16 September 2023 12:10

3. Find Duplicate Number (Leetcode-287)

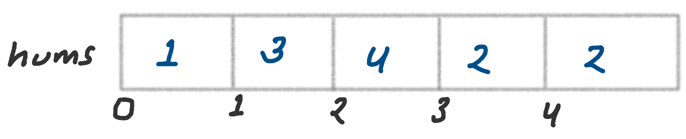
```
Example 1:
Input: nums = [1,3,4,2,2]
Output: 2

Example 2:
Input: nums = [3,1,3,4,2]
Output: 3
```

```
Example 3:
Input: nums = [2,2,2,2,2]
Output: 2

Example 4:
Input: nums = [1,2,3,4]
Output: -1
```

APPROACH:

Method 01: Sorting approach
Method 02: Negative marking approach
Method 03: Position and swapping approach

Method 01: Sorting approach

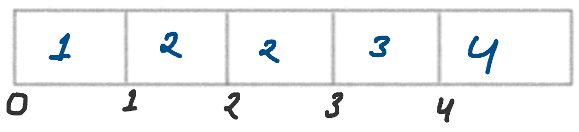


① nums[i] ∈ [1,N]
② size = N+1

Output: 2

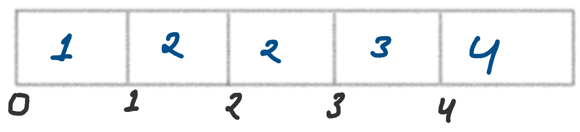
DRY RUN

Step 01: Sort the array



Step 02: Iterate the sorted array

Iteration 0



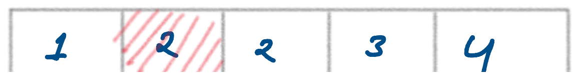
index = 0

(nums[index] != nums[index + 1])

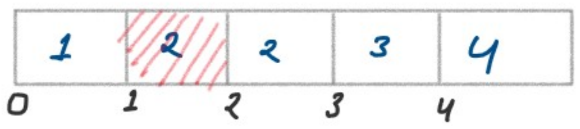
$\hookrightarrow$ 1 != 2 True

index++;

Iteration 1



Iteration 1



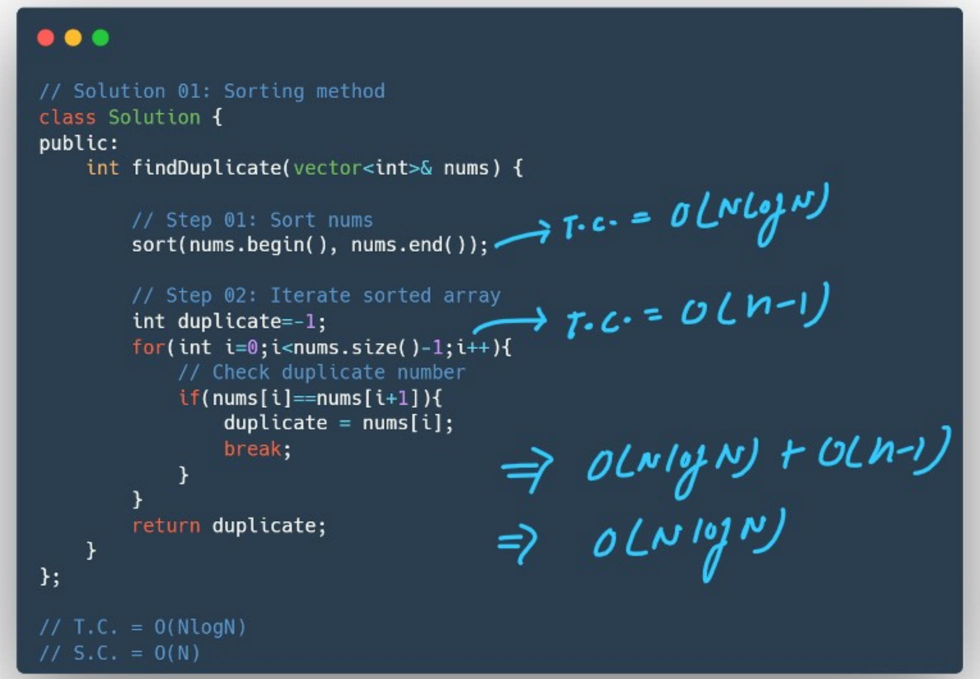
index = 1

(nums[index] != nums[index + 1])

$\hookrightarrow$ 2 != 2 false

return nums[index];

