XOR

Thursday, 30 May 2024 10:36 AM

- Rule of XOR -
  - 0 XOR 0 = 0
  - 0 XOR 1 = 1
  - 1 XOR 0 = 1
  - 1 XOR 1 = 0

- **Use & to check for conflicts** → If usedBits & nums[r] is **non-zero**, there's a bit overlap.
- **Use ^ to remove the leftmost number** → Helps shrink the window when a conflict arises.
- **Use | to add the new number** → Expands the window when no conflict is found.
- The **AND (&) check** ensures uniqueness of bits in the current subarray.
- The **XOR (^) operation** efficiently removes old numbers from the window.
- The **OR (|) operation** adds new numbers.

Properties of XOR:
  - **Identity:** a XOR 0 = a
  - **Self-inverse:** a XOR a = 0
  - **Commutative:** a XOR b = b XOR a
  - **Associative:** (a XOR b) XOR c = a XOR (b XOR c)

- XOR can be used to swap two variables without a temporary variable:
  - a = a XOR b
  - b = a XOR b
  - a = a XOR b
- xor & -xor isolates the rightmost set bit in xor. This is because -xor is the two's complement of xor, which is ~xor + 1.

- (num&mask)!=0 compares the bit wise AND for both the numbers.
- In an array where every element appears twice except for one, XOR can find the unique element.
  - unique = 0
  - For(int num:nums)
    - unique ^= num

- In an array of numbers:
  - If we need to find the subarray which has XOR of 0, it means that all the elements are nullified, for e.g [2,3,4,5,5,2,4,3] all the elements appear twice so the entire array has XOR of 0.
  - In an array if the we need to find triplet where the XOR values are equal, then we first need to find the subarray where XOR is 0, then all the elements between the subarray will be triplets as explained in this :
  - 
    - https://leetcode.com/problems/count-triplets-that-can-form-two-arrays-of-equal-xor/discuss/5228547/GOD-Level-Detailed-Explanation-oror-2-Approaches
- consider there are 2 arrays num1 and num2, if I need to do xor for all the values in num1 to all the values of num2.. there are m x n xor values and I want to do xor of all the elements in the resultant array.
  - Solution:
    Understand Repetition:
    - Check how many times each element contributes to the final XOR.
    - If elements are repeated an even number of times, they cancel out.

    Simplify Using Lengths:
    - Use the size of the arrays to determine whether contributions cancel out.

- Odd lengths retain contributions, while even lengths cancel them.

Key Pattern:

- XOR problems often involve detecting repetition, symmetry, or patterns. Think about how XOR behaves when elements appear repeatedly.

https://leetcode.com/problems/bitwise-xor-of-all-pairings/?envType=daily-question&envId=2025-01-16

- Bit manipulation basics:
  - To find the 1 and 0 in a number

```
for (int bit = 0; bit < 32; bit++) {
                if ((nums & (1 << bit)) != 0) {
                    bitCount[bit]++;
                }
            }
```