齿轮与相遇问题

小圆滚滚

1 相遇问题

1.1 直线上的相遇

同步并行，就是两个人速度一样，肩并肩的前进。而现实生活中，总会有停顿、休息、再出发、起步、加速、减速、停顿……

假设两个人的速度不同，并且匀速运动，笨鸟先飞，速度慢的在速度快的开始前，就已经在速度快的前面领先了n米。那么就会存在一次相遇问题。

```python
# x = 30  # 甲的速度，单位：米/秒
# y = 22  # 乙的速度
#
# d_value = 400  # 甲和乙之间的距离是400米

jia = int(input("请输入甲的速度："))
yi = int(input("请输入乙的速度："))
distance = int(input("请输入甲乙两地的距离："))

"""
相遇问题求解函数
"""


def get_time(a, b, c):
    if a-b <= 0:
        return 0
    t = c / (a - b)  # 甲乙共同运动的时间
    print("甲走了{}米".format(jia * t))
    print("乙走了{}米".format(yi * t))
    return t


if __name__ == "__main__":
    time_float = get_time(jia, yi, distance)
    if time_float == 0:
        print("甲追不上乙")
    else:
        time = int(time_float)
        print("甲乙会在{}秒之后相遇。".format(time))
```

当跑道从直道变成环道时，就会存在多次相遇的问题。

两个齿数不同的齿轮互相咬合，标记起始点的两个齿轮的咬合赤，当它们再次相遇的时候，需要多少圈？

1.2 齿轮的相遇

```python
count = 0
for i in range(1, 1000):
    # print("两齿轮同时转动，走{}格".format(i))
    gear_1 = i % 30
    gear_2 = i % 18
    if gear_1 == 0:
        count += 1
        print("齿轮1碰到了齿轮2的{}，此时i={}，count={}".format(gear_2, i, count))

    # if gear_2 == 0:
    #     print("齿轮2碰到了齿轮1的{}，此时i={}".format(gear_1, i))

    if gear_1 == 0 and gear_2 == 0:
        print("齿轮相遇，此时i={}".format(i))
        exit()

```

Run: 齿轮比例-20220727-pre2

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.exe D:/Users/21/Pychar
齿轮1碰到了齿轮2的12，此时i=30，count=1
齿轮1碰到了齿轮2的6，此时i=60，count=2
齿轮1碰到了齿轮2的0，此时i=90，count=3
齿轮相遇，此时i=90

Process finished with exit code 0
```

两个齿轮的齿数如果互为质数，也就是他们的最大公约数是1，那么以其中一个齿轮接触另一个齿轮的齿为基准，转动，它会遍历另一个齿轮的所有的齿。换句话讲，两个齿轮的齿数差如果小于任一齿轮的齿数，如果这个齿数差与较小的齿轮齿数互质，那么就可以找到较小齿轮的逆，从而求得线性同余方程的解。依据就是辗转相减法求最大公约数。辗转相除法就是快速的辗转相减法。也就欧几里德算法。扩展欧几里德算法可以求得ax+by=gcd(a,b)的整数解。

写成方程：

$$\begin{cases} i \equiv x \ (mod\, 30) \\ i \equiv y \ (mod\, 18) \\ x = y = 0 \end{cases}$$

方程的解：

$$\begin{cases} i = x + k_1 \cdot 30 \\ i = y + k_2 \cdot 18 \\ x = y = 0 \end{cases}$$

$\Rightarrow \frac{k_1}{k_2} = \frac{18}{30}$

也就是（含义）：说当齿轮有最大公约数的时候，齿轮中固定的齿，只能触碰到对面齿轮固定的齿。他们的比例就是对面的齿数除以最大公约数。

若齿轮的齿数互质，则

$$\begin{cases} i \equiv x \ (mod\, 30) \\ i \equiv y \ (mod\, 17) \\ x - y = 1 \end{cases}$$

方程的解：

$$\begin{cases} i = x + k_1 \cdot 30 \\ i = y + k_2 \cdot 17 \\ \quad x - y = 1 \end{cases}$$

$\Rightarrow 1 = k_2 \cdot 17 - k_1 \cdot 30$

1.2.1 裴蜀定理

即裴蜀定理（或贝祖定理）得名于法国数学家艾蒂安·裴蜀，说明了对任何整数a、b和它们的最大公约数d，关于未知数x和y的线性不定方程（称为裴蜀等式）：若a,b是整数,且gcd(a,b)=d，那么对于任意的整数x,y,ax+by都一定是d的倍数，特别地，一定存在整数x,y，使ax+by=d成立。

它的一个重要推论是：a,b互质的充分必要条件是存在整数x,y使ax+by=1.

即

$$ax \equiv 1 \ (mod\, b) \tag{1}$$

含义是：两个齿轮齿数如果互质，那么其中一个齿轮的一个齿，必定会遍历另一个齿轮的所有齿。

(1)式称为线性同余方程的标准方程。

1.3 青蛙的相遇

齿轮比例-20220727-pre2.py　青蛙问题-20220729-pre1-2.py

