Count the no. of subarray with xor k.

Problem Statement Suggest Edit

Given an array ***'A'*** consisting of ***'N'*** integers and an integer ***'B'***, find the number of subarrays of array 'A' whose bitwise XOR($\oplus$) of all elements is equal to 'B'.

A subarray of an array is obtained by removing some(zero or more) elements from the front and back of the array.

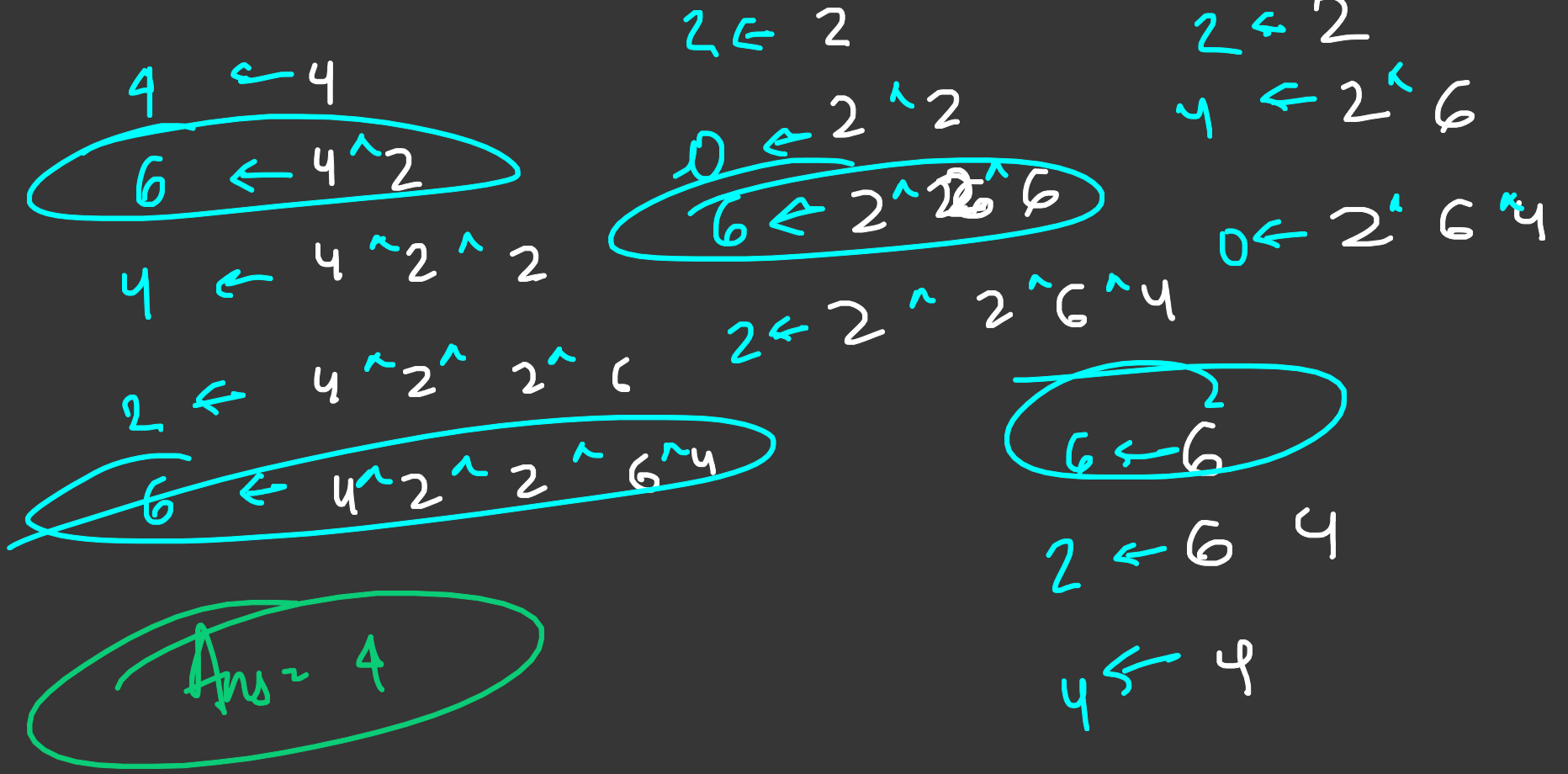
Example:

