位运算

数组中两个数的最大异或值

> 求非空数组中两个数异或的最大值

关键

- 异或运算的性质 如果三个数中的其中两个异或可以得到第三个，那么他们的顺序可以任意调换
- 贪心
    1. 二进制越高位是 '1'，则这个数越大
    2. 专注最大异或值 只用关心最大异或值需要满足什么性质，进而推出这个最大值是什么，而不必关心这个异或值是由哪两个数得来的
- 使用前缀找出异或值

思路

1. 假设最大异或值是 S，那么必有两个数异或等于 S，根据异或运算的性质，S 与其中一个数异或的值必然在数组中。

2. 根据这个特点，从最高位，找到第一个不是 0 的前缀，贪心的认为是 S 的前缀，然后与数组中所有元素对应前缀异或，若找到存在于数组中的元素，则认为假设正确，继续往下

3. 若不存在，赋为 0，一直到最右

复杂度分析

- 时间复杂度：$O(N)$，把整个数组搜索了 32 次，即 $O(32N) = O(N)$
- 空间复杂度：$O(N)$，这里的 N 是哈希表的长度，具体长度是多少，与输入的规模、扩容策略、负载因子和冲突策略等有关。

```
def findMaximumXOR(nums):
    res, mask = 0, 0
    for i in range(31, -1, -1):
        # 生成掩码用于得到前缀
        # 掩码与原数依次与操作，得到前缀
        # 放入当前哈希表中
        mask |= (1 << i)
        s = set()
        for num in nums:
            s.add(mask & num)
        # 先“贪心地”假设这个数位上是 “1” ，如果全部前缀都看完，都不符合条件，这个数位上就是 “0”
        temp = res | (1 << i)
        for prefix in s:
            if temp ^ prefix in s:
                res = temp
                break
    return res
```

只出现一次的数系列

> 除了某个元素只出现一次以外，其余每个元素均出现两次

直接运用异或的性质

> 除了某个元素只出现一次以外，其余每个元素均出现三次

步骤

- 将数字表示为二进制，统计每一位上 1 的个数
- 针对每一位,累加值只有两种情况:
  - 是 3 的倍数
  - 不是 3 的倍数，但对 3 取模一定等于 1
- 对累加值模 3，能整除置 0，不能整除置 1
- 将二进制转换回十进制
- python 对负数表示有问题，因为 python 将 int 当作对象，需要特殊处理

解释

- 由于除了一个数之外都出现 3 次，所以当某个位置为时 1，它的出现次数一定是 3 的倍数，而 0 不会产生贡献，
- 所以 1 的出现次数累加起来必然是 3N 或者 3N+1，3N 表示对应位是 0，3N + 1 表示对应位为 1

时间复杂度: $32 * O(N)$

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        res = 0
        for i in range(0, 32):
            mask = 1 << i
            count = 0
            # 统计与的数目
            for j in range(len(nums)):
                if nums[j] & mask:
                    count += 1
            if count % 3:
                res = res | mask
        return self.convert(res)
    ## python 类型也是对象，所以负数没法获得
    def convert(self,x):
        if x >= 2**31:
            x -= 2**32
        return x
```

法2: 三进制加法，类似于电路图

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
```

```
        a = 0
        b =0
        for num in nums:
            a = (a ^num) & ~b
            b = (b ^ num) & ~a
        return a
```

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

思路

- 首先全部异或，得到的是A、B 异或的结果，对应二进制中的 1 是 A 和 B 出现位置
- 取第一个 1 所在的位数，假设是第 3 位，接着把原数组分成两组，分组标准是第 3 位是否为 1。如此，相同的数肯定在一个组，因为相同数字所有位都相同，而不同的数，肯定不在一组
- 然后把这两组按照最开始的思路，依次异或，剩余的两个结果就是这两个只出现一次的数字
- a & (-a) 可以获得a最低的非0位 比如 a 的二进制是 10000，-a 等价于取反后加1，取反就是01111，加1就是10000，最后整体相与的结果就是 10000