**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：2022年11月6日星期日**

一、实验目的（包括实验环境、实现目标等等）

实验环境

Windows11 , Python 3.9.12

二、方案设计

（包括背景、原理、必要的公式、图表、算法步骤等等）

实验背景

在《孙子算经》中有这样一个问题：“今有物不知其数，三三数之剩二（除以3余2），五五数之剩三（除以5余3），七七数之剩二（除以7余2），问物几何？”这个问题称为“孙子问题”，该问题的一般解法国际上称为“中国剩余定理”

即求满足以下条件的整数：除以3余2，除以5余3，除以7余2。

实验原理

中国剩余定理

若正整数  两两互素，对 的同余式组为



有整数解。并且在 且



的情况下有唯一解，解为



其中，而为模的逆元

算法步骤

1) 计算所有模数的积;

2) 对于第个方程:

a. 计算;

b. 计算在模意义下的逆元;

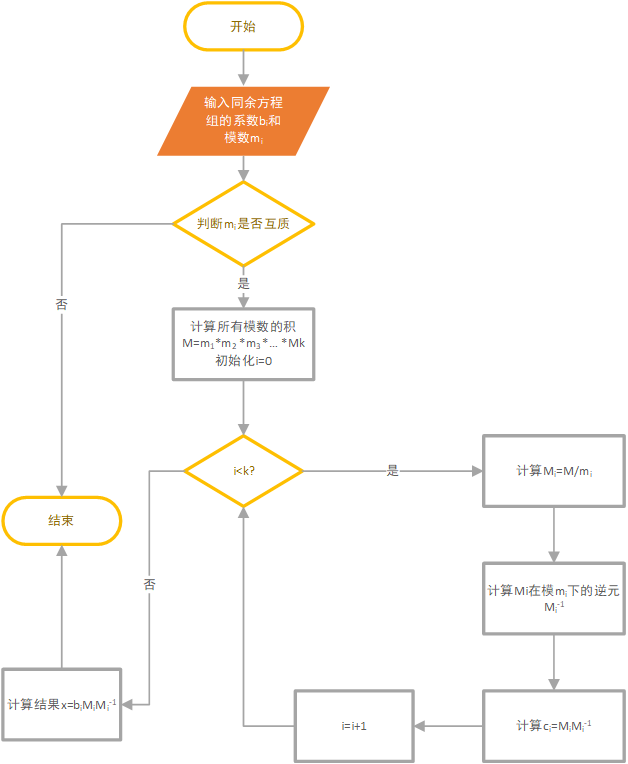
c. 计算。

3) 方程组在模意义下的唯一解为: 

