Bit Manipulation

```
0b101         # 5
eval('0b101') # 5
bin(5)        # 输出 '0b101'
>>, <<, &, |, ^, ~  # 右移, 左移, 位与, 位或, 位异或, 非
```

Single Number I (2+1) `xor`
Single Number III (2+1+1) `xor, any set bit, two groups`
Single Number II (3+1) `base-3 counter`
Missing Number `(index, value) 转化为 single number`

Number of 1 Bits `bin(n).count('1')`
Hamming Distance `bin(x^y).count('1')`
Total Hamming Distance

Number Complement `位翻转`

```
def singleNumber_2+1(nums):
    a = 0
    for b in nums: a ^= b
    return a

def singleNumber_2+1+1(nums):
    xor = 0
    for num in nums: xor ^= num
    mask = 1
    while mask & xor == 0: mask <<= 1  # find any set bit
    a = b = 0
    for num in nums:
        if num & mask: a ^= num
        else: b ^= num
    return [a, b]

def singleNumber_3+1(nums):
    a = b = 0
    for c in nums:
        a, b = (~a&b&c)|(a&~b&~c), (~a&~b&c)|(~a&b&~c)
    return a|b

def missingNumber(nums):
    missing = len(nums)
    for i, num in enumerate(nums): missing ^= i ^ num
    return missing

def countOneBits(n)
    count = 0
    while (n != 0):
        count++
        n &= (n - 1) # delete lsb
    return count

def findComplement(n):
    mask = 1
    while n >= mask:
        n ^= mask
        mask <<= 1
    return n
```

没做

Majority Element

Sum of Two Integers
Maximum XOR of Two Numbers in an Array

Counting Bits
Reverse Bits

Repeated DNA Sequences

Power of Two
Power of Four

Find the Difference
Bitwise AND of Numbers Range
Maximum Product of Word Lengths
Binary Watch

Letter Case Permutation
UTF-8 Validation
Generalized Abbreviation
Binary Number with Alternating Bits

IP to CIDR
Convert a Number to Hexadecimal
Prime Number of Set Bits in Binary Representation
Integer Replacement
Pyramid Transition Matrix
Minimum Unique Word Abbreviation
Bitwise ORs of Subarrays