```python
modulo = 18
x_0 = 0
y_0 = 0
m = 5
n = 7
for i in range(100):
    # print("两齿轮同时转动，走{}格".format(i))
    frog_1 = (x_0 + m * i) % modulo
    frog_2 = (y_0 + n * i) % modulo
    if frog_1 == frog_2:
        print("两青蛙相遇，此时i={}，相遇时在环的{}位置'

```

Run: 青蛙问题-20220729-pre1-2

```
C:\Users\21\AppData\Local\Programs\Python\Python
两青蛙相遇，此时i=0，相遇时在环的0位置
两青蛙相遇，此时i=9，相遇时在环的9位置
两青蛙相遇，此时i=18，相遇时在环的0位置
两青蛙相遇，此时i=27，相遇时在环的9位置
两青蛙相遇，此时i=36，相遇时在环的0位置
两青蛙相遇，此时i=45，相遇时在环的9位置
两青蛙相遇，此时i=54，相遇时在环的0位置
```

在一个体育场跑道上有两只青蛙A和B，两只青蛙的起始位置分别为x, y，A每次跳m,B每次跳n，环总长为modula.两只青蛙同时出发，两只青蛙落在同一点(肩并肩)视为相遇，问最少经过几次（k）跳跃两只青蛙相遇。

根据条件，我们列出解题方程： $(x + m \times k) \equiv (y + n \times k)\ (mod\ L)$，其中≡代表两边取模.

展开，为:

$x + m \times k = y + n \times k + L \times k'$

平移后转换为

$x - y = (n - m) \times k + L \times k'$

即

$(x-y) \equiv ((n-m) \times k) \ (mod\, L)$

令n-m=a, x-y=b可得$a \times k \equiv b \ (mod\, L)$,这便形如标准的一元线性同余方程

定义：a, b是整数，形如$a \times k \equiv b \ (mod\, M)$，且x是未知整数的同余式称为一元线性同余方程，其中≡ 及(mod M)表示两边对M取模.

1.4 五度相生律是加法同余

一根琴弦，按到三分之二处，就能得到此音纯五度高音

一根琴弦，按到二分之一处，就能得到此音纯八度高音

$x + 4 \equiv y \ (mod\, 7)$（加法同余）

printWuDuXiangSheng-20210807-pre3.py

```python
# pan = int(input("::请输入转盘大小: "))
# step = int(input("::请输入步长: "))
# # times = int(input("::请输入循环次数: "))

pan = 7
step = 4
seed = 0
ls = []
ls_num = []
dict = {0:'C', 1:'D', 2:'E', 3:'F', 4:'G', 5:'A', 6:'B'}

for i in range(pan):
    seed = (seed + step) % pan
    if seed in ls:
        print("列表中元素重复。")
        break
    else:
        ls.append(dict[seed])
        ls_num.append(seed)
print(ls)
print(ls_num)
# 只要pan和step互质就能遍历。

```

for i in range(pan) › else

Run: printWuDuXiangSheng-20210807-pre3

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.
 D:/Users/21/PycharmProjects/五度相生律/printWuDuXiangSheng-
['G', 'D', 'A', 'E', 'B', 'F', 'C']
[4, 1, 5, 2, 6, 3, 0]

Process finished with exit code 0
```

齿轮比例-20220727-互质的齿轮-pre3.py

```python
count = 0
for i in range(1, 1000):
    # print("两齿轮同时转动，走{}格".format(i))
    gear_1 = i % 4
    gear_2 = i % 7
    if gear_1 == 0:
        count += 1
        print("齿轮1碰到了齿轮2的{}，此时i={}，count={}".format(gear_2, i, count))

    # if gear_2 == 0:
    #     count += 1
    #     print("齿轮2碰到了齿轮1的{}，此时i={}，count={}".format(gear_1, i, count))

    if gear_1 == 0 and gear_2 == 0:
        print("齿轮相遇，此时i={}".format(i))
        exit()
```

Run: 齿轮比例-20220727-互质的齿轮-pre3

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.exe D:/Users/21/Pycharm
 质的齿轮-pre3.py
齿轮1碰到了齿轮2的4，此时i=4，count=1
齿轮1碰到了齿轮2的1，此时i=8，count=2
齿轮1碰到了齿轮2的5，此时i=12，count=3
齿轮1碰到了齿轮2的2，此时i=16，count=4
齿轮1碰到了齿轮2的6，此时i=20，count=5
齿轮1碰到了齿轮2的3，此时i=24，count=6
齿轮1碰到了齿轮2的0，此时i=28，count=7
齿轮相遇，此时i=28

Process finished with exit code 0
```

1.5 倍数齿轮

$$3x \equiv y \ (mod\, 7)$$

printWuDuXiangSheng-20210807-pre5.py

```python
# pan = int(input("::请输入转盘大小："))
# step = int(input("::请输入步长："))
# # times = int(input("::请输入循环次数："))

pan = 7
step = 0
seed = 3
ls = []
ls_num = []
dict = {0: 'C', 1: 'D', 2: 'E', 3: 'F', 4: 'G', 5: 'A', 6: 'B'}

for i in range(pan):
    seed = (seed * 3 + step) % pan
    if seed in ls:
        print("列表中元素重复。")
        break
    else:
        ls.append(dict[seed])
        ls_num.append(seed)
print(ls)
print(ls_num)
# 只要pan和step互质就能遍历。
```

for i in range(pan) › else

Run: printWuDuXiangSheng-20210807-pre5

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.exe D:
 /printWuDuXiangSheng-20210807-pre5.py
['E', 'B', 'G', 'A', 'D', 'F', 'E']
[2, 6, 4, 5, 1, 3, 2]

Process finished with exit code 0
```

