西安电子科技大学网信院

信息安全基础与密码学综合实验

实 验 报 告（二）

中国剩余定理

班级：

姓名：

学号：

日期：

一、实验目的

1.实验环境：Windows11，tdm64-gcc-10.3.0，VSCode

2.实验目标：通过编程实现中国剩余定理;加深对于中国剩余定理的理解与运用，体会密码学与数论的紧密联系；提高逻辑思维能力

二、方案设计

1.背景

一千多年前的《孙子算经》中，有这样一道算术题：“今有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二，问物几何？”按照今天的话来说：个数除以三余二，除以五余三，除以七余二，求这个数。《孙子算经》给出了一个非常有效的巧妙解法。术曰：“三、三数之剩二，置一百四十；五、五数之剩三，置六十三；七、七数之剩二，置三十，并之，得二百三十三。以二百一十减之，即得。凡三、三数之剩一，则置七十；五、五数之剩一，则置二十一；七、七数之剩一，则置十五。一百六以上，一百五减之，即得。

后来流传的《孙子歌》中所说“七十稀”、“廿一枝”和“正半月”，就是暗指这三个关键的数字。《孙子算经》没有说明这三个数的来历。实际上，它们具有如下特性：也就是说，这三个数可以从最小公倍数 M=3×5×7=105 中各约去模数 3、5、7 后，再分别乘以整数 2、1、1 而得到。假令 $k_1=2$，$k_2=1$，$k_3=1$，那么整数 k_i（i=1，2，3）的选取使所得到的三数 70、21、15 被相应模数相除的时候余数都是 1。由此出发，立即可以推出，在余数是 r_1、r_2、r_3 的情况下的情况。应用上述推理，可以完全类似地把孙子算法推广到一般情形：设有一数 n，分别被两两互素的几个数 a_1，a_2，…，a_n 相除得余数 r_1，r_2，…，r_n，即 $n\equiv r_i$（mod a_i）（i=1，2，…，n），只需求出一组数 k，使满足 1（mod a_i）（i=1，2，…，n），那么适合已给一次同余组的最小正数解是 p 是整数，$m=a_1\times a_2\times\cdots\times a_n$)，就是现代数论中著名的剩余定理。

2.原理：中国剩余定理

设正整数 m_1，m_2，…，m_k 两两互素，对任意整数 a_1，a_2，…，a_k，一次同余方程

$$\begin{cases} x \equiv a_1 (mod\ m_1) \\ x \equiv a_2 (mod\ m_2) \\ \quad\quad \dots \\ \quad\quad \dots \\ x \equiv a_k (mod\ m_k) \end{cases}$$ 在模 m 意义下有唯一解，该解可表示为

$$x \equiv M_1 M_1^{-1} a_1 + \ M_2 M_2^{-1} a_2 + \dots + M_k M_k^{-1} a_k (mod\ m)$$

其中 $m=m_1m_2\cdots m_k$， $M_j=m/m_j$， $M_jM_j^{-1}\equiv 1(mod\ m_j)$， $j=1,2,\cdots,k$

