电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221302006

姓 名 周玉川

（实验） 课程名称 信息安全数学基础

理论教师 陈大江

实验教师 陈大江

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：周玉川 学号：2017221302006 指导教师：陈大江**

**实验地点：信软楼西304 实验时间：2018.12.06**

**一、实验名称：有限域上运算实现。**

**二、实验学时：2。**

**三、实验目的：**

1. 学习有限域概念，已经有限域相关运算的算法。
2. 学习多项式，如何表示多项式，多项式求余，加减运算，除法乘法，以及求逆运算。
3. 复习扩展欧几里得算法，重复平方算法。
4. 编程实现多项式在有限域下的相关运算，包括加减乘除，求逆，平方等。

**四、实验原理：**

1. 有限域F是指只包含有限个元素的域，F的阶为元素个数，有限域又被称为伽罗瓦域。有限域可以用多项式以及本原元表示。
2. 有限域求余运算即多项式求余，因为是F28,多项式加减运算即每一位异或运算，多项式求逆类比数字的扩展欧几里得算法，多项式的幂运算则类比于数字的重复平方算法。

**五、实验内容：**

1. 实现28域上元素的多项式基表示，实现模多项式的乘法运算和求逆运算，从而 实现28域上元素乘法运算和逆元运算。
2. 构造指数对数表，从而通过查表实现28域上元素乘法运算和逆元运算。

