

⚡ Two No. with odd occurrences.

You are given an array '*arr*' of size '*n*'.

It contains an even number of occurrences for all numbers except two numbers.

Find those two numbers which have odd occurrences and return in decreasing order.

Example:

For '*arr*' = {2, 4, 1, 3, 2, 4}, '*n*' = 6.

Answer is {3, 1}.

Here, numbers 1, 3 have occurrence 1 i.e. odd and numbers 2, 4 have occurrence 2 i.e. even.

→ Naive Approach :-

↳ Store the count of numbers.

↳ Return the two no. with odd no. of occurrence.

→ Optimized Approach :-

2	4	1	3	2	4
---	---	---	---	---	---

1. Run a loop : $2 \oplus 4 \oplus 1 \oplus 3 \oplus 2 \oplus 4 = 1 \oplus 3$

$= 10$

↑

set bit indicates that this particular bit is different in 1 & 3 when written in form of binary.

2. Calculate rightmost set bit

XOR & -XOR

↳ 2's complement.

3. Now traverse the array & get the value of both the elements.

```
vector<int> twoOddNum(vector<int> arr) {
    int XOR = 0;
    for (int num : arr) {
        XOR ^= num;
    }

    int rightmostSetBit = XOR & -XOR;
    int num1 = 0, num2 = 0;
    for (int num : arr) {
        if (num & rightmostSetBit) {
            num1 ^= num;
        } else {
            num2 ^= num;
        }
    }
    if (num1 < num2) {
        swap(num1, num2);
    }
    return {num1, num2};
}
```