■ Subarray XOR Equal to K

Given an array of integers and an integer K, count the number of subarrays such that the XOR of all elements in the subarray is equal to K.

■ Intuition: - XOR behaves like addition in prefix-sum problems but with different properties. - If we know the XOR of prefix [0..i] and prefix [0..j], then XOR of subarray [i+1..j] can be computed as: prefixXOR[j] ^ prefixXOR[i] - We maintain a running XOR (`xr`) as we traverse the array. - If `xr == k`, then the subarray from 0 to current index has XOR = k. - Otherwise, we check if there exists a prefix with XOR = (xr ^ k). If yes, then some subarray between that prefix and the current index has XOR = k. - We store the frequency of each prefix XOR in a hashmap (dictionary).

■ Algorithm: 1. Initialize count = 0, xr = 0, and a hashmap to store frequencies of prefix XORs. 2. Traverse the array: - Update xr with current element. - If xr == k, increment count. - Compute rem = xr ^ k. - If rem exists in hashmap, add its frequency to count. - Store current xr in hashmap (increase its frequency). 3. Return count at the end.

Python Code:

```
from collections import defaultdict

class Solution:
    def subarrayXor(self, arr, k):
        n = len(arr)
        count = 0
        xr = 0
        mpp = defaultdict(int)

        for i in range(n):
            xr ^= arr[i]

            if xr == k:
                count += 1

            rem = xr ^ k
            count += mpp.get(rem, 0)

            mpp[xr] += 1

        return count
```