**六、实验器材（设备、元器件）：**

Pc一台。

**七、实验步骤：**

1. 分析多项式加减乘除，求逆，幂运算，以及指数对数表。
2. 编写python代码。
3. 各种运算测试一次。
4. 输出指数对数表，根据表做乘法和求逆运算。

**八、实验结果与分析（含重要数据结果分析或核心代码流程分析）**

程序代码（python编写，带注释可直接运行）

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """  Created on Thu Dec 6 11:00:28 2018  @author: 鑫鑫玉川  """    """  名字：Mod(a,b,n),a,b表示列表，长度都为n  功能：返回a模b的列表  """  def Mod(a,b,n):  result=a[:] #c拷贝a  d=b[:]  a\_len=n  b\_len=n  for i in range(n):  if(a[i]!=0):  break  a\_len-=1  for i in range(n):  if (b[i]!=0):  break  b\_len-=1  #print(a\_len,b\_len)#测试  if (a\_len >= b\_len):  for i in range(b\_len,a\_len):  d.append(0)  for i in range(a\_len-1,b\_len-2,-1):  #print(n-i-1,a\_len, b\_len,n,a,b,result)#测试  if(result[n-i-1]==1):  result=AddSub(result,d[i-b\_len+1:n+i-b\_len+1],n)  #print("Mod",result)#测试代码  return result  """  名字：AddSub(a,b,n),a,b为列表，长度为n  功能：返回a减b或者a加b  说明：无论是加运算,还是减运算，在模2运算中算法相同  """  def AddSub(a,b,n):  c=[]  for i in range(n):  c.append(a[i]^b[i])  return c  """  名字：Multiply(a,b,n),a,b为列表，长度为n  功能：返回a乘b的元素  """  def Multiply(a, b, n, mod):  c=a[:]  while (n<len(mod)):  c.insert(0,0)  b.insert(0,0)  n+=1  result=[0 for i in range(n)]  for i in range(n-1, -1 , -1):  if ( b[i]==1 ):  result = AddSub(result,c,n)  c.append(0)  c.pop(0) #相当于左移一位  c = list(Mod(c, mod, n))#得到c模基准的多项式  #print("Multi",c)#测试  return result  """  名字：Divide(a,b,n),a,b为列表，长度为n  功能：返回a整除b后的多项式  """  def Divide(a , b , n):  c=a[:]  d=b[:]  result=[0 for i in range(n)]  a\_len=n  b\_len=n  for i in range(n):  if(a[i]!=0):  break  a\_len-=1  for i in range(n):  if (b[i]!=0):  break  b\_len-=1  if (a\_len >= b\_len):  for i in range(b\_len,a\_len):  d.append(0)  for i in range(a\_len-1,b\_len-2,-1):  if(c[n-i-1]==1):  result[n-i+b\_len-2]=1  c=AddSub(c,d[i-b\_len+1:n+i-b\_len+1],n)  #print(c,result),测试代码  return result      """  名字：getRes(a,b,n,mod),a,b为列表，长度为n  功能：求多项式a关于b的逆元  原扩展欧几里得算法  def OGLD\_pro(a,b):  if (b==0):  return {'d':a,'x':1,'y':0}  else:  t=OGLD\_pro(b,a%b)  temp=t['x']  t['x']=t['y']  t['y']=temp-int(a/b)\*t['y']  return t  """  def getRes(a,b,n,mod):  if (max(b)==0):#证明c中，即b中全为0  x=[0 for i in range(n-1)]  x.append(1)  #print("end",x)#测试  return {'d':a[:],'x':x,'y':[0 for i in range(n)]}  else:  t=getRes(b,Mod(a,b,n),n,mod)  #print(t)#测试  temp=list(t['x'])  t['x']=list(t['y'])  t['y']=AddSub(temp,Multiply(Divide(a,b,n),t['y'],n,mod),n)    return t  """  名字：ExpOfMod(a,k,myMod,n):列表a,k为指数，myMod为要模的多项式，n为列表长度  功能：模的指数运算，重复平方算法  """  def ExpOfMod(a,k,mod,n):  result=[0 for i in range(n-1)]  result.append(1)  if ( k!= 0):  A = a[:]  if ( k%2 == 1):  result = a[:]  k = k //2 #注意整除  while (k != 0):  A = Multiply(A,A,n,mod)  if (k%2==1):  result = Multiply(A,result,n,mod)  k = k // 2  #print(result,k)#用于测试  return result  """  名字：transList(lst),lst为列表  功能:用于把列表中二进制数转化为10进制  """  def transList(lst):  #n=len(lst)  result=0  for i in lst:  result\*=2  result+=i    return result  #用列表中元素，从低位到高位依次表示二进制数的高位到低位  #列表，可以进行的操作包括索引，切片，加，乘，检查成员。  """  名字：基准，即要模得多项式，用多项式基表示  功能：作为全局变量  注意：多项式一定不能拆分，即不可约多项式  """  myMod = [1,0,1,1,0,0,0,1,1] #基准表示要模的元素，即f(x)=x^8 + x^6 + x^5 + x + 1  testMod =[1,0,0,1,1] #类似myMod，用于测试,f(x)=x^4 + x + 1  a=[0,0,0,0,0,0,1,1,1] #f(x)=x^2+x+1  b=[0,1,0,1,0,1,1,0,1] #f(x)=x^7+x^5+x^3+x^2+1  print("a=",a)  print("b=",b)  print("a+b=a-b=",AddSub(a,b,len(a))) #输出相加减结果  print("b/a=",Divide(b,a,len(a))) #输出相除结果  print("a\*b=",Multiply(a,b,len(a),myMod))#输出相乘结果  t=getRes(a,myMod,9,myMod)  a\_=t['x'] #a的逆  print("a得逆元为：",a\_) #输出逆元  base = [0,0,0,0,0,0,0,1,0] #本原元  ExpList=[0 for i in range(256)]#指数表,初始化为0  LogList=[0 for i in range(256)]#对数表，初始化为0  LogList[0]=255  for i in range(255): #循环得到指数对数表  index=transList(ExpOfMod(base,i,myMod,len(base)))  ExpList[i]=index  LogList[index]=i  print("\n\t\t\t\t\t指数对数表\n\n")  i=0  while(i<256):  while(True):  if (i%16==0):  print("{:8s}".format("序号"),end=' ')  print("{:4d}".format(i),end=' ')  if ((i+1)%16==0):  print('\n')  i=i-15  break;  i+=1  while(True):  if (i%16==0):  print("{:9s}".format("(02)^i"),end=' ')  print("{:4d}".format(ExpList[i]),end=' ')  if ((i+1)%16==0):  print('\n')  i=i-15  break;  i+=1  while(True):  if (i%16==0):  print("{:7s}".format("log(02)^i"),end=' ')  print("{:4d}".format(LogList[i]),end=' ')  if ((i+1)%16==0):  print('\n\n')  i+=1  break;  i+=1 |

