

密码学实验实验报告一

18374480-黄翔

2021 年 3 月 15 日

1 实验目的

1. 通过本次实验, 了解 $GF(2^n)$ 上元素的性质及其四则运算
2. 掌握 $GF(2^n)$ 上的本原多项式的判定和生成算法

2 实验环境

python 3.9.1+

3 实验内容

3.1 四则运算

3.1.1 算法流程

$GF(2^8)$ 加减法 $GF(2^8)$ 上的加法与减法流程一致 如表 1 可见, 结果符合

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

表 1: 按位加减

异或运算 \oplus 。只需按位异或即可得到结果

$GF(2^8)$ **乘法** 模拟十进制乘法, 对 $a \times b$, 将 b 的每一个非零位乘 a , 结果相加即可。由于选择 $GF(2^8)$ 本原多项式选择为 $x^8 + x^4 + x^3 + x^1 + x^0$, 当 $a \ll i$ 出现 x^8 项时, 只需将 $+x^8$ 改为 $+x^4 + x^3 + x^1 + x^0$ 即可

$GF(2)$ **带余除法** 模拟长除法, 对 $\frac{a}{b}$, 当 a 最高位不小于 b 最高位, b 左移相差位后被 a 减, 结果作为 a 重复上述操作。最后最高位小于 b 的 a 为余数, 每次差异位组成商

3.1.2 算法伪代码

Algorithm 1 $GF(2^8)$ 上加、减法

Input: a, b

Output: $s = a + b$

1: **return** $a \oplus b$

Algorithm 2 $GF(2^8)$ 上乘法

Input: a, b

Output: $s = a \times b$

```

1:  $s \leftarrow 0$ 
2: for  $i \leftarrow 0$  to 7 do
3:   if  $b \& 1$  is 1 then
4:      $s \leftarrow s \oplus a$ 
5:   end if
6:    $Mark \leftarrow a \& 80$ 
7:    $a \leftarrow (a \ll 1) \& ff$ 
8:   if  $Mark$  is 80 then
9:      $a \leftarrow a \oplus 1b$ 
10:  end if
11: end for
12: return  $s$ 
```

Algorithm 3 $GF(2)$ 上带余除法

Input: a, b

Output: $q, m : a = q \times b + m$

```

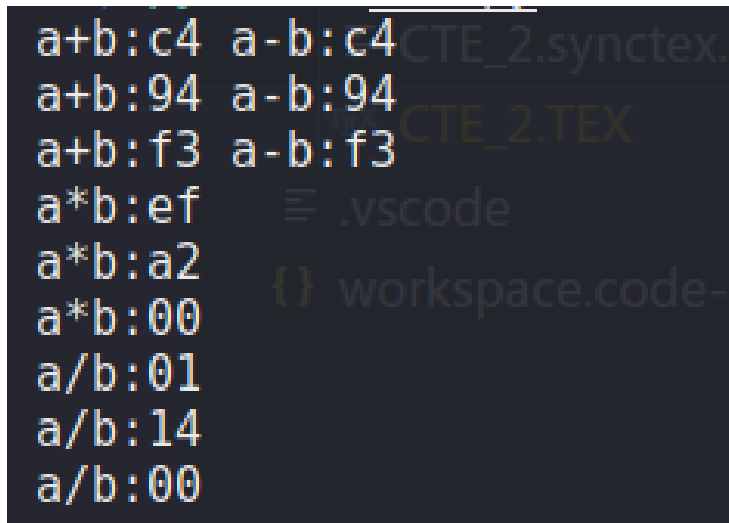
1: function DIV( $a, b$ )
2:   if The degree of  $a <$  the degree of  $b$  then
```

```

3:      return  $q \leftarrow 0, m \leftarrow a$ 
4:  else
5:       $diff \leftarrow a.degree - b.degree$ 
6:       $a \leftarrow a - (b \ll diff)$ 
7:       $q, m \leftarrow DIV(a, b)$ 
8:      return  $q \leftarrow (1 \ll diff)|q, m \leftarrow m$ 
9:  end if
10: end function