Duplicate = 2 at index 1



```cpp
// Solution 01: Sorting method
class Solution {
public:
    int findDuplicate(vector<int>& nums) {

        // Step 01: Sort nums
        sort(nums.begin(), nums.end());

        // Step 02: Iterate sorted array
        int duplicate=-1;
        for(int i=0;i<nums.size()-1;i++){
            // Check duplicate number
            if(nums[i]==nums[i+1]){
                duplicate = nums[i];
                break;
            }
        }
        return duplicate;
    }
};

// T.C. = O(NlogN)
// S.C. = O(N)
```

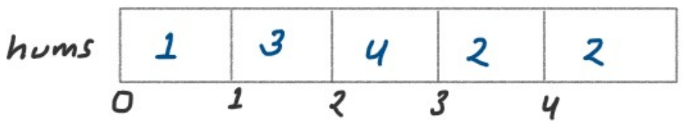
$\rightarrow$ T.C. = $O(N\log N)$

$\rightarrow$ T.C. = $O(n-1)$

$\Rightarrow O(N\log N) + O(n-1)$

$\Rightarrow O(N\log N)$

Method 02: *Negative marking approach*



① $nums[i] \in [1, N]$

② size = N+1

Output: 2

Step 01: Iterate the array
Step 02: Mark visited
Step 03: Already visited position then return duplicate

DRY RUN

Iteration 0

Element ko as a position Index man lo

1	3	4	2	2
0	1	2	3	4

$\uparrow$
i=0

nums[i] = position = 1

nums[nums[i]]=Element = 3

index=1

index = abs(nums[i])

(nums[index] >0)

3 >0

hums[index] *= -1;

nums[1] = -3

Iteration 1

1	-3	4	2	2
0	1	2	3	4

$\uparrow$
i=1

index=2

index = abs(nums[i])

(nums[index] >0)

4 >0

hums[index] *= -1;

nums[2] = -4

Iteration 2

1	-3	-4	2	2
0	1	2	3	4

$\uparrow$
i=2

index=4

index = abs(nums[i])

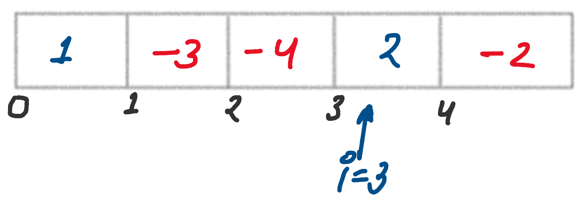
(nums[index]> 0)

2 >0

hums[index] *= -1;

nums[4] = -2

Iteration 3



index = 2

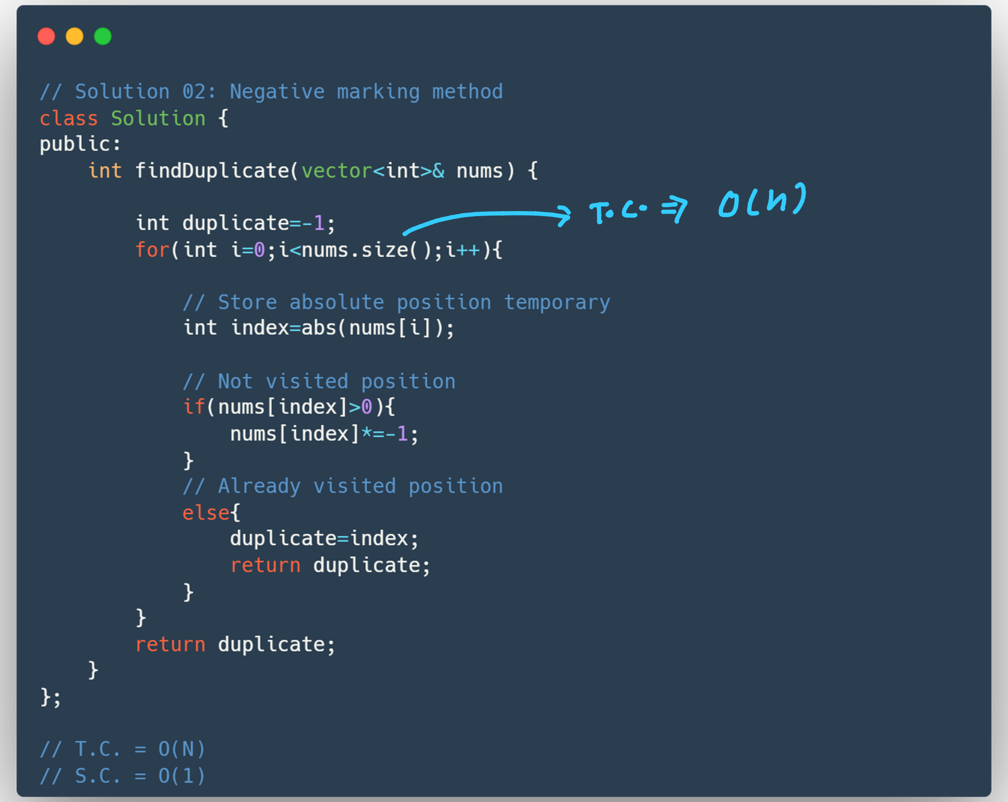
index = abs(nums[i])