3.算法步骤

a.判断正整数 m_1，m_2，…，m_k 是否两两互素；是，则继续，否则跳出，输出“不能直接利用中国剩余定理”；

b.计算 $m=m_1m_2\cdots m_k$，$M_j=m/m_j$；

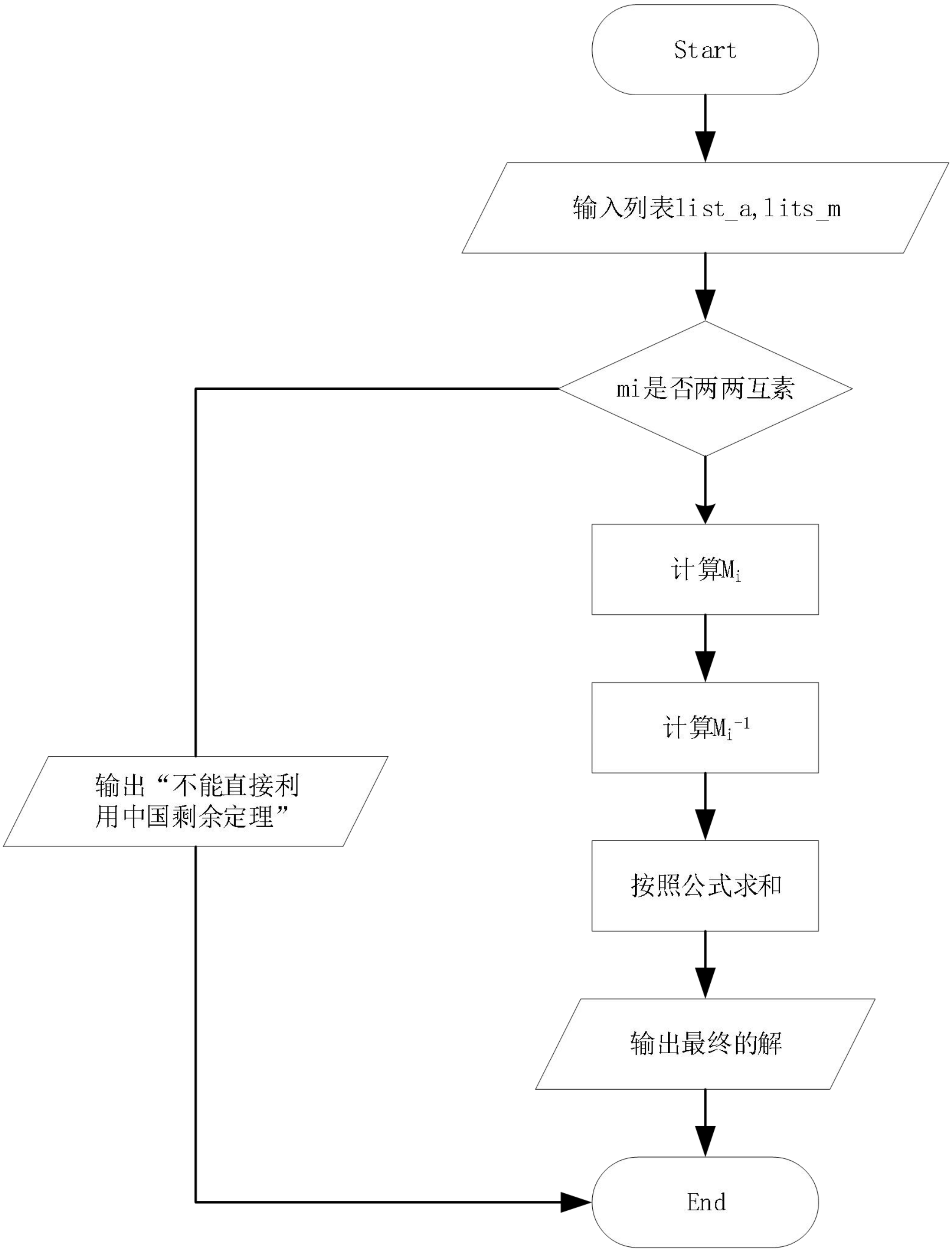
c.计算 $M_j^{-1}(mod\ m_j)$；

d.计算 $x_j\equiv M_jM_j^{-1}a_j(mod\ m)$；

e.计算 $x \equiv \sum_{j=1}^{k} x_j\ (mod\ m)$。

三、方案实现

1.算法流程图



2.主要函数介绍

a. gcd(a, b)：求 a 和 b 的最大公约数，若结果为 1，则两数互质

```
def gcd(a, b):
    if a < b:  # 调整大小顺序
        a, b = b, a
    else:
```

```
        pass
    while b != 0:
        return gcd(b, a % b)
    return a
```

b. compare(list)：判断一个列表里的数是否两两互质，互质则满足 CRT 的使用条件

```
def compare(list):
    for i in range(0, len(list)):
        for j in range(i + 1, len(list)):
            if gcd(list[i], list[j]) != 1:
                print('不能直接利用中国剩余定理')
                exit()
```

