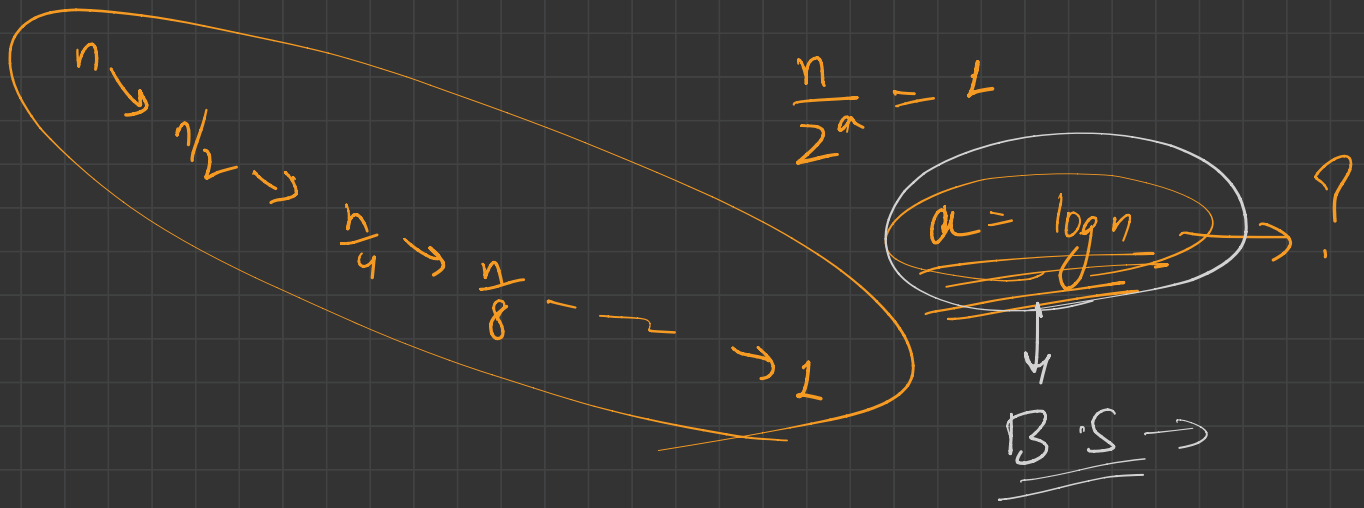
$$T(n) = K + T(n/2)$$

$$T\left(\frac{n}{2}\right) = K + T\left(\frac{n}{4}\right)$$

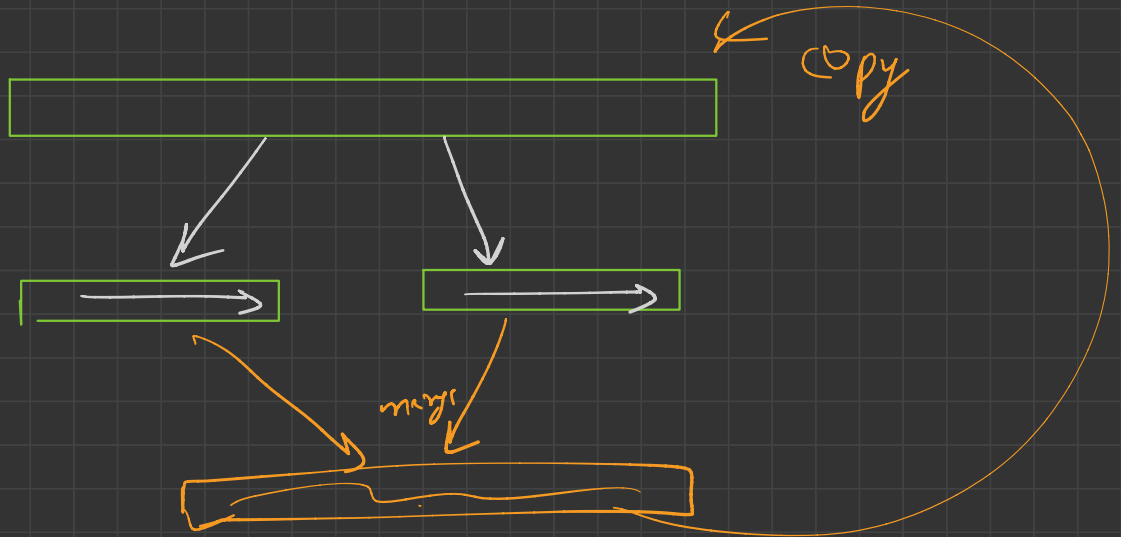
$$T\left(\frac{n}{4}\right) = K + T\left(\frac{n}{8}\right)$$

a times


T(2) = K + T(1)
T(1) = K
T(n) = a * K
T(n) = K * log n
= log n
∴ T.C → O(log n)


n → n/2 → n/4 → n/8 → 1
n/2^a = 1
a = log n
?
B.S →

→ Merge Sort




Steps

→ break 2 array left & right

→ Rec → sort (left / Right)

→ new array → merge → K3 n

→ copy new array content → original array → K4 n

left

right part

$$T(n) = K_1 + K_2 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + K_3 n + K_4 n$$

