**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：2118039**

**姓名：**

**学号：**

**日期：2023.11.25**

一、实验目的

1. 实验环境：
   1. Windows11
   2. Python3.12
2. 实现目标：

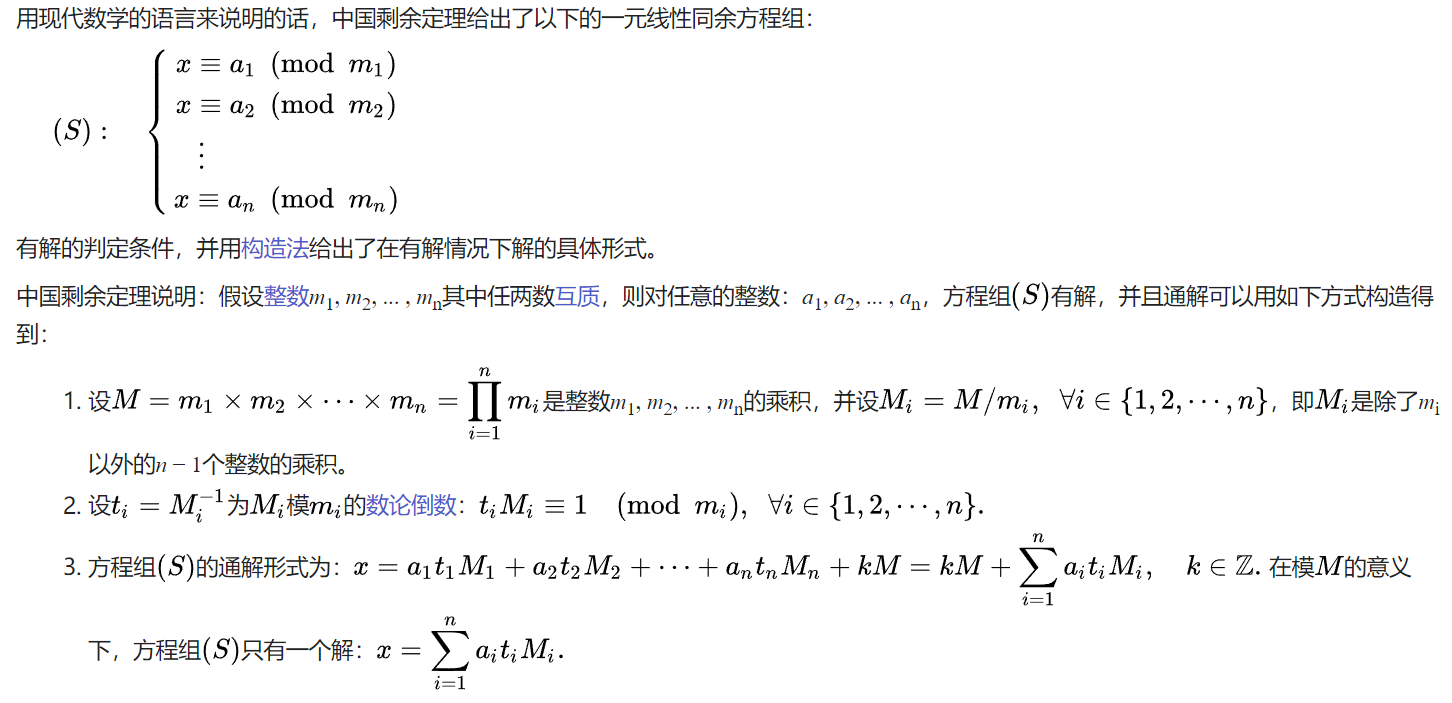
编程实现利用中国剩余定理求解一次同余方程组的解。

二、方案设计

1. 背景：

中国剩余定理，又称孙子定理或中国余数定理，是数论中的一个关于一元线性同余方程组的定理，说明了一元线性同余方程组有解的准则以及求解方法。

1. 原理



1. 算法步骤
   1. 判断正整数m1,m2,……,mk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”；
   2. 计算m=m1m2……mk, Mj = m/mj;
   3. 计算Mj-1(mod mj)
   4. 计算xj MjMj-1aj(mod m)
   5. 计算x xj(mod m)

