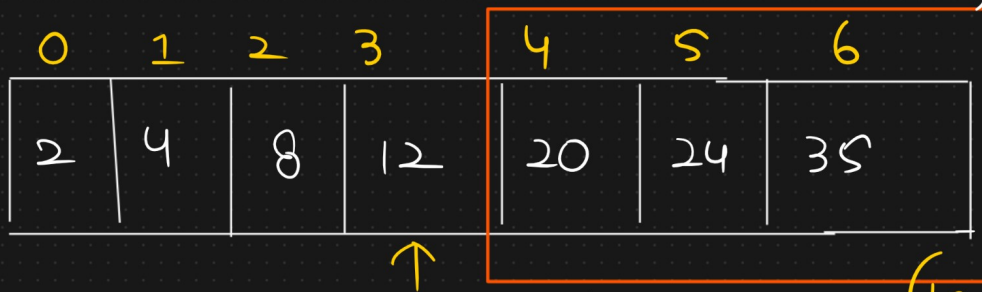
Searching & Sorting

Searching

- Linear Search
- Binary Search
- Ternary Search

1) Sorted Manner

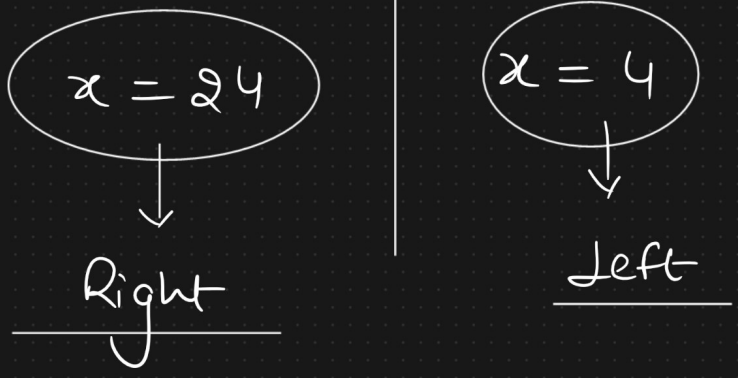


① (low + high)/2

low = 0, high = 6

② mid = low + (high - low)/2

= 3



Iterative approach

binarySearch(arr, low, high, x)

```
while (low <= high) {
    if (arr[mid] == x)
        return mid;                    // (1)

    if (arr[mid] < x)                  // 12 < 24   (2)
        low = mid + 1;

    if (arr[mid] > x)
        high = mid - 1;                // (3)
}

}
```

Recursive approach

```
binarySearch (arr, low, high, x) {
    while (low <= high) {
        mid = low + (high - low)/2;
        if (arr(mid) == x) {
            return mid;
        }
        if (arr(mid) < x) {
            return binarySearch (arr, mid+1, high, x)
        }
        if (arr(mid) > x) {
            return binarySearch (arr, low, mid-1, x);
```

C

T(n/2)

Right side

Recursive function call

low

T(n/2)