$K_1 + K_2 = K$

$$T(n) = K + 2T\left(\frac{n}{2}\right) + n(K_3 + K_4)$$

$K_3 + K_4 = K_5$

$$T(n) = K + 2T\left(\frac{n}{2}\right) + nK_5$$

$K \approx 0$

$$T(n) = 2T\left(\frac{n}{2}\right) + nK_5$$

$$T(n) = 2T\left(\frac{n}{2}\right) + nk$$

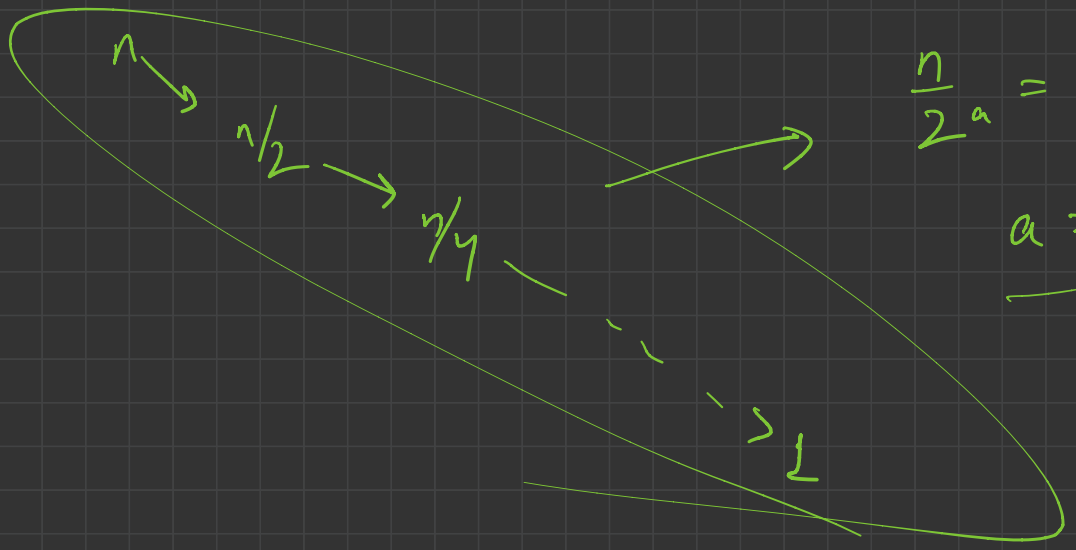
$$2\times\left[T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}k\right] = 4T\left(\frac{n}{4}\right) + nk$$

$$4\times\left[T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}k\right] = 8T\left(\frac{n}{8}\right) + nk$$

a fimes

$$T'(1) = K$$

$$T(n) = a * nK$$




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

$$a = \log n$$

$$T(n) = an$$

$$= \log n \times n$$

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

T.C

→ Fibonacci Series

```
fib( int n)
{
    // B.C
    if (n == 0 || n == 1)
        return n;

    // R.C
    return fib(n-1) + fib(n-2);
}
```

