3289. The Two Sneaky Numbers of Digitville

Solved

Easy | Topics | Companies | Hint

In the town of Digitville, there was a list of numbers called `nums` containing integers from `0` to `n - 1`. Each number was supposed to appear **exactly once** in the list, however, **two** mischievous numbers sneaked in an *additional time*, making the list longer than usual.

As the town detective, your task is to find these two sneaky numbers. Return an array of size **two** containing the two numbers (in *any order*), so peace can return to Digitville.

Example 1:

> **Input:** `nums = [0,1,1,0]`
>
> **Output:** `[0,1]`
>
> **Explanation:**
>
> The numbers 0 and 1 each appear twice in the array.

Example 2:

> **Input:** nums = [0,3,2,1,3,2]
>
> **Output:** [2,3]
>
> **Explanation:**
>
> The numbers 2 and 3 each appear twice in the array.

Example 3:

> **Input:** nums = [7,1,5,4,3,4,6,0,9,5,8,2]
>
> **Output:** [4,5]
>
> **Explanation:**
>
> The numbers 4 and 5 each appear twice in the array.

Constraints:

- `2 <= n <= 100`
- `nums.length == n + 2`
- `0 <= nums[i] < n`
- The input is generated such that `nums` contains **exactly** two repeated elements.

Python:

```
class Solution:
    def getSneakyNumbers(self, nums):
        xor_sum = 0
        total_size = len(nums)
        actual_size = len(nums) - 2

        # XOR all elements of the array
        for num in nums:
```

```
        xor_sum ^= num

    # XOR all numbers from 0 to actual_size - 1
    for i in range(actual_size):
        xor_sum ^= i

    # Find the rightmost set bit in xor_sum
    rightmost_set_bit = xor_sum & ~(xor_sum - 1)

    first_sneaky_number = 0
    second_sneaky_number = 0

    # Separate the numbers into two groups based on the rightmost set bit
    for num in nums:
        if num & rightmost_set_bit:
            first_sneaky_number ^= num
        else:
            second_sneaky_number ^= num

    # XOR the range of numbers from 0 to actual_size - 1
    for i in range(actual_size):
        if i & rightmost_set_bit:
            first_sneaky_number ^= i
        else:
            second_sneaky_number ^= i

    return [first_sneaky_number, second_sneaky_number]
```

JavaScript:

```
const getSneakyNumbers = nums => {
    let xor = 0;
    const n = nums.length - 2;

    for (let num of nums) xor ^= num;
    for (let i = 0; i < n; i++) xor ^= i;

    const diffBit = xor & -xor;

    let a = 0, b = 0;
    for (let num of nums) {
        if ((num & diffBit) === 0) a ^= num;
        else b ^= num;
    }
    for (let i = 0; i < n; i++) {
```

```
        if ((i & diffBit) === 0) a ^= i;
        else b ^= i;
    }

    return [a, b];
};
```

Java:

```
class Solution {
    public int[] getSneakyNumbers(int[] nums) {
        int n = nums.length;
        int[] res = new int[2];
        HashSet<Integer> set = new HashSet<>();
        for(int i=0,j=0;i<n;i++){
            if(!set.isEmpty() && set.contains(nums[i])){
                res[j] = nums[i];
                j++;
            }
            else{
                set.add(nums[i]);
            }
        }
        return res;
    }
}
```