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.

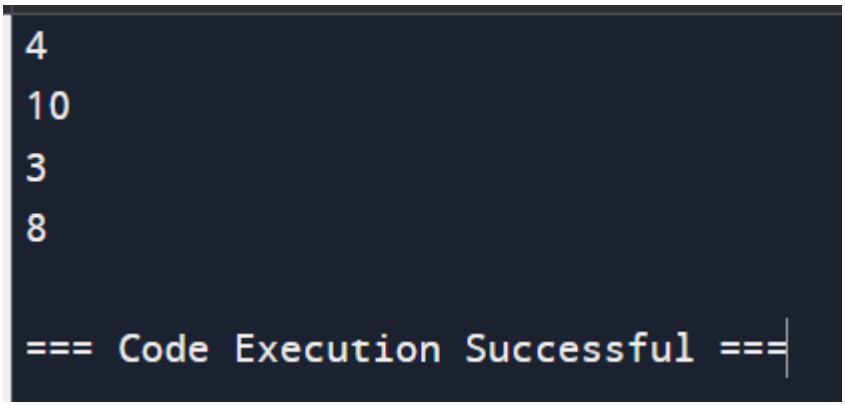
Return the number of triplets (i, j and k) Where a == b.

Program:

```
def count_triplets(arr):
    n = len(arr)
    prefix_xor = [0] * (n + 1)

    # Calculate prefix XOR array
    for i in range(n):
        prefix_xor[i + 1] = prefix_xor[i] ^ arr[i]

    count = 0
    # Map to store the number of times a particular prefix_xor has occurred and the sum of indices
    xor_count = {}
    index_sum = {}

    for j in range(n):
        if prefix_xor[j + 1] in xor_count:
            count += xor_count[prefix_xor[j + 1]] * j - index_sum[prefix_xor[j + 1]]

        # Update the count and sum of indices for the current prefix_xor
        if prefix_xor[j] in xor_count:
            xor_count[prefix_xor[j]] += 1
            index_sum[prefix_xor[j]] += j
        else:
```

```
            xor_count[prefix_xor[j]] = 1
            index_sum[prefix_xor[j]] = j

    return count

# Example usage
arr = [2, 3, 1, 6, 7]
print(count_triplets(arr))  # Output: 4

arr = [1, 1, 1, 1, 1]
print(count_triplets(arr))  # Output: 10

arr = [1, 3, 5, 7, 9]
print(count_triplets(arr))  # Output: 3

arr = [7, 11, 12, 9, 5, 2, 7, 17, 22]
print(count_triplets(arr))  # Output: 8
```

Output:



Time complexity:O(n^2)