齿轮比例-20220727-互质的齿轮-pre5.py

```python
count = 0
gear_2 = 1
for i in range(1, 7):
    # print("两齿轮同时转动，走{}格".format(i))
    gear_1 = i % 7
    gear_2 = i * 3 % 7
    print("此时i={}，gear_1={}，gear_2={}".format(i, gear_1, gear_2))
    if gear_1 == 0:
        count += 1
        print("齿轮1碰到了齿轮2的{}，此时i={}，count={}".format(gear_2, i, count))

    # if gear_2 == 0:
    #     count += 1
    #     print("齿轮2碰到了齿轮1的{}，此时i={}，count={}".format(gear_1, i, count))

    # if gear_1 == 0 and gear_2 == 0:
    #     print("齿轮相遇，此时i={}".format(i))
    #     exit()
```

for i in range(1, 7)

Run: 齿轮比例-20220727-互质的齿轮-pre5

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.exe D:/Users/21/Pycharm
 轮比例-20220727-互质的齿轮-pre5.py
此时i=1，gear_1=1，gear_2=3
此时i=2，gear_1=2，gear_2=6
此时i=3，gear_1=3，gear_2=2
此时i=4，gear_1=4，gear_2=5
此时i=5，gear_1=5，gear_2=1
此时i=6，gear_1=6，gear_2=4