三、方案实现

1. 算法流程图

图示

描述已自动生成

1. 主要函数的介绍
   1. 求最大公约数：

屏幕上有字

描述已自动生成

* 1. 判断是否两两否互质

文本

描述已自动生成

* 1. 求模m的a的逆元

手机屏幕的截图

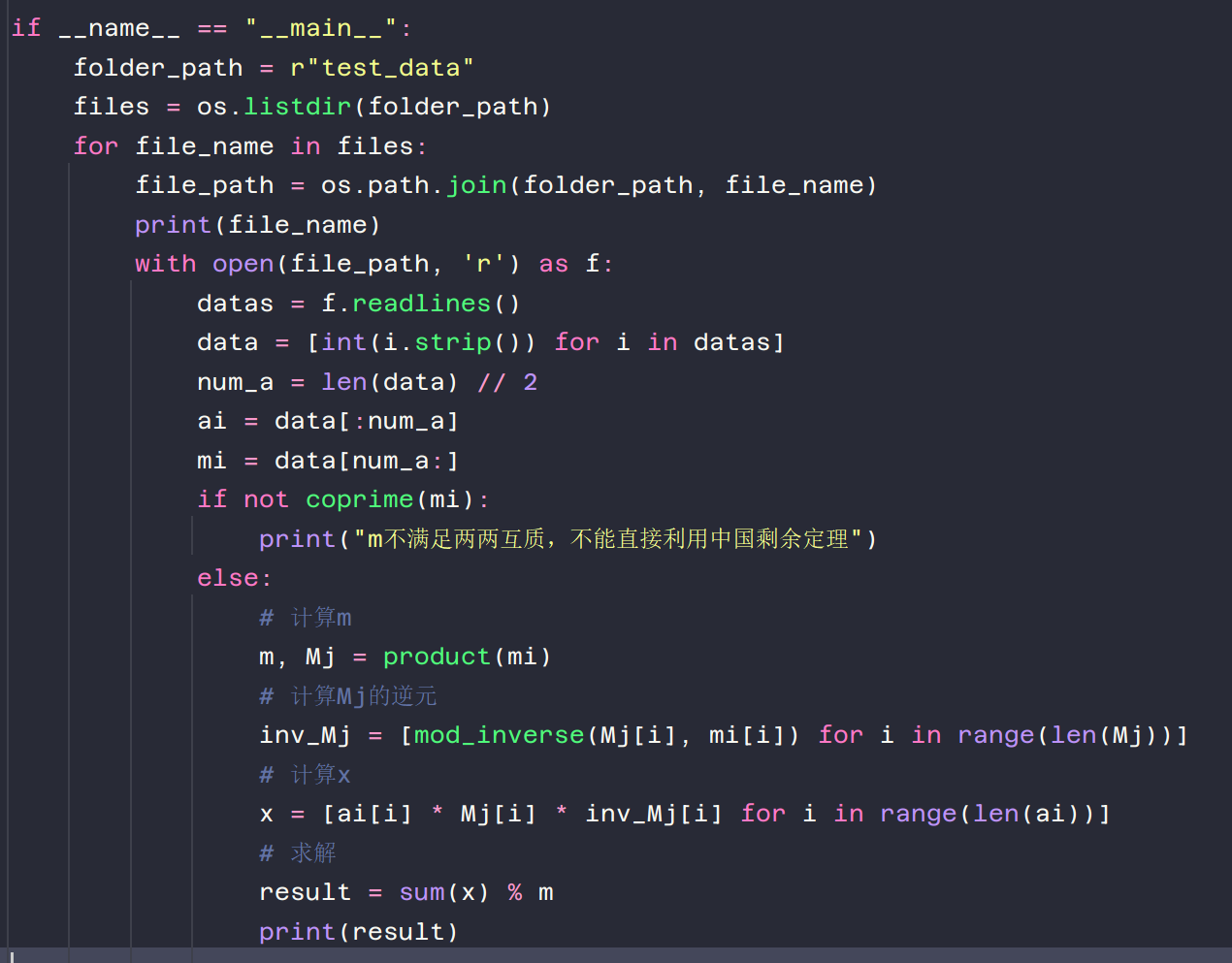
描述已自动生成

* 1. 计算乘积，包含m和Mj

文本

描述已自动生成

* 1. 主函数，进行文件的读取和求解



1. 算法实现的主要代码

import os  
  
  
def gcd(*a*, *b*):  
 while *b*:  
 a, b = *b*, *a* % *b* return *a*def coprime(*x*):  
 for i in range(len(*x*)):  
 for j in range(i + 1, len(*x*)):  
 if gcd(*x*[i], *x*[j]) != 1:  
 return False  
 return True  
  