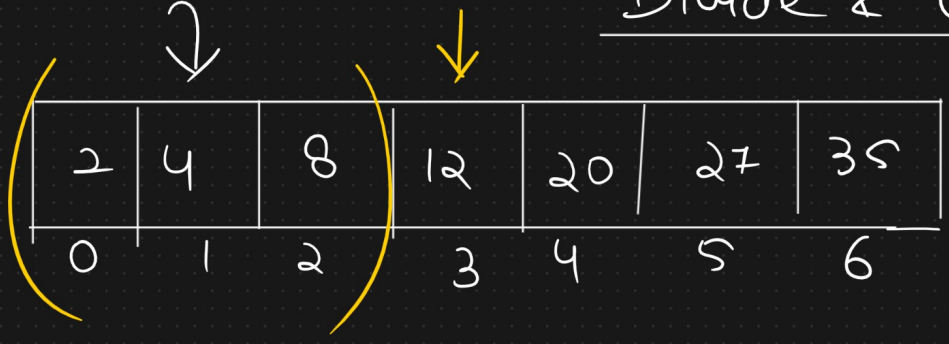
Left side

Recurrence Relation

$$T(n) \Rightarrow T(n/2) + c$$

$$\Rightarrow O(\log n)$$

Divide & Conquer



low = 0, high = 6

mid = 3

x = 27

low = 4, high = 6

mid = 5

arr(mid) == x

27 27

result = 5

$x = 4$

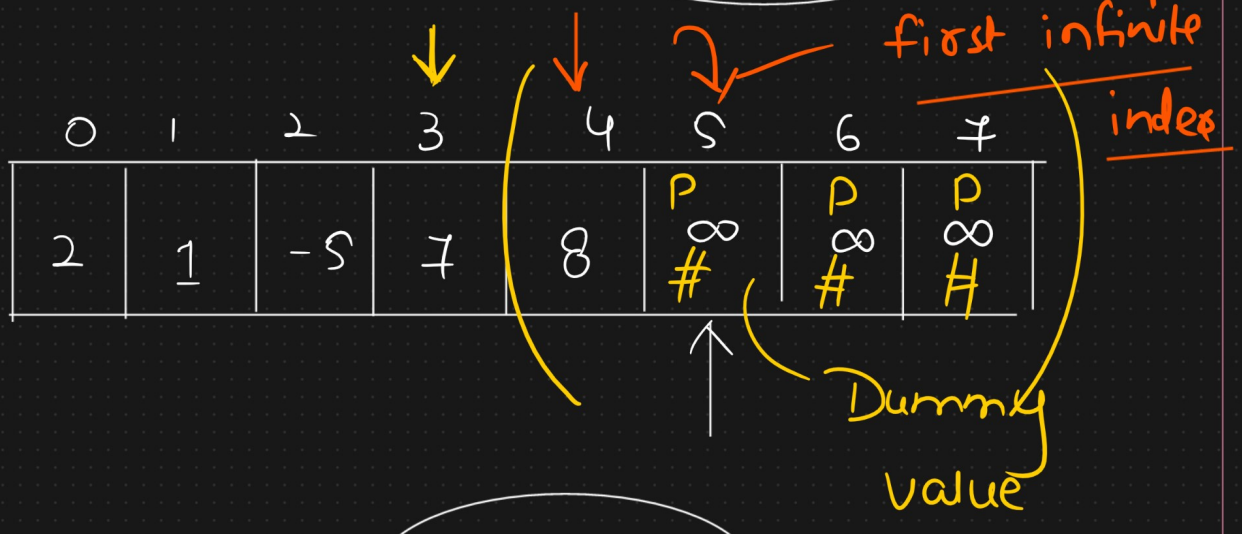
low = 0, high = 6

mid = 3

low = 0, high = 2

mid = 1

result = 1



result = 5

low = 0, high = 7

mid = 3

low = 4, high = 7

mid = 5

---

x < 2 → return x;

Sqrt(x)

x = 16

low = 2

high = 16/2 = 8

x = 16

num = 25

2 3 4 5 6 7 8

while (low <= high)

mid = 5

num = 5 * 5 = 25

---

if num > x

→ left

→ high = mid - 1

if num < x

→ Right

low = mid + 1

if num == x

return mid;

}

return high;

→ Lower bound

x = 8

Expected Result = 2

low = 2 high = 4

2 3 4

mid = 3

num = 9

num > x :

low = 2 right = 2

mid = 2

num = 4

num < x

↳ right

low = mid + 1;