$K_1$ → T.C → $O(2^n)$

$K_2$

$$T(n) = K_1 + K_2 + T(n-1) + T(n-2)$$
$$= K + T(n-1) + T(n-2)$$

$$T(n) = K + T(n-1) + T(n-2)$$

$$T(n-1) = K + T(n-2) + T(n-3)$$

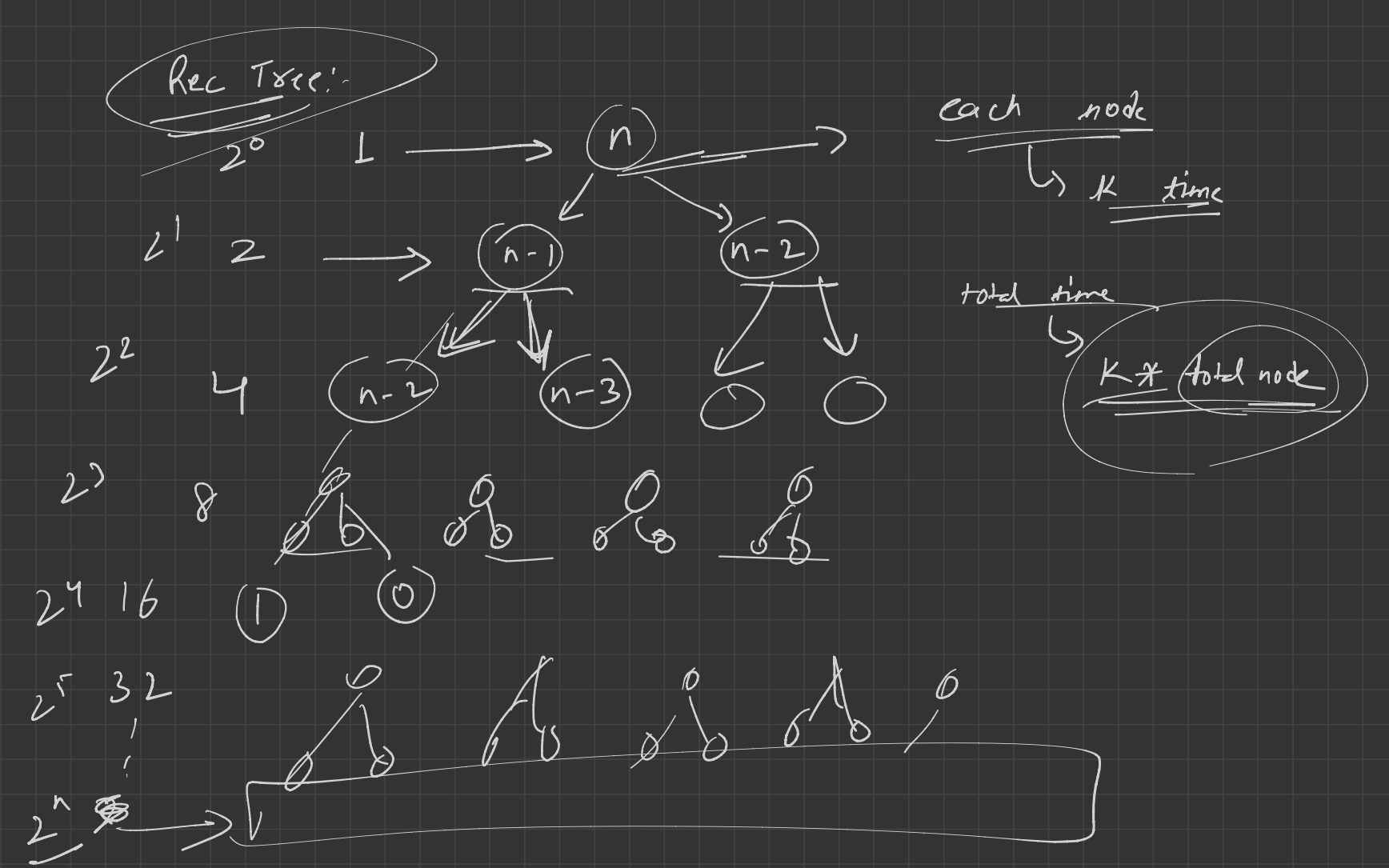
$$T(n-2) = K + T(n-3) + T(n-4)$$

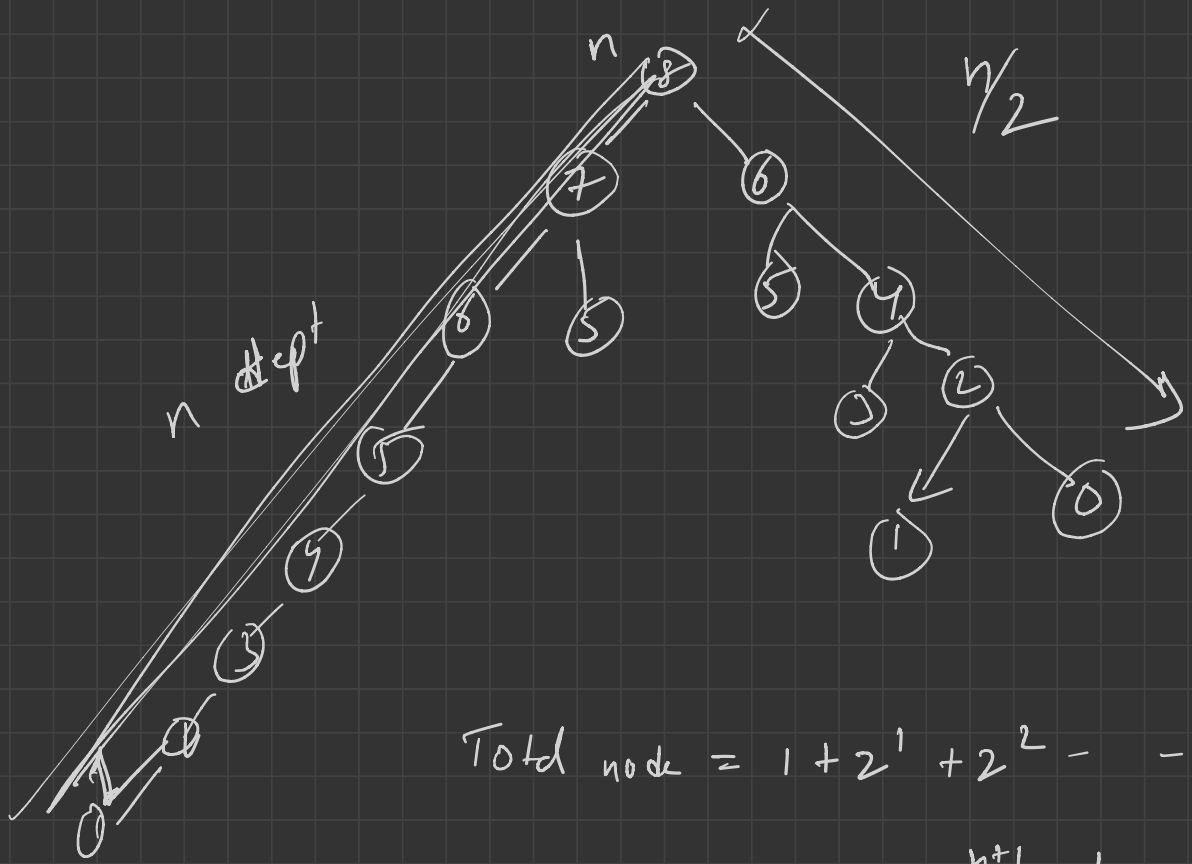
Solve - ?

by Hw

$$T(1) = K_1$$

$$T(0) = K_2$$


Rec Tree:-
$2^0$ 1
n
each node
K time
$2^1$ 2
n-1
n-2
total time
K * total node
$2^2$ 4
n-2
n-3
$2^3$ 8
$2^4$ 16
1
0
$2^5$ 32
$2^n$