c. product_m(list)：求出输入的 $m_1, m_2, \cdots, m_k$ 的乘积即 m

```
def product_m(list):
    m = 1
    for i in list:
        m *= i
    return m
```

d. get_divsion(list, m)：求 $M_1, M_2, ..., M_K$ 的值，其中 $M_j=m/m_j$

```
def get_divsion(list, m):
    div = []
    for i in list:
        div.append(m // i)
    return div
```

e. get_inverse(a, m)：先求数 a 的逆，再求其模 m 的值

```
def get_inverse(a, m):
    if gcd(a, m) != 1:
        return None
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m
    while v3 != 0:
        q = u3 // v3
        v1, v2, v3, u1, u2, u3 = (u1 - q * v1), (u2 - q * v2),
(u3 - q * v3), v1, v2, v3
    return u1 % m
```

f. get_x(M: int, M_inverse: int, a: int, m: int)：求 X_j，算法为：$X_j = (M * M\_inverse * a) \% m_j$

```
def get_x(M: int, M_inverse: int, a: int, m: int):
    product_x = (M * M_inverse * a) % m
```

```
    return product_x
```

g. get_solution(list_m, list_a)：算出最终答案 X = $X_1+X_2+\cdots+X_k$

```
def get_solution(list_m, list_a):
    compare(list_m)
    m = product_m(list_m)

    list_M = get_divsion(list_m, m)

    list_M_inverse = []
    list_X = []
    total = 0

    for i in range(0, len(list_M)):
        list_M_inverse.append(get_inverse(list_M[i],
list_m[i]))

    for i in range(len(list_M)):
        list_X.append(get_x(list_M[i], list_M_inverse[i],
list_a[i], m))

    for x in list_X:
        total += x

    return total % m
```

3. 算法实现主要代码

```
f = open('4.txt', 'r', encoding='utf-8')
data = f.readlines()
list_a = list(map(int, data[0:3]))
list_m = list(map(int, data[3:6]))
print(get_solution(list_m, list_a))
```

四、数据分析(包括算法测试数据的分析，运行结果截图等等)

1. 输入读取“1. txt”

```python
f = open('1.txt', 'r', encoding='utf-8')
data = f.readlines()
list_a = list(map(int, data[0:3]))
list_m = list(map(int, data[3:6]))
print(get_solution(list_m, list_a))
```

问题 输出 JUPYTER 调试控制台 终端

```
PS C:\Users[redacted]cd "c:\Users
PS C:\Users[redacted]\实验2-CRT> py
不能直接利用中国剩余定理
PS C:\Users[redacted]\实验2-CRT>
```

2. 输入读取“2. txt”

```python
f = open('2.txt', 'r', encoding='utf-8')
data = f.readlines()
list_a = list(map(int, data[0:3]))
list_m = list(map(int, data[3:6]))
print(get_solution(list_m, list_a))
```

问题 输出 JUPYTER 调试控制台 终端

```
PS C:\Users[redacted]\实验2-CRT>
PS C:\Users[redacted]\实验2-CRT>
不能直接利用中国剩余定理
PS C:\Users[redacted]\实验2-CRT>
```

3. 输入读取“3. txt”

```python
f = open('3.txt', 'r', encoding='utf-8')
data = f.readlines()
list_a = list(map(int, data[0:3]))
list_m = list(map(int, data[3:6]))
print(get_solution(list_m, list_a))
```

问题 输出 JUPYTER 调试控制台 终端

```
PS C:\Users[redacted]实验2-CRT> python -u "c:\User
s[redacted]"
542864850248289073548572880894113698650347984625880005580337658354523403026
79356524892639345005568791319529360375069880910979199010559372463516459785072
10948874326419139803862900995488633097366604024618914096512453530894104022741
69919095197787890140641285415865549062204056152902924813039655647442366945921
57536889399052980756296286674968903855409045308143102611163019809571142018059
12929802887923461535807982091468405059138427896837702468444682049234483850013
77059670569639320089842324850040612342341303479453334918963112460638687646659
08369312889053995280922084157488759177250559310275929015078988
PS C:\Users[redacted]实验2-CRT>
```

