**287. Find the Duplicate Number: -**

Medium Accepted: 1.3M Submissions: 2.3M Acceptance Rate: 59.2%

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.

There is only **one repeated number** in nums, return *this repeated number*.

You must solve the problem **without** modifying the array nums and uses only constant extra space.

**Example 1:**

**Input:** nums = [1,3,4,2,2]

**Output:** 2

**Example 2:**

**Input:** nums = [3,1,3,4,2]

**Output:** 3

**Constraints:**

* 1 <= n <= 105
* nums.length == n + 1
* 1 <= nums[i] <= n
* All the integers in nums appear only **once** except for **precisely one integer** which appears **two or more** times.

**Follow up:**

* How can we prove that at least one duplicate number must exist in nums?
* Can you solve the problem in linear runtime complexity?

**Code: -**

class Solution {

public:

    int findDuplicate(vector<int>& nums) {

      // using FLOYD's Cycle Detection Algorithm

      // Chain (containing cycle) is formed using INDEX

      int slow = nums[0];

      int fast = nums[0];

      do{

        slow = nums[slow];

        fast = nums[nums[fast]];

      }while(slow != fast);

      cout<<fast;

      slow = nums[0];

      while(slow != fast){

        slow = nums[slow];

        fast = nums[fast];

      }

      return slow;

    }

};