low = 3 high = 2

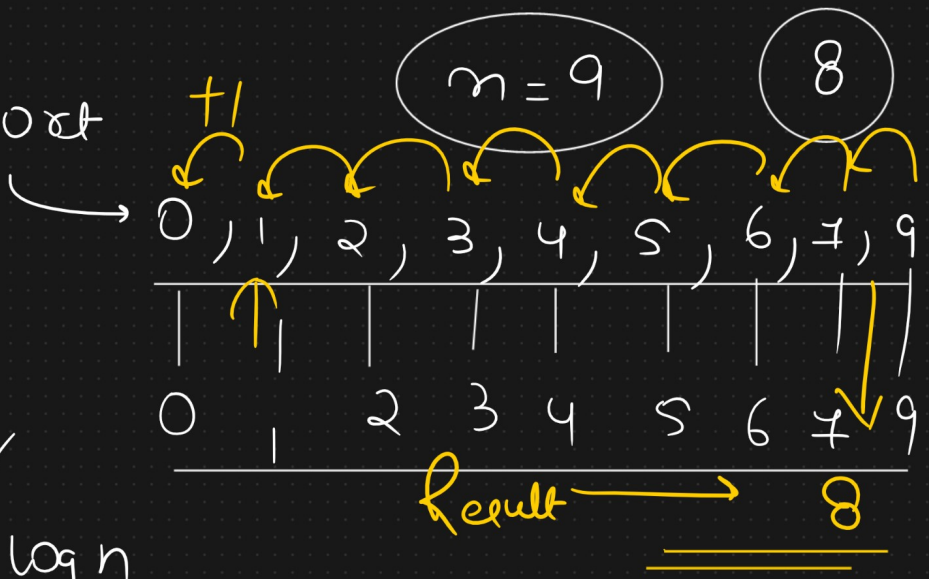
Missing number

$n = 9$

9	7	6	4	2	0	1	3	5
0	1	2	3	4	5	6	7	8

Output = 8

① Sort



$n \log n$

Space → $O(n)$

② Gauss' formula

Sum of n natural numbers

↳ $n(n+1)/2$

Expected Sum $= \frac{10 * 11}{2} = 55$

c ←

n

```
for (num : nums) {
    actualSum += num
}
```

↳ 47

$55 - 47 = 8$

ExpectedSum − actualSum = 8

- Time complexity = O(n)
- Space complexity = O(1)

(3) XOR Operation

↳ Bitwise manipulation

Truth Table

A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	0

$A \wedge 0 = A$

$n = 9$

0, 1, 2, 3, 4, 5, 6, 7, 9

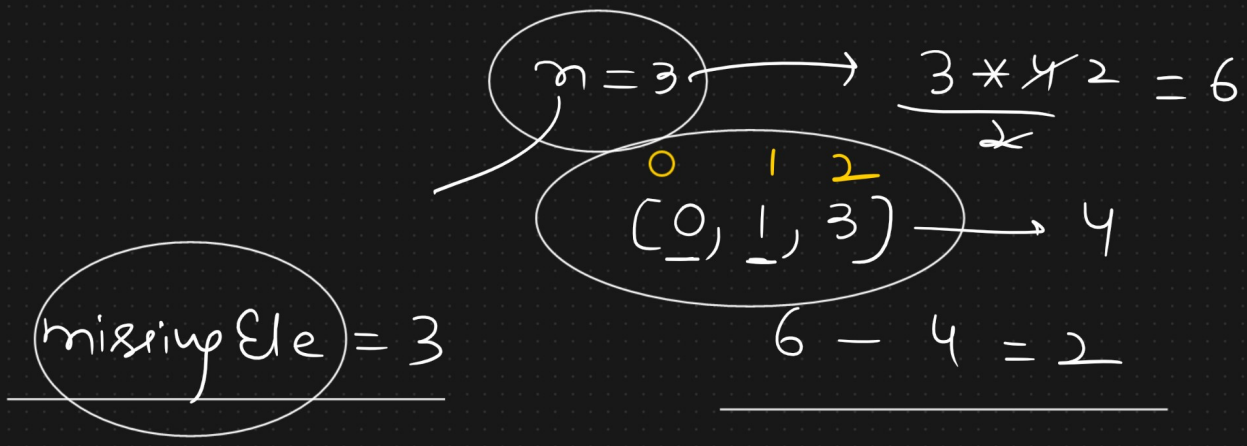
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

