136 Single Number

2018年4月2日 21:39

Question:

Given an array of integers, every element appears *twice* except for one. Find that single one.

Note:

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

来自 <https://leetcode.com/problems/single-number/description/>

给定一个整数数组，除了某个元素外其余元素均出现两次。请找出这个只出现一次的元素。

备注：

你的算法应该是一个线性时间复杂度。 你可以不用额外空间来实现它吗？

Solution for Python3:

```python
# bit位操作
# If we take XOR of zero and some bit, it will return that bit
#   a⊕0=a
# If we take XOR of two same bits, it will return 0
#   a⊕a=0
# a⊕b⊕a=(a⊕a)⊕b=0⊕b=b
class Solution1:
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        for i in range(1,len(nums)):
           nums[0] ^= nums[i]
        return nums[0]

# 哈希方法
class Solution2:
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        hash_table = {}
        for i in nums:
           try:
               hash_table.pop(i)
           except:
               hash_table[i] = 1
        return hash_table.popitem()[0]

# 2*(a+b+c)-(a+a+b+b+c)=c
class Solution3:
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        return 2 * sum(set(nums)) - sum(nums)
```

Solution for C++:

```cpp
class Solution1 {
public:
    int singleNumber(vector<int>& nums) {
        for (int i = 1; i < nums.size(); i++) {
           nums[i] ^= nums[i - 1];
        }
```

```
7           return nums[nums.size() - 1];
8       }
9   };
10
11  class Solution2 {
12  public:
13      int singleNumber(vector<int>& nums) {
14          set<int> s(nums.begin(),nums.end());
15          return 2 * accumulate(s.begin(), s.end(), 0) - accumulate(nums.begin(), nums.end(), 0);
16      }
17  };
```

Appendix:

C++: accumulate(nums.begin(), nums.end(), int):

1) 累加函数，累加nums数组中每一个元素到第三个参数上。
2) 第三个参数为0则按数字累加到0上。
3) 若为'0'，则取字符'0'的ascii码值进行累加。