三、方案实现

（包括算法流程图、主要函数的介绍、算法实现的主要代码等等）

算法流程图



代码

1. import gmpy2
2. *#中国剩余定理*
3. base1=[6280258886892748634510684834296247689424750823228354964160873376765560927915079545499575080926643252043513075581338938780210530201651149065459075478297277788632711131392808177265297439703089763909138271911063920511008763414512745717273222737303967477,
4. 2736455869987068844584034882444748612965398275988826044932737248592878515276968779821471672366818284147603550093051571475753154039709010825232379058045334742208275432609121820269662549238034880323083843369151525570918813848018286858914177008224217206,
5. 8665126284864072440947476564232984766333105693011561831898491426661651434340473891198905931381862265805424160730440725550800572483330354664557524172011914157644900357295071162702239552997535972800970708461746430958035283032573132400329945780527723417,
6. 609916182836072484510697159537216800485064532931203818192093495124315918904439014622612727220476235725126553345875741633730843866349537244461426321686087680899484019494078852071610347391771997488392342853457097016366707737987999959109863373656950889794421097435800768714724065649675835948070598583722,
7. 224336317877529001149272849982106490385645395876269500102785121650693648627855012654138654451845478851221868980007235885372503912039900653542541783164037171051747566455191972487984952149112992774981659893762066991494739368109862018399264488871741275064433039072393043564109001911012944866948999832660,
8. 407126738349599334657470858931171583281602505536465495530791442859387960980187708107744781025700247140792608003156364489978288333356567794926123496028029668907918489320256438631771516315239465357990150358440476827564338763951213110570335391415231761148217699293807010375389912306536017206927462681523]
9. base2=[1703400778910483820422805008255510202952780108580687282406204991102977355497819362037336119731002913846662180847077020740236290693493738835482271882051497287782803150523222405080338334271801792866482004568188646373815031989672512987923656390452397739,
10. 2724305591776277437698018538336328525368755596343722369278379045862095348399405909803531445622923586434858075923794289199772931203020662107251938261421134535489303223863021432862007558144614961478557772158790865755237609603854734189485065702894533007,
11. 5054062422109780731533257906013120348277952566924151532670827347964065027024283070553070224352222147708139041781535542580186174743209933967784491580267630649415317930613025621277091068103021482556687957739478553608686399952265973926562315865941811106,
12. 529012089683106027540972464242844528071743673743373492337127252712791226562406546874415951553948390892712915241165589135143177845694531817071616643290537045923226008758702728170159550568805907850622424020015265726841001559668865110779486144192282420641364473883446730980837401121533551908678272444684,
13. 543215757827618377203473258275047104979822158102404878687148537849799550247381564365578590422056394278257158069356234349542268988578552880243969652244111766224512688426426706878276641902193985933655384799228976467016700686667749302166453794272633505983640121880738193066243430412939374407346361018673,
14. 910409201715205002347478646685130446423660152260507358434979331439581396068284694915012061028789908792252336550127518559297299352963181057834931040933258962452367236215393076418627949218921266472636232832254258117915197059091231844804196176368468920201615703783443596364327788081668519651744410101292]
15. base3=[5963533518323759992554607177430867358486851282034825468613195536233474229748227497105997065615017822867937430259877578637549250133081396303549824197515580096551093277283532310046598789715143661789924175678686515394584704002698912400360544349584105053,
16. 7452147498938949476823918277259388511208314795517506630131639522869476931664319331999287065159813243159220658294359000834379144325299074839188183145751865959915395944123238452146817998590504704409144177253981983496408422851304571062742872070053597781,
17. 9791553188843350711911320803153869584042064903057744938846414791049993340267839296583573667229339667423378978993973059594193467223787912526408229270166113282331353592137037653549309808543886438135335025330344634355726311631692287157836010187549697033,
18. 736026894520013145053959893041907103801786277875074808043094679021048591904886572241063599195186281186045410287408340438501951106621035951386659621670588327954137474402459141720940937736412438057497469266151389553294232028234435749687861420081325499806705696003250405278239467097419352576463355243047,
19. 822133543134400325816236583808204657546644575118717165226138158817090803490095076646561687767071855511032867989547540228102384603044671232082169916159817973775601566972952016094980712601001651606854901098178067592170128901984368176059683470221479398450343749445851362633660071533865039629449293643181,
20. 666862214073229347457363746613972922639274081936663778780666180616990715201034304434823629318786074033107938413965200245547739283841506412360219950769580551911651386910043030289964724519173587731635947680115173597573559458235634593301646388624380695138506650044135228891892580966398311773634400428043]
22. base4=[8695039453824585971327803091640327439264591005655488011745393293145693657952152999306407259261770227769126705900977747911851437578989508123510979745347100821709337665424897266934977753636498274478498325810960608950295083895917256121194946489862590531,
23. 2843586888776659243226129768596946248650199985912875849363349934062940237054802099741704753655572936241294947678141581242450053030988467105995226180947812728014932325811696214659355368088498823535203466513124053882630083938669706033914334272897725827,
24. 7823172232329967143253080336216473133559059471949452135263320655253223177876623709115617602222829735561588048570850030793579858516821809715240795109686396100253463538575629486987673937099056108251105593664390258183737270501095307514377495715221620851,
25. 116017853623276268534139398907929878877787793651980551492831128460152265817955210396589291631422367361262415115121837412178585592287169858210020969161771830842579745360462792019608011639872984330943977063269966943474772865114131255717447905531474229802532044068660360456893054575436566915238490309883,
26. 512447157769228304131256323740835904819759921673225154827094436771226096325013237418054628602994336075764089916228418424306832707597853204209771112842072708289338794604081011427030629039302332957025365167149233660135959038245478352895374943727331531173885009715832998171386546136367093686989913049831,
27. 969850455230698192652261474515788687418941545125366868561633605148672698351493104516166153304063487537678627945130392397105464048856577407056079471349982831692582998623711436409450154130174976181809303563157487873504380599234293660083773732553256060917095555110841495582128346683005794423459984157739
28. ]
29. def chinese\_remainder(data):
30. *#判断是否互质*
31. for i in range(len(data)):
32. for j in range(i+1,len(data)):
33. if gmpy2.gcd(data[i][1],data[j][1])!=1:
34. return -1,-1
35. #计算n的乘积
36. N=1
37. for a,m in data:
38. N\*=m
39. *#计算Ni*
40. Ni=[]
41. for a,m in data:
42. Ni.append(N//m)
43. *#计算Ni的逆元*
44. Ni\_inv=[]
45. for i in range(len(Ni)):
46. Ni\_inv.append(gmpy2.invert(Ni[i],data[i][1]))
47. *#计算x*
48. x=0
49. for i in range(len(Ni)):
50. x+=data[i][0]\*Ni[i]\*Ni\_inv[i]
51. x=x%N
52. return N,x
53. def crt\_data(base):
54. data=[]
55. *#重构为a\_i mod m\_i 数据*
56. for i in range((len(base)//2)):
57. data.append((base[i],base[len(base4)//2+i]))
58. n,x=chinese\_remainder(data)
59. if n==-1 or x==-1:
60. print("not coprime")
61. else:
62. print(str(x)+'\n'+"mod "+str(n))
63. if \_\_name\_\_ == '\_\_main\_\_':
64. basename="base"
65. for i in range(1,5):
66. print("base"+str(i)+":")
67. exec("crt\_data(base"+str(i)+")")