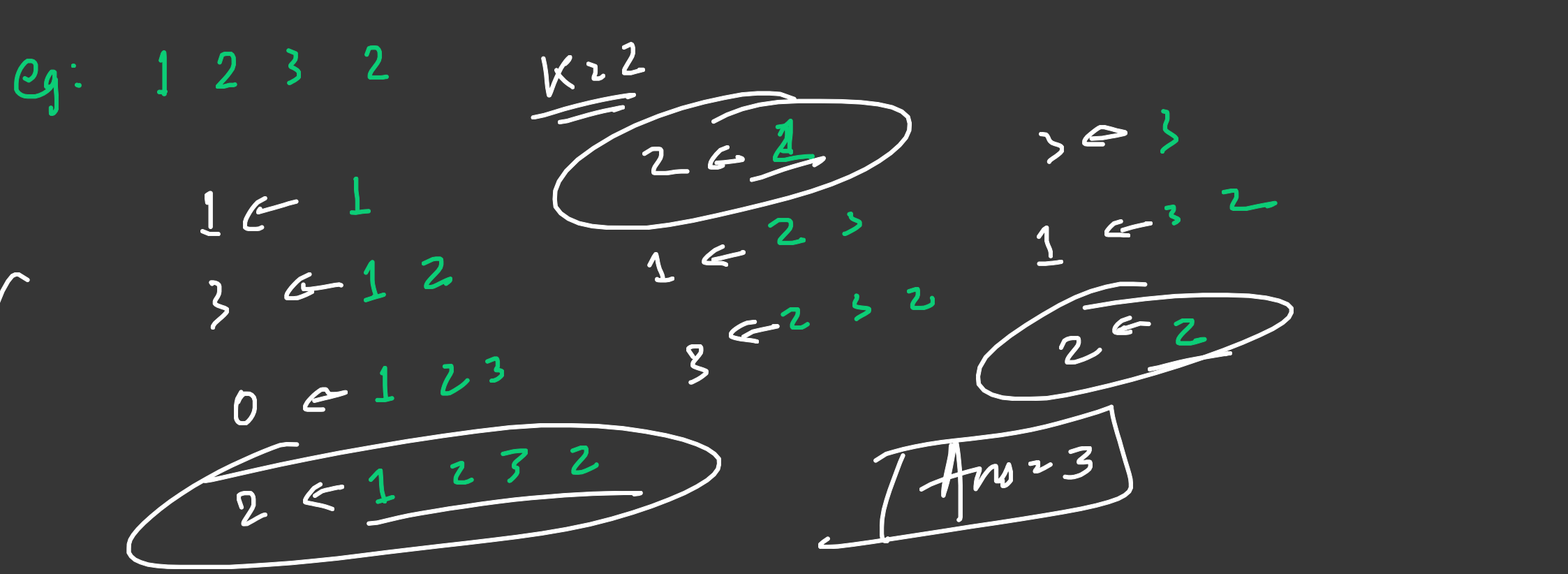
```
Input: 'N' = 4 'B' = 2
'A' = [1, 2, 3, 2]

Output: 3

Explanation: Subarrays have bitwise xor equal to '2' are: [1, 2, 3, 2], [2], [2].
```

arr[] : { 4, 2, 2, 6, 4 } K = 6

Brute / Naive approach:

Generate all Subarray and XOR them to get K=6



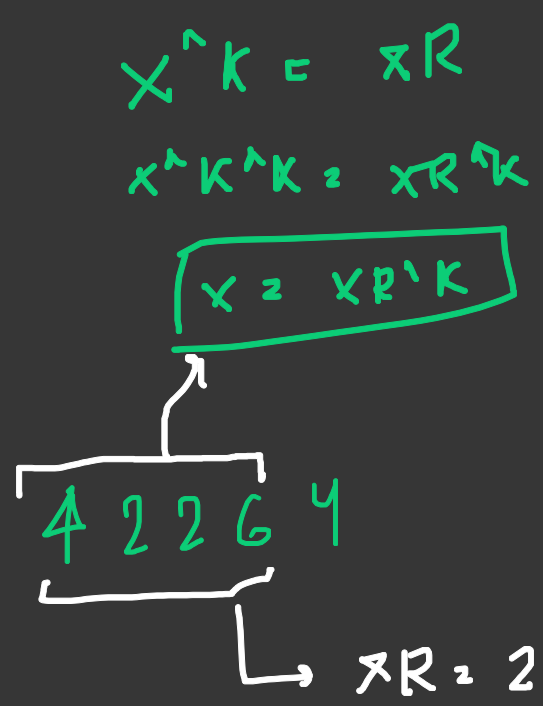


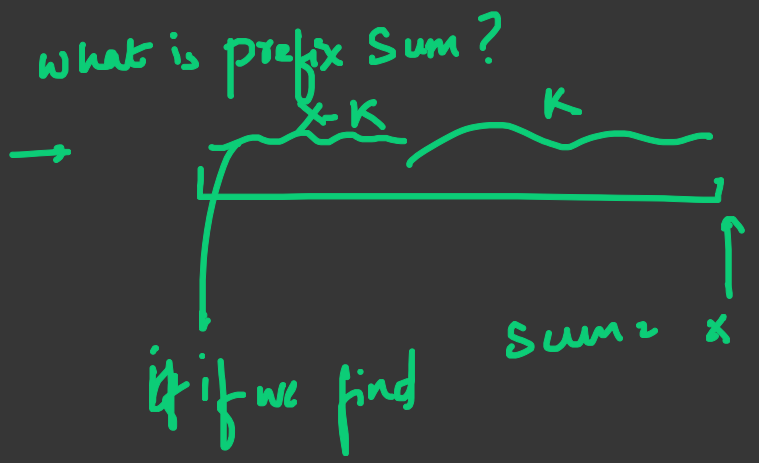
So, Brute force is simple. but. Time Complexity is $O(N^2)$.

Optimal Approach:

4 2 2 6 4

K = 2





Hashmap to keep the track of XOR

$K = 6$

4 2 2 6 4

$x = XR \oplus K$

$x = 4 \oplus 6 = 2$

$x = 6 \oplus 6 = 0$

$x = 4 \oplus 6 = 2$

$x = 2 \oplus 6 = 4$

$x = 6 \oplus 6 = 0$

XR = 4, 6, 4, 2, 6

count = 1, 3, 4

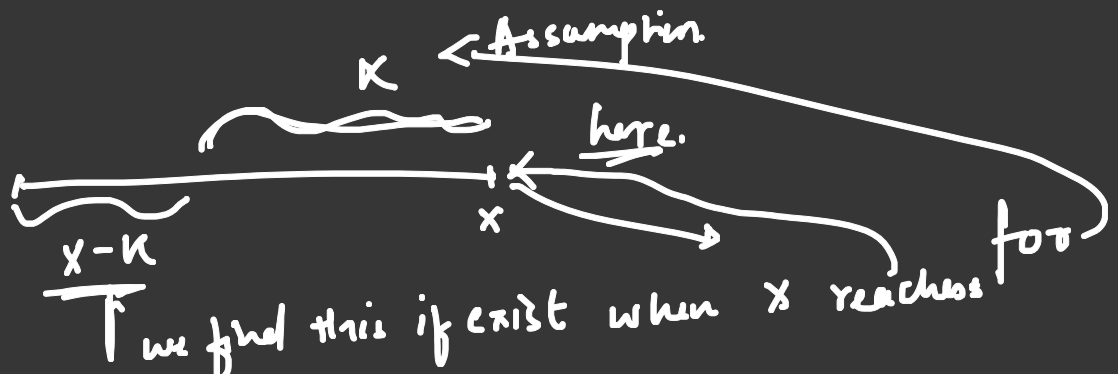
Ans = 4

check

XOR	cnt
2	1
6	1
4	2
0	1

Explanation of the intution and Algo:

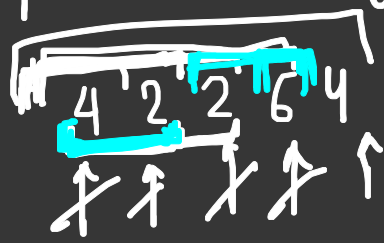
- Idea is to use prefix Sum Concept



eg: 4 2 2 6 4

Dry Run

→ To keep the track of the pre XOR value we use map.



XR: 4 6 4 2 6

cnt: 0 0 0 1

x = XR ^ K

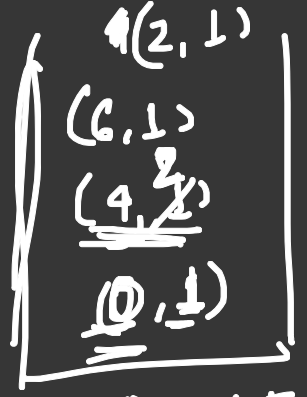
= 4 ^ 6 = 2

6 ^ 6 = 0

4 ^ 6 = 2

2 ^ 6 = 4

6 ^ 6 = 0



4 2 2 6 4

1 0 0
0 1 0
1 1 0

Pseudocode:

```
Initial the map.
    XR = 0
map[XR]++;
```

means.


```
for (i=0; i<n; i++)
    XR ^= arr[i];

    X = XR^K;
    Ans count += mp[X];
    mp[XR] ++;

}
return count;
```

Time Complexity:

ifmap <int, int> ⟶ operation O(LogN) ⟶ Worst Case.

Unordered <int, int> ⟶ operation Best Case O(1)
Worst Case O(N) ⟶ if there is many collision

T.C. = O(N) * O(log(N)) or O(N) * O(1) ⟶ O(N²)

O(N) ⟶ iteration
O(log(N)) ⟶ map
O(1) ⟶ unordered map

S.C. = O(N) ⟶ at worst it will store all array element