8

2) $8 \wedge 0 = 8$

time complexity = $O(n)$

Space complexity = $O(1)$

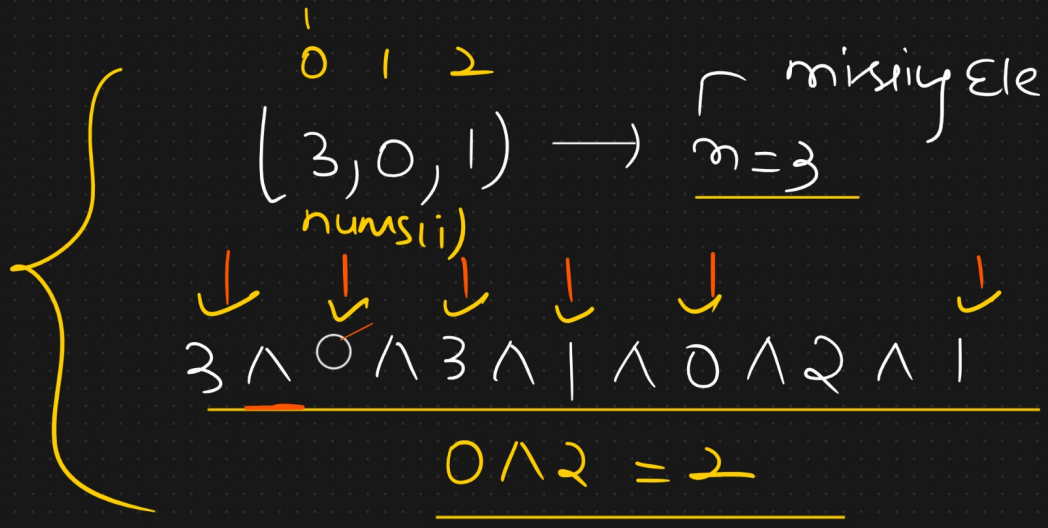


i = 0	i = 1	i = 2
0 ^ 0 = 0	1 ^ 1 = 0	i nums(i) 2 ^ 3

missingEle = 3 ^ 2 ^ 3

⇒ 0 ^ 2 = 2

missingEle = missingEle ^ i ^ nums(i)



Approach 1

Set
↳ Not any duplicates

0	1	2	3	4
1	3	4	2	2

↑ ↑ ↑ ↑ ↑ → (2)

time complexity = O(n)

Set
{ 1, 3, 4, 2 }

Space
↳ O(n)

Approach 2

↳ Sort → O(n log n)

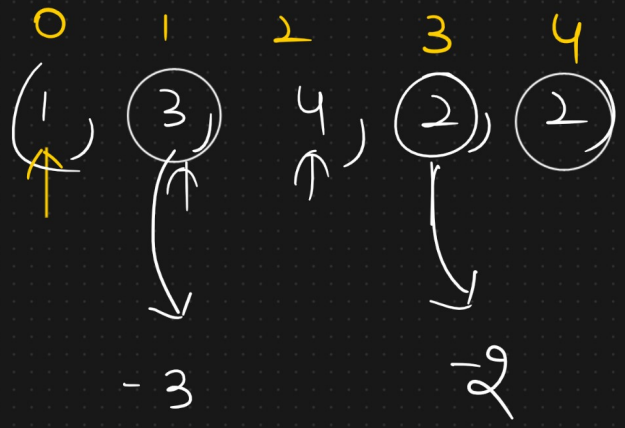
n = 5

0	1	2	3	4
1	2	2	3	4

(2)

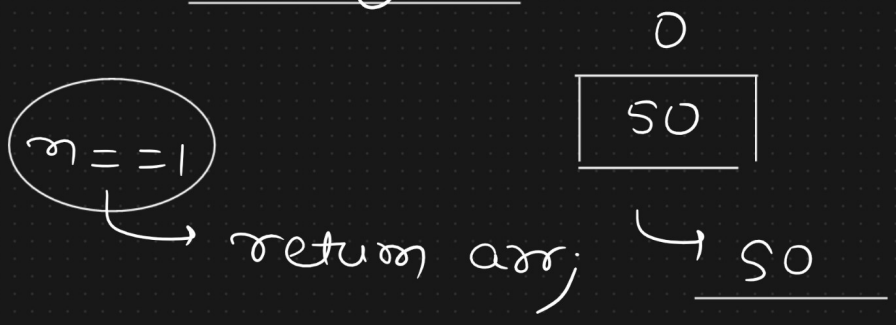
Space → O(n)

