3.Hard\06.Subarrays_with_xor_k.cpp

```cpp
/*QUESTION:
Given an array 'A' consisting of 'N' integers and an integer 'B', find the number of subarrays of array 'A' whose bitwise XOR of all elements is equal to 'B'.

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].

APPROACH:
To find the number of subarrays with bitwise XOR equal to B, we can use the technique of prefix XOR along with a hashmap.
1. Initialize a variable `prefixXOR` to keep track of the prefix XOR while iterating through the array.
2. Initialize a variable `count` to keep track of the count of subarrays with XOR equal to B.
3. Initialize a hashmap `xorCount` to store the frequency of prefix XOR values encountered so far.
4. Set the initial prefix XOR to 0 and set its count to 1 in the `xorCount` hashmap.
5. Iterate through the array and update the prefix XOR by XOR-ing each element.
6. Check if the current prefix XOR is equal to B. If it is, increment the `count` variable.
7. Check if the XOR of the current prefix XOR with B exists in the `xorCount` hashmap. If it does, add the count of that XOR value to the `count` variable.
8. Increment the count of the current prefix XOR in the `xorCount` hashmap.
9. Finally, return the `count` variable as the number of subarrays with XOR equal to B.

CODE:
*/

int subarraysWithSumK(vector<int> a, int b) {
    int pref_xr = 0;
    int ans = 0;
    unordered_map<int, int> mp;

    for(int i = 0; i < a.size(); i++){
        pref_xr = pref_xr ^ a[i];

        if(pref_xr == b)
            ans++;

        if(mp.find(pref_xr ^ b) != mp.end()){
            ans += mp[pref_xr ^ b];
        }

        mp[pref_xr]++;
    }

    return ans;
}

/*
TIME COMPLEXITY: O(n), where n is the size of the input array a.
```

```
SPACE COMPLEXITY: O(n), as we are using a hashmap to store the prefix XOR values and their corresponding counts.
*/


```