[leetcode刷题笔记]数学与位运算

[leetcode刷题笔记]数学与位运算

> 位运算是二进制中比较常见的运算，包括按位与&，按位或|，非～，异或^ 等，本文记录LeetCode刷题一些知识点，水平有限还望多多指正

1.只出现一次的数字

> 给定一个非空整数数组，除了某个元素只出现一次以外，其余每个元素均出现两次。找出那个只出现了一次的元素。

根据异或运算的三个特点：

- 1.交换律：$a \wedge b \wedge c = a \wedge c \wedge b$
- 2.任何数于0异或为任何数: $0 \wedge n = n$
- 3.相同的数异或为0: $n \wedge n = 0$

若输入$[4,1,2,1,2]$，则$4 \wedge 1 \wedge 2 \wedge 1 \wedge 2 = 4 \wedge (1 \wedge 1) \wedge (2 \wedge 2) = 4 \wedge 0 \wedge 0 = 4$即为所求

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        res = 0
        for i in nums:
            res^=i
        return res
```

2.只出现一次的数字II

> 给定一个非空整数数组，除了某个元素只出现一次以外，其余每个元素均出现了三次。找出那个只出现了一次的元素。

在上一题中，除了某元素外其他元素均出现两次，消除规则为0 -> 1 -> 0，因此使用一个二进制位即可。在本题中其他数字出现三次，需要两个二进制位完成 00 -> 01 -> 10 -> 00。如何使某位在0与1之间转换，可以根据以下公式。

$$x\& \sim x = 0$$
$$x\& \sim 0 = x$$

设b,a代表两个二进制位，当出现3次是，其他元素归零，最后仅剩b,a = 0,1，即a此时位元素值。

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        a, b = 0, 0
        for num in nums:
            a = a ^ num & ~b
            b = b ^ num & ~a
        return a
```

3.只出现一次的数字III

> 给定一个整数数组 nums，其中恰好有两个元素只出现一次，其余所有元素均出现两次。 找出只出现一次的那两个元素。

首先通过异或运算能够求出这两个数字的异或$diff$，可以用 $diff$作为标记来分离 x 和y。

如何求数字x,考虑异或结果的某个非 0 位如最后一个非 0 位, 因为我们知道只有当 x、y在该位不一样的时候才会出现异或结果为 1,我们通过 $diff\&(-diff)$保留$diff$最右边的 1，这个 1 要么来自x，要么来自y,所以我们以该位是否为 1 对数组进行划分, 只要该位为 1 就和 x 异或, 只要该位为 0就和 y异或, 这样最终得到就是只出现过一次的两个数。

下面清楚的说明$diff\&(-diff)$的作用，因为 $-diff = \sim diff + 1$，故:

$diff\&(-diff) = diff\&(\sim diff + 1)$

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        diff, x,y = 0,0,0
        for i in nums:
            diff ^=i
        diff&=(~ diff+1)
        for i in nums:
            if diff&i==0:
                x ^=i
            else:
                y ^=i
        return x,y
```

4.缺失数字

> 给定一个包含 0, 1, 2, ..., n 中 n 个数的序列，找出 0 .. n 中没有出现在序列中的那个数。

若输入$[0,1,3,4]$,其中i为下标，num为数值，将其异或运算，该题就变成了数组中只有一个数字出现一次，如表

i	0	1	2	3

i	0	1	2	3
num	0	1	3	4

可以将结果的初始值设为n，再对数组中的每一个数以及它的下标进行一个异或运算，即：

$$\begin{aligned} res &= 4 \wedge (0 \wedge 0) \wedge (1 \wedge 1) \wedge (2 \wedge 3) \wedge (3 \wedge 4) \\ &= (4 \wedge 4) \wedge (0 \wedge 0) \wedge (1 \wedge 1) \wedge (3 \wedge 3) \wedge 2 \\ &= 0 \wedge 0 \wedge 0 \wedge 0 \wedge 2 \\ &= 2 \end{aligned}$$

即为所求

```
class Solution:
    def missingNumber(self, nums: List[int]) -> int:
        res = len(nums)
        for i,num in enumerate(nums):
            res ^= i^num
        return res
```

5.消失的两个数字

> 给定一个数组，包含从 1 到 N 所有的整数，但其中缺了两个数字。你能在 O(N) 时间内只用 O(1) 的空间找到它们吗?

将问题转换为nums数组的所有数和1~N的数一同构成的集合特征为——该集合内只有2个数字出现过1次，而其余数字都出现2次，请找出这2个只出现1次的数字。

