

```

1  /*
2  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。
3
4  示例 1:
5
6  输入: [1,1,2,3,3,4,4,8,8]
7  输出: 2
8
9  示例 2:
10
11 输入: [3,3,7,7,10,11,11]
12 输出: 10
13
14 注意: 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。
15
16 来源: 力扣 (LeetCode)
17 链接: https://leetcode-cn.com/problems/single-element-in-a-sorted-array
18 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
19 */

```

分析:

- 方法一:位运算异或法,时间复杂度为 $O(n)$;
- 方法二:逐二遍历查找法,时间复杂度为 $O(n)$
- 方法三:二分查找法,时间复杂度为 $O(\log n)$
 - 按照题意输入数组的长度一定是奇数,则一定有中间元素,另当前元素的索引为 mid , $mid > 0$.
 - 若 mid 为数组头或数组尾巴,则直接返回 $nums[mid]$;
 - 若 mid 是奇数
 - 若 $nums[mid] == nums[mid - 1]$,表明断点在右半部分, $left = mid + 1$;
 - 若 $nums[mid] \neq nums[mid - 1]$
 - 若 $nums[mid] \neq num[mid + 1]$,则已经找到了数量为1的元素;
 - 若 $nums[mid] == num[mid + 1]$,表明断点在左半部分, $right = mid - 1$;
 - 若 k 是偶数
 - 若 $nums[mid] == nums[mid + 1]$,则表明断点在右半部分, $left = mid + 1$
 - 若 $nums[mid] \neq num[mid + 1]$
 - 若 $nums[mid] \neq nums[mid - 1]$,则已经找到了数量为1的元素;
 - 若 $nums[mid] == nums[mid - 1]$,则表明果断点在左半部分, $right = mid - 1$.

方法一:C++_位运算异或法

```

1  class Solution
2  {
3      public:
4          int singleNonDuplicate(vector<int>& nums)
5          {
6              /*
7              时间复杂度  $O(n)$ 

```

```

8          空间复杂度  $O(1)$ 
9
10         */
11         int ret_val = 0x00;
12         int i      = 0;
13         for(i = 0 ; i < nums.size() ; i++)
14         {
15             ret_val ^= nums[i];
16         }
17         return ret_val;
18     }
19 };
20
21 /*
22 执行结果:
23 通过
24 显示详情
25 执行用时 :12 ms, 在所有 cpp 提交中击败了76.04% 的用户
26 内存消耗 :9.5 MB, 在所有 cpp 提交中击败了9.73%的用户
27 */

```

方法二:逐二遍历查找法_C++

```

1  class Solution
2  {
3      public:
4          int singleNonDuplicate(vector<int>& nums)
5          {
6              /*
7              时间复杂度  $O(n)$ 
8              空间复杂度  $O(1)$ 
9              */
10             int ret_val = 0x00          ;
11             int i      = 0              ;
12             int left   = 0              ;
13
14             for(i=0;i<nums.size();i+=2)
15             {
16                 if(i+1<nums.size())
17                 {
18                     if(nums[i]!=nums[i+1])
19                     {
20                         ret_val = nums[i];
21                         break;
22                     }
23                 }
24                 else
25                 {
26                     ret_val = nums[i];
27                     break;
28                 }
29             }
30             return ret_val;
31         }
32     };
33

```

```
34  /*
35  执行结果:
36  通过
37  显示详情
38  执行用时 :12 ms, 在所有 cpp 提交中击败了76.04% 的用户
39  内存消耗 :9.5 MB, 在所有 cpp 提交中击败了6.19%的用户
40  */
```

方法三:二分查找法_C++

```

1 class Solution
2 {
3     public:
4         int singleNonDuplicate(vector<int>& nums)
5         {
6             int left    = 0                ;
7             int right   = nums.size()-1    ;
8             int mid     = 0                ;
9
10
11         while(left <= right)
12         {
13             mid = left + (right - left ) / 2;
14
15             /*最右边*/
16             if( (mid == nums.size()-1)
17                || (mid == 0)
18            )
19            {
20                break;
21            }
22            else
23            {
24                /*mid 是奇数*/
25                if(mid & 0x01)
26                {
27                    if(nums[mid] == nums[mid-1])
28                    {
29                        left = mid + 1;
30                    }
31                    // nums[mid] != nums[mid-1]
32                    else
33                    {
34                        if(nums[mid] != nums[mid+1])
35                        {
36                            //找到了
37                            break;
38                        }
39                        else
40                        {
41                            right = mid - 1;
42                        }
43                    }
44                }
45                /*mid是偶数*/
46                else

```

```

47         {
48             if(nums[mid] == nums[mid + 1])
49             {
50                 left = mid + 1;
51             }
52             else
53             {
54                 if(nums[mid] == nums[mid-1])
55                 {
56                     right = mid - 1;
57                 }
58                 else
59                 {
60                     // 找到了
61                     break;
62                 }
63             }
64         }
65     }
66 }
67 }
68
69 return nums[mid];
70
71
72 }
73 };
74
75 /*
76 执行结果:
77 通过
78 显示详情
79 执行用时 :8 ms, 在所有 cpp 提交中击败了97.08% 的用户
80 内存消耗 :9.6 MB, 在所有 cpp 提交中击败了5.31%的用户
81 */

```