LeetCode #540 有序数组中的单一元素

作者：matthewhan
原文链接：https://ld246.com/article/1593745065936
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

#540 Single Element in a Sorted Array

Problem Description

给定一个只包含整数的有序数组，每个元素都会出现两次，唯有一个数只会出现一次，找出这个数。

note

您的方案应该在 O(log n)时间复杂度和 O(1)空间复杂度中运行。

e.g.

- **示例 1:**

输入: [1,1,2,3,3,4,4,8,8]
输出: 2

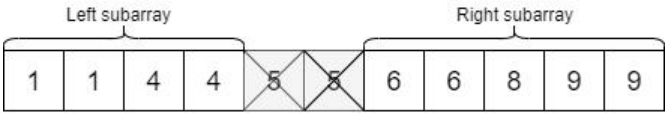
- **示例 2:**

输入: [3,3,7,7,10,11,11]
输出: 10

Solution

题目很简单，但是题干中重点强调了O(logn)的时间复杂度和O(1)空间复杂度，所以一切会伴随输入组大小变化的额外空间和遍历也不行哦，即使是用双指针优化到O(logn/4)也是不满足的。一眼看到O(ogn)的时间复杂度就想到了题目的本意应该是让我们利用二分法来解。

因为是有序数组，并且每个元素都是出现两次，除了唯一的一个元素。所以相同的元素一定是连续的且长度为奇数，所以我们可以每次取中间的元素，和左右两边比较（注意边界），找出相邻元素相同情况，将整个数组分隔开，左边的子串和右边子串一定有一个是奇数长度（该数组一定有个唯一元素，单个元素就一定藏在这个子串里。




```
public static int singleNonDuplicate(int[] nums) {
    int left = 0;
    int right = nums.length - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (mid - 1 >= 0 && nums[mid] == nums[mid - 1]) {
            // 奇数在左边
            if (((mid - 1) & 1) == 1) {
                right = mid - 2;
            } else {
                left = mid + 1;
            }

        } else if (mid + 1 < nums.length && nums[mid] == nums[mid + 1]) {
            // 奇数在左边
            if ((mid & 1) == 1) {
                right = mid - 1;
            } else {
                left = mid + 2;
            }
        } else {
            return nums[mid];
        }
        System.out.println("left = " + left);
        System.out.println("right = " + right);

    }
    return 0;
}
```