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

Program:

```
def countTriplets(arr):
    n = len(arr)
    prefixXor = [0] * (n + 1)
    for i in range(n):
        prefixXor[i + 1] = prefixXor[i] ^ arr[i]
    count = 0
    for i in range(n):
        for k in range(i + 1, n):
            if prefixXor[i] == prefixXor[k + 1]:
                count += (k - i)
    return count
```

arr = [2, 3, 1, 6, 7]
print(countTriplets(arr))
Output:

Time complexity:
O(n^2)