8 → Remove Element –

1 → Question –

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The relative order of the elements may be changed.

(Result to be placed in first part of the array nums.) — ---- .

If there are k elements after removing the duplicate, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements. — nums ---- × ×

Return k after placing the final result

Donot allocate extra space for another array. Modifying the input array in-place with O(1) extra memory.

2 → Examples –

nums = [3, 2, 2, 3], val = 3

→ 2, [2, 2, -, -]

nums = [0, 1, ~~2~~, ~~2~~, 3, 0, 4, ~~2~~], val = 2

→ 5, [0, 1, 4, 0, 3 ----]

3 → Intuition –

Two pointer? Why?

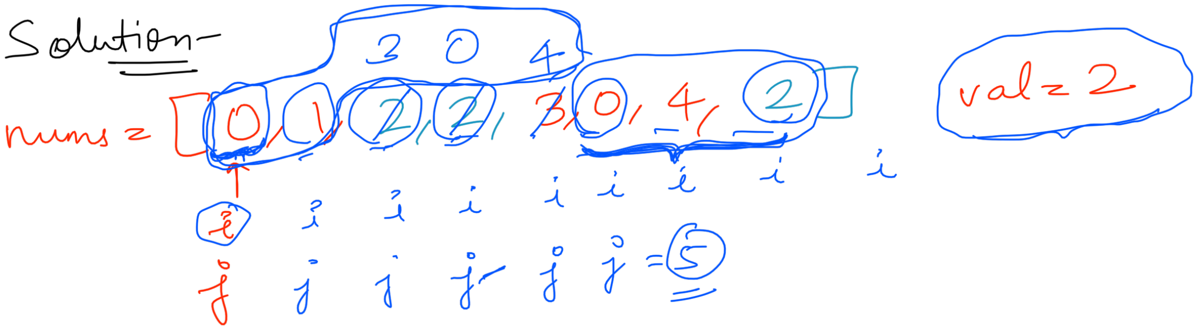
First → To run through the entire length of array

Second → To hold elements in first part that are to be kept in the array...

i

j

④ Solution-



5, [0, 1, 3, 0, 4]

order is not import

⑤ Time/ Space Complexity-

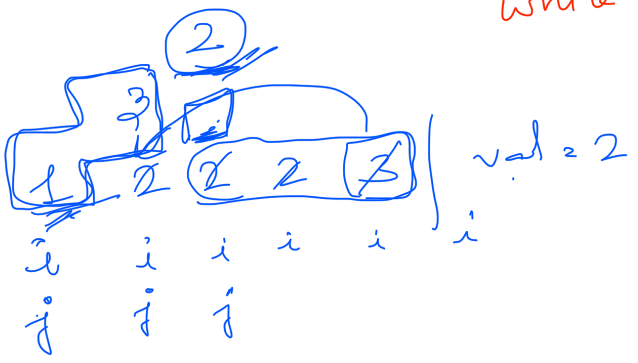
T = O(n)   i → n

S = O(1)

⑥ Code Walkthrough-

```
int removeElement (int[] nums, int val) {

        i = 0, j = 0
        while (i < nums.length) {
              if (nums[i] != val) {
                   nums[j] = nums[i];
                   j += 1;
              }
              i += 1;
        }
        return j;
}
```


2
3
1 2 2 2 3
val = 2
i i i i i i
j j j