Single Number

Problem

Given a non-empty array of integers `nums`, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Examples

Input: nums = [2, 2, 1] → **Output:** 1
Input: nums = [4, 1, 2, 1, 2] → **Output:** 4
Input: nums = [1] → **Output:** 1

Constraints

$1 \leq$ nums.length $\leq 3 \times 10^4$
$-3 \times 10^4 \leq$ nums[i] $\leq 3 \times 10^4$
Each element appears twice except one that appears only once.

Approach 1: Brute Force (Hash Map)

Create an empty hash map to store counts of each element.
Loop through the array, update the count for each element.
Loop again to find the element with count 1 and return it.

Dry Run

Input: [4, 1, 2, 1, 2]

Step 1: Counting frequency

Insert 4 → hash[4] = 1
Insert 1 → hash[1] = 1
Insert 2 → hash[2] = 1
Update 1 → hash[1] = 2
Update 2 → hash[2] = 2

Step 2: Find element with count 1

4 → hash[4] = 1 → **Return 4**

Time and Space Complexity

Time Complexity: O(n)
We traverse the array twice: once for counting and once for checking.

Space Complexity: O(n)
The hash map may store counts for up to `n` elements in the worst case.

JavaScript | C++ | C | Java | Python

```javascript
var singleNumber = function(nums) {
    let hash = {};
    for (let i = 0; i < nums.length; i++) {
        if (!hash[nums[i]]) {
            hash[nums[i]] = 1;
        } else {
            hash[nums[i]]++;
        }
    }
    for (let i = 0; i < nums.length; i++) {
        if (hash[nums[i]] === 1) {
            return nums[i];
        }
    }
};
```

Approach 2 – Optimal using XOR

XOR of two same numbers is 0: `a ^ a = 0`

XOR of a number with 0 is the number itself: `a ^ 0 = a`

So, if all elements occur twice except one, XOR-ing all gives that unique number.

Dry Run

Input: [2, 3, 5, 2, 3]

Start `xor = 0`

xor = 0 ^ 2 = 2
xor = 2 ^ 3 = 1
xor = 1 ^ 5 = 4
xor = 4 ^ 2 = 6
xor = 6 ^ 3 = 5

Final answer: 5

Time and Space Complexity

Time Complexity: O(n)
where n is the number of elements in the array

Space Complexity: O(1)
no extra space used

JavaScript | C++ | C | Java | Python

```javascript
var singleNumber = function(nums) {
    let xor = 0;
    for (let i = 0; i < nums.length; i++) {
        xor = xor ^ nums[i];
    }
    return xor;
};
```