

《信息安全基础综合实验》课程实验报告

实验题目：中国剩余定理

班级：1718039 学号 1：17189110002 姓名 1：祝佳磊

班级：1718039 学号 2：17180210027 姓名 2：李欣

一、实验目的

实验环境：

1. Visual Studio 2017

2. miracl 库

实现目标：使用 miracl 库对通过中国剩余定理计算基于大数的同余方程

二、方案设计

背景：中国剩余定义最早出现在《孙子算经》中，1247 年在秦九韶的《数书九章》给出了一次同余方程组的一般性解法——大衍求一术

原理：

定理 中国剩余定理

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

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

在模 m 意义下有唯一解，该解可表示为

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

其中 $m = m_1 m_2 \dots m_k$ ， $M_j = m / m_j$ ， $M_j M_j^{-1} \equiv 1 \pmod{m_j}$ ， $j = 1, 2, \dots, k$ 。

算法步骤：

求解一次同余方程组 $\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_k \pmod{m_k} \end{cases}$ **的解。**

- (1) 判断正整数 m_1, m_2, \dots, 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”
- (2) 计算 $M_j^{-1} \pmod{m_j}$
- (3) 计算 $x_j \equiv M_j M_j^{-1} a_j \pmod{m}$
- (4) 计算 $x \equiv \sum_{j=1}^k x_j \pmod{m}$ 。

三、方案实现

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include "miracl.h"
4  #include <windows.h>
5
6  #define SIZE 50
7
8  int main() {
9      FILE *fp;
10     big a[SIZE], m[SIZE], t[SIZE], x[SIZE];
11     big Xul, Mul, GCD, Mi, Mj, Mm, re;
12     int i, k, b, j, flag;
13
14     miracl *mip = mirsys(10000, 10); //500位十进制空间
15     mip->IOBASE = 10;
16     Xul = mirvar(0);
17     Mul = mirvar(1);
18     GCD = mirvar(0);
19     Mi = mirvar(1);
20     Mj = mirvar(1);
21     Mm = mirvar(0);
22     re = mirvar(0);
23
24     if ((fp = fopen("1.txt", "r")) == NULL) {
25         printf("打开文件失败...\n");
26         return -1;
27     }
28     //将文件内的数据读入数组
29     for (b = 0; !feof(fp); b++) {
30         t[b] = mirvar(0); //先初始化
31         cinnum(t[b], fp); //读入数据
32     }
33     //再分别将数据放入a和m数组，前一半a，后一半m
34     k = (b - 1) / 2;
35     printf("a数组:\n");
36     for (i = 0; i < k; i++) {
37         a[i] = mirvar(0);
38         a[i] = t[i];
39         cotnum(t[i], stdout);
40         x[i] = mirvar(1); //顺便初始化
41     }
42     printf("m数组:\n");
43     for (i = k; i < b - 1; i++) {
44         m[i - k] = mirvar(0);
45         m[i - k] = t[i];
46         cotnum(t[i], stdout);
47     }
48 }
```

这部分代码的主要作用是导入 miracl 库，初始化变量，并从测试文件中读取测试数据并将其按行放入数组中，以便后续进行计算

```
48
49 //比较m数组的数据，判断是否达到中国剩余定理的条件（两两互素）
50 flag = 1;
51 for (i = 0; i < k; i++) {
52     for (j = 0; j < k; j++) {
53         if (i == j) {
54             continue;
55         }
56         else {
57             egcd(m[i], m[j], GCD); //计算最大公因数
58             if (mr_compare(GCD, mirvar(1))) { //不为1跳出
59                 flag = 0;
60                 break;
61             }
62         }
63     }
64     if (!flag) {
65         break;
66     }
67 }
68
69 if (!flag) {
70     printf("不能直接利用中国剩余定理...");
71 }
```

这部分的代码主要是从 m 数组中将各个 m 提取出来，然后通过 miracl 库中的

egcd 函数去计算它们之间的最大公因数，比较最大公因数是否为 1，如果不为 1，说明 m 之间不是两两互素的，无法使用中国剩余定理。

