merge 2 sorted arrays

nums1 → [1, 2, 3, 0, 0, 0] m+n
(m = 3, n = 3)

nums2 → [2, 5, 6] n = 3

update nums1

no. of elements in nums2

num1 → [ ]m

nums2 → [ ]n

↳ new array [ ]m+n

[1, 2, 2, 3, 5, 6] m+n

nums1 → [1, 2, 3]m

nums2 → [2, 5, 6]n

i → exhausted

] → merge the 2 sorted arrays in a new array

O(m+n)

↳ result → [1 2 2 3 5 6]m+n

0 1 2 3 4 5

i

j

k

↳ sorted in asc order

On the 0th index of result, we will be having the smallest element.

i & j represent the best cand elements for felley for k.

In every iter of while loop we resolve 1 elem.

Brute force

result → num1 + num2

sorted

k

i

j

→ num1 → [1, 2, 3, 0, 0, 0]

num 2 → [2, 5, 6]

O(m+n)

Base Case

Array of length ≤ 1 is already sorted

Recursion

Merge Sort

Bloomberg London

start

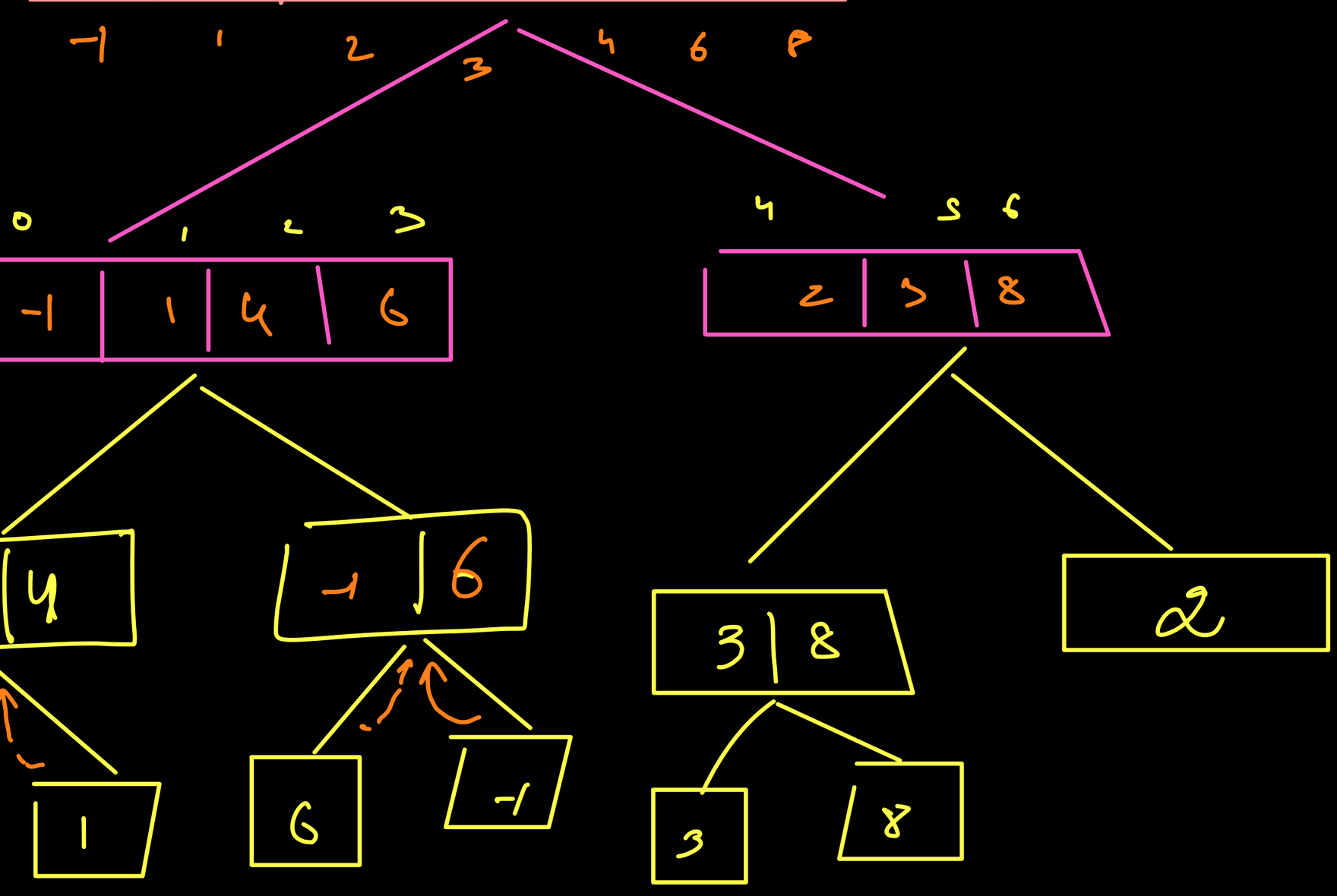
end

0 1 2 3 4 5 6

4	1	6	-1	3	8	2

-1 1 2 3 4 6 8

↙ entire array



$f(arr, start, end) =$

$$mid = \frac{start + end}{2}$$

$$left = f(arr, start, mid)$$

$$right = f(arr, mid+1, end)$$

$$merge(left, right)$$

merge sort

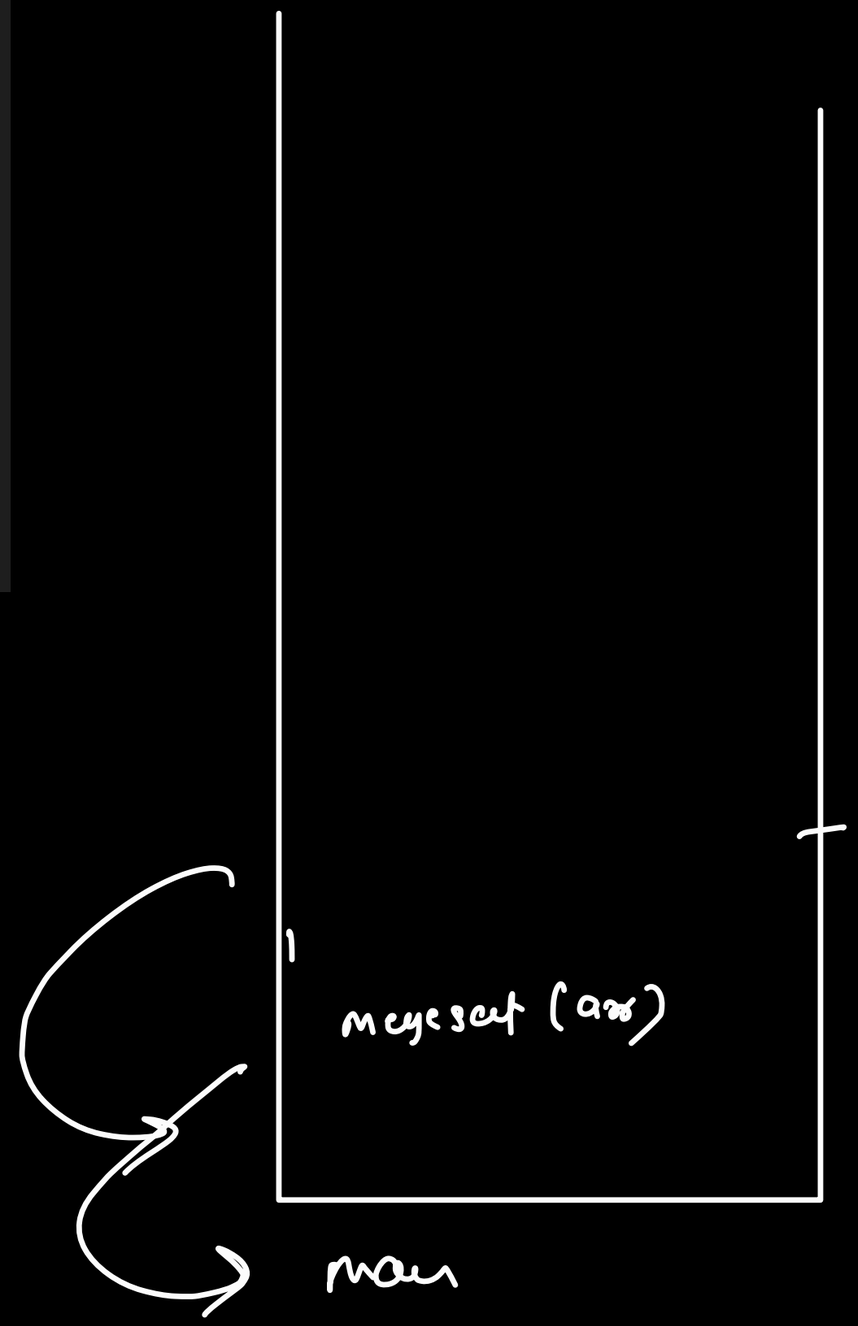
apply merge sort on arr [start, end]