```
class Solution:
    def missingTwo(self, nums: List[int]) -> List[int]:
        diff, x,y = 0,0,0
        N = len(nums) + 2
        for i in range(1, N+1):
            diff ^= i
        for i in nums:
            diff ^= i

        diff&=~ diff

        for i in nums:
            if i & diff:
                x ^= i
            else:
                y ^= i

        for i in range(1, N+1):
            if i & diff:
                x ^= i
            else:
                y ^= i

        return [x, y]
```

6.不用加减乘除做加法

> 写一个函数，求两个整数之和，要求在函数体内不得使用 “+”、“-”、“*”、“/” 四则运算符号。

∧异或 ----相当于 无进位的求和， 想象10进制下的模拟情况： （如:19+1=20；无进位求和就是10，而非20；因为它不管进位情况）

& 与 ----相当于求每位的进位数， 先看定义：$1\&1=1$；$1\&0=0$；$0\&0=0$；即都为1的时候才为1，正好可以模拟进位数的情况,还是想象10进制下模拟情况： （9+1=10，如果是用&的思路来处理，则9+1得到的进位数为1，而不是10，所以要用<<1向左再移动一位，这样就变为10了）；

这样公式就是：$(a \wedge b) \wedge ((a\&b) << 1)$ 即：每次无进位求 + 每次得到的进位数--------我们需要不断重复这个过程，直到进位数为0为止；

```
class Solution:
    def add(self, a: int, b: int) -> int:
        while b:
            a, b = (a ^ b) & 0xFFFFFFFF, ((a & b) << 1) & 0xFFFFFFFF
        return a if a <= 0x7FFFFFFF else ~ a ^ 0xFFFFFFFF
```

[注]为什么要加 `0xFFFFFFFF`,在leetcode解析上,作者jyd给出了如下解释。
由于 Python 的数字存储特点，需要做一些特殊处理，以下详细介绍。
Python 负数的存储：
Python / Java 中的数字都是以 补码 形式存储的。但 Python 没有 int , long 等不同长度变量，即没有变量位数的概念。
获取负数的补码： 需要将数字与十六进制数 `0xffffffff` 相与。可理解为舍去此数字 32 位以上的数字，从无限长度变为一个 32位整数。
返回前数字还原： 若补码 a 为负数（`0x7fffffff` 是最大的正数的补码 ），需执行`~(a ^ x)`操作，将补码还原至 Python 的存储格式。`a ^ x`运算将 1至 32 位按位取反； ~ 运算是将整个数字取反；因此，`~(a ^ x)` 是将 32 位以上的位取反，即由 0 变为 1， 1 至 32 位不变。

```
print(hex(1)) # = 0x1 补码
print(hex(-1)) # = -0x1 负号 + 原码 （ Python 特色，Java 会直接输出补码）

print(hex(1 & 0xffffffff)) # = 0x1 正数补码
print(hex(-1 & 0xffffffff)) # = 0xffffffff 负数补码

print(-1 & 0xffffffff) # = 4294967295 （ Python 将其认为正数）
```

7.数组中数字出现的次数

> 一个整型数组 nums 里除两个数字之外，其他数字都出现了两次。请写程序找出这两个只出现一次的数字。要求时间复杂度是O(n)，空间复杂度是O(1)。

```
class Solution:
    def singleNumbers(self, nums: List[int]) -> List[int]:
        ret=0
        for num in nums:
            ret ^= num
        div = 1
```

```
        while div & ret == 0:
            div <<= 1
        a, b = 0, 0
        for num in nums:
            if num & div:
                a ^= num
            else:
                b ^= num
        return [a, b]
```

8.3的幂

给定一个整数，写一个函数来判断它是否是 3 的幂次方。你能不使用循环或者递归来完成本题吗?

使用log,若n为3的幂次方，则$\log_3(n)$为整数，根据换底公式

$$n = 3^i$$
$$i = \log_3(n) = \frac{\log_b(n)}{\log_b(3)}$$

判断是否为整数只需判断i%1是否为0即可

```
class Solution:
    def isPowerOfThree(self, n: int) -> bool:
        from math import log10

        if n <= 0:
            return False

        return (log10(n)/log10(3))%1 == 0
```

再次注意，由于精度问题，在此使用log10()而非log()。

9.2的幂

给定一个整数，编写一个函数来判断它是否是 2 的幂次方。

可以使用上题求对数，也可根据二进制的性质，若n为2的幂，则$n\&(n-1) = 0$

```
class Solution:
    def isPowerOfTwo(self, n: int) -> bool:

        if n < 1:
            return False

        return n&(n-1) == 0
```

题目和部分解析来源力扣（LeetCode）,更多内容后续补充，欢迎指正。

写下你的评论...

评论0
赞2
...
抽奖

2赞3赞
赏
赞赏
更多好文