Total node $= 1 + 2^1 + 2^2 - - - 2^n$

$= 2^{n+1} - 1$

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

$$= K \times 2^{n+1}$$

$$= K \times 2 \times 2^{n}$$

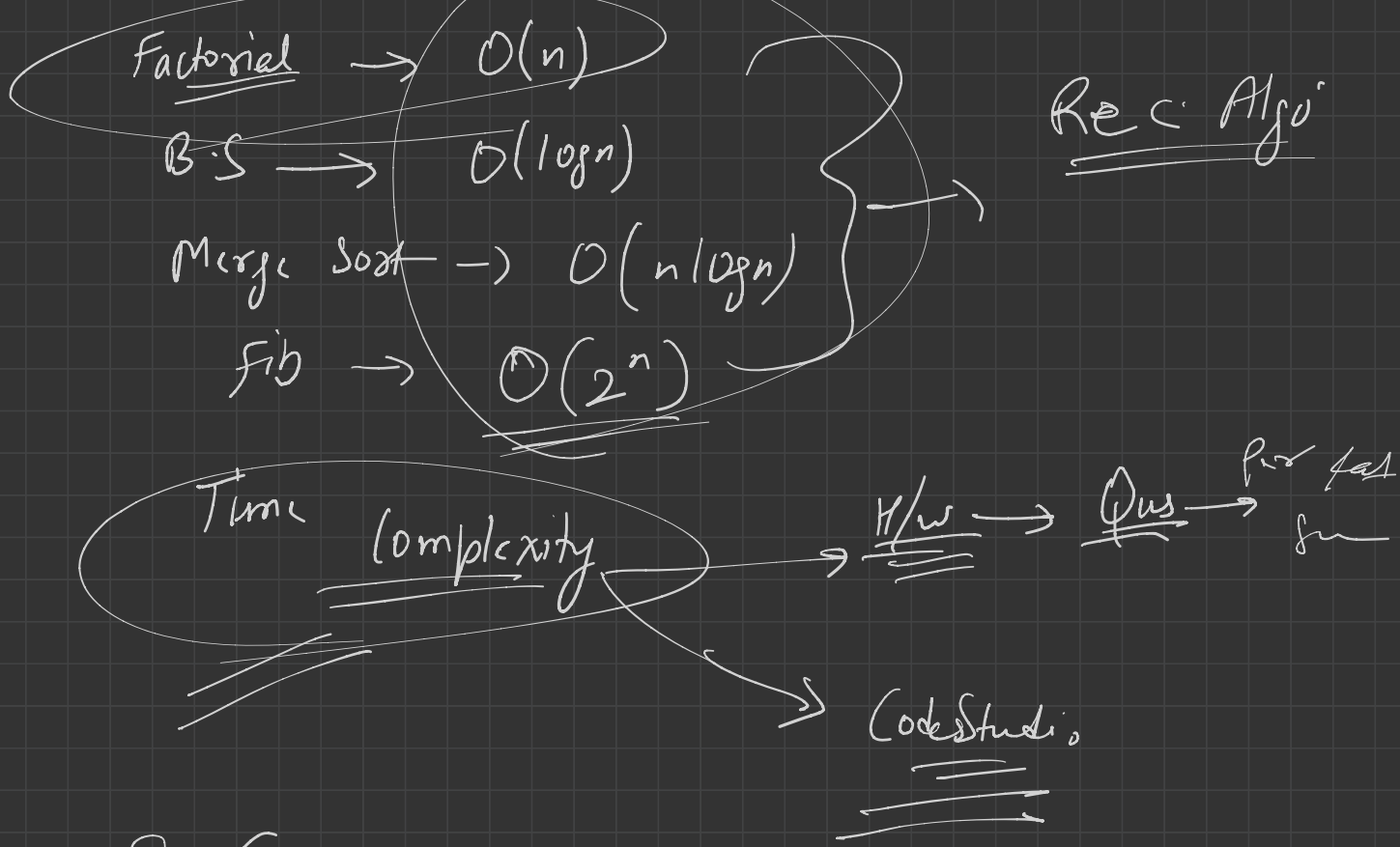
$$= K \times 2^{n}$$

$$= 2^{n}$$

Fib

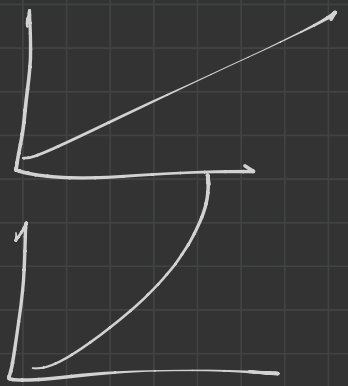
T.C $\longrightarrow O(2^n)$

exponential

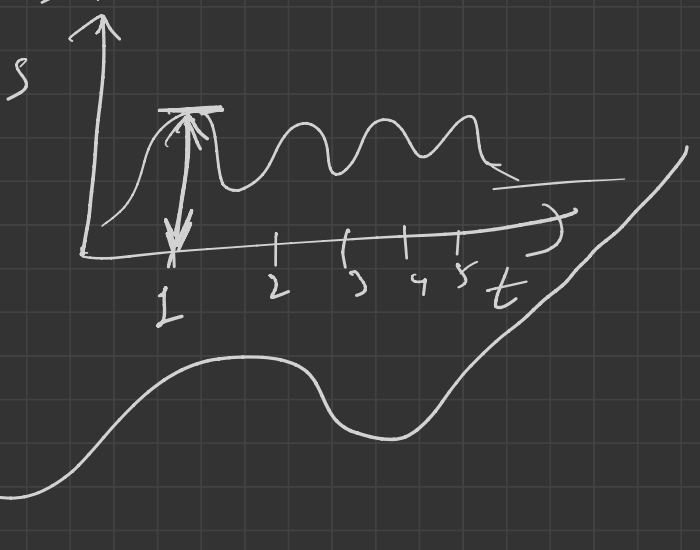

Factorial → O(n)
B.S → O(logn)
Merge Sort → O(nlogn)
Fib → O(2^n)
Rec Algo
Time complexity
H/w → Qus → For fast
CodeStudio
S.C