代码分析

1. 首先理解有限域里面的运算，把学了十几年的数的加减乘除先抛到脑后，在有限域内人为定义新的运算，称为该有限域内的加减乘除。
2. Python代码先定义了4个函数，Mod为多项式模运算，AddSub为多项式的加减运算，因为是模2所以加和减的运算是一样的，Multiply是多项式乘法，Divide是多项式除法，这是4个最基本的运算。
3. 利用2中4个基本运算，完成求逆操作，数字的扩展欧几里得算法已经学过，多项式也是扩展的欧几里得算法，只是把数字的乘除加减求模运算改成有限域多项式的加减乘除求模运算，算法思想是一样的，下面是他们的比较。

数字的扩展欧几里得算法

|  |
| --- |
| def OGLD\_pro(a,b):  if (b==0):  return {'d':a,'x':1,'y':0}  else:  t=OGLD\_pro(b,a%b)  temp=t['x']  t['x']=t['y']  t['y']=temp-int(a/b)\*t['y']  return t |

多项式的扩展欧几里得算法

|  |
| --- |
| """  名字：getRes(a,b,n,mod),a,b为列表，长度为n  功能：求多项式a关于b的逆元  """  def getRes(a,b,n,mod):  if (max(b)==0):#证明c中，即b中全为0  x=[0 for i in range(n-1)]  x.append(1)  #print("end",x)#测试  return {'d':a[:],'x':x,'y':[0 for i in range(n)]}  else:  t=getRes(b,Mod(a,b,n),n,mod)  #print(t)#测试  temp=list(t['x'])  t['x']=list(t['y'])  t['y']=AddSub(temp,Multiply(Divide(a,b,n),t['y'],n,mod),n)    return t |

可以看出，这两种算法在形式上是一摸一样的，符号减‘-’对应函数AddSub，符号求模‘%’对应函数Mod，符号乘‘\*’对应函数Multiply，符号除‘/’对应函数Divide。

1. 重复平方算法和3中思路一样，可以把多项式的幂运算代码写出来，从而求出指数对数表。

|  |
| --- |
| 关键代码  """  名字：ExpOfMod(a,k,myMod,n):列表a,k为指数，myMod为要模的多项式，n为列表长度  功能：模的指数运算，重复平方算法  """  def ExpOfMod(a,k,mod,n):  result=[0 for i in range(n-1)]  result.append(1)  if ( k!= 0):  A = a[:]  if ( k%2 == 1):  result = a[:]  k = k //2 #注意整除  while (k != 0):  A = Multiply(A,A,n,mod)  if (k%2==1):  result = Multiply(A,result,n,mod)  k = k // 2  #print(result,k)#用于测试  return result |

