1. 136只出现一次的数字

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

t=0

for i in nums:

t=t^i

return t

1. 149直线上最多的点数

class Solution:

def maxPoints(self, points: List[List[int]]) -> int:

from collections import Counter, defaultdict

# 所有点统计

points\_dict = Counter(tuple(point) for point in points)

print(points\_dict)

# 把唯一点列举出来

not\_repeat\_points = list(points\_dict.keys())

print(not\_repeat\_points)

n = len(not\_repeat\_points)

if n == 1: return points\_dict[not\_repeat\_points[0]]

res = 0

# 求最大公约数

def gcd(x, y):

if y == 0:

return x

else:

return gcd(y, x % y)

for i in range(n - 1):

# 点1

x1, y1 = not\_repeat\_points[i][0], not\_repeat\_points[i][1]

# 斜率

slope = defaultdict(int)

for j in range(i + 1, n):

# 点2

x2, y2 = not\_repeat\_points[j][0], not\_repeat\_points[j][1]

dy, dx = y2 - y1, x2 - x1

# 方式一 利用公约数

g = gcd(dy, dx)

if g != 0:

dy //= g

dx //= g

slope["{}/{}".format(dy, dx)] += points\_dict[not\_repeat\_points[j]]

print(slope)

res = max(res, max(slope.values()) + points\_dict[not\_repeat\_points[i]])

return res

1. 166分数到小数

输入：1,2 输出：”0.5” 输入：2,1 输出：1 输入20,3 输出：0.(6)

class Solution:

def fractionToDecimal(self, numerator: int, denominator: int) -> str:

if numerator == 0: return "0"

res = []

# 首先判断结果正负, 异或作用就是 两个数不同 为 True 即 1 ^ 0 = 1 或者 0 ^ 1 = 1

if (numerator > 0) ^ (denominator > 0):

res.append("-")

numerator, denominator = abs(numerator), abs(denominator)

# 判读到底有没有小数

a, b = divmod(numerator, denominator)

res.append(str(a))

# 无小数

if b == 0:

return "".join(res)

res.append(".")

# 处理余数

# 把所有出现过的余数记录下来

loc = {b: len(res)}

while b:

b \*= 10

a, b = divmod(b, denominator)

res.append(str(a))

# 余数前面出现过,说明开始循环了,加括号

if b in loc:

res.insert(loc[b], "(")

res.append(")")

break

# 在把该位置的记录下来

loc[b] = len(res)

return "".join(res)

1. 172阶乘后的0

def trailingZeroes(self, n: int) -> int:

p = 0

while n >= 5:

n = n // 5

p += n

return p

1. 190颠倒二进制位

（1）解法一

class Solution:

# @param n, an integer

# @return an integer

def reverseBits(self, n):

return int(bin(n)[2:].zfill(32)[::-1], base=2)

（2）解法2

class Solution:

# @param n, an integer

# @return an integer

def reverseBits(self, n):

res = 0

count = 32

while count:

res <<= 1

# 取出 n 的最低位数加到 res 中

res += n&1

n >>= 1

count -= 1

return int(bin(res), 2)

1. 191位1的个数

def hammingWeight(self, n):

count = 0

while n:

count += n&1

n >>= 1

return count

1. 204计数质数