6317. Count the Number of Beautiful Subarrays

My Submissions (/contest/weekly-contest-336/problems/count-the-number-of-beautiful-subarrays/submissions/) Back to Contest (/contest/weekly-contest-336/)

User Accepted:	4242
User Tried:	5463
Total Accepted:	4448
Total Submissions:	10807
Difficulty:	Medium

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 2^k 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.

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

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

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 2^1 from both numbers. The subarray becomes [1, 1, 0].
  - Choose [1, 1, 0] and k = 0. Subtract 2^0 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 2^2 from both numbers. The subarray becomes [0, 3, 1, 2, 0].
  - Choose [0, 3, 1, 2, 0] and k = 0. Subtract 2^0 from both numbers. The subarray becomes [0, 2, 0, 2, 0].
  - Choose [0, 2, 0, 2, 0] and k = 1. Subtract 2^1 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 <= nums.length <= 10^5`
- `0 <= nums[i] <= 10^6`

Python

```python
class Solution(object):
    def beautifulSubarrays(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        count = 0
        xor_sum = 0
        freq = {0: 1}
        for i in range(len(nums)):
            xor_sum ^= nums[i]
            if xor_sum in freq:
                count += freq[xor_sum]
            freq[xor_sum] = freq.get(xor_sum, 0) + 1
        return count
```

Custom Testcase Use Example Testcases

Run Submit