

Problem 2588: Count the Number of Beautiful Subarrays

Problem Information

Difficulty: Medium

Acceptance Rate: 52.89%

Paid Only: No

Tags: Array, Hash Table, Bit Manipulation, Prefix Sum

Problem Description

You are given a **0-indexed** integer array `nums`. In one operation, you can:

* Choose two different indices `i` and `j` such that `0 ≤ i, j < nums.length`. * Choose a non-negative integer `k` such that the `k`th bit (**0-indexed**) in the binary representation of `nums[i]` and `nums[j]` is `1`. * Subtract `2k` from `nums[i]` and `nums[j]`.

A subarray is **beautiful** if it is possible to make all of its elements equal to `0` after applying the above operation any number of times (including zero).

Return the number of **beautiful subarrays** in the array `nums`.

A subarray is a contiguous **non-empty** sequence of elements within an array.

Note : Subarrays where all elements are initially 0 are considered beautiful, as no operation is needed.

Example 1:

Input: `nums = [4,3,1,2,4]` **Output:** 2 **Explanation:** There are 2 beautiful subarrays in `nums`: `[4,3,1,2,4]` and `[4,3,1,2,4]`. - We can make all elements in the subarray `[3,1,2]` equal to 0 in the following way: - Choose `[3, 1, 2]` and `k = 1`. Subtract 2 from both numbers. The subarray becomes `[1, 1, 0]`. - Choose `[1, 1, 0]` and `k = 0`. Subtract 1 from both numbers. The subarray becomes `[0, 0, 0]`. - We can make all elements in the subarray `[4,3,1,2,4]` equal to 0 in the following way: - Choose `[4, 3, 1, 2, 4]` and `k = 2`. Subtract 4 from both numbers. The subarray becomes `[0, 3, 1, 2, 0]`. - Choose `[0, 3, 1, 2, 0]` and `k`

= 0. Subtract 20 from both numbers. The subarray becomes [0, 2, 0, 2, 0]. - Choose [0, __, 0, __, 0] and k = 1. Subtract 21 from both numbers. The subarray becomes [0, 0, 0, 0, 0].

Example 2:

Input: nums = [1,10,4] **Output:** 0 **Explanation:** There are no beautiful subarrays in nums.

Constraints:

$1 \leq \text{nums.length} \leq 10^5$ $0 \leq \text{nums}[i] \leq 10^6$

Code Snippets

C++:

```
class Solution {
public:
    long long beautifulSubarrays(vector<int>& nums) {

    }
};
```

Java:

```
class Solution {
    public long beautifulSubarrays(int[] nums) {

    }
}
```

Python3:

```
class Solution:
    def beautifulSubarrays(self, nums: List[int]) -> int:
```