Single Number II

Problem Description

Given an integer array nums where every element appears three times except for one, which appears exactly once. Find the single element and return it.

Requirements

- Implement a solution with a linear runtime complexity.
- Use only constant extra space.

Example 1:

Input: nums = [2,2,3,2]

Output: 3

Example 2:

Input: nums = [0,1,0,1,0,1,99]

Output: 99

Constraints

- (1 leq text{nums.length} leq 3 times 10^4)
- (-2^{31} leq text{nums[i]} leq 2^{31} 1)
- Each element in nums appears exactly three times except for one element which appears once.

Solution Explanation

To solve this problem, we use a bitwise approach to keep track of the counts of each bit position across all numbers in the array. The main idea is to use two variables, ones and twos, to track the bits that appear once and twice respectively. The key operations are:

- **XOR Operation (^):** This helps in flipping the bits.
- **AND Operation (&):** This is used to reset the bits.
- **NOT Operation (~):** This helps in inverting the bits.

By iterating through the array, we update ones and twos to reflect the counts of the bits modulo 3. After processing all the numbers, the variable ones will hold the bits of the number that appears exactly once.

This approach ensures that we achieve linear runtime complexity ((O(n))) and use constant extra space ((O(1))).