60. Count Triplets That Can Form Two Arrays of Equal XOR

Given an array of integers arr.

We want to select three indices i, j and k where (0 <= i < j <= k < arr.length).

Let's define a and b as follows:

- a = arr[i] ^ arr[i + 1] ^ ... ^ arr[j - 1]
- b = arr[j] ^ arr[j + 1] ^ ... ^ arr[k]

Note that ^ denotes the bitwise-xor operation.

Return the number of triplets (i, j and k) Where a == b.

Example 1:

Input: arr = [2,3,1,6,7]

Output: 4

Explanation: The triplets are (0,1,2), (0,2,2), (2,3,4) and (2,4,4)

Example 2:

Input: arr = [1,1,1,1,1]

Output: 10

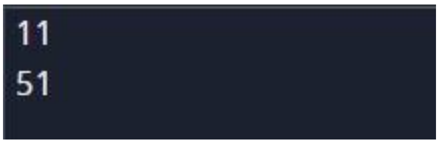
AIM: To Count Triplets That Can Form Two Arrays of Equal XOR

PROGRAM:

```
def countTriplets(arr):
    count = 0
    xor_prefix = [0]
    for num in arr:
        xor_prefix.append(xor_prefix[-1] ^ num)


    xor_count = {}
    for i in range(len(arr) + 1):
        for j in range(i + 1, len(arr) + 1):
            xor_val = xor_prefix[i] ^ xor_prefix[j]
            if xor_val in xor_count:
                count += xor_count[xor_val]
            if xor_val in xor_count:
                xor_count[xor_val] += 1
            else:
                xor_count[xor_val] = 1


    return count
arr1 = [2, 3, 1, 6, 7]
print(countTriplets(arr1))
arr2 = [1, 1, 1, 1, 1]
print(countTriplets(arr2))
```

OUTPUT: 

```
11
51
```

TIME COMPLEXITY: O(n^2)