算法讨论班第二十期------李玥珮

2016 年 1 月 15 日星期五

136. Single Number

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?

解题思路：

思路一、

统计每个数字出现的次数，然后从中找出只出现一次的数字。Python 中可用字典实现，算法的时间复杂度 O(n)，空间复杂度 O(n)。

```
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        dictionary = {}
        for element in nums:
            if element in dictionary:
                dictionary[element] += 1
            else:
                dictionary[element] = 1

        for key in dictionary.keys():
            if dictionary[key] == 1:
                return key
```

思路二、

首先对数组排序，然后两个一组的遍历数组。同一组的两个数字如果不相等，则该组的第一个数组就是结果。如果没有，则结果是最后一个数字。

```python
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        nums.sort()
        for i in range(1, len(nums), 2):
            if nums[i] != nums[i-1]:
                return nums[i-1]
        return nums[-1]
```

Python 中 sort 的算法实现是 Timsort 算法。时间复杂度平均 O(nlogn)，空间复杂度最坏 O(n/2)。所以整个算法的时间复杂度是 O(nlogn)，空间复杂度 O(n/2)。

思路三、利用 XOR 运算。

因为 A XOR A = 0，XOR 运算满足交换律。所以对于实例{2 1 4 5 2 4 1}

```
(2^1^4^5^2^4^1) => ((2^2)^(1^1)^(4^4)^(5)) => (0^0^0^5) => 5
```

算法时间复杂度 O(n)，不需要额外的空间。

```python
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        result = 0
        for i in nums:
            result ^= i
        return result
```

137. Single Number II

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

Note:

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

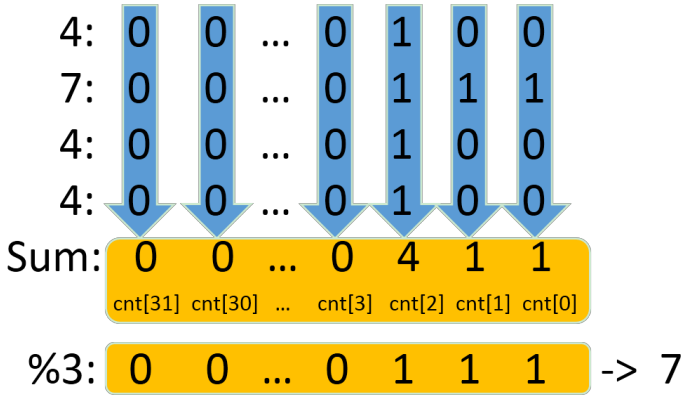
思路一、利用字典统计每个数字出现的次数。

思路二、先排序，然后每三个数字一组遍历。

以上两种思路与上题一样，只要对代码稍作修改即可。

思路三、利用位运算。

1、分别统计每位上 1 出现的次数，如果是 3 的倍数则该位设为 0，否则设为 1。



算法代码：c++。时间复杂度 O(32n)，空间复杂度 O(1)。

```cpp
[cpp] view plain copy print ?
int singleNumberII_36(int A[], int n)
{
    int ans = 0;
    for (int i = 0; i < 32; i++)
    {
        int c = 0, d = 1<<i;
        for (int j = 0; j < n; j++)
            if (A[j] & d) c++;

        if (c%3) ans |= d;
    }
    return ans;
}
```

2、位运算。首先只看一位的情况，代码实际上统计的是 1 出现的次数。32 位运算结果与 1 位是类似的。位运算过程如下：

```
twos |= ones & item   # ho
ones ^= item   # hold item
threes = ones & twos   # h

ones ^= threes   # once so
twos ^= threes   # once so
```

状态	THREE	TWO	ONE	出现 1 的次数
初态 0	0	0	0	0
状态 1	0	0	1	1
状态 2	0	1	0	2
状态 3	1	0	0	3
状态 1	0	0	1	4
……				

代码如下：

```python
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        ones, twos, threes = 0, 0, 0
        for item in nums:
            twos |= ones & item  # hold item which occur twice
            ones ^= item  # hold item which occur once
            threes = ones & twos  # hold item which occur three times

            ones ^= threes  # once some item occur three items, reset ones
            twos ^= threes  # once some item occur three items, reset twos
        return ones
```

思路四：用 set 处理数组，求和，与原数组的和做差比较得出结果。

```python
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        a = set(nums)
        a = sum(a) * 3 - sum(nums)
        a = a / 2
        return a
```

260. Single Number III

Given an array of numbers `nums`, in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once.

For example:

Given `nums = [1, 2, 1, 3, 2, 5]`, return `[3, 5]`.

Note:

1. The order of the result is not important. So in the above example, `[5, 3]` is also correct.
2. Your algorithm should run in linear runtime complexity. Could you implement it using only constant space complexity?

解题思路：

思路一、利用字典统计每个数字的个数。

思路二、先排序，再求解。

思路三、位运算。

因为要选出两个数字，这里考虑将原序列分成两个序列来做。分配的过程要满足两点：

1、 相同的两个数字必须分在一个序列中。

2、 只出现一次的两个数字（设为 a、b）要分属两个不同的序列。

为了满足第一条，可以用数据中的某一位来把数据分类。这样就能保证相同的数字被分入同一组。为了满足第二条，我们要选择 a、b 中的不同的一位来分类。算法代码如下：

```
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        xor = 0
        a = 0
        b = 0
        for num in nums:
            xor ^= num
        mask = 1
        while(xor & mask == 0):
            mask = mask << 1
        for num in nums:
            if num & mask:
                a ^= num
            else:
                b ^= num
        return [a, b]
```