```
72     else {
73         for (i = 0; i < k; i++) {
74             multiply(Mul, m[i], Mul); //计算Mul=M1*M2*...*Mn
75         }
76         for (i = 0; i < k; i++)
77         {
78             fdiv(Mul, m[i], Mi); //Mi=Mul/m[i]
79             xgcd(Mi, m[i], Mj, Mj, Mj); //Mj=invers(Mi,m[i]); 即求Mi的模逆 (Example: xgcd(x,p,x,x,x); //x=x^-1 mod p)
80             //printf("/nMj:");
81             //cotnum(Mj, stdout);
82             multiply(Mi, Mj, Mn); //Mn=Mi*Mj
83             multiply(Mn, a[i], x[i]); //x[i]=Mi*Mj*a[i]
84         }
85         for (i = 0; i < k; i++)
86         {
87             add(Xul, x[i], Xul); //累加
88         }
89
90         powmod(Xul, mirvar(1), Mul, re); //re=Xul^1 mod Mul; 即re=Xul%Mul
91         printf("结果为:\n");
92         cotnum(re, stdout);
93     }
94
95     mirkill(Xul);
96     mirkill(Mul);
97     mirkill(GCD);
98     mirkill(Mi);
99     mirkill(Mj);
100    mirkill(Mn);
101    mirkill(re);
102
103    system("pause");
104    return 0;
105 }
```

如果 m 之间两两互素，那么就使用中国剩余定理来计算 x。首先先通过 multiply 累乘函数，计算所有 m 的乘积，存入 Mul。

然后计算 Mul 除 m[i] 的值，存入 Mi 中。并通过 xgcd 函数去计算 Mi 的模逆，存入 Mj 中，然后计算 $x[i] = Mi * Mj * a[i]$ 。

计算出所有的 x[i] 后通过 add 累加函数去累加 x[i]，存入 Xul 中。

最后求 Xul 模 Mul 的值，得出 x。

四、数据分析

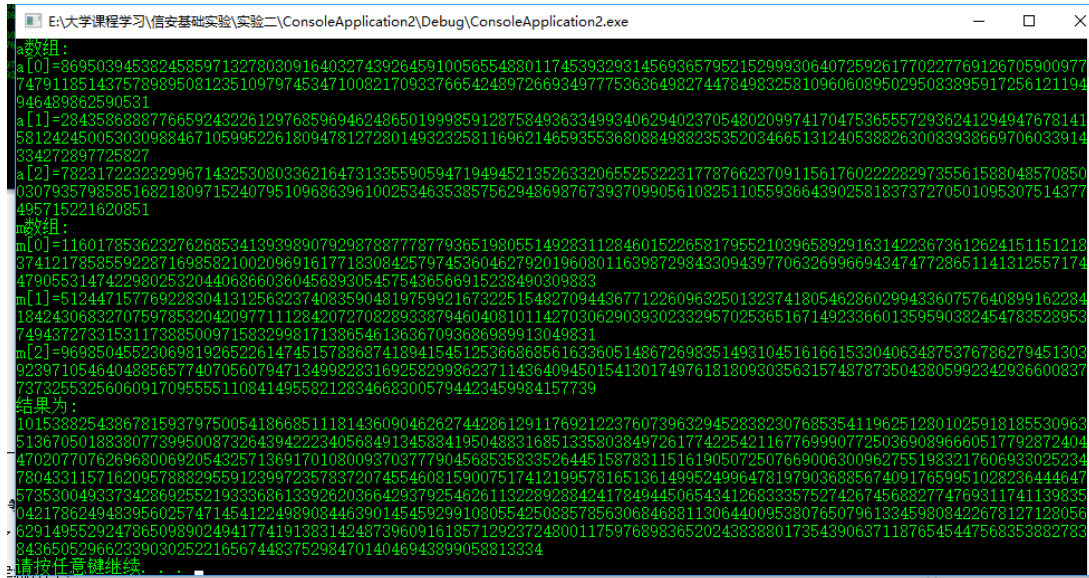
测试数据 1:



```
E:\大学课程学习\信息安全实验\实验二\ConsoleApplication2\Debug\ConsoleApplication2.exe
a数组:
a[0]=6813235941703376655326877259456765345944674477348694400431478065190540398329898045051943099710064607893291238536350
803906521190043509681107373431914896534849472572018314246368059514481985974756802602917677211932317480314364208827846554
2049176622489843
a[1]=5329358793645372171872844022646807978923298258962562871017555004646749332435126785512091799731320284332783330569218
3043442356465418663948758694034454210226722014741853068963165828428561626632437848851704858856750515863500344238617281204
9424103514951577
a[2]=2257909091213371927783887882236153443884213794425001522554955003226761884415214074620836856719371739848873415048006
100984173237519321380321235233453717683579510056340621287447538012660312045598794033165173102568393996326193272735393676
116019520160015
m数组:
m[0]=8641988732614118864892019630290497813378990961597339594544995095638205033584800830271773019567574600821044880021654
934294588343316661657506440221435209848067394394902170258149768583688124546247286239819499461862058920334935440097853077
51964072678853661013191733761971231979915702569701099292470225565
m[1]=4640470254946816235194211631305513024345105242420762140473286171511773428338328590710957609119666054365023728160756
535330446745807317758113576136851344158795067158447683047846760676699683924945721873403893935769032345501769934332682109
741276243390105081895017358893813467661897761825666141343401805201
m[2]=1664370259470810900723808123724405799940690401133490086155458193133490950842602990170987972879355537756069568692574
915732850554623872126380236609025890998777211918286779260373433013849130377510838502241832882928290316153290015598738671
92108036844243283614392218194941020705192773964213850946909270666
不能直接利用中国剩余定理...请任意键继续...
```

可以看出，由于 m 不满足两两互素的条件，所以我们不能直接使用中国剩余定理。

测试数据 14:



```
EA大学课程学习\信安基础实验\实验二\ConsoleApplication2\Debug\ConsoleApplication2.exe
a数组:
a[0]=8695039453824585971327803091640327439264591005655488011745393293145693657952152999306407259261770227769126705900977
74791185143757898950812351097974534710082170933766542489726693497753636498274478498325810960608950295083895917256121194
946489862590531
a[1]=284358688776659243226129768596946248650199985912875849363349934062940237054802099741704753655572936241294947678141
581242450053030988467105995226180947812728014932325811696214659355368088498823535203466513124053882630083933669706033914
334272897725827
a[2]=7823172232329967143253080336216473133559059471949452135263320655253223177876623709115617602222829735561588048570850
930793579858516821809715240795109686396100253463538575629486987673937099056108251105593664390258183737270501095307514377
495715221620851
m数组:
m[0]=1160178536232762685341393989079298788777877936519805514928311284601522658179552103965892916314223673612624151151218
374121785855922871698582100209691617718308425797453604627920196080116398729843309439770632699669434747728651141312557174
47905531474229802532044068660360456893054575436566915238490309883
m[1]=5124471577692283041312563237408359048197599216732251548270944367712260963250132374180546286029943360757640899162284
184243068327075978532042097711128420727082893387946040810114270306290393023329570253651671492336601359590382454783528953
74943727331531173885009715832998171386546136367093686989913049831
m[2]=9698504552306981926522614745157836874189415451253668685616336051486726983514931045161661533040634875376786279451303
923971054640483565774070560794713499828316925829986237114364094501541301749761818093035631574878735043805992342936600837
7373255325606091709555110841495582128346683005794423459984157739
结果:
101538825438678159379750054186685111814360904626274428612911769212237607396329452838230768535411962512801025918185530963
513670501883807739950087326439422234056849134588419504883168513358038497261774225421167769990772503690896660517792872404
470207707626968006920543257136917010800937037779045685358335264451587831151619050725076690063009627551983217606933025234
780433115716209578882955912399723578372074554608159007517412199578165136149952499647819790368856740917659951028236444647
573530049337342869255219333686133926203664293792546261132289288424178494450654341268333575274267456882774769311741139835
942178624948395602574714541224989084463901454592991080554250885795630684683113064400953807650796133459808422678127128056
629149552924786509890249417741913831424873960916185712923724800117597689836520243838801735439063711876545447568353882783
84365052966233903025221656744837529847014046943899058813334
请按任意键继续...
```

我们使用中国剩余定理成功计算出了 x 的值

五、总结

一开始，我们用 1.txt 去测试程序的时候一切顺利，判断出了 1.txt 中的方程组无法使用中国剩余定理去计算。但是在测试 14.txt 的时候报出了溢出的错误。经过排查，我们发现了导致问题的点

```
73 for (i = 0; i < k; i++) {
74     multiply(Mul, m[i], Mul); //计算Mul=M1*M2*...*Mn
75 }
```

在这里计算所有 m 的累乘结果，结果超过了我们设置的空间

```
14 miracl *mip = mirsys(500, 10); //500位十进制空间
```

一开始我们只设置了 500 位十进制的空间，后来我们将空间设置为 10000 位解决了问题。最后测试了一下大概 1500 位十进制空间就可以了。