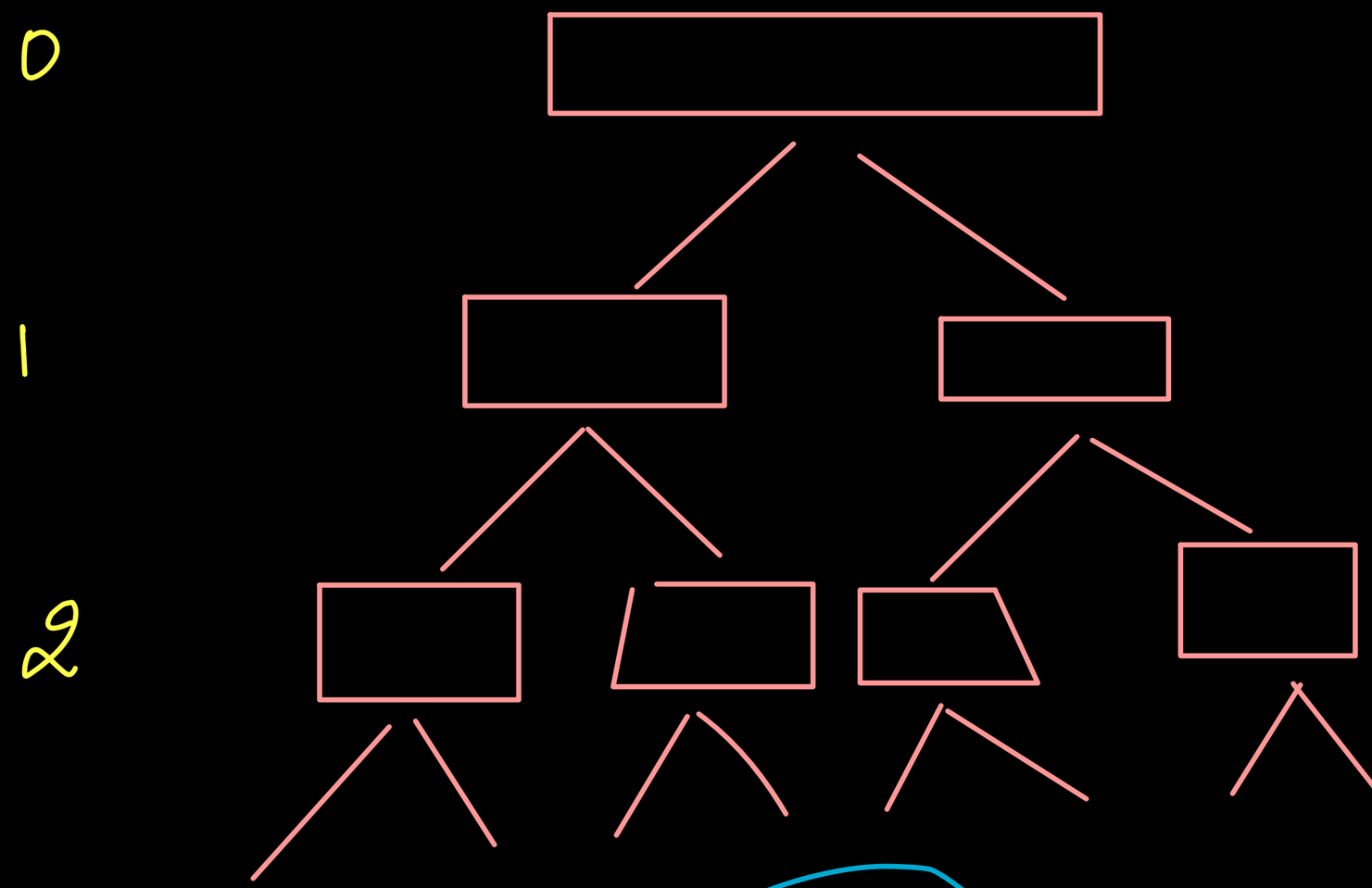
```java
    public static int[] mergeSortHelper(int[] arr, int start, int end) {
        if(start == end) {
            int[] result=new int[1];
            result[0] = arr[start];
            return result;
        }
        int mid = (start + end) / 2;
        int[] left = mergeSortHelper(arr, start, mid);
        int[] right = mergeSortHelper(arr, mid + 1, end);
        return merge(left, right);
    }

    public static int[] mergeSort(int[] array) {
        return mergeSortHelper(array, start:0, array.length - 1);
    }
```
Run | Debug