  
def mod\_inverse(*a*, *m*):  
 return pow(*a*, -1, *m*)  
  
  
def product(*x*):  
 result = 1  
 M = []  
 for i in *x*:  
 result \*= i  
 for i in range(len(*x*)):  
 temp = 1  
 for j in range(len(*x*)):  
 if i != j:  
 temp \*= *x*[j]  
 M.append(temp)  
 return result, M  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 folder\_path = r"test\_data"  
 files = os.listdir(folder\_path)  
 for file\_name in files:  
 file\_path = os.path.join(folder\_path, file\_name)  
 print(file\_name)  
 with open(file\_path, 'r') as f:  
 datas = f.readlines()  
 data = [int(i.strip()) for i in datas]  
 num\_a = len(data) // 2  
 ai = data[:num\_a]  
 mi = data[num\_a:]  
 if not coprime(mi):  
 print("m不满足两两互质，不能直接利用中国剩余定理")  
 else:  
 # 计算m  
 m, Mj = product(mi)  
 # 计算Mj的逆元  
 inv\_Mj = [mod\_inverse(Mj[i], mi[i]) for i in range(len(Mj))]  
 # 计算x  
 x = [ai[i] \* Mj[i] \* inv\_Mj[i] for i in range(len(ai))]  
 # 求解  
 result = sum(x) % m  
 print("x ≡ {} \nmod {}".format(result, m))

四、数据分析

电脑萤幕的截图

描述已自动生成

5.txt

5.txt

m不满足两两互质，不能直接利用中国剩余定理

6.txt

m不满足两两互质，不能直接利用中国剩余定理

7.txt

x ≡ 10632493914901792690327911096391815563767912989793771534274098749291522857005262301857540196215274100328919290471839190515388637264165372946660696514542244925527237924055084686375854183216455018682135450505851147307598160556215676682267643202026043195127938130183670427563910447574712299957354288222213067487326731735783244241070904904979688306814928997242678815001095425413806773539756466990369936070189776393809968384661247883116940518014243387802799599418584622394999173360287324260822716568502553196932684386891825531584317163616535000465391339743819974631195488159315327175269972819224655159622141620775179929624920415987207552656501748927747012480083870729307292480776060048765401863109165123630071060491170503808880323816497762331524278791689163245539218653717481115015118978071962624450398822999266405413073010150959805105029844248726692222465347663417088015738354375044879961853321699872419

mod 11648688456152601308621618357407287075158652544392004591607616935924699060389752590864729891827925421872581048102633894609733753007618581653895488574719854650701094872520511683762575738065786662275808940136150180528795535318479605310517456259936835269482137102795380970323377065419488634734841527579428538878534535903267618943094739627156819120205736421350842235034797483529712636640678517439631434564837143687930205168292497576668659810873224794714884734246227965351760679026459378836208157102609533128640180641133772996626179142534983434439839601890396974979471547315581048517719872319279942914371704919363722854211506255187369456685047155987548931658092335315960109499191639756503465011245810848767883888355188342878138717181555965470573915718356671334033443211518078508732149467536522149524741901972897830167138630616708695817659138128335693629167168975836846044941218230486674325391835599068503