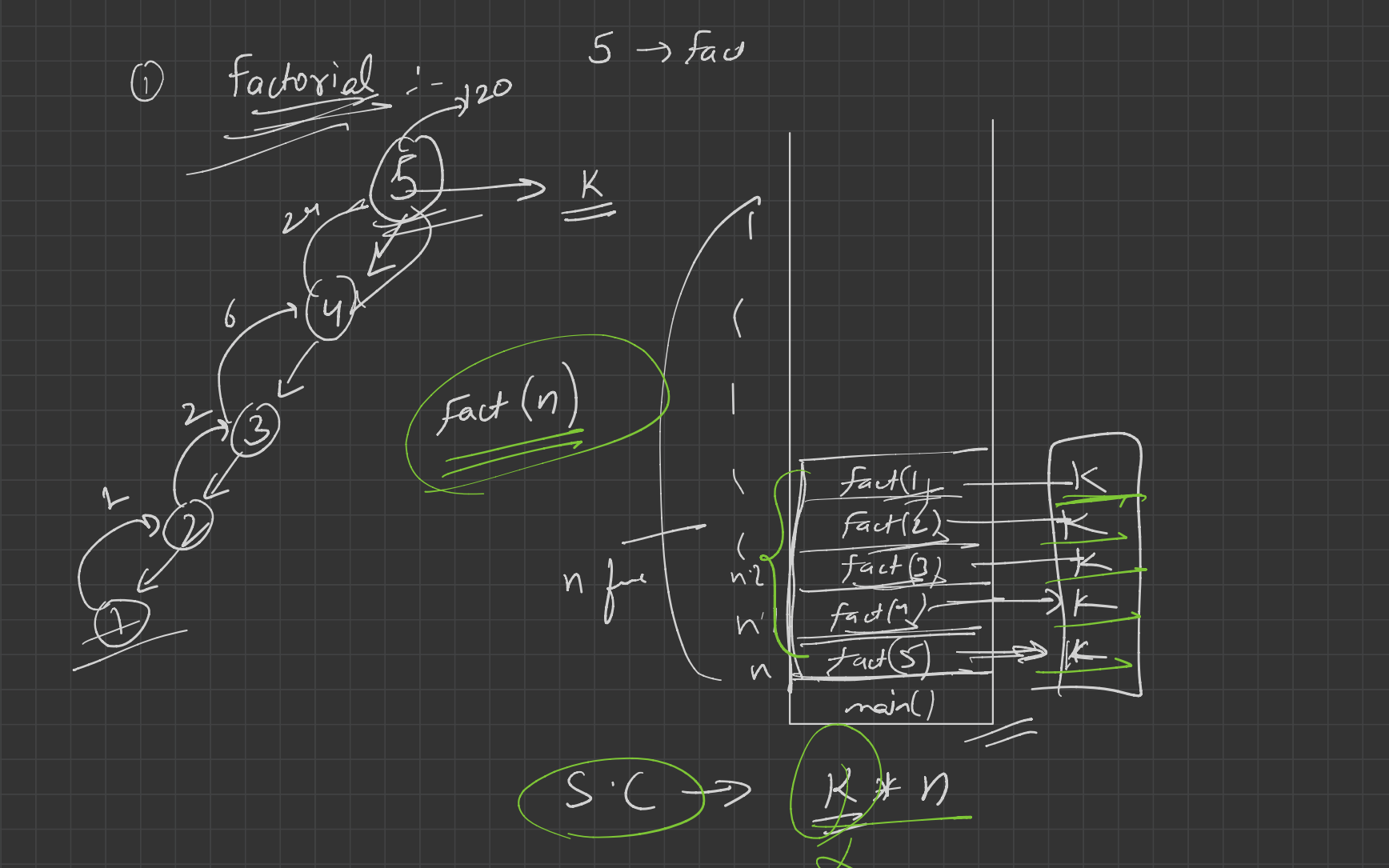
Space Complexity :- space required as $f(n)$

T.C



max. space req. at any instant

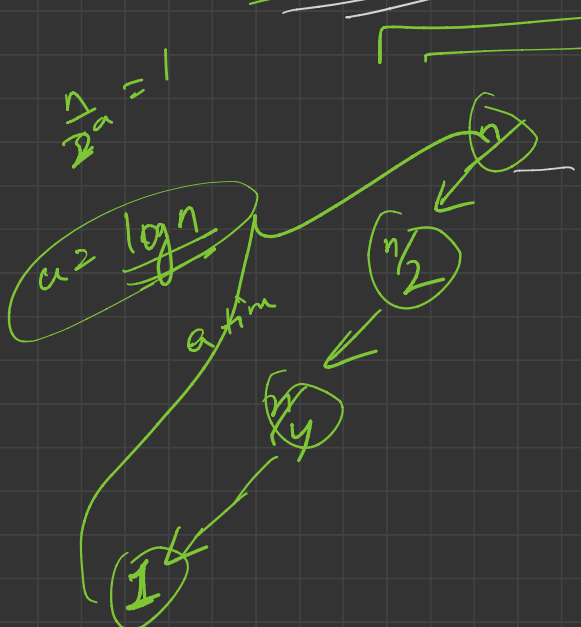


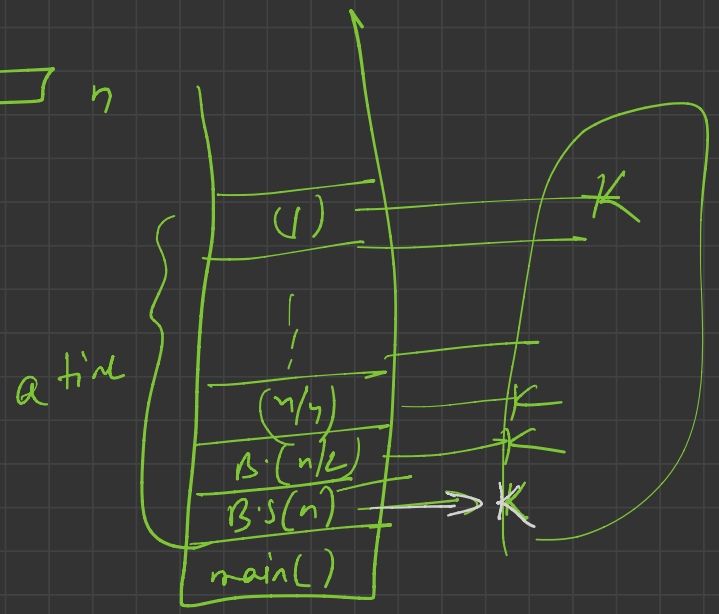

5 → fact
① Factorial :-
120
5
K
24
6
4
2
3
2
1
1
Fact (n)
1
2
6
n fact
n-2
n-1
n
fact(1)
fact(2)
fact(3)
fact(4)
fact(5)
main()
K
K
K
K
K
S.C → K * n

factorial

↳ O(n)