(nums[index]> 0)

-4 > 0 False

Return Duplicate

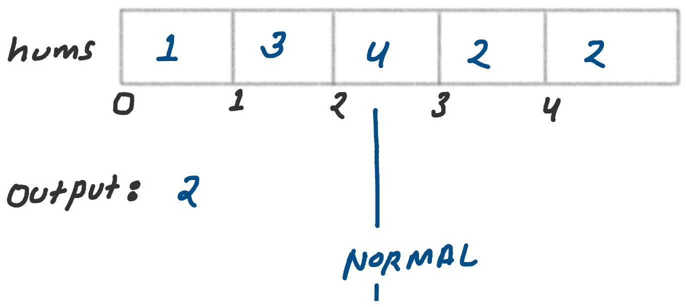
return INDEX =2



```
// Solution 02: Negative marking method
class Solution {
public:
    int findDuplicate(vector<int>& nums) {

        int duplicate=-1;
        for(int i=0;i<nums.size();i++){

            // Store absolute position temporary
            int index=abs(nums[i]);

            // Not visited position
            if(nums[index]>0){
                nums[index]*=-1;
            }
            // Already visited position
            else{
                duplicate=index;
                return duplicate;
            }
        }
        return duplicate;
    }
};

// T.C. = O(N)
// S.C. = O(1)
```

$T.C. \Rightarrow O(n)$

Method 03: Position and swapping approach (Position index==Element)



Output: 2

① $nums[i] \in [1,N]$

② $size = N+1$

↓

0	1	3	4	2
0	1	2	3	4

size = 5

Agar 0th index par zero ke alawa koi or element dobara se aata hai to iska matlb wo ek duplicate number hona chaiye

DRY RUN

num

1	3	4	2	2
0	1	2	3	4

While(num[0] != num[num[0]])

iter:0

1 != 3

swap(1,3)

num

3	1	4	2	2
0	1	2	3	4

iter:1

3 != 2

swap(3,2)

num

2	1	4	3	2
0	1	2	3	4

iter:2

2 != 4

swap(2,4)

num

4	1	2	3	2
0	1	2	3	4

iter:3

4 != 2

swap(4,2)

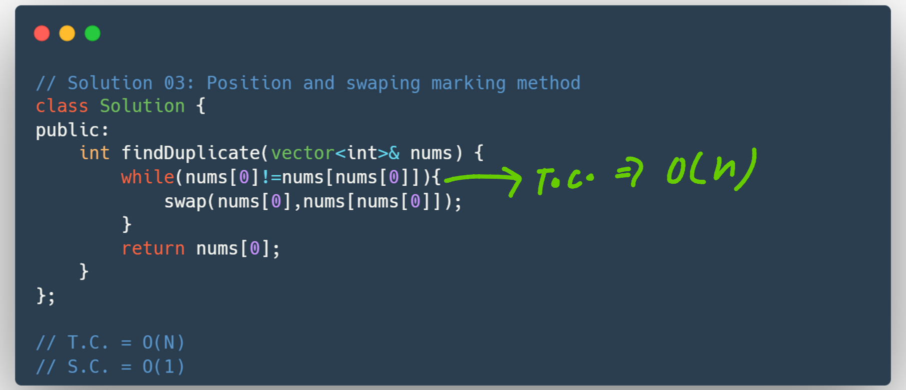
num

2	1	2	3	4
0	1	2	3	4

iter:4

2 != 2 False

return num[0] 2 is Duplicate



```
// Solution 03: Position and swaping marking method
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        while(nums[0]!=nums[nums[0]]){ ⟶ T.C. ⇒ O(n)
            swap(nums[0],nums[nums[0]]);
        }
        return nums[0];
    }
};

// T.C. = O(N)
// S.C. = O(1)
```