Documentation: Single Number Problem

Objective

Given a non-empty array of integers nums, every element appears twice except for one. The task is to find that single unique element.

Requirements

- The solution must have a linear runtime complexity, i.e., O(n).
- Only constant extra space can be used.

Example 1

- **Input:** nums = [2, 2, 1]
- **Output:** 1

Example 2

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

Example 3

- **Input:** nums = [1]
- **Output:** 1

Constraints

- The length of the array nums is between 1 and 30,000 (inclusive).
- Each element in the array is an integer between -30,000 and 30,000 (inclusive).
- Every element appears exactly twice except for one element which appears only once.

Approach

The approach to solving this problem utilizes the XOR bitwise operation to find the single unique element in the array. The XOR operation has the following properties which make it useful for this problem:

1. a ^ a = 0: Any number XORed with itself results in 0.
2. a ^ 0 = a: Any number XORed with 0 remains unchanged.
3. XOR is both commutative and associative, meaning the order of operations does not matter.

Steps

1. Initialize a variable single_num to 0. This will hold the result.
2. Iterate through each number in the array.
3. XOR each number with single_num. Due to the properties of XOR, pairs of the same number will cancel each other out (result in 0), and the unique number will be left.
4. Return the result stored in single_num.

Complexity

- **Time Complexity:** O(n), where n is the length of the array. This is because each element is processed exactly once.
- **Space Complexity:** O(1), since only a constant amount of extra space is used regardless of the input size.