Process finished with exit code 0
```

齿轮种子以3开始的时候，右侧相当于查表，可得2.齿轮1调为2时，齿轮2对应6，依次向下查表，可得左侧乘法同余的序列。

模运算性质1参见章节1.8

可得齿轮相加

$3x + 4 \equiv y \ (mod\, 7)$

1.6 求解一元线性同余方程（展开同余方程的表达式）

求解一元线性同余方程，首先需要将其一步步做如下变换：

1. 标准的一元线性同余方程 $a \times k \equiv b \ (mod\, M)$等价于
   $a \times x + M \times y = b$
2. 假设d为a与M的最大公约数，记为gcd(a, M)=d,易知若x，y有解,d必为b的因子。
   设$a = a_0 \times d$, $M = M_0 \times d$, $b = b_0 \times d$,
   因此等式变换为$a_0 \times x + M_0 \times y = b_0$
   其中a_0, M_0 互质，即$gcd(a_0, M_0) = 1$
3. 令$x = x_0 \times b_0, y = y_0 \times b_0$ 方程变换为$a_0 \times x_0 + M_0 \times y_0 = 1$ 即$a_0 \times x_0 \equiv 1 \ (mod\, M_0)$

最后运用模逆计算出x_0

定理：如果a和M为互素的整数，$M > 1$，则存在a的模M的逆.而且这个逆模M是唯一.

这个定理的证明也比较简单，推导过程如下：
若a与m互质，必存在$s \times a + t \times M = 1$（若a,M有大于1的公因子d，则$s \times a + t \times M = d \times k$而不可能等于1），两边取M的模，可得$s \times a \equiv 1 \ (mod\, M)$,即s为a的模M的逆
若记$\bar{a}$为a的模M的逆，即$x_0 \equiv \bar{a} \ (mod\, M_0)$,也就是 $x_0 \equiv \bar{a} + k \times M_0$

举个简单的例子：

求解$5 \times x \equiv 7 \ (mod\, 9)$

1. 由于$gcd(5, 9) = 1$,可知存在5模9的逆，易知$2 \times 5 + (-1) \times 9 = 1$
2. 得出2为5模9的逆，方程两边乘以2有$2 \times 5 \times x \equiv 2 \times 7 \ (mod\, 9)$
3. 化简为$x \equiv x + 9x \equiv 10 \times x \ (mod\, 9) \equiv 14 \ (mod\, 9) \equiv 5 \ (mod\, 9)$
4. 得出 $x = 5 + 9k$

综合以上，我们将普通线性同余方程$a \times x \equiv b \ (mod\, L)$转换为标准方程(1)式$a_0 \times x_0 \equiv 1 (mod\, M_0)$进行求解，求解标准方程完毕后需要将解空间映射回去，若标准方程的解为$x_0 \in \{x = p + q \times M_0\}$

则由于$x = x_0 \times b_0, y = y_0 \times b_0$且$b = b_0 \times d$，得出$x \in \{x = b/d \times p + q' \times M_0\}$ 为方程$a \times x + M \times y = b$的解

1.7 基本性质

1. 若$p|(a - b)$，则$a \equiv b \ (mod\, p)$。例如 $11 \equiv 4 (mod\, 7)$， $18 \equiv 4 (mod\, 7)$.解释a-b可以被p整除。
2. $(a \ \% \ p) = (b \ \% \ p)$意味$a \equiv b \ (mod\, p)$
3. 对称性：$a \equiv b \ (mod\, p)$等价于$b \equiv a \ (mod\, p)$
4. 传递性：若$a \equiv b \ (mod\, p)$且$b \equiv c \ (mod\, p)$，则$a \equiv c \ (mod\, p)$

1.8 模运算的性质

若$(a-b)\%m == 0$,就称a,b关于m同余，或者说a,b对模数m同余。

e.g. $(100-60)\%8 == 0$,我们就说100和60对于模数8同余。

它的另一层含义就是说，100和60除以8的余数相同。

a和b对m同余，我们记作$a \equiv b\ (mod\, m)$

性质：

1. 如果$a \equiv b\ (mod\, m), x \equiv y\ (mod\, m)$，则$a+x \equiv b+y\ (mod\, m)$//两边分别相加
2. 如果$a \equiv b\ (mod\, m), x \equiv y\ (mod\, m)$，则$ax \equiv by\ (mod\, m)$//两边分别相乘
3. 如果$ac \equiv bc\ (mod\, m)$，且c和m互质，则$a \equiv b\ (mod\, m)$//c,m互质时，去掉c

1.9 模线性方程组

输入正整数a,b,n 解方程$a \equiv b\ (mod\, m).a, b, n <= 10^9$

方程可理解为ax-b是n的正整数倍，设倍数为y，则ax-b=ny;即ax-ny=b 解此不定方程—-扩展欧几里得算法