// arr → [3, 2, 1, 0] //

[0, 1, 2, 3)





recursion

call stack

log n

$2^n < 3^n$

$K < 3$

3^n

$2^n + \log n$

10^3 10^6

$10^6 + \log 10^6 \rightarrow 10^6 + 20$

$10^3 + \frac{\log 10^3}{\log 10}$

$Kn + \log n$

$1 + \log n$

$O(n)$

$\Rightarrow$ Space

level	length	# of parts	work done
0	n	1	n
1	$\frac{n}{2}$	2	n
2	$\frac{n}{4}$	4	n
3	$\frac{n}{8}$	8	n
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	$\frac{n}{2^k}$	2^k	n

if k is the final level, then length of the array is 1

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

$$n = 2^k \longrightarrow \text{take log both sides}$$

$$\log_2 n = \log_2 2^k$$

$$k \log_2 2 = \log_2 n$$

$$k = \log_2 n$$

$\longrightarrow$ Total levels in worst case

$\hookrightarrow \log_2 n$

$O(n \times \log n)$

Time

$\Omega(n \log n)$

$\Theta(n \log n)$

Space

$O(n)$

↳

$\log_a n$ → easily convert into any other base

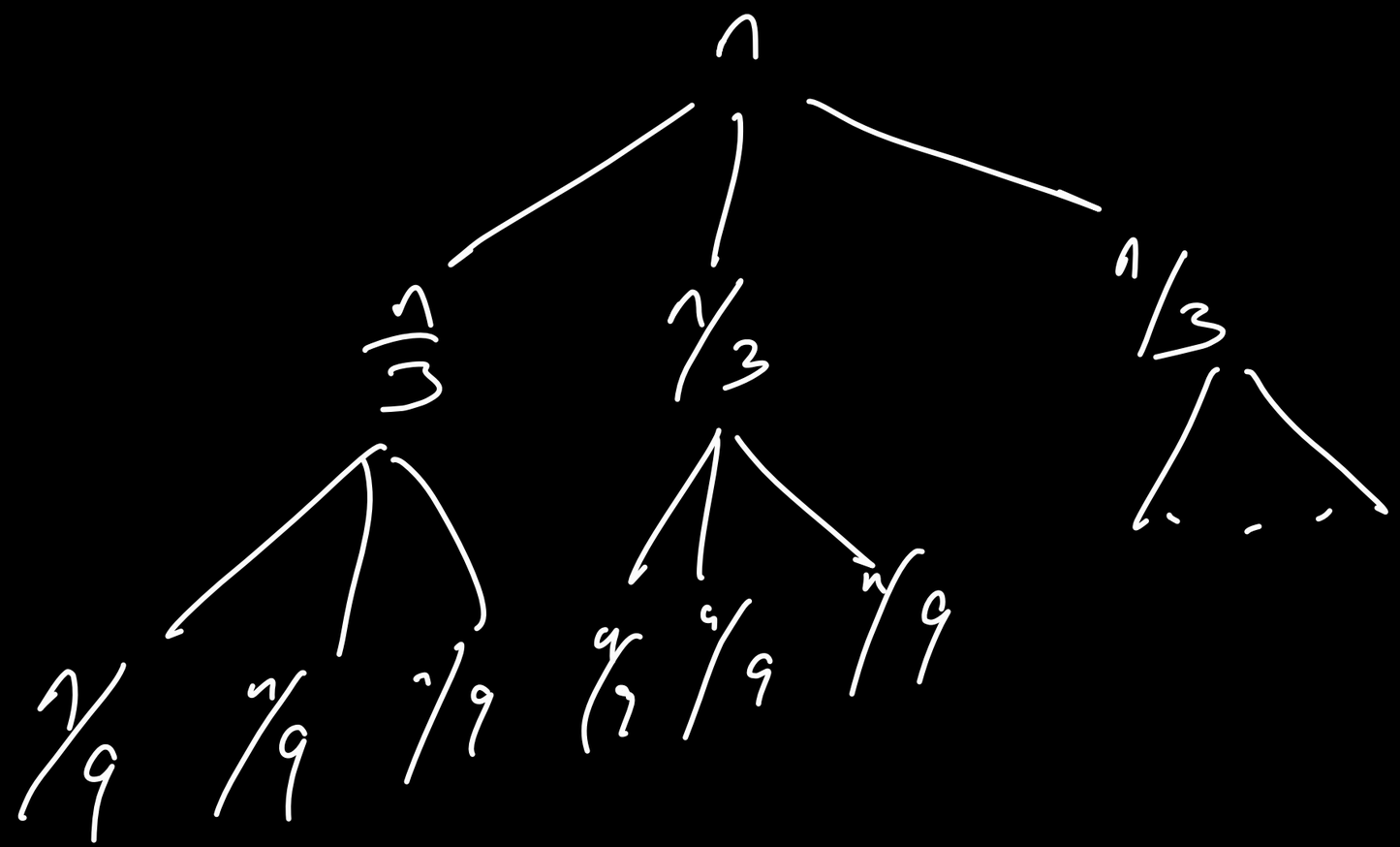
$\log_2 n \rightarrow \log_{10} n$

$\log_{10} 2$ → K const

$$\log_a n = \frac{\log_b n}{\log_b a}$$

$a \rightarrow b$

$$\log_4 n \longrightarrow \frac{\log_5 n}{\log_5 4}$$



$$\frac{n}{3^k} = 1$$

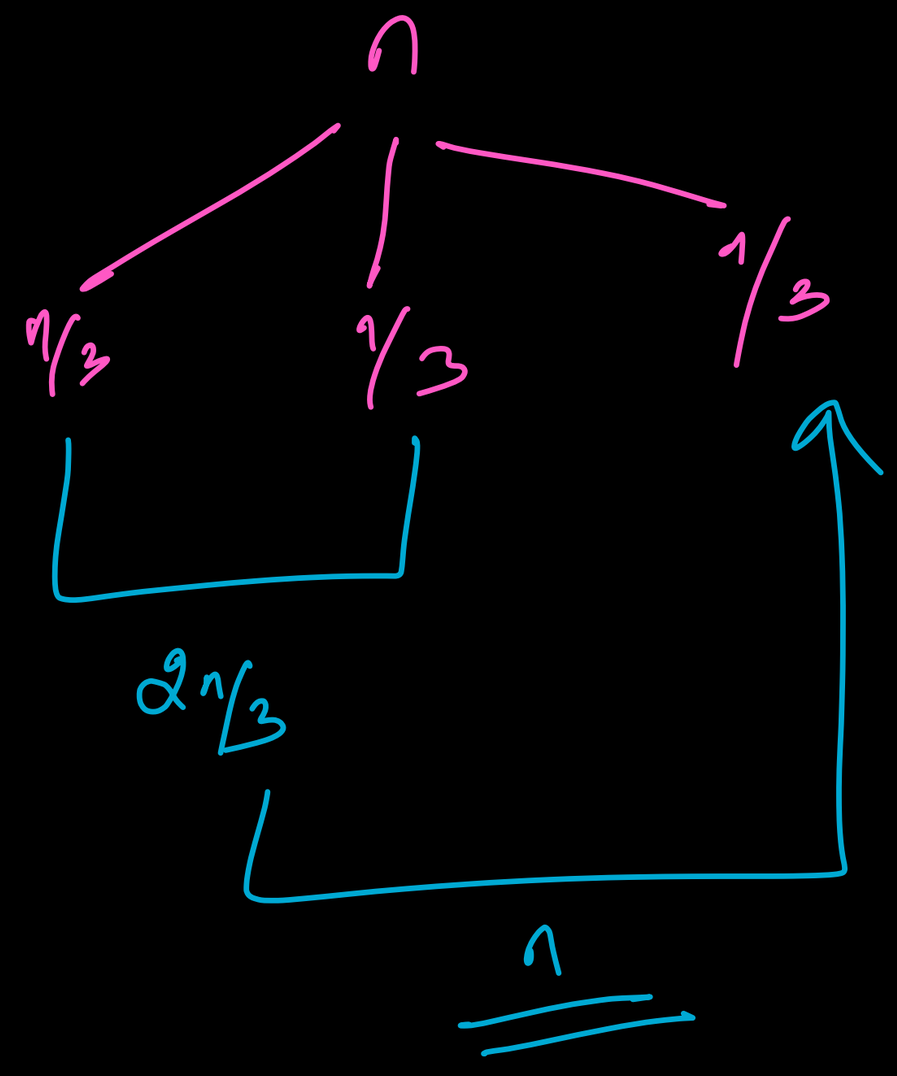
$$n = 3^k$$