4. 输入读取“4. txt”

```python
f = open('4.txt', 'r', encoding='utf-8')
data = f.readlines()
list_a = list(map(int, data[0:3]))
list_m = list(map(int, data[3:6]))
print(get_solution(list_m, list_a))
```

问题 输出 JUPYTER 调试控制台 终端

```
PS C:\Users[redacted]\实验2-CRT> python -u "c:\User
[redacted]"
10153882543867815937975005418668511181436090462627442861291176921223760739632
94528382307685354119625128010259181855309635136705018838077399500873264394222
34056849134588419504883168513358038497261774225421167769990772503690896660517
79287240447020770762696800692054325713691701080093703777904568535833526445158
78311516190507250766900630096275519832176069330252347804331157162095788829559
12399723578372074554608159007517412199578165136149952499647819790368856740917
65995102823644464757353004933734286925521933368613392620366429379254626113228
92884241784944506543412683335752742674568827747693117411398350421786249483956
02574714541224989084463901454592991080554250885785630684688113064400953807650
79613345980842267812712805662914955292478650989024941774191383142487396091618
57129237248001175976898365202438388017354390637118765454475683538827838436505
2966233903025221656744837529847014046943899058813334
PS C:\Users[redacted]\实验2-CRT>
```

五、思考与总结

1. 求一次同余方程$\begin{cases} x \equiv a_1 \ mod\ m_1 \\ x \equiv a_2 \ mod\ m_2 \\ \cdots \\ x \equiv a_k \ mod\ m_k \end{cases}$组的解，若正整数$\boldsymbol{m_1}$，$\boldsymbol{m_2}$，…，$\boldsymbol{m_k}$不是两两互素，是否能直接用中国剩余定理求解？例如方程组 $\begin{cases} 7x \equiv 5 \ mod\ 18 \\ 13x \equiv 2 \ mod\ 15 \end{cases}$，需要如何求解？

答：不能直接求解，需要先进行变形：

$$7x \equiv 5(mod\ 18) \ {-}{-}{>} \begin{cases} x \equiv 1(mod\ 2) \\ 7x \equiv 5(mod\ 9) \end{cases}$$

$$13x \equiv 2(mod\ 15) \ {-}{-}{>} \begin{cases} x \equiv 2(mod\ 3) \\ 3x \equiv 2(mod\ 5) \end{cases}$$

综合可得：$\begin{cases} x \equiv 1(mod\ 2) \\ x \equiv 4(mod\ 5) \\ x \equiv 2(mod\ 9) \end{cases}$

运用中国剩余定理，得：

$$\begin{cases} M_1 = 45, M_1^{-1} \equiv \ 1\ mod\ 2 \\ M_2 = 18, M_2^{-1} \equiv \ 2\ mod\ 5 \\ M_3 = 10, M_3^{-1} \equiv \ 1\ mod\ 9 \end{cases}$$

则$x \equiv 45 \times 1 \times 1 + 18 \times 2 \times 4 + 10 \times 1 \times 2(mod\ 2 \times 5 \times 9) = 45 + 144 + 20(mod\ 90) \ \equiv 29(mod\ 90)$，求解完毕。

2. 实验过程中还遇到了什么问题，如何解决的？通过该实验有何收获？

a.　不熟悉同余运算。由于没有学过信息安全数学基础，对于同余方程组的性质不是很了解，在手动进行同余运算时频繁出错，对算法步骤理解不清。

b. 测试代码时在Python中反复输入数据，过于麻烦。通过学习Python对文件的打开与读函数以及格式转化，使用readlines()函数将文件中的数据读进列表后进行拆分，便于进行计算。

c.　Word中流程图格式问题。排版总是很乱，最后下载Visio并在其中画图后导入Word，解放双手。