同余方程的解是一个同余等价类

b=1时,$a \equiv 1\ (mod\, m)$的解称a关于模n的逆invers—-乘法逆元

有解条件，a,n互质gcd（a,n）=1，此时，方程只有唯一解

2 扩展欧几里德算法

扩展欧几里得算法（英语：Extended Euclidean algorithm）是欧几里得算法（又叫辗转相除法）的扩展。已知整数a、b，扩展欧几里得算法可以在求得a、b的最大公约数的同时，能找到整数x、y（其中一个很可能是负数），使它们满足贝祖等式

$ax + by = gcd(a, b).$

如果a是负数，可以把问题转化成$|a|(-x) + by = gcd(|a|, b)$，然后令x'=(-x)。

通常谈到最大公约数时，我们都会提到一个非常基本的事实：给予二个整数a、b，必存在整数x、y使得ax + by = gcd(a,b)。

有两个数a,b，对它们进行辗转相除法，可得它们的最大公约数——这是众所周知的。然后，收集辗转相除法中产生的式子，倒回去，可以得到ax+by=gcd(a,b)的整数解。

扩展欧几里得算法可以用来计算模反元素(也叫模逆元)，而模反元素在RSA加密算法中有举足轻重的地位。

2.1 例子

用类似辗转相除法，求二元一次不定方程47x+30y=1的整数解。过程可以用矩阵表示（其中q表示商，r表示余数）。

$\begin{pmatrix} a \\ b \end{pmatrix} = \prod_{i=0}^{N} \begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} r_{N-1} \\ 0 \end{pmatrix}$

$\begin{pmatrix} 47 \\ 30 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 47 & 11 \\ 30 & 7 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

详细过程：

$$\begin{pmatrix}47\\30\end{pmatrix}=\begin{pmatrix}1&1\\1&0\end{pmatrix}\begin{pmatrix}30\\17\end{pmatrix}$$
$$\begin{pmatrix}30\\17\end{pmatrix}=\begin{pmatrix}1&1\\1&0\end{pmatrix}\begin{pmatrix}17\\13\end{pmatrix}$$
$$\begin{pmatrix}17\\13\end{pmatrix}=\begin{pmatrix}1&1\\1&0\end{pmatrix}\begin{pmatrix}13\\4\end{pmatrix}$$
$$\begin{pmatrix}13\\4\end{pmatrix}=\begin{pmatrix}3&1\\1&0\end{pmatrix}\begin{pmatrix}4\\1\end{pmatrix}$$
$$\begin{pmatrix}4\\1\end{pmatrix}=\begin{pmatrix}4&1\\1&0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}$$

矩阵变换（同时左乘 $\begin{pmatrix}47&11\\30&7\end{pmatrix}$逆矩阵$\begin{pmatrix}-7&11\\30&47\end{pmatrix}$）：

$$\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}-7&11\\30&47\end{pmatrix}\begin{pmatrix}47\\30\end{pmatrix}$$

或者用初等变换

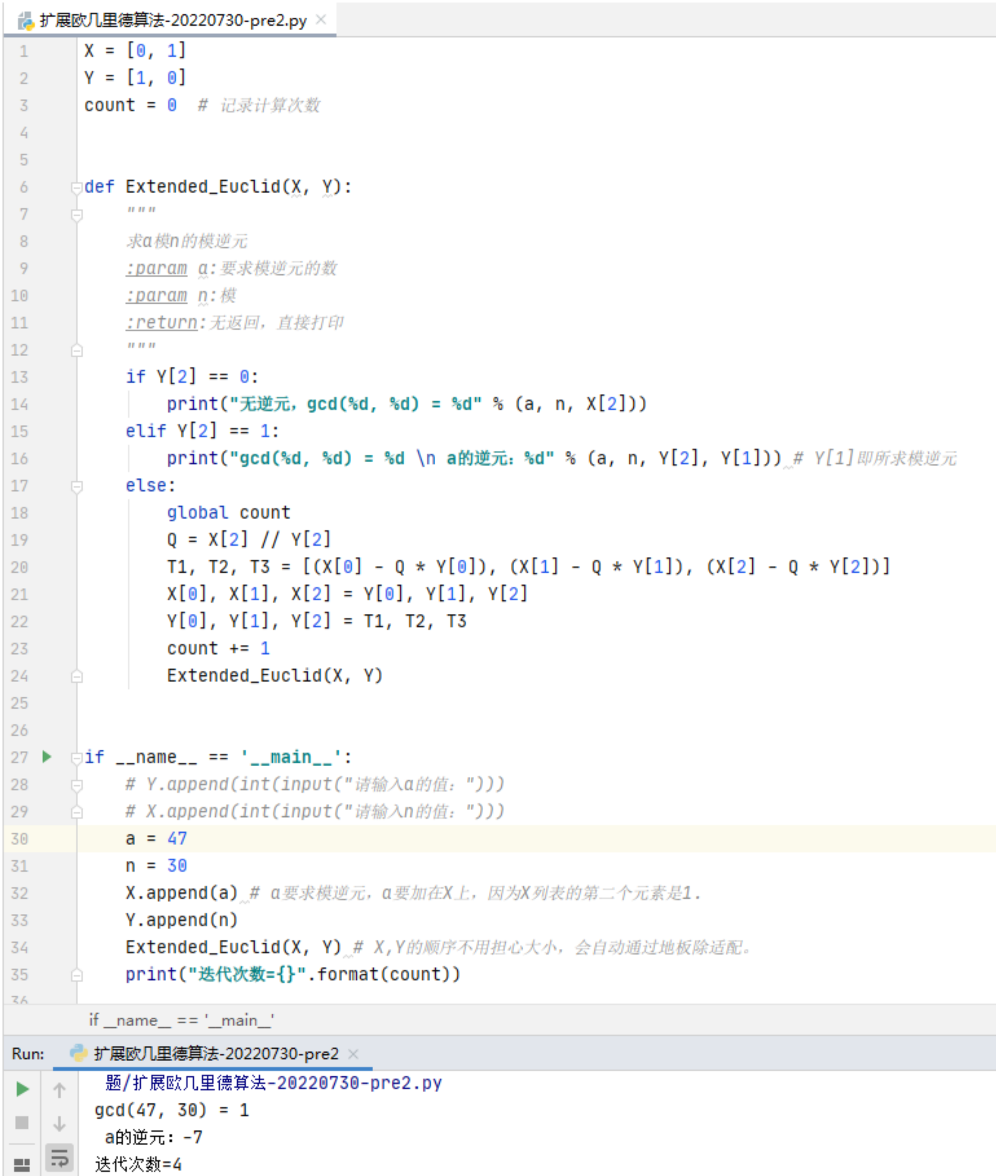
$$\begin{pmatrix}47&30\\1&0\\0&1\end{pmatrix}\to\begin{pmatrix}17&30\\1&0\\-1&1\end{pmatrix}\to\begin{pmatrix}17&13\\1&-1\\-1&2\end{pmatrix}\to\begin{pmatrix}4&13\\2&-1\\-3&2\end{pmatrix}\to\begin{pmatrix}4&1\\2&-7\\-3&11\end{pmatrix}\Rightarrow 1=47(-7)+30(11)$$

递归伪代码

定义列表$(X1,X2,X3)\leftarrow(0,1,a);(Y1,Y2,Y3)\leftarrow(1,0,n)$，跟在0,1后面的是要求模逆元的数，跟在1,0后面的是模。

定义函数Extended EUCLID(X, Y):

1. 如果Y3=0返回X3=gcd(a,n);无逆元
2. 如果Y3=1返回Y3=gcd(a,n);$Y2=a^{-1}\,mod\,n$即所求模逆元
3. $Q=\lfloor X3/Y3\rfloor$
4. $(T1,T2,T3)\leftarrow(X1-Q\cdot Y1,X2-Q\cdot Y2,X3-Q\cdot Y3)$
5. $(X1,X2,X3)\leftarrow(Y1,Y2,Y3)$
6. $(Y1,Y2,Y3)\leftarrow(T1,T2,T3)$
7. Extended EUCLID(X, Y)



