1 原码、反码和补码

二进制有三种不同的表示形式：原码、反码和补码，计算机内部使用**补码**来表示。

原码：就是其二进制表示（注意：有一位符号位）。

```
00 00 00 11 -> 3
10 00 00 11 -> -3
```

反码：正数的反码就是原码，负数的反码是符号位不变，其余位取反。

```
00 00 00 11 -> 3
11 11 11 00 -> -3
```

补码：正数的补码就是原码，负数的补码是反码+1。

```
00 00 00 11 -> 3
11 11 11 01 -> -3
```

符号位：最高位为符号位，0表示正数，1表示负数。在位运算中符号位也参与运算。

2 位运算

2.1 按位非操作~

~把num的补码中的0和1全部取反，符号位同样取反。

```
~1 = 0
~0 = 1
```

2.2 按位与操作&

只有两个对应位都为1时才为1。

```
1 & 1 = 1
1 & 0 = 0
0 & 1 = 0
0 & 0 = 0
```

2.3 按位或操作 |

只要两个对应位中有一个为1时就为1。

```
1 | 1 = 1
1 | 0 = 1
0 | 1 = 1
0 | 0 = 0
```

2.4 按位异或操作^

只有两个对应位不同时才为1。

```
1 ^ 1 = 0
1 ^ 0 = 1
0 ^ 1 = 1
0 ^ 0 = 0
```

异或操作的性质：满足交换律和结合律

2.5 按位左移操作<<

`num << i` 将 `num` 的二进制表示向左移动 `i` 位所得的值。

```
00 00 10 11 -> 11
11 << 3
--
01 01 10 00 -> 88
```

2.6 按位右移操作>>

`num >> i` 将 `num` 的二进制表示向右移动 `i` 位所得的值。

```
00 00 10 11 -> 11
11 >> 2
--
00 00 00 10 -> 2
```

2.7 利用位运算实现快速计算

通过<< ，>> 快速计算2的倍数问题。

```
n << 1 -> 计算 n*2
n >> 1 -> 计算 n/2，负奇数的运算不可□
n << m -> 计算 n*(2^m)，即乘以 2 的 m 次□
n >> m -> 计算 n/(2^m)，即除以 2 的 m 次□
1 << n -> 2^n
```

通过^ 快速交换两个整数。

```
a ^= b
b ^= a
a ^= b
```

通过a&(-a) 快速获取a的最后为1位置的整数。

```
00 00 01 01 -> 5
&
11 11 10 11 -> -5
--
00 00 00 01 -> 1

00 00 11 10 -> 14
&
11 11 00 10 -> -14
--
00 00 00 10 -> 2
```

2.8利□位运算实现整数集合

一个数的二进制表示可以看作是一个集合（0表示不在集合中，1表示在集合中）。

比如集合{1, 3, 4, 8}，可以表示成01 00 01 10 10， 而对应的运算也就可以看成是对集合进行的操作。

```
# 元素与集合的操作
a | (1<<i) -> 把 i 插□到集合中
a & ~(1<<i) -> 把 i 从集合中删除
a & (1<<i) -> 判断 i 是否属于该集合（零不属于，□零属于）

# 集合之间的操作
a 补 -> ~a
a 交 b -> a & b
a 并 b -> a | b
a 差 b -> a & (~b)
```

3 python的bin()函数

正数在内存中是以补码的形式存在的，输出自然也是按照补码输出，

【例1】C#语言输出负数

```
class Program
{
  static void Main(string[] args)
  {
    string s1 = Convert.ToString(-3, 2);
    Console.WriteLine(s1);
    // 11111111111111111111111111111101

    string s2 = Convert.ToString(-3, 16);
    Console.WriteLine(s2);
    // fffffffd
  }
}
```

【例2】Python的bin()输出

```
print(bin(3)) # 0b11
print(bin(-3)) # -0b11
```

```
print(bin(-3 & 0xffffffff))
# 0b11111111111111111111111111111101

print(bin(0xfffffffd))
# 0b11111111111111111111111111111101

print(0xfffffffd) # 4294967293
```

【注意】

1. Python中`bin`一个负数（十进制表示），**输出的是它的原码的二进制表示加上一个负号**。
2. Python中的整型是补码形式存储的。
3. Python中整型是不限长度的，不会超出范围溢出。
4. 所以为了获得**负数（十进制表示）的补码**，需要手动将其和十六进制数`0xffffffff`进行按位与操作，再交给bin()进行输出。

4 练习题

leetcode 136.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。（尝试用位运算解决此题）

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