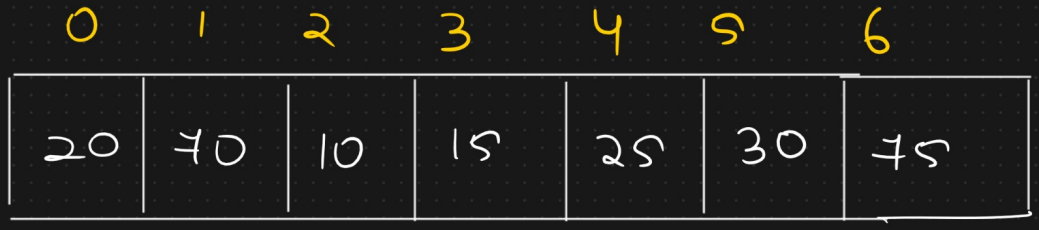
Approach 3 — Negative Marking

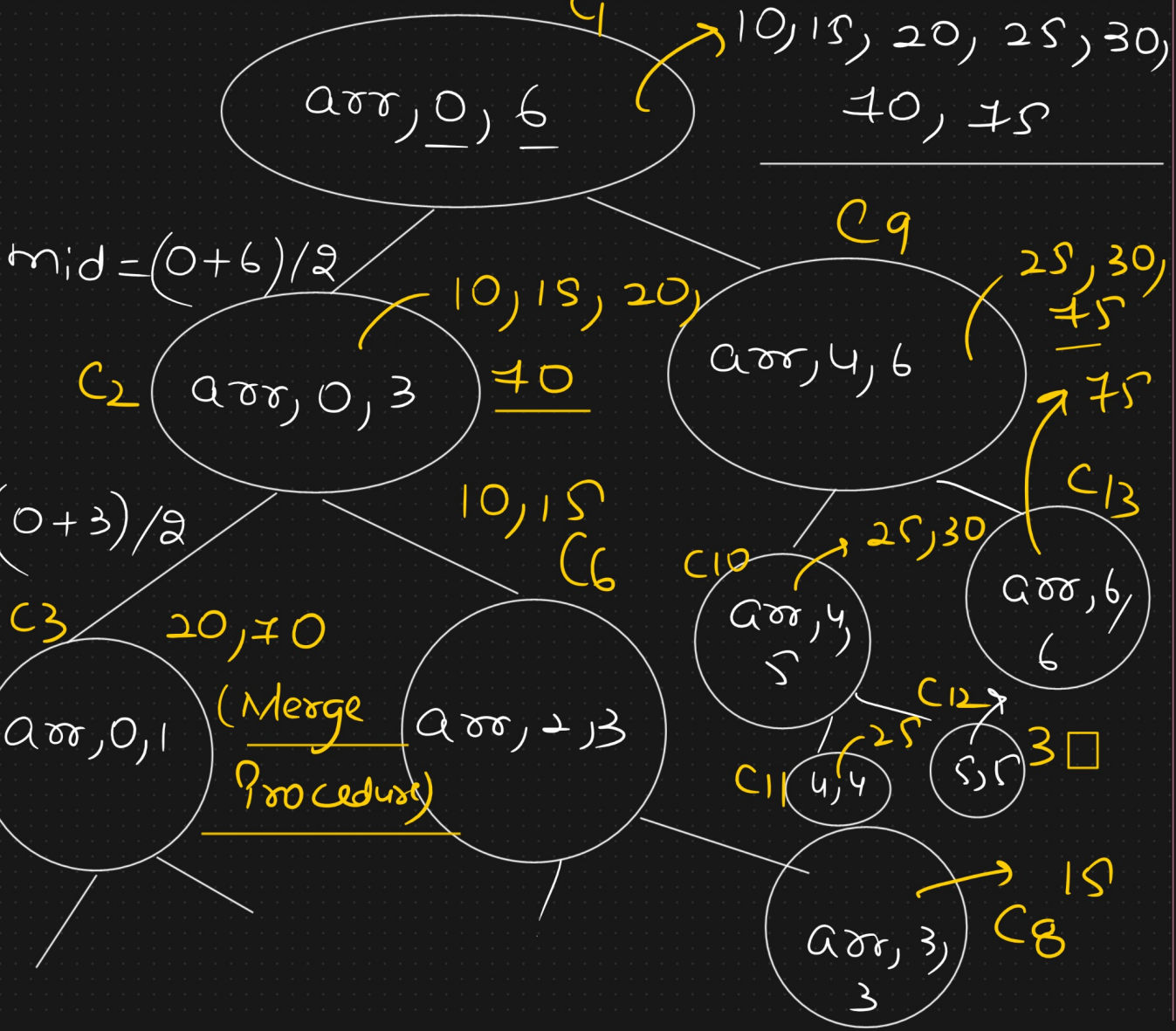


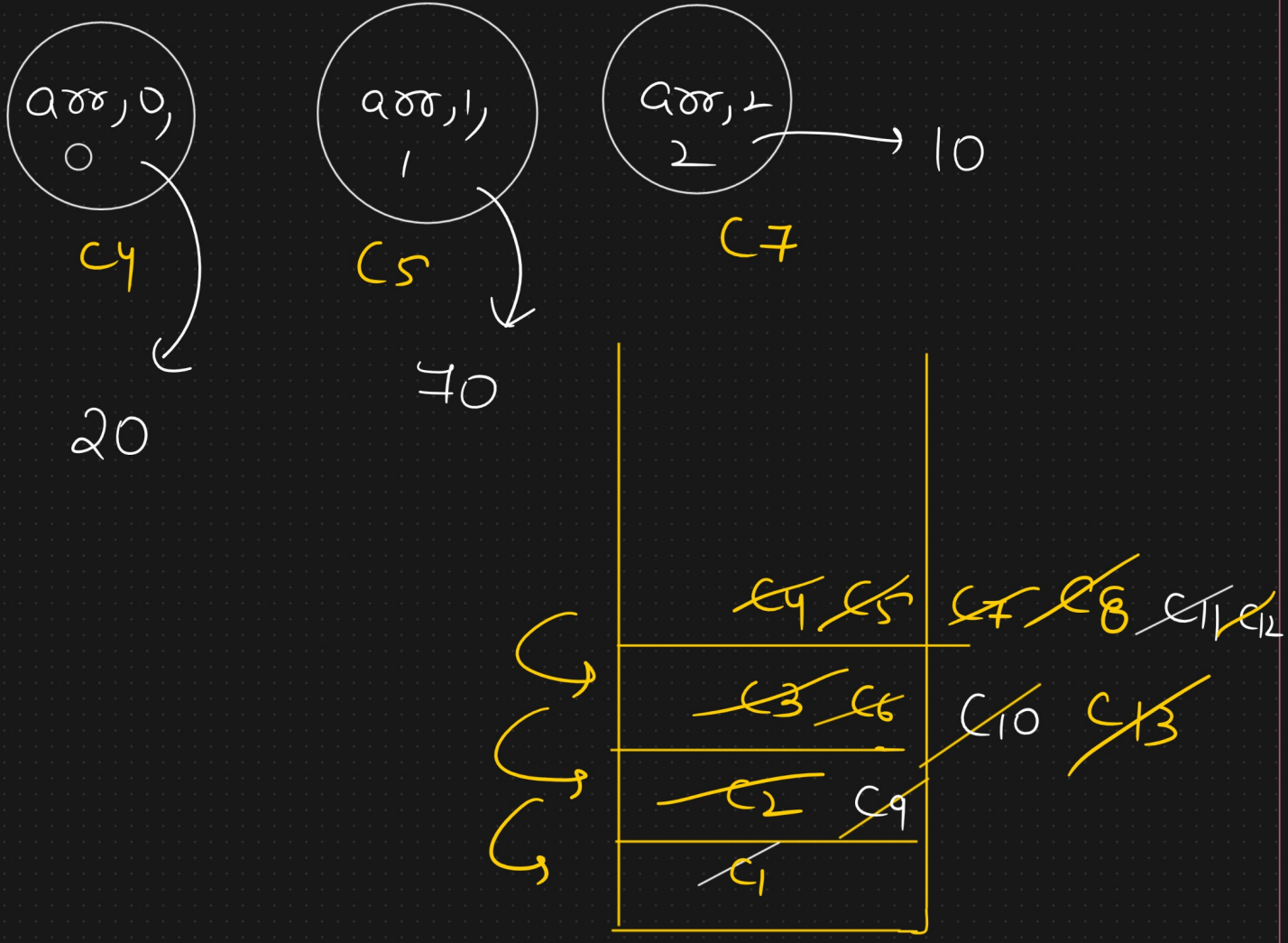
Divide & conquer

↳ MergeSort









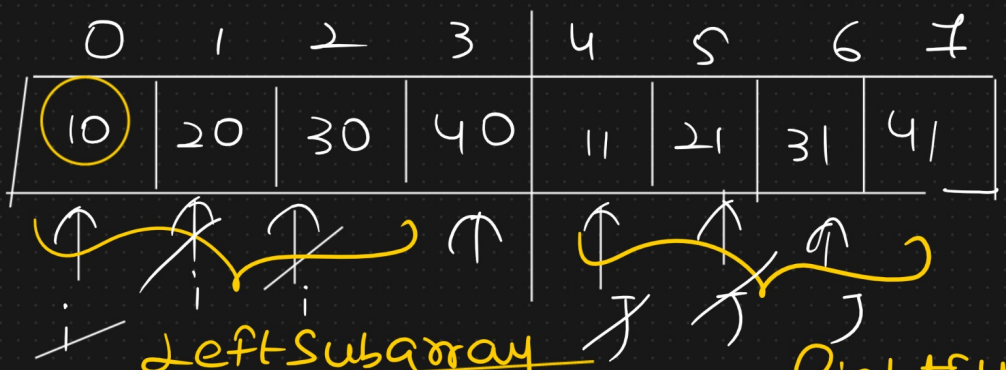
MergeSort

① Divide the bigger Problem into smaller subproblems

② Conquer → Recursion

③ Combine → mergeProcedure