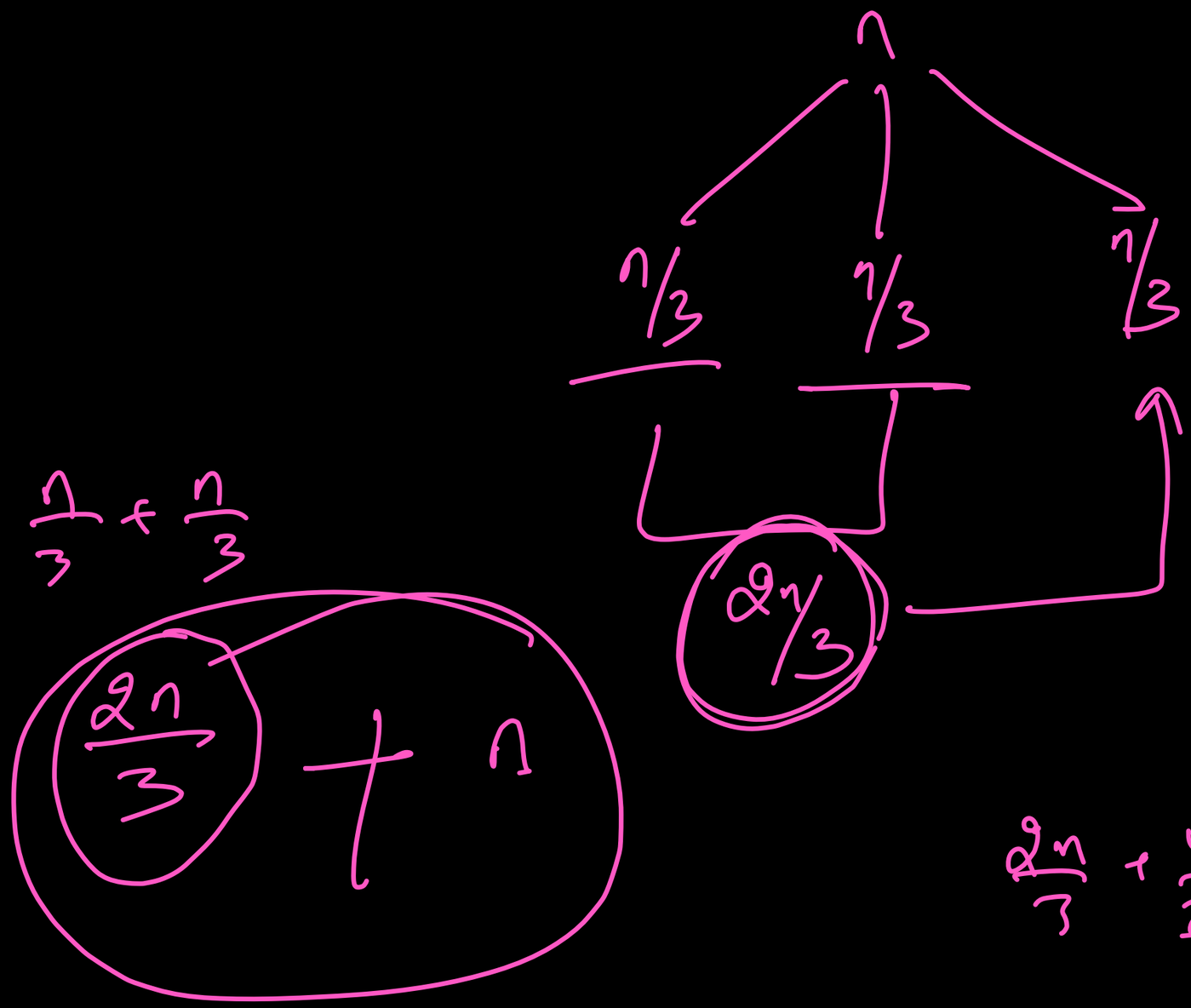
$$\log_3 n = \log_3 3^k$$

Pak $\log_3 \cdot$ bo n

$k = \log_3 n$

$\log_3 n < \log_2 n$



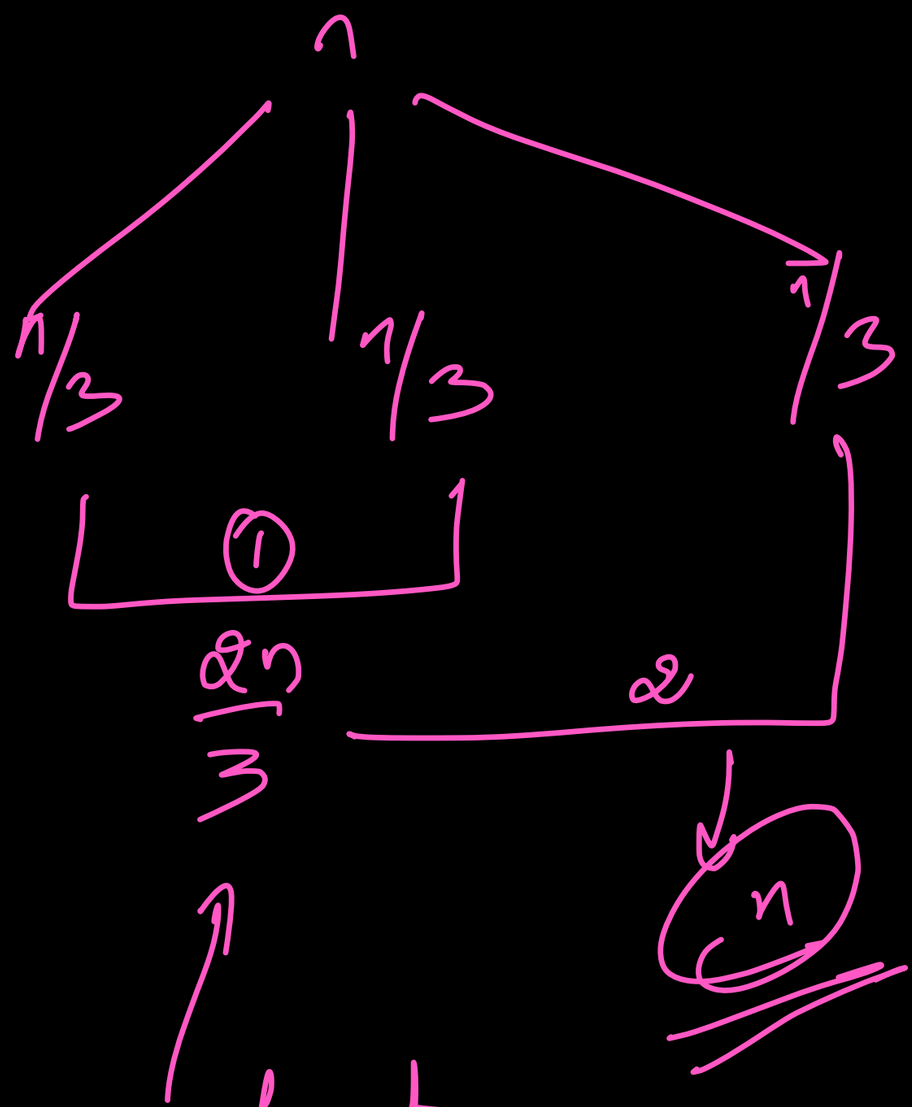


$$\frac{2n}{3} + \frac{n}{3} \to n$$

① → $\frac{2n}{3}$

② → n

$\frac{2n}{3} + n$

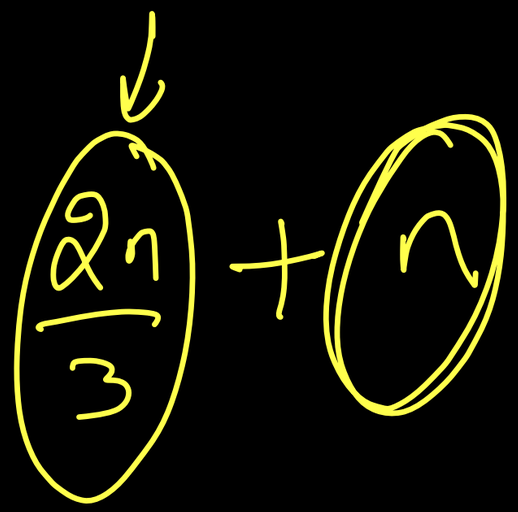


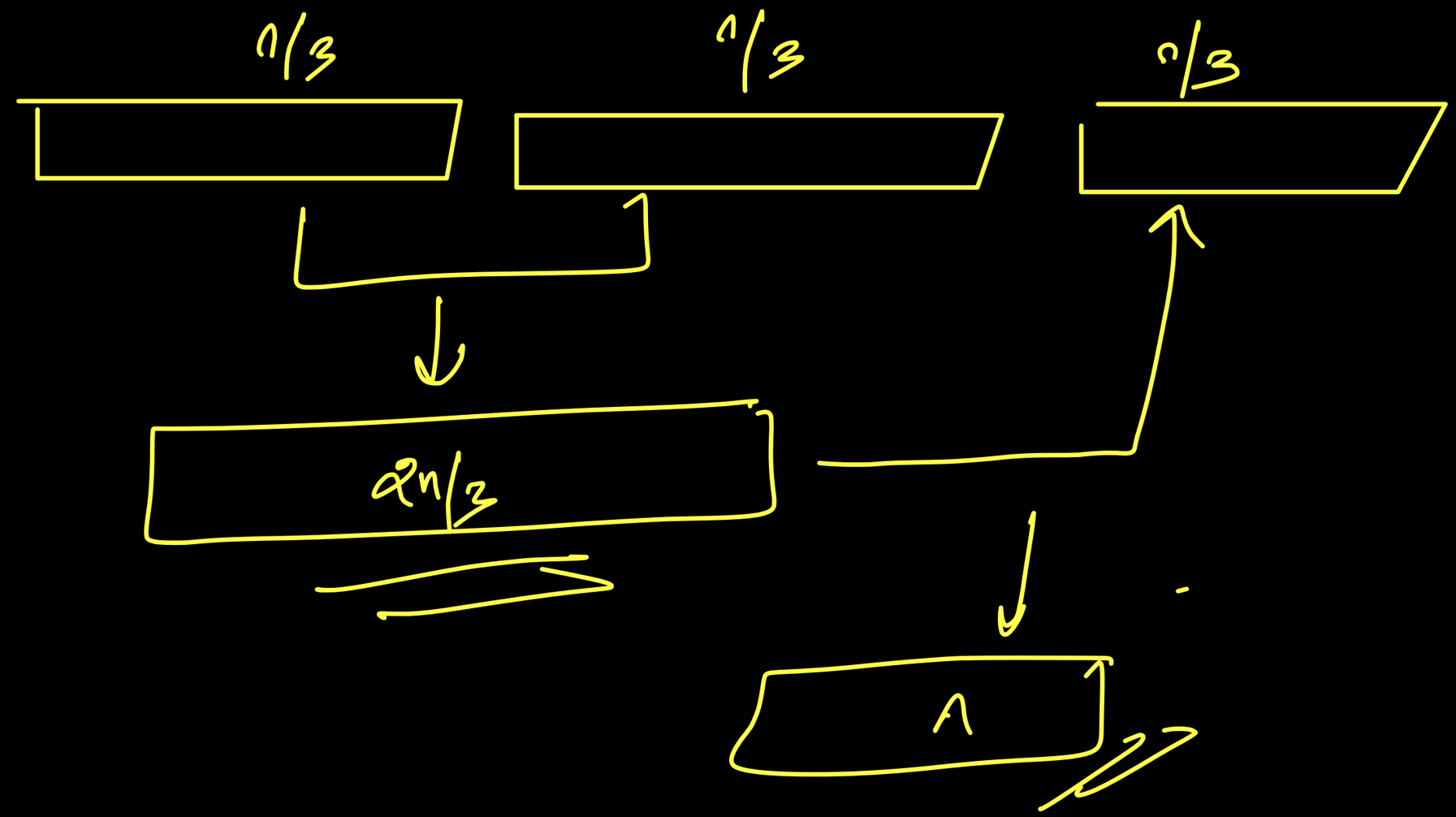
Divide N Conquer → merge sort
quick sort
G.S
⋮