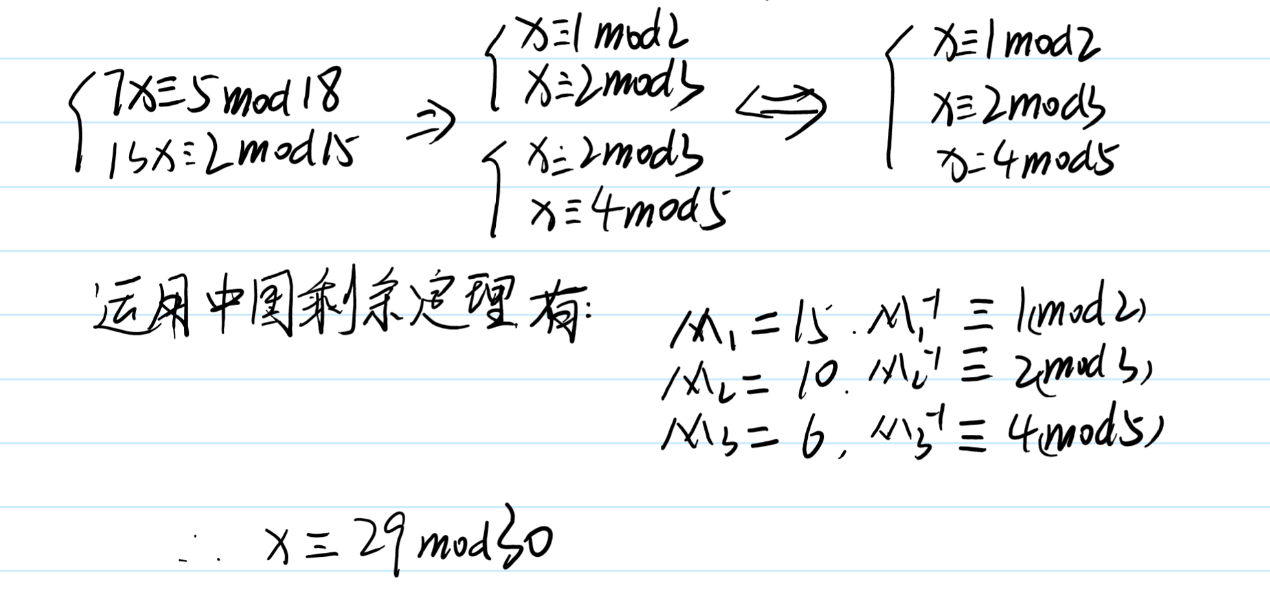
8.txt

x ≡ 14755452957737139265654242910528893109631194116487435180445089160957067162740176124220975820197723806870710862748092411886010979264836075732250607721863725728551741267702384055180763902835517945510105998781774435296914858483349905617704917401172906601399919224256175523406826473918574690575708876326958138743676371080398665374700323263843422916317788639126792411951155802621463404631106486062366317103866092693215555707110096550958572849241639109586817394897221504168228180356055670574606942445104578257955019737268274727890972429339934981545675146330440090630006735344118780372639767356463818749411819915862884226114256069825419185677271314052604663902157857283452160929063159271268310438430542621981250244236447251263312164016882221607545182817804863112778905560420584538275897151346940004501558333606368596908004755637933629709277395928202550794740305129702112664376555379064917005816111576394316

mod 20727523227224053913795641275227637216903778358063234047757745358227129696977582032250687951735421964100453599828167662716366040650768628520756498019327350144687698086924996951534757593783058318349849852081586297555179884297992592714883906495081446090592312651098882236402082355770881483792653592144970734991027766125378815268745784769808256765734426850157979152695831389257414659204133522105197119586839756612574185906568946177725190129258556874943203145809400369156773876721343900610875469535044528518278544048484437974686276816901482569343238146133599394889560721519749721607111636446629626893065312886415102936688649783558883313302570522164865327090760426954607075132837653142524124787468317463640820936528369392582335744313029665663460966406896618193646341026666356473879108773039746595130655164881063960177546079262615920787557616592322721798349645540221216691419073821213345234723223841335003

五、思考与总结

1. 求一次同余方程组的解，若正整数𝒎𝟏，𝒎𝟐，…，𝒎𝒌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，需要如何求解？

不能。需要先将其转换为中国剩余定理所需要的形式，即x前面系数为一，正整数𝒎𝟏，𝒎𝟐，…，𝒎𝒌两两互素，具体过程如下：

使用程序验证，是正确的：

电脑萤幕的截图

描述已自动生成

1. 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？

文本

描述已自动生成

Python的除法操作溢出了，转换为乘法，即在计算m的过程中算出Mj来，而不是通过m/mj得到。