```python
X = [0, 1]
Y = [1, 0]
count = 0  # 记录计算次数


def Extended_Euclid(X, Y):
    """
    求a模n的模逆元
    :param a:要求模逆元的数
    :param n:模
    :return:无返回，直接打印
    """
    if Y[2] == 0:
        print("无逆元，gcd(%d, %d) = %d" % (a, n, X[2]))
    elif Y[2] == 1:
        print("gcd(%d, %d) = %d \n a的逆元：%d" % (a, n, Y[2], Y[1]))  # Y[1]即所求模逆元
    else:
        global count
        Q = X[2] // Y[2]
        T1, T2, T3 = [(X[0] - Q * Y[0]), (X[1] - Q * Y[1]), (X[2] - Q * Y[2])]
        X[0], X[1], X[2] = Y[0], Y[1], Y[2]
        Y[0], Y[1], Y[2] = T1, T2, T3
        count += 1
        Extended_Euclid(X, Y)


if __name__ == '__main__':
    # Y.append(int(input("请输入a的值：")))
    # X.append(int(input("请输入n的值：")))
    a = 47
    n = 30
    X.append(a)  # a要求模逆元，a要加在X上，因为X列表的第二个元素是1.
    Y.append(n)
    Extended_Euclid(X, Y)  # X,Y的顺序不用担心大小，会自动通过地板除适配。
    print("迭代次数={}".format(count))
```

