Find the Duplicate Number - Notes

Problem: Given an array of integers nums containing n + 1 integers where each integer is between 1 and n (inclusive), prove that at least one duplicate number must exist. Assume that there is only one duplicate number, find the duplicate one. Constraints: - You must not modify the array (assume the array is read-only). - You must use only constant extra space. - Your runtime complexity should be less than O(n^2).

Approach (Floyd's Tortoise and Hare Algorithm - Cycle Detection): 1. The array values can be treated as pointers to the next index, forming a cycle due to the duplicate number. 2. Use two pointers (slow and fast): - Move slow pointer by one step, fast pointer by two steps until they meet (detecting a cycle). 3. Reset fast pointer to the start, then move both one step at a time until they meet again. - The meeting point is the duplicate number. Time Complexity: O(n) Space Complexity: O(1)

```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int slow = nums[0];
        int fast = nums[0];

        do {
            slow = nums[slow];
            fast = nums[nums[fast]];
        } while (slow != fast);

        fast = nums[0];
        while (slow != fast) {
            slow = nums[slow];
            fast = nums[fast];
        }

        return slow;
    }
};
```