Find the Duplicate Number

> Given an array 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.
>
> Note: You must not modify the array (assume the array is read only). You must use only constant, O(1) extra space. Your runtime complexity should be less than O(n2). There is only one duplicate number in the array, but it could be repeated more than once.

哈希表法

复杂度

时间 O(N) 空间 O(N)

思路

遍历数组时，用一个集合记录已经遍历过的数，如果集合中已经有了说明是重复。但这样要空间，不符合。

暴力法

复杂度

时间 O(N^2) 空间 O(1)

思路

如果不用空间的话，最直接的方法就是选择一个数，然后再遍历整个数组看是否有跟这个数相同的数就行了。

排序法

时间 O(NlogN) 空间 O(1)

思路

更有效的方法是对数组排序，这样遍历时遇到前后相同的数便是重复，但这样要修改原数组，不符合要求。

二分法

复杂度

时间 O(NlogN) 空间 O(1)

思路

实际上，我们可以根据抽屉原理简化刚才的暴力法。我们不一定要依次选择数，然后看是否有这个数的重复数，我们可以用二分法先选取n/2，按照抽屉原理，整个数组中如果小于等于n/2的数的数量大于n/2，说明1到n/2这个区间是肯定有重复数字的。比如6个抽屉，如果有7个袜子要放到抽屉里，那肯定有一个抽屉至少两个袜子。这里抽屉就是1到n/2的每一个数，而袜子就是整个数组中小于等于n/2的那些数。这样我们就能知道下次选择的数的范围，如果1到n/2区间内肯定有重复数字，则下次在1到n/2范围内找，否则在n/2到n范围内找。下次找的时候，还是找一半。

注意

- 我们比较的 `mid` 而不是 `nums[mid]`
- 因为mid是下标，所以判断式应为 `cnt > mid` ，最后返回 `min`

代码

```
public class Solution {
    public int findDuplicate(int[] nums) {
        int min = 0, max = nums.length - 1;
        while(min <= max){
            // 找到中间那个数
            int mid = min + (max - min) / 2;
            int cnt = 0;
            // 计算总数组中有多少个数小于等于中间数
            for(int i = 0; i < nums.length; i++){
                if(nums[i] <= mid){
                    cnt++;
                }
            }
            // 如果小于等于中间数的数量大于中间数，说明前半部分必有重复
            if(cnt > mid){
                max = mid - 1;
            // 否则后半部分必有重复
            } else {
                min = mid + 1;
            }
        }
```

```
    }
```

映射找环法

复杂度

时间 O(N) 空间 O(1)

思路

假设数组中没有重复，那我们可以做到这么一点，就是将数组的下标和1到n每一个数一对一的映射起来。比如数组是 `213` ,则映射关系为 `0->2, 1->1, 2->3` 。假设这个一对一映射关系是一个函数f(n)，其中n是下标，f(n)是映射到的数。如果我们从下标为0出发，根据这个函数计算出一个值，以这个值为新的下标，再用这个函数计算，以此类推，直到下标超界。实际上可以产生一个类似链表一样的序列。比如在这个例子中有两个下标的序列， `0->2->3` 。

但如果有重复的话，这中间就会产生多对一的映射，比如数组 `2131` ,则映射关系为 `0->2, {1, 3}->1, 2->3` 。这样，我们推演的序列就一定会有环路了，这里下标的序列是 `0->2->3->1->1->1->1->...` ，而环的起点就是重复的数。

所以该题实际上就是找环路起点的题，和Linked List Cycle II一样。我们先用快慢两个下标都从0开始，快下标每轮映射两次，慢下标每轮映射一次，直到两个下标再次相同。这时候保持慢下标位置不变，再用一个新的下标从0开始，这两个下标都继续每轮映射一次，当这两个下标相遇时，就是环的起点，也就是重复的数。对这个找环起点算法不懂的，请参考Floyd's Algorithm。

注意

第一次找快慢指针相遇用do-while循环

代码

```
public class Solution {
    public int findDuplicate(int[] nums) {
        int slow = 0;
        int fast = 0;
        // 找到快慢指针相遇的地方
        do{
            slow = nums[slow];
            fast = nums[nums[fast]];
        } while(slow != fast);
        int find = 0;
        // 用一个新指针从头开始，直到和慢指针相遇
        while(find != slow){
            slow = nums[slow];
            find = nums[find];
        }
        return find;
    }
}
```

你可能感兴趣的文章

leetcode 算法解析（一）：260. Single Number III 3.3k 浏览

Leetcode解题报告：Remove Element 607 浏览

LeetCode 394: DecodeString (Java) 1.4k 浏览

7 条评论

默认排序 | 时间排序

tjut5547 · 2016年09月29日

nlogn时间复杂度的算法有问题，可能你在leetcode上面过了，但是如果测试序列是[1,2,2,2,5,6,7,8]或者是[1,2,3,4,6,7,7,8]你如何判断？重复元素是小于mid还是大于mid？

赞 回复

同意 可能出现两边相等的情况
— **yuryant** · 2017年02月09日

@yurya输入不合法
— **在梅边** · 2017年02月27日

@yuryant 但这个算法实现确实有问题
— **在梅边** · 2017年02月27日

添加回复 | 显示更多

hitwlh · 2017年04月26日

没有仔细看楼主的二分代码，但是这里的复杂度分析你应该错了，二分的复杂度是O(n)的
下面是我的，也ac了
class Solution {
public:

```
int findDuplicate(vector<int>& nums) {
    int n = nums.size() - 1;
    int left = 1, right = n;
    while(left < right){
        int middle = (left + right) / 2;
        int lt = 0, rt = 0, mt = 0;
        for(auto i: nums){
```

```
                if(i > middle && i <= right) rt ++;
            }
            if(mt > 1) return middle;
            if(rt > lt){
                left = middle + 1;
            }else{
                right = middle - 1;
            }

        }
        return left;
    }


};
```

赞　回复

抱歉我说错了，这个的时间复杂度是o(nlongn)，我的代码应该没问题

— **hitwlh** · 2017年04月27日

添加回复

文明社会，理性评论

发布评论

ethannnli

740 声望

关注作者

发布于专栏

Ethan Li 的技术专栏

全栈工程师的修炼旅途

系列文章

[Leetcode] Repeated Substring Pattern 重复子串格式 144 浏览

[Leetcode] Matchsticks to Square 火柴拼方形 143 浏览