```
题/扩展欧几里德算法-20220730-pre2.py
gcd(47, 30) = 1
 a的逆元：-7
迭代次数=4
```

3 Hull-Dobell Theorem

3.1 混合同余法

混合同余法产生随机数的公式

$$x_i \equiv ax_{i-1} + c \; (mod\, m) \tag{2}$$

$c \neq 0$

The sequence defined by the congruence relation (2) has full period m, provided that

1. c is relatively prime to m;
2. $a \equiv 1 \; (mod\, p)$ if p is a prime factor of m;
3. $a \equiv 1 \; (mod\, 4)$ if 4 is a factor of m;

Thus with m a power of 2, as is natural on a binary machine, we need only have c odd, and $a \equiv 1 \ (mod\, 4)$. With m a power of 10 we need only have c not divisible by 2 or 5, and $a \equiv 1 \ (mod\, 20)$.

定理1翻译：

$c \neq 0$

1. m和c互质
2. a-1 能被m的所有质因数整除
3. 如果m能被4整除，那么a-1也要能被4整除

3.2 乘同余法

乘同余法产生随机数公式

$$x_i \equiv a x_{i-1} \ (mod\, m) \tag{3}$$

The sequence defined by taking c = 0 in the congruence relation (2) has maximal period, provided that

1. x_0 is relatively prime to m;
2. a is a primitive root for p^α, if p^α is a factor of m, with p odd and a as large as possible, or with $p = 2$ and $\alpha = 1$ or 2;
3. a belongs to $2^{\alpha-2}$, if 2^α is a factor of m, with $\alpha > 2$. Moreover, for any m, there exist values of a satisfying these conditions, and, finally, the maximal period is the lowest common multiple of the periods, $(p-1)p^{\alpha-1}$ or $2^{\alpha-2}$, with respect to the prime power factors.

定理2翻译：

1. x_0 与m互质

2. 如果p的α次幂是m的一个因数，并且（p是奇数，并且α尽可能的大），或者并且（p=2并且（α=1或者2）），那么a需是p的α次幂的原根。

3. 如果2的α次幂是m的一个因数，a需属于2的（α-2）次幂，并且$\alpha > 2$. 此外，对于任何m，存在满足这些条件的值，最后，关于素数幂的因数们，最大周期是最小周期的公倍数，（p-1）倍的p的（α-1）次幂或者2的（α-2）次幂，

4 欧拉定理

4.1 欧拉函数

欧拉函数$\varphi(n)$ 是「小于n 的正整数中和n 互质的数」的个数.

例如

$\varphi(100) = 100 \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{5}\right) = 40$

欧拉定理是数论中一个非常重要的定理：

设$a, m \in N^+$，且$gcd(a, m) = 1$则：

$a^{\varphi(m)} \equiv 1 \pmod m$

含义，欧拉函数说的是素数在该素数之前的所有的数都和它互质，所以缩系的元素个数就是该数减1.费马小定理说的是如果p是一个质数，而整数a不是p的倍数，则有a的(p-1)次幂模p等于1.欧拉定理说的是a的缩系的元素个数次方模m等于1.当m为质数时，则欧拉定理退化为费马小定理.

这里的$\varphi(m)$ 是欧拉函数，即小于或等于m 且与m 互质的正整数个数。当m 是质数p 时，欧拉定理退化成费马小定理$a^{p-1} \equiv 1 \pmod p$ 。

我们稍后再来证明欧拉定理。在算法竞赛中，我们常常会用到它的一个重要的推论：若正整数a与m 互质，则

$a^b \equiv a^{b \bmod \varphi(m)} \pmod m$

这是因为: $a^b = a^{\varphi(m)\lfloor b/\varphi(m)\rfloor + b \bmod \varphi(m)} \equiv 1 \cdot a^{b \bmod \varphi(m)} \pmod m$

利用这个推论，即使b 比较大，我们也可以轻松地计算$a^b \bmod m$ 的值，但需要满足a 与m 互质的前提。

4.2 r模n的阶

原根的定义要从整数的阶说起.

如果正整数r, n满足$gcd(r, n) = 1$，那么根据欧拉定理方程$r^x \equiv 1\ (mod\, n)$必然存在正整数解；设所有正整数解当中最小的是x_0，那么若$r^x \equiv 1\ (mod\, n)$则$x_0|x$.

(这个可以由带余除法定理证明，我就偷懒略去了.)

这个最小的正整数成为r模n的阶，记为$ord_n(r) = x_0$

「证明」原根相关性质及证明

先定义阶的概念：如果$gcd(a, p) = 1$，那么对于方程$a^r \equiv 1(mod\ p)$来说，

首先根据欧拉定理$a^{\varphi(p)} \equiv 1(mod\ p)$，解一定存在所以$r \le \phi(p)$，最小的$r$称为$a$关于$p$的阶，记作$ord_p(a)$.阶总是可以整除欧拉函数在该模下的值。

4.3 原根相关性质及证明

定义原根概念：一个模p意义下的$0 \sim p-1$次幂各不相同，取遍$[0, p-1]$,也就是说$ord_p(a) = \varphi(p)$。那么a就是p的原根。

例如：模$p = 9$，9的质因数是1，3.那么从1到8，有1、2、3、4、5、7、8，6个数与9是互质的，即欧拉函数等于$\varphi(9) = 6$（$9 \times (1 - \frac{1}{3}) = 6$）.

取遍[0,8]，$2^1 \equiv 2 \pmod 9, 2^2 \equiv 4 \pmod 9, 2^3 \equiv 8 \pmod 9, 2^4 \equiv 7 \pmod 9, 2^5 \equiv 5 \pmod 9, 2^6 \equiv 1 \pmod 9, 2^7 \equiv 2 \pmod 9, 2^8 \equiv 4 \pmod 9 \cdots$，可以看出6是9的阶。6又等于欧拉函数在9的值，阶和欧拉函数相等，所以2是9的原根。

先说一下什么样的数具有原根。

结论是:对于奇质数p，有原根的数是:$1, 2, 4, p, 2p, p^n$证明比较麻烦，Niven和Zuckerman证明《An Introduction to the Theory of Numbers》，略去过程。

因为最小原根一般都比较小，所以可以直接枚举出来，而这种方法有时候就显得过于慢。

怎么更快。

原根判定定理：设$m \ge 3, gcd(a, m) = 1$ ，则a 是模m 的原根的充要条件是，对于 $\varphi(m)$的每个素因数p ，都有$a^{\frac{\varphi(m)}{p}} \not\equiv 1 \pmod m$

翻译为：有一个结论可以用：对于一个有原根的数 p，如果 g 的 $\phi(p)$ 的所有因子次方在 $mod\ p$ 条件下均不为 1，那么 g 是 p 的原根。

证明：首先结论可以转化为，如果对于任意的 $b|\phi(p)$, 均不满足 $g^b \equiv 1(mod\ p)$ 那么，对于任意的 $1 \le b \le \phi(p) - 1$ (b 不满足 $b|\phi(p)$), 均不满足 $g^b \equiv 1(mod\ p)$。

反证：

假设存在一个 b,(b 不满足 $b|\phi(p)$)，满足 $g^b \equiv 1(mod\ p)$，设其中小的为 c, 那么 $g^c \equiv 1(mod\ p)$ 成立。

令 $d = \phi(p) - c, d >= c$

根据欧拉定理。

$g^d \equiv g^{\phi(p)-c} \equiv g^{-c} \equiv 1(mod\ p)$

引理：$c|d$ 不成立。

反证：假设成立。

令 $d = kc$

那么：$\phi(p) = d + c = (k+1)c$

不满足 $c|\phi(p)$。

所以假设不成立。

引理 $c|d$ 不成立得证。

那么 $gcd(c,d) \leq c$

因为:

$g^c \equiv 1(mod\ p)$

$g^d \equiv 1(mod\ p)$

那么：

$g^{c-d} \equiv 1(mod\ p)$

模拟更相损益术重复相减得到最终的 gcd 的过程，发现最终结果是：

$g^{gcd(d,c-d)} \equiv g^{gcd(c,d)} \equiv 1(mod\ p)$;

因为 $gcd(c,d) < c$ 与假设的 c 是最小的 b 不成立。

所以假设不成立。

证毕。

所以用这种方式可以较快的验证原根。

4.4 原根的性质

原根的性质

1. 对于任意正整数a,m,如果gcd(a,m) = 1,存在最小的正整数 d 满足$a^d \equiv 1 \pmod{m}$，则有 d 整除$\varphi(m)$ ，因此$Ord_m(a)$整除$\varphi(m)$。这里的d被称为a模m的阶，记为$Ord_m(a)$.
   例如:求3模7的阶时，我们仅需要验证 3 的 1、2、3 和 6 次方模 7 的余数即可。
   因为4、5不能整除6，而阶一定小于等于$(7-1)$.
2. 记$\delta = Ord_m(a)$，则$a^1, \cdots, a^{(\delta-1)}$模m两两不同余。因此当$a$是m的原根时，$a^0, a^1, \cdots, a^{(\delta-1)}$构成模 m 的简化剩余系。
3. 模m有原根的充要条件是$m = 1, 2, 4, p, 2p, p^n$，其中p是奇质数，n是任意正整数。
4. 对正整数gcd(a,m) = 1，如果 a 是模 m 的原根，那么 a 是整数模n乘法群这里应该是整数模m乘法群（即加法群 Z/mZ的可逆元，也就是所有与 m 互素的正整数构成的等价类构成的乘法群）Z_n的一个生成元。由于Z_n有 $\varphi(m)$个元素，而它的生成元的个数就是它的可逆元个数，即 $\varphi(\varphi(m))$个，因此当模m有原根时，它有$\varphi(\varphi(m))$个原根。

4.5 缩系

与模m互素的剩余类：

剩余类：用一个正数代表在 $\mod m$情况下的同余集合。

比如10的$[1] = \{1, 11, 21, \cdots, -9, -19, \cdots\}$

剩余类集：显然就是剩余类的集合

例如：$Z10 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$，其中每个数字都是一个剩余类

简化剩余类：是剩余类集的一个子集，在与模m互素的全体剩余类中，从每一个类中各任取一个数作为代表组成的集合，叫做模m的一个简化剩余系。

例如模5的一个简化剩余系是：1、3、7、9

模m的完系乘以一个与m互素的数a后，得到的集合仍然是完系。对于模m的缩系有同样的性质。

打印判断是否原根2.py × 模10的完系乘以一个素数.py ×

