LeetCode:- 287

✅ Problem Understanding

I'm given an array `nums` of size `n + 1`, where:

- Each element is in the range `[1, n]`.
- Only one number is repeated (it may appear more than twice).
- I **cannot modify the array**, and I have to use **O(1) extra space** with **linear time complexity**.

Example:

- `nums = [1,3,4,2,2]` → Output: `2`.

The key idea is that because there are `n + 1` numbers and only `n` possible values, **pigeonhole principle** guarantees at least one duplicate.

✅ My Current Approach in Code

```
class Solution {
public:
    int findDuplicate(vector<int>& arr) {
        int i = 0;
        int len = arr.size();
        while(i < len) {
            int correct_index = arr[i];
            if(arr[i] == arr[correct_index]) return arr[i];
            else swap(arr[i], arr[correct_index]);
        }
        return 100;
    }
};
```

✅ How My Code Works (Step-by-Step Thought Process)

1. Variables

- `i = 0` → I'll start from the first index.
- `len = arr.size()` → Total elements = `n + 1`.

2. While Loop Logic

- Run `while(i < len)` to traverse the array.

3. Calculate the Correct Index

- I assume that if the array were perfectly arranged, each number `x` would be at index `x`.
- So `correct_index = arr[i]`.

4. Check for Duplicate

- If `arr[i] == arr[correct_index]`, then **the current number is already in its correct position AND it matches another value → this means duplicate found**.
- So I `return arr[i]`.

5. Otherwise, Swap

- If `arr[i] != arr[correct_index]`, I swap them to move `arr[i]` to its correct position.

✅ Issues in My Code

- This approach **modifies the array**, which violates the condition.
- The loop has no increment for `i` in the else part (but since swap changes elements, eventually `i` will progress, still it's risky).
- It's based on **Cyclic Sort technique**, which works when modification is allowed.

✅ Key Learnings

- **Current code solves the problem if modification is allowed**, but **does not meet the original constraints** (cannot modify array).
- For the actual constraint solution, I should use **Floyd’s Cycle Detection (Tortoise and Hare)** because the array values represent a linked list structure.