→ Binary Search:-

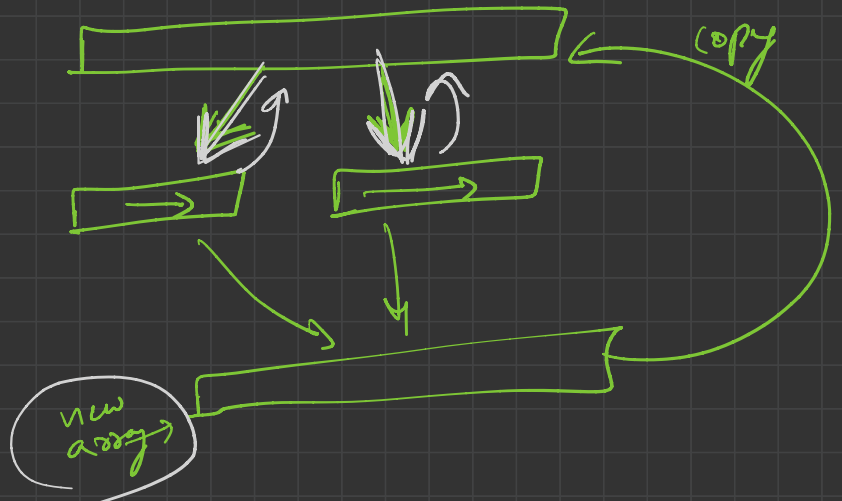


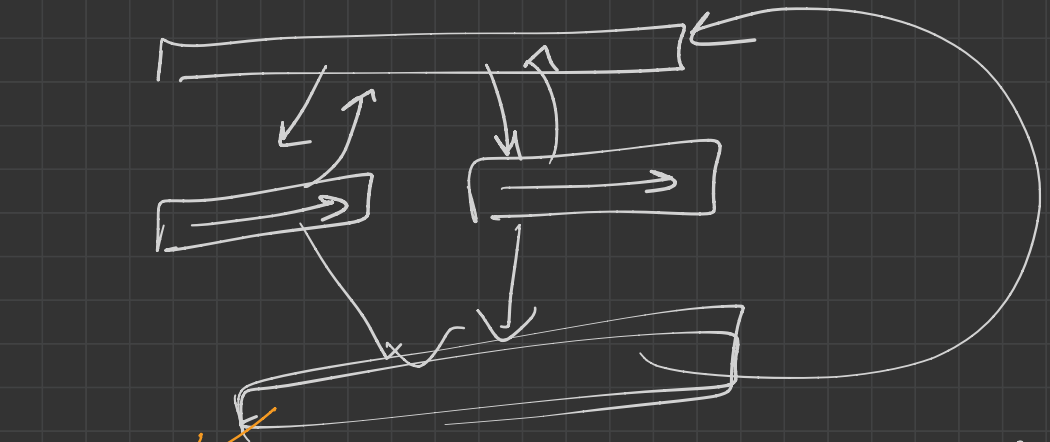



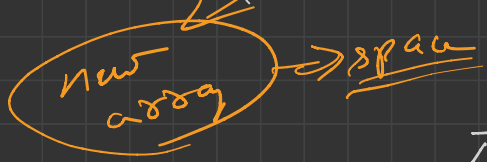
S.C → $\frac{K}{2} * \log n$

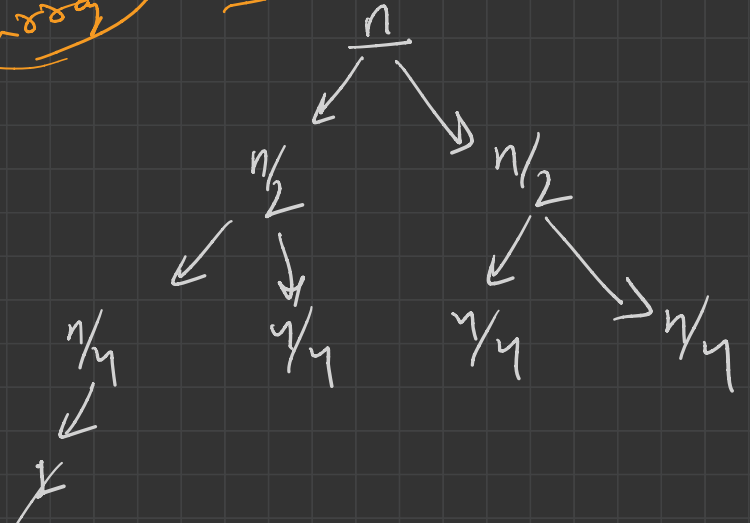
$= O(\log n)$

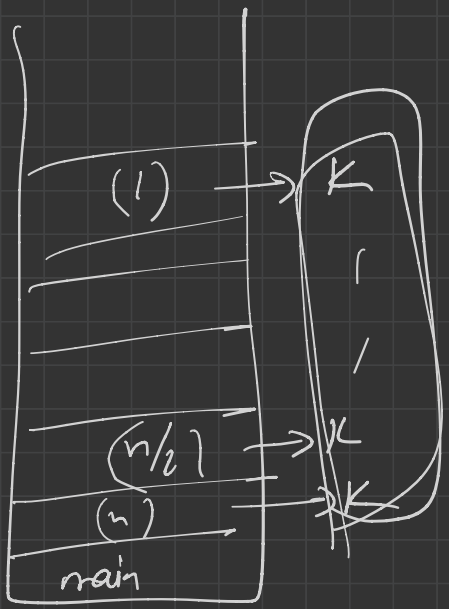
