6. 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]

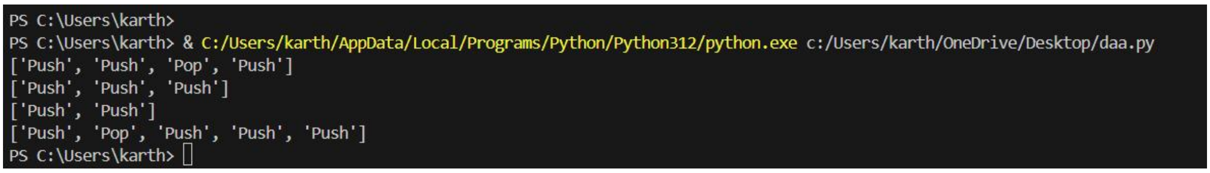
Note that ^ denotes the bitwise-xor operation.

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

Code:

```
def countTriplets(arr):
    n=len(arr)
    count=0
    prefix_xor=[0]*(n + 1)
    for i in range(n):
        prefix_xor[i + 1]=prefix_xor[i]^arr[i]
    for i in range(n):
        for k in range(i + 1, n):
            if prefix_xor[i]==prefix_xor[k + 1]:
                count += k - i
    return count
arr=[2, 3, 1, 6, 7]
print(countTriplets(arr))
arr=[1, 1, 1, 1, 1]
print(countTriplets(arr))
arr=[2, 3, 1, 6, 7, 2, 7]
print(countTriplets(arr))
```

output:



```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Desktop/daa.py
['Push', 'Push', 'Pop', 'Push']
['Push', 'Push', 'Push']
['Push', 'Push']
['Push', 'Pop', 'Push', 'Push', 'Push']
PS C:\Users\karth> 
```

time complexity:

$f(n)=o(n)$