运行结果

|  |
| --- |
| 多项式的加减乘除  a= [0, 0, 0, 0, 0, 0, 1, 1, 1]  b= [0, 1, 0, 1, 0, 1, 1, 0, 1]  a+b=a-b= [0, 1, 0, 1, 0, 1, 0, 1, 0]  b/a= [0, 0, 0, 1, 1, 1, 0, 0, 1]  a\*b= [0, 1, 1, 1, 0, 0, 1, 1, 0]  a得逆元为： [0, 1, 0, 0, 0, 1, 1, 0, 0] |
| 指数对数表    指数对数表  序号 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  (02)^i 1 2 4 8 16 32 64 128 99 198 239 189 25 50 100 200  log(02)^i 255 0 1 197 2 139 198 108 3 50 140 192 199 37 109 81  序号 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31  (02)^i 243 133 105 210 199 237 185 17 34 68 136 115 230 175 61 122  log(02)^i 4 23 51 238 141 216 193 234 200 12 38 247 110 134 82 124  序号 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47  (02)^i 244 139 117 234 183 13 26 52 104 208 195 229 169 49 98 196  log(02)^i 5 66 24 145 52 115 239 76 142 206 217 209 194 189 235 244  序号 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63  (02)^i 235 181 9 18 36 72 144 67 134 111 222 223 221 217 209 193  log(02)^i 201 45 13 220 39 180 248 164 111 176 135 212 83 30 125 158  序号 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79  (02)^i 225 161 33 66 132 107 214 207 253 153 81 162 39 78 156 91  log(02)^i 6 100 67 55 25 129 146 227 53 225 116 118 240 154 77 120  序号 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95  (02)^i 182 15 30 60 120 240 131 101 202 247 141 121 242 135 109 218  log(02)^i 143 74 207 242 218 162 210 156 195 106 190 79 236 232 245 122  序号 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111  (02)^i 215 205 249 145 65 130 103 206 255 157 89 178 7 14 28 56  log(02)^i 202 172 46 8 14 87 221 102 40 18 181 69 249 94 165 57  序号 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127  (02)^i 112 224 163 37 74 148 75 150 79 158 95 190 31 62 124 248  log(02)^i 112 186 177 27 136 34 213 131 84 91 31 148 126 151 159 229  序号 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  (02)^i 147 69 138 119 238 191 29 58 116 232 179 5 10 20 40 80  log(02)^i 7 171 101 86 68 17 56 93 26 185 130 33 147 90 228 150  序号 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159  (02)^i 160 35 70 140 123 246 143 125 250 151 77 154 87 174 63 126  log(02)^i 54 99 226 128 117 224 119 153 241 73 155 161 78 105 121 231  序号 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175  (02)^i 252 155 85 170 55 110 220 219 213 201 241 129 97 194 231 173  log(02)^i 144 65 75 114 208 205 243 188 219 44 163 179 211 175 157 29  序号 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191  (02)^i 57 114 228 171 53 106 212 203 245 137 113 226 167 45 90 180  log(02)^i 196 254 107 138 191 49 80 36 237 22 233 215 246 11 123 133  序号 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207  (02)^i 11 22 44 88 176 3 6 12 24 48 96 192 227 165 41 82  log(02)^i 203 63 173 42 47 252 9 20 15 169 88 183 222 97 103 71  序号 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223  (02)^i 164 43 86 172 59 118 236 187 21 42 84 168 51 102 204 251  log(02)^i 41 62 19 251 182 168 70 96 250 61 95 167 166 60 58 59  序号 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239  (02)^i 149 73 146 71 142 127 254 159 93 186 23 46 92 184 19 38  log(02)^i 113 64 187 204 178 43 28 174 137 253 35 48 214 21 132 10  序号 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255  (02)^i 76 152 83 166 47 94 188 27 54 108 216 211 197 233 177 0  log(02)^i 85 170 92 16 32 184 149 89 127 98 152 223 160 72 230 104 |

**九、总结及心得体会：**

1. 这次实验很有难度，相当于前面各种算法的汇总以及超强升级版，包括扩展欧几里得，重复平方算法，多项式求逆
2. 兴趣对学习的帮助是极大的，很大可能事半功倍。
3. 代码能力占一定重要性，关键在于理解力。
4. 第一次实验用C语言编写，第二，三次实验用java编写，这次完成承诺用python编写。

**十、对本实验过程及方法、手段的改进建议：**

1. 实验内容较少，除了实验要用的算法，书上还有Garner算法（中国剩余定理），以及同余方程，勒让德符号等均可以用代码实现，可以适当增加前面实验的内容。
2. 鼓励学生用C语言之外的编程语言完成要求。
3. 数学知识尤其是和算术相关的定理，用计算机语言很容易实现，若有兴趣，可以把代码写出来，很有趣，也有成就感，对信安基础的学习也大有帮助。
4. 鼓励老师布置作业。

**报告评分：**

**指导教师签字：**