③ Merge sort →

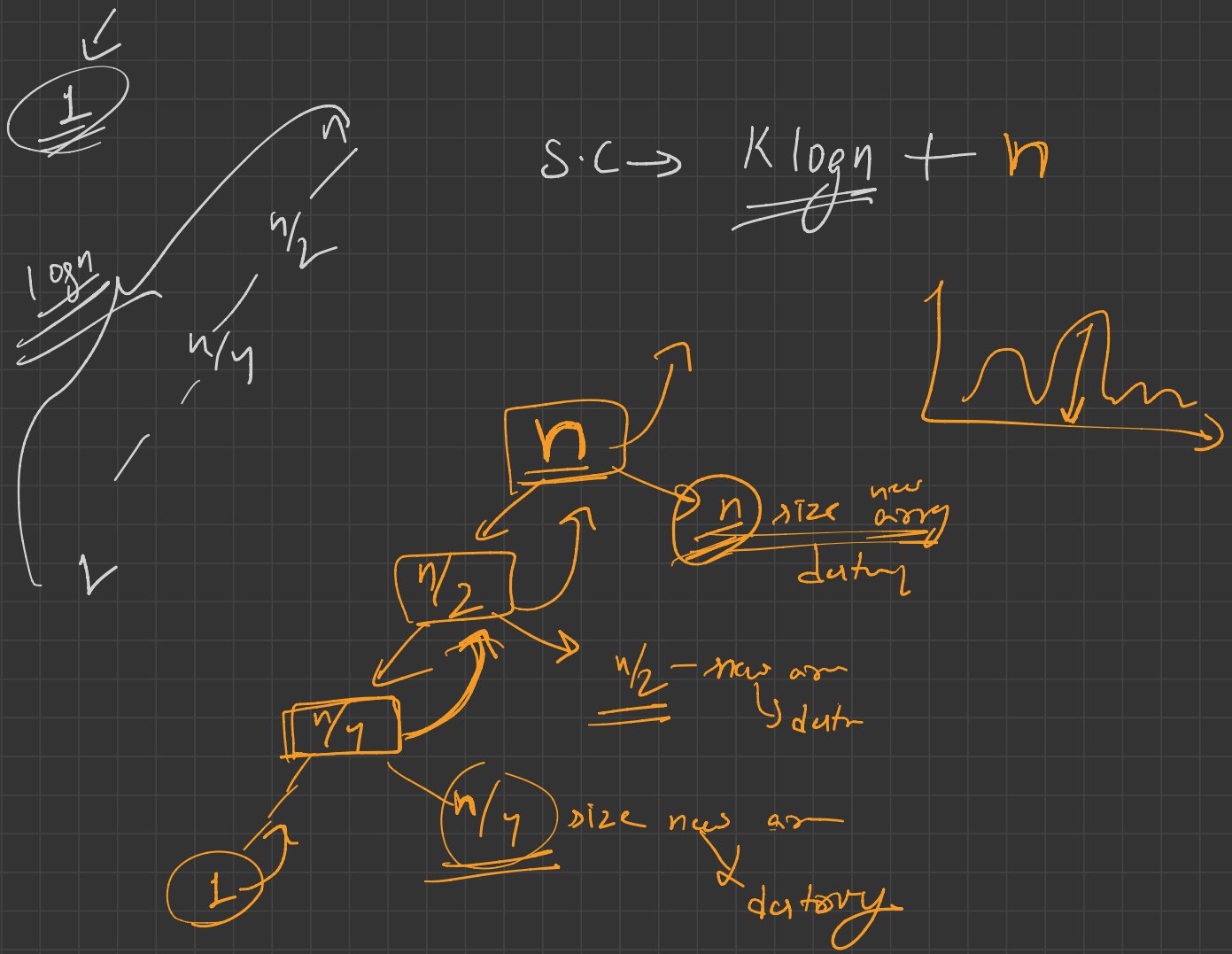





New array
space


n
n/2
n/2
n/4
n/4
n/4
n/4
K


(1)
K
(n/2)
K
(n)
K
main


1
n
n/2
n/4
1
log n
S.C → K log n + n
n
n
size new array
destroy
n/2
n/2 — new arr
destroy
n/4
n/4
size new arr
destroy
1