主要函数介绍

1).crtdata()函数负责数据的清洗输入

2). chinese\_remainder()是中国剩余定理的主干

四、数据分析(包括算法测试数据的分析，运行结果截图等等)

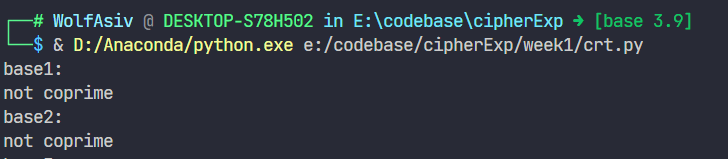
测试数据分析

数据1

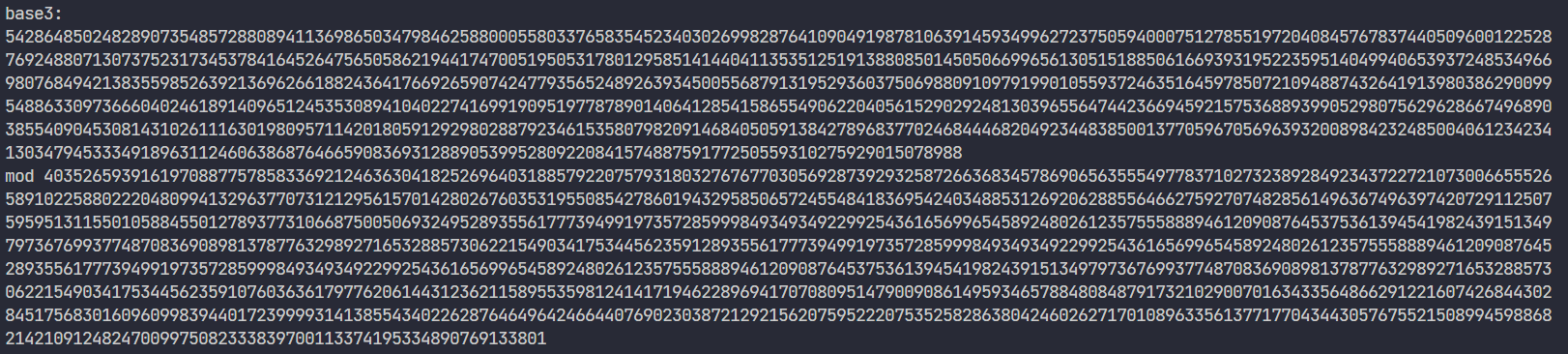
不互素

数据2

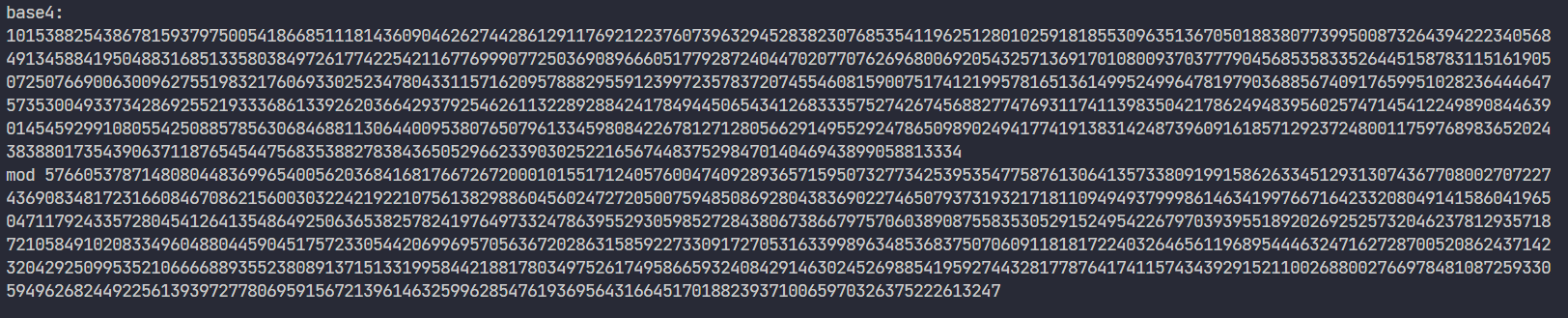
不互素



数据3



数据4



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？

答: 若不是两两互素，不能直接用中国剩余定理求解，我这里用扩展的中国剩余定理



等价于



移一下项



由于则该同余式有解

令 显然p1,p2是互质的

因此上式可变为，通过扩展的欧几里得求解可得



于是

【定理】若有特解 , 那么  的通解是:  , 亦即



因此通解为

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？

答:数据输入时搞错了a,m的位置,通过该实验我理解了中国剩余定理在求解同余方程组的优势和局限，并扩展了该定理。