```python
##phi = int(input("请输入欧拉函数值:"))
##mod = int(input("请输入模的大小:"))
##a = int(input("请输底:"))
"""
欧拉定理：若gcd(a,n)=1，则a^{\varphi(n)} \equiv 1 (mod\;p)
"""
phi = 6
mod = 9
a = 2
ls = []
for i in range(1, (phi+1)*2):
    ls.append(a**i%mod)
print(ls)
ls = list(set(ls)) # 去重
ls.sort() # 排序
print(ls)
print("列表长度：", len(ls))


a = 5
ls = []
for i in range(1, (phi+1)*2):
    ls.append(a**i%mod)
print(ls)
ls = list(set(ls)) # 去重
ls.sort() # 排序
print(ls)
print("列表长度：", len(ls))
```

Run: 打印判断是否原根2 ×

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.
[2, 4, 8, 7, 5, 1, 2, 4, 8, 7, 5, 1, 2]
[1, 2, 4, 5, 7, 8]
列表长度： 6
[5, 7, 8, 4, 2, 1, 5, 7, 8, 4, 2, 1, 5]
[1, 2, 4, 5, 7, 8]
列表长度： 6

Process finished with exit code 0
```

打印判断是否原根2.py　模10的完系乘以一个素数.py

```python
ls = []
# 完系乘以素数
for i in range(20):
    ls.append(i * 3 % 10)
print(ls)
ls = list(set(ls))
ls.sort()
print(ls)

# 缩系乘以素数
ls1 = [1, 3, 7, 9]
ls.clear()
for i in ls1:
    ls.append(i * 7 % 10)
print(ls)
ls = list(set(ls))
ls.sort()
print(ls)
```

Run: 模10的完系乘以一个素数

```
C:\Users\21\AppData\Local\Programs\Python\Python38\python.exe D:/Users/21/
[0, 3, 6, 9, 2, 5, 8, 1, 4, 7, 0, 3, 6, 9, 2, 5, 8, 1, 4, 7]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[7, 1, 9, 3]
[1, 3, 7, 9]

Process finished with exit code 0
```

4.6 举个例子

例如：$a = 2, m = 1000003, \varphi(m) = 1000002$则：

1000002的质因数：2、3、166667

$2^{3\times166667} \not\equiv 1 \pmod{1000003}$

$2^{2\times166667} \not\equiv 1 \pmod{1000003}$

$2^{2\times3} \not\equiv 1 \pmod{1000003}$

所以2是1000003的原根。$2^x \pmod{1000003}$将遍历 $\varphi(m)$，其中$x \in \varphi(m)$

例2：$a = 2, m = 17, \varphi(m) = 16, p = 2$，且p只有2.则

$2^{\frac{\varphi(m)}{p}} = 2^{\frac{16}{2}} \equiv 1 \pmod{17}$

$3^{\frac{16}{2}} \equiv 16 \pmod{17} \not\equiv 1 \pmod{17}$所以3是17的原根。

求9的原根：9的质因数有1，2，4，5，7，8.所以$\varphi(9)=6$，则有

$2^1\equiv 2 \pmod 9, 2^2\equiv 4 \pmod 9, 2^3\equiv 8 \pmod 9, 2^4\equiv 7 \pmod 9, 2^5\equiv 5 \pmod 9, 2^6\equiv 1 \pmod 9$

至此找到阶。不可能继续，因为不能超过$\varphi(9)=6$.阶和欧拉函数值相等，所所以2是9的原根。右侧产生的数就是缩系（没有与9不互质的数：3、6）。

同样有：$5^1\equiv 5 \pmod 9, 5^2\equiv 7 \pmod 9, 5^3\equiv 8 \pmod 9, 2^5\equiv 4 \pmod 9, 5^5\equiv 2 \pmod 9, 5^6\equiv 1 \pmod 9$，可知5是9的另一个原根。

如果用源根判定定理，因为6的素因数有2，3.不包含1. $2^{\frac{6}{2}}\not\equiv 1 \pmod 9, 2^{\frac{6}{3}}\not\equiv 1 \pmod 9$，所以2是9的原根。

例3：$a=3, m=10, \varphi(10)=4, p=2$，$3^2\not\equiv 1 \pmod{10}$，所以3是10的原根。缩系：$3^1\equiv 3 \pmod{10}, 3^2\equiv 9 \pmod{10}, 3^3\equiv 7 \pmod{10}, 3^4\equiv 1 \pmod{10}$是$\{1,3,7,9\}$

5 程序员如何设计一个游戏，看起来像上帝

天地不仁，以万物为刍狗。仁慈就是偏向，不仁慈就是公正。如何公正？就是均匀分布。满足均匀分布就是随机的第一步。

如何随机，起始种子不一样。然后线性同余。

取模如何大于设定的所有条件的个数。因为大自然的变量是无穷的，所以只要取模足够大，那么满足均匀分布的分配就是随机的。

5.1 游戏中的抽奖如何规避犯法