✅ Tumhara Approach: Priyansh ka Logic

1. Array `arr` me elements `0` se `n` ke beech hain, length = `n`.
2. Ideal position har element ka `arr[i]` ka index = `arr[i]` hona chahiye (agar value < n).
3. Tum kya kar rahe ho:
   - Jab tak element apni jagah par nahi hai, swap karte raho.
   - Jab sahi position aa jaye, next index par move.
4. End me, jo index aur value mismatch kare, wahi missing number hoga.
5. Agar koi mismatch nahi mila, missing number = `n`.

🔍 Tumhara Code ka Explanation Line by Line:

```
int missingNumber(vector<int>& arr) {
    int i = 0;
    int n = arr.size();

    // Step 1: Place every element at its correct index
    while(i < n) {
        if(arr[i] < n && arr[i] != i) {
            swap(arr[i], arr[arr[i]]);
        } else {
            i++;
        }
    }

    // Step 2: Check which index is incorrect
    for(int i = 0; i < n; i++) {
        if(arr[i] != i) return i; // Found the missing number
```

```
    }

    // Step 3: If all are correct, missing number is n
    return n;
}
```

✅ Dry Run Priyansh ke Perspective se:

Example: `nums = [3, 0, 1]`

`n = 3`

- `i = 0`: arr[0] = 3 → condition `(arr[i] < n)` fail → i++
- `i = 1`: arr[1] = 0 → condition true → swap(arr[1], arr[arr[1]])

  Swap(0th and 1st) → `[0, 3, 1]`
- i stays 1: arr[1] = 3 → fail → i++
- `i = 2`: arr[2] = 1 → true → swap(arr[2], arr[arr[2]])

  Swap(2nd and 1st) → `[0, 1, 3]`
- i stays 2: arr[2] = 3 → fail → i++
- Done.

Now second loop:

- i=0 → arr[0] = 0 ✅
- i=1 → arr[1] = 1 ✅
- i=2 → arr[2] = 3 ❌ → return 2 ✅

Answer = **2**.

✅ Complexity:

- **Time:** O(n) (kyunki har element max 1-2 swaps lega).
- **Space:** O(1).

🔥 Priyansh ka tareeqa kyun OP hai?

- Tumne **extra space use nahi kiya**, unlike boolean array method.
- Tumne **sum method** ya **XOR trick** avoid kiya, but ye approach **more generalized** hai (ye logic aur problems me bhi kaam aata hai like "First Missing Positive").
- Tumhara code neat hai aur interview friendly hai.
