3583. Count Special Triplets

Medium | Topics | Companies | Hint

You are given an integer array `nums`.

A **special triplet** is defined as a triplet of indices `(i, j, k)` such that:

- `0 <= i < j < k < n`, where `n = nums.length`
- `nums[i] == nums[j] * 2`
- `nums[k] == nums[j] * 2`

Return the total number of **special triplets** in the array.

Since the answer may be large, return it **modulo** $10^9 + 7$.

Example 1:

> **Input:** `nums = [6,3,6]`
>
> **Output:** 1
>
> **Explanation:**
>
> The only special triplet is `(i, j, k) = (0, 1, 2)`, where:
>
> - `nums[0] = 6`, `nums[1] = 3`, `nums[2] = 6`
> - `nums[0] = nums[1] * 2 = 3 * 2 = 6`
> - `nums[2] = nums[1] * 2 = 3 * 2 = 6`

Example 2:

> **Input:** `nums = [0,1,0,0]`
>
> **Output:** 1
>
> **Explanation:**
>
> The only special triplet is `(i, j, k) = (0, 2, 3)`, where:
>
> - `nums[0] = 0`, `nums[2] = 0`, `nums[3] = 0`
> - `nums[0] = nums[2] * 2 = 0 * 2 = 0`
> - `nums[3] = nums[2] * 2 = 0 * 2 = 0`

Example 3:

Input: nums = [8,4,2,8,4]

Output: 2

Explanation:

There are exactly two special triplets:

- (i, j, k) = (0, 1, 3)
  - nums[0] = 8, nums[1] = 4, nums[3] = 8
  - nums[0] = nums[1] * 2 = 4 * 2 = 8
  - nums[3] = nums[1] * 2 = 4 * 2 = 8
- (i, j, k) = (1, 2, 4)
  - nums[1] = 4, nums[2] = 2, nums[4] = 4
  - nums[1] = nums[2] * 2 = 2 * 2 = 4
  - nums[4] = nums[2] * 2 = 2 * 2 = 4

Constraints:

- $3 <= n == nums.length <= 10^5$
- $0 <= nums[i] <= 10^5$

Python:

```
class Solution:
    def specialTriplets(self, A: List[int]) -> int:
        n = len(A)
        left, right = Counter(), Counter(A)
        res = 0
        for a in A:
```

```
            right[a] -= 1
            res += left[a * 2] * right[a * 2]
            left[a] += 1
        return res % (10 ** 9 + 7)
```

JavaScript:

```
/**
 * @param {number[]} nums
 * @return {number}
 */
var specialTriplets = function(nums) {
    const MOD = 1_000_000_007n;

    const left = new Map();
    const right = new Map();

    for (const x of nums) {
        right.set(x, (right.get(x) || 0n) + 1n);
    }

    let ans = 0n;

    for (const x of nums) {
        right.set(x, right.get(x) - 1n);

        const need = BigInt(x * 2);
        const lc = left.get(Number(need)) || 0n;
        const rc = right.get(Number(need)) || 0n;

        ans = (ans + lc * rc) % MOD;

        left.set(x, (left.get(x) || 0n) + 1n);
    }

    return Number(ans);
};
```

Java:

```
class Solution {
        public int specialTriplets(int[] A) {
        int mod = 1_000_000_007, res = 0;
        Map<Integer, Integer> left = new HashMap<>(), right = new HashMap<>();
        for (int a : A) {
            right.put(a, right.getOrDefault(a, 0) + 1);
        }
```

```
        for (int a : A) {
            right.put(a, right.get(a) - 1);
            int ci = left.getOrDefault(a * 2, 0);
            int ck = right.getOrDefault(a * 2, 0);
            res = (int)((res + 1L * ci * ck) % mod);
            left.put(a, left.getOrDefault(a, 0) + 1);
        }
        return res;
    }
}
```