Brute force

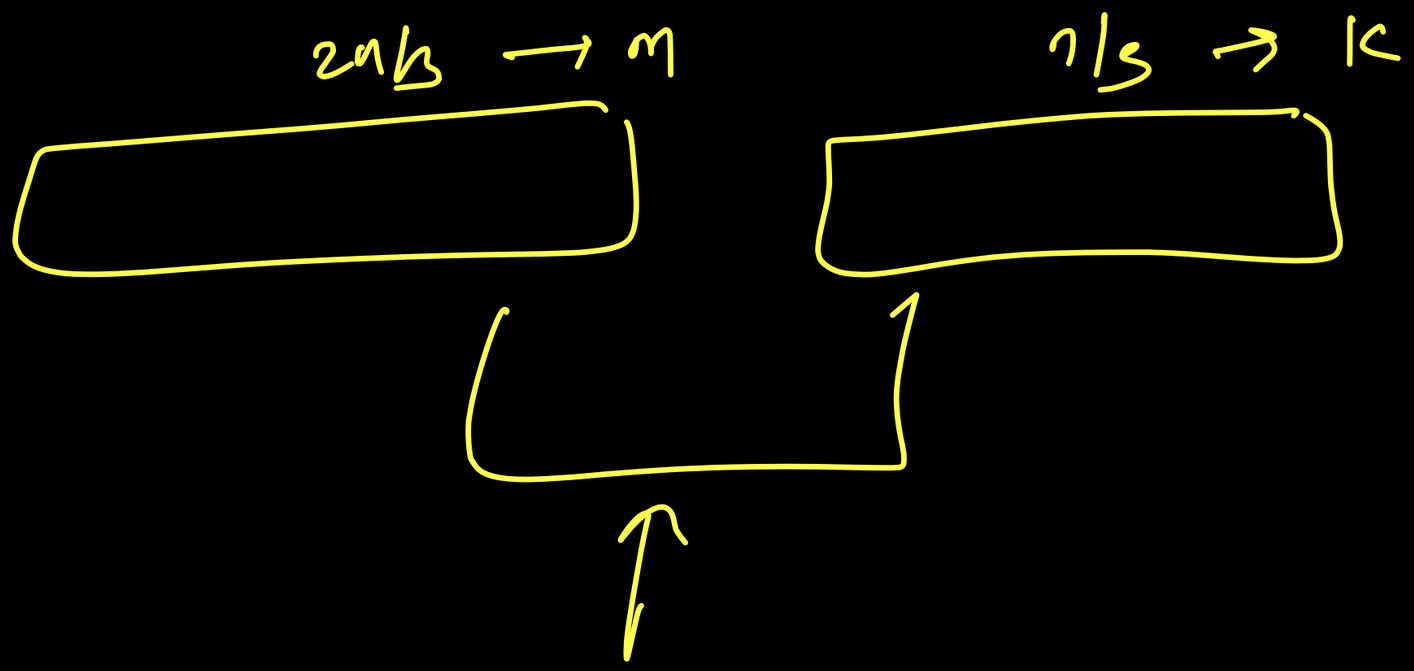
greedy

dynamic prog





$$\frac{S_1}{3}$$



$O(m+k)$

$O\left(\frac{2n}{3} + \frac{3n}{3}\right)$

$\rightarrow n$