Merge Procedure



30, 31 → 30

30, 21 → 21

leftSubarray

RightSubarray

40, 31 = 31

40, 41 = 40

10, 11 → 10 | 20, 11 → 11 | 20, 21 = 20

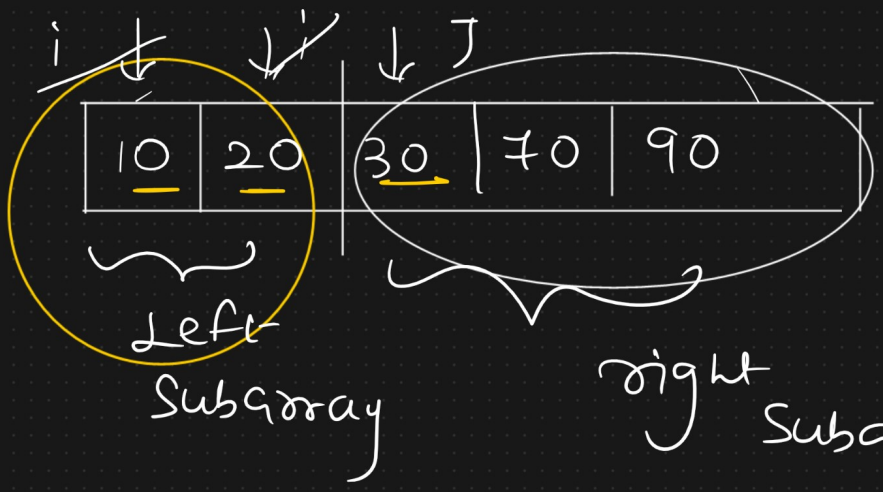
0	1	2	3	4	5	6	7
10	11	20	21	30	31	40	41

$m = 4$ | $m = 4$

Worst Case

Comparisons $= m + n - 1$

$4 + 4 - 1 = 7$



2 comparisons $\begin{cases} 10 < 30 = 10 \\ 20 < 30 = 20 \end{cases}$

10	20	30	40	90

Best case $= \min(m, n)$

num of Comparisons $= \min(2, 3)$

$= 2$

mergesort(arr, low, high) {

if (low < high) {

Divide ——— mid = low + (high - low)/2

Conquer { mergesort(arr, low, mid)
mergesort(arr, mid+1, high)

mergeProcedure(arr, low, mid, high)

↳ Combine

↳ Bottom to up

↳ Merge Sorted array