

快慢指针找环 的入口

题目描述：

287. 寻找重复数

难度 中等

1379



给定一个包含 $n + 1$ 个整数的数组 `nums`，其数字都在 `1` 到 `n` 之间（包括 `1` 和 `n`），可知至少存在一个重复的整数。

假设 `nums` 只有一个重复的整数，找出这个重复的数。

你设计的解决方案必须不修改数组 `nums` 且只用常量级 $O(1)$ 的额外空间。

题目分析：

这里简单解释为什么后面将 `slow` 放置起点后移动相遇的点就一定是答案了。假设环长为 L ，从起点到环的入口的步数是 a ，从环的入口继续走 b 步到达相遇位置，从相遇位置继续走 c 步回到环的入口，则有 $b + c = L$ ，其中 L 、 a 、 b 、 c 都是正整数。根据上述定义，慢指针走了 $a + b$ 步，快指针走了 $2(a + b)$ 步。从另一个角度考虑，在相遇位置，快指针比慢指针多走了若干圈，因此快指针走的步数还可以表示成 $a + b + kL$ ，其中 k 表示快指针在环上走的圈数。联立等式，可以得到

$$2(a + b) = a + b + kL$$

解得 $a = kL - b$ ，整理可得

$$a = (k - 1)L + (L - b) = (k - 1)L + c$$

从上述等式可知，如果慢指针从起点出发，快指针从相遇位置出发，每次两个指针都移动一步，则慢指针走了 a 步之后到达环的入口，快指针在环里走了 $k - 1$ 圈之后又走了 c 步，由于从相遇位置继续走 c 步即可回到环的入口，因此快指针也到达环的入口。两个指针在环的入口相遇，相遇点就是答案。

常规代码：

```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int slow = 0, fast = 0;
        do {
            slow = nums[slow];
            fast = nums[nums[fast]];
        } while (slow != fast);
        slow = 0;
        while (slow != fast) {
            slow = nums[slow];
            fast = nums[fast];
        }
        return slow;
    }
};
```

作者：LeetCode-Solution

链接：<https://leetcode-cn.com/problems/find-the-duplicate-number/solution/xun-zhao-zhong-fu-shu-by-leetcode-solution/>

来源：力扣（LeetCode）

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

位运算解法：

- 如果测试用例的数组中 *target* 出现了两次，其余的数各出现了一次，且 *target* 的第 *i* 位为 1，那么 *nums* 数组中第 *i* 位 1 的个数 *x* 恰好比 *y* 大一。如果 *target* 的第 *i* 位为 0，那么两者相等。
- 如果测试用例的数组中 *target* 出现了三次及以上，那么必然有一些数不在 *nums* 数组中了，这个时候相当于我们用 *target* 去替换了这些数，我们考虑替换的时候对 *x* 的影响：
 - 如果被替换的数第 *i* 位为 1，且 *target* 第 *i* 位为 1: *x* 不变，满足 $x > y$ 。
 - 如果被替换的数第 *i* 位为 0，且 *target* 第 *i* 位为 1: *x* 加一，满足 $x > y$ 。
 - 如果被替换的数第 *i* 位为 1，且 *target* 第 *i* 位为 0: *x* 减一，满足 $x \leq y$ 。
 - 如果被替换的数第 *i* 位为 0，且 *target* 第 *i* 位为 0: *x* 不变，满足 $x \leq y$ 。



```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int n = nums.size(), ans = 0;
        // 确定二进制下最高位是多少
        int bit_max = 31;
        while (!(n - 1) >> bit_max) {
            bit_max -= 1;
        }
        for (int bit = 0; bit <= bit_max; ++bit) {
            int x = 0, y = 0;
            for (int i = 0; i < n; ++i) {
                if (nums[i] & (1 << bit)) {
                    x += 1;
                }
                if (i >= 1 && (i & (1 << bit))) {
                    y += 1;
                }
            }
            if (x > y) {
                ans |= 1 << bit;
            }
        }
        return ans;
    }
}
```

```
};
```

作者: LeetCode-Solution

链接: <https://leetcode-cn.com/problems/find-the-duplicate-number/solution/xun-zhao-zhong-fu-shu-by-leetcode-solution/>

来源: 力扣 (LeetCode)

著作权归作者所有。商业转载请联系作者获得授权, 非商业转载请注明出处。