```

3.1.3 测试样例及结果截图



```

a+b:c4 a-b:c4
a+b:94 a-b:94
a+b:f3 a-b:f3
a*b:ef
a*b:a2
a*b:00
a/b:01
a/b:14
a/b:00

```

3.1.4 总结

与整数的四则运算相比，二元有限域上的运算有很大的不同，与本原多项式密切相关。例如对于乘法，需要考虑结果超出有限域时模上本原多项式。但是同时，有限域上的乘法以及多项式域的带余除法可以仿造整数乘除进行模拟。

3.2 $GF(2^8)$ 快速模幂运算

3.2.1 算法流程

注意需要使用有限域上运算。初始化 d 为 1，对 n 每次除以 2 的余数，若为 1 则 $d = d^x \bmod m$ ，再整除 2，否则只需整除 2。 $x = x^2 \bmod m$ ，若此时 $n > 0$ ，继续上述流程。

3.2.2 算法伪代码

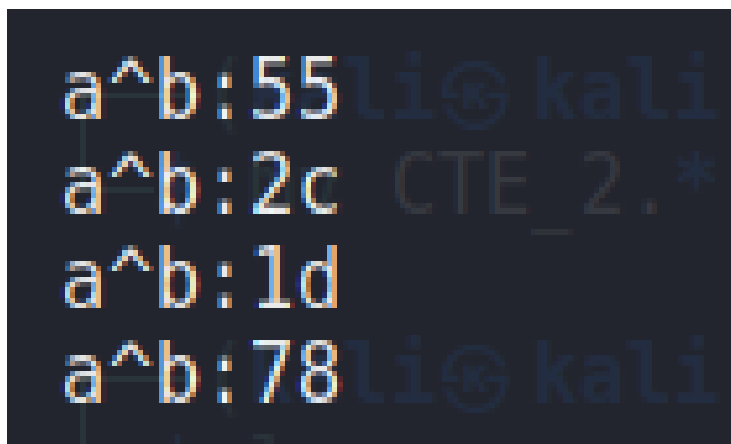
Algorithm 4 快速幂取模

Input: x, n, m

Output: $x^n \bmod m$

```
1:  $d \leftarrow 1$ 
2: while  $n > 0$  do
3:   if  $n \bmod 2$  is 1 then
4:      $d \leftarrow (d^x) \bmod m$ 
5:      $n \leftarrow \frac{n-1}{2}$ 
6:   else
7:      $n \leftarrow \frac{n}{2}$ 
8:   end if
9:    $x \leftarrow (x \times x) \bmod m$ 
10: end while
11: return  $d$ 
```

3.2.3 测试样例及结果截图



3.2.4 总结

算法正确性 对于快速模幂算法，其关键是对幂的二进制展开来做模乘运算，因此对有限域仍成立。

算法复杂度 快速模幂算法时间复杂度为 $O(\log n)$ ，空间复杂度为 $O(1)$

3.3 $GF(2^8)$ 上欧几里得算法

3.3.1 算法流程

注意事项 所有运算都是有限域上的运算

流程 先求 $p = (a, b)$ 。构造解 (x_n, y_n)

$$\begin{cases} x_n = \frac{c}{p} \\ \forall y_n \in Z \end{cases}$$

在递归的回溯过程中，利用公式：

$$\begin{cases} x_{i-1} = & y_i \\ y_{i-1} = x_i - \lfloor a_{i-1}/b_{i-1} \rfloor \times y_i \end{cases}$$

倒推每一组 (a_i, b_i) 的解 (x_i, y_i) 。最后得到 (a, b) 和原方程 $a \times x_0 + b \times y_0 = c$ 的 x_0, y_0

3.3.2 算法伪代码

Algorithm 5 $GF(2^8)$ 上欧几里得算法

Input: a, b

Output: $x = \gcd(a, b)$

```
1: while  $b \neq 0$  do  
2:    $temp \leftarrow b$   
3:    $b \leftarrow a \bmod b$   
4:    $a \leftarrow temp$   
5: end while  
6: return  $x \leftarrow a$ 
```

