

■ ■ XOR of Numbers in Range

Bit Manipulation • Medium (Prefix XOR Pattern)

■ The Hook

Given a range $[L, R]$, find XOR of all integers from L to R . Naively, loop from L to R — $O(n)$. But XOR from 0 to n has a repeating 4-cycle pattern. Once you spot it, you compute XOR of any range in $O(1)$. This pattern appears in competitive programming ranges and interview problems alike.

■ The 4-Cycle Pattern — XOR(0..n)

XOR of all numbers from 0 to n follows a cycle of 4 based on $n \% 4$:

$n \% 4$	XOR(0..n)	Example
0	n	XOR(0..4) = 4
1	1	XOR(0..5) = 1
2	$n+1$	XOR(0..6) = 7
3	0	XOR(0..7) = 0

■ ■ Range XOR Formula

$$\text{XOR}(L..R) = \text{XOR}(0..R) \wedge \text{XOR}(0..L-1)$$

This works because XOR is its own inverse. $\text{XOR}(0..L-1) \wedge \text{XOR}(0..R)$ cancels everything before L , leaving only $L..R$. Both prefix XORs are computed using the 4-cycle lookup above.

■ Dry Run — XOR(4..7)

■ XOR(0..7): $7 \% 4 = 3 \rightarrow \text{result} = 0$

■ XOR(0..3): $3 \% 4 = 3 \rightarrow \text{result} = 0$

■ XOR(4..7) = $0 \wedge 0 = 0$

Verify: $4^5^6^7 = 1^6^7 = 7^7 = 0 \checkmark$

■ Java Implementation

```

public class XORRange {
    // XOR of all numbers from 0 to n in O(1)
    public static int xorUpTo(int n) {
        switch (n % 4) {
            case 0: return n;
            case 1: return 1;
            case 2: return n + 1;
            default: return 0; // case 3
        }
    }

    // XOR from L to R inclusive
    public static int xorRange(int L, int R) {
        return xorUpTo(R) ^ xorUpTo(L - 1);
    }
}

```

■ Complexity

Case	Time	Space	Why
Brute force	O(n)	O(1)	Loop from L to R
Optimized	O(1)	O(1)	4-cycle lookup + one XOR

■ Practice Problems

Problem	Difficulty
XOR Queries of a Subarray (LeetCode 1310)	Medium
Find XOR of numbers in range [L,R]	Medium
Count Triplets (LeetCode 1442)	Medium
Maximum XOR With an Element From Array (LeetCode 1707)	Hard

■ Quick Revision

- $XOR(L..R) = XOR(0..R) \oplus XOR(0..L-1)$ (prefix XOR)
- $XOR(0..n)$ follows a 4-cycle: $n, 1, n+1, 0$ based on $n\%4$
- O(1) for any range after recognizing the pattern

■ Mental Model

Think of $\text{XOR}(0..n)$ as a prefix sum but for XOR. Just like $\text{sum}(L..R) = \text{prefixSum}[R] - \text{prefixSum}[L-1]$, $\text{XOR}(L..R) = \text{prefixXOR}[R] \oplus \text{prefixXOR}[L-1]$. The 4-cycle is a shortcut to compute $\text{prefixXOR}[n]$ in $O(1)$.