287 寻找重复数

Label：双指针、二分法

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 。
nums 中 只有一个整数 出现 两次或多次 ，其余整数均只出现 一次

- set

```java
class Solution {
    public int findDuplicate(int[] nums) {
       Set<Integer> set = new HashSet<>();
       for (int i : nums) {
           if (set.contains(i)) {
               return i;
           }
           set.add(i);
       }
       return -1;
    }
}
```

- 快慢指针

```java
class Solution {
    public int findDuplicate(int[] nums) {
        int slow = 0, fast = 0;
        do {
            slow = nums[slow];
            fast = nums[nums[fast]];
        } while (slow != fast);

        slow = 0;
        while (slow != fast) {   // 没太看懂这步
            slow = nums[slow];
            fast = nums[fast];
        }
        return slow;
    }
}
```

- 标志位

```
class Solution {
    public int findDuplicate(int[] nums) {
        int len = nums.length;
        for (int i = 0; i < len; i++) {
            nums[nums[i] % len] += len;  // 设置加len的标志
        }
        for (int i = 0; i < len; i++) {
            if(nums[i] > 2*len) {
                return i;
            }
        }
        return -1;
    }
}
```

- 二分法（抽屉原则）

```
public class Solution {

    public int findDuplicate(int[] nums) {
        int len = nums.length;
        int left = 1;
        int right = len - 1;

        while (left < right) {
            // 在 Java 里可以这么用，当 left + right 溢出的时候，无符号右移保证结果依然正
确
            int mid = (left + right + 1) >>> 1;

            int cnt = 0;  // 计算num中小于mid的数的个数
            for (int num : nums) {
                if (num < mid) {
                    cnt += 1;
                }
            }

            // 根据抽屉原理，严格小于 4 的数的个数如果大于等于 4 个，
            // 此时重复元素一定出现在 [1, 3] 区间里
            if (cnt >= mid) {
                // 重复的元素一定出现在 [left, mid - 1] 区间里
                right = mid - 1;
            } else {
                // if 分析正确了以后，else 搜索的区间就是 if 的反面
                // [mid, right]
                // 注意：此时需要调整中位数的取法为上取整
                left = mid;
            }
        }
        return left;
    }
}
```