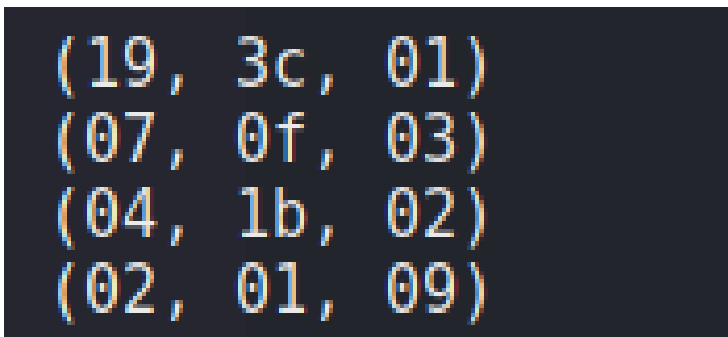
Algorithm 6 $GF(2^8)$ 上拓展欧几里得算法

Input: a, b

Output: $\gcd(a, b), x, y: x \times a + y \times b = \gcd(a, b)$

```
1: function EXGCD( $a, b$ )  
2:   if  $b$  is 0 then  
3:     return  $\gcd(a, b) \leftarrow a, x \leftarrow 1, y \leftarrow 0$   
4:   else  
5:      $d, x_1, y_1 = \text{EXGCD}(b, a \bmod b)$   
6:      $x_0, y_0 = y_1, x_1 - [a/b] \times y_1$   
7:     return  $d, x_0, y_0$   
8:   end if  
9: end function
```

3.3.3 测试样例及结果截图



```
(19, 3c, 01)  
(07, 0f, 03)  
(04, 1b, 02)  
(02, 01, 09)
```

3.3.4 总结

编程相关 起初对于 $GF(2)$ 类只重定义了 `__str__` 魔法函数，但是由于拓展欧几里得输出为元组类型，导致输出结果一直不是字符串，通过覆盖魔法函数 `__repr__` 最终解决问题

other 有限域上欧几里得算法流程与整数上一致，但是要注意运算为有限域上的运算，取模与求商为 $GF(2)$ 上的运算。

3.4 $GF(2^8)$ 上求逆元

3.4.1 算法流程

方法 1 遍历有限域上所有元素与 a 相乘，若结果为乘法单位元 1，则此时的元素就是 a 的逆元

方法 2 在有限域 $GF(p)$ 上，对于任一非零元 g ，有 $g \times x + p \times y = \gcd(g, p) = 1$ ，求出 x 即是 g 的逆元

3.4.2 算法伪代码

Algorithm 7 $GF(2^8)$ 上元素求逆元方法 1

Input: a

Output: $ans : a \times ans = 1$

```
1: if  $a$  is 0 then
2:   WRONG INPUT
3: else
4:   for  $e \leftarrow 1$  to 255 do
5:     if  $e \times a$  is 1 then
6:       return  $ans \leftarrow e$ 
7:     end if
8:   end for
9: end if
```

Algorithm 8 $GF(2^8)$ 上元素求逆元方法 2

Input: a , Function $EXGCD$

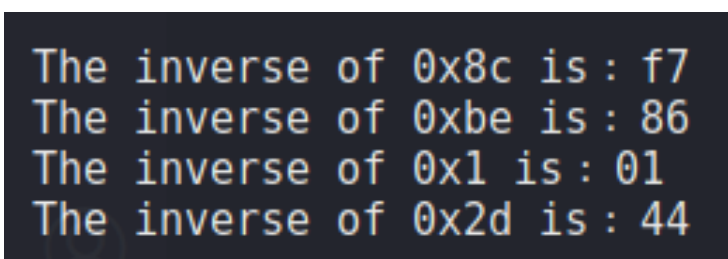
Output: $ans : a \times ans = 1$

```

1: if  $a$  is 0 then
2:   WRONG INPUT
3: else
4:    $d, x, y = EXGCD(a, 11b)$ 
5:   return  $ans \leftarrow x$ 
6: end if