S.C → $K \log n + n$

$n >>> \log n$

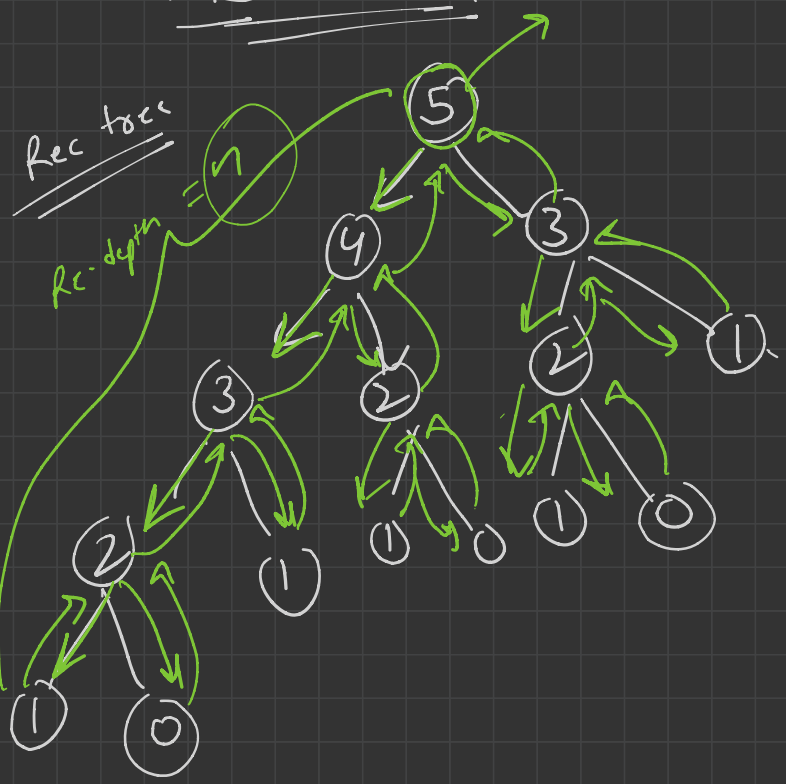
S.C → $O(n)$

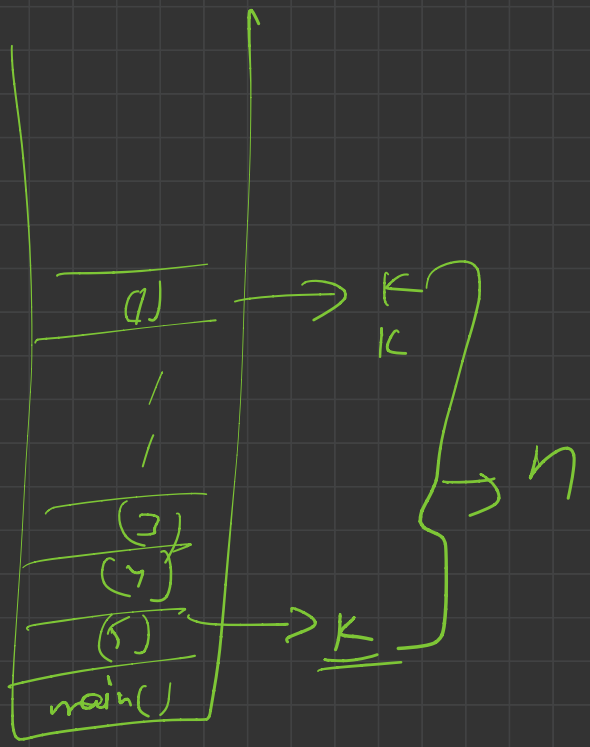
Recc Merge sort

$n = 1024$

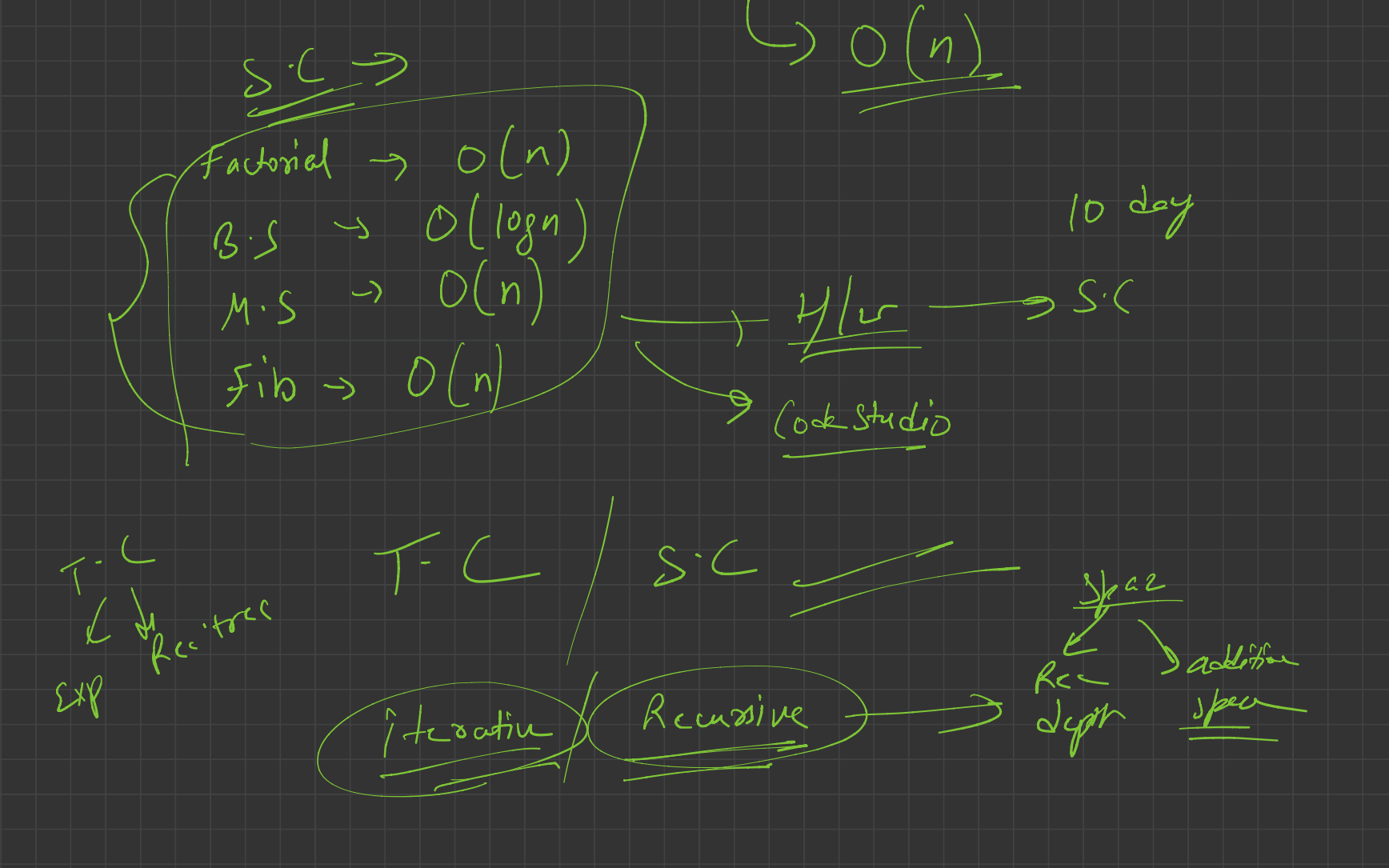
$\log n = 10$

# Fib Series:-




S.C → n*k


O(n)
S.C →
Factorial → O(n)
B.S → O(log n)
M.S → O(n)
Fib → O(n)
10 day
H/Ir
S.C
Codestudio
T.C / S.C
T.C
Recursive
EXP
Iterative
Recursive
Space
Rec
depth
addition
space