```

3.4.3 测试样例及结果截图



```

The inverse of 0x8c is : f7
The inverse of 0xbe is : 86
The inverse of 0x1 is : 01
The inverse of 0x2d is : 44

```

3.4.4 总结

算法正确性 有限域（除去零元）乘法构成群，因此乘法群中元素必定有逆元。遍历乘法群中元素，必定可以找到逆元

算法复杂度 对于方法 1, 算法时间复杂度约为 $O(n)$ ，对于方法 2, 算法时间复杂度即欧几里得算法时间复杂度 $O(\log n)$

3.5 $GF(2)$ 上 n 次本原多项式判定与生成

3.5.1 算法流程

遍历所有 $degree$ 为 n 的多项式 $f(x)$ ，若 $f(x)$ 满足：

1. $f(x)$ 为不可约多项式
2. $f(x)$ 可整除 $x^m + 1$, $m = 2^n - 1$
3. $\forall q < m$, $x^q + 1$ 不能整除 $f(x)$

则返回 $f(x)$ 为 $GF(2)$ 的一个 n 次本原多项式

3.5.2 算法伪代码

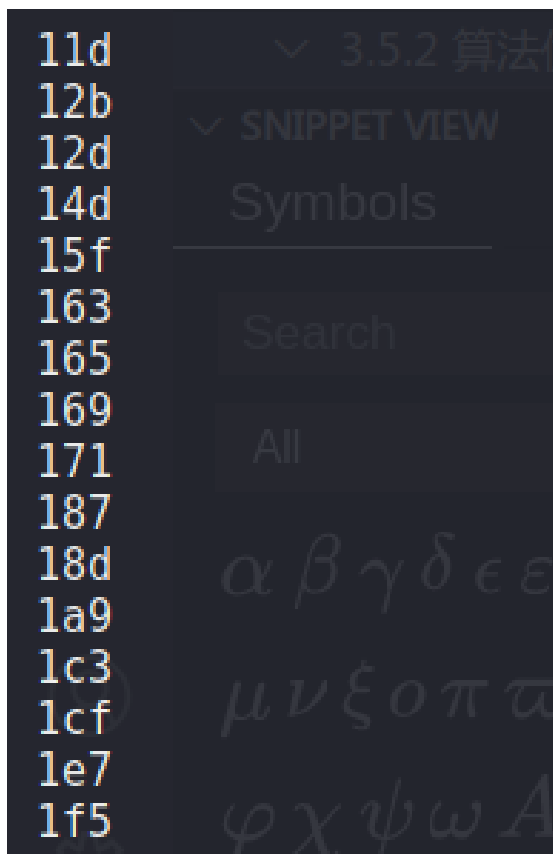
Algorithm 9 $GF(2)$ 上 n 次本原多项式判定与生成

Input: n

Output: L : the list of all primitive polynomials of degree n

```
1:  $m \leftarrow x^{2^n-1} + 1$ 
2: for  $f(x) \leftarrow x^n$  to  $x^n + \dots + 1$  do
3:   for  $g(x) \leftarrow x$  to  $f(x) - 1$  do
4:     if  $f(x) \equiv 0 \pmod{g(x)}$  then
5:       goto line2
6:     end if
7:   end for
8:   if  $m \equiv 0 \pmod{f(x)}$  then
9:     for  $q \leftarrow 1$  to  $2^n - 2$  do
10:      if  $x^q + 1 \equiv 0 \pmod{f(x)}$  then
11:        goto line2
12:      end if
13:    end for
14:     $L.add(f(x))$ 
15:  end if
16: end for
17: return  $L$ 
```

3.5.3 测试样例及结果截图



3.5.4 总结

注意事项 编写代码时对 $x^{x^n-1} + 1$ 等多项式通过位操作获得

4 总结

本次实验实现了 $GF(2^8)$ 上的加法、乘法，快速模幂算法，欧几里得算法，逆元求解。以及 $GF(2)$ 上的带余除法及 n 次本原多项式判定生成算法。进一步体会了有限域上运算的独特性以及普适性。同时进一步了解了 python 语言弱类型的语言特性。