云南大学数学与与统计学院

上机实践报告

课程名称：信息论基础实验	**年级**：2013	**上机实践成绩**：
指导教师：陆正福	**姓名**：金洋	
上机实践名称：循环码实验	**学号**：20131910023	**上机实践日期**：2016/6
上机实践编号：No. 8	**组号**：	**上机实践时间**： **12:47**

一、实验目的

（1） 熟悉循环码的基本原理

（2） 通过具体的 CRC 码的实现以及纠错能力的考察，熟悉检错码的概念、术语、设计思路和局限性

（3） 熟悉位运算对于二元循环码的实现程序的优化作用

二、实验内容

(1) 设计以 $x^{16}+x^{12}+x^5+1$ 为生成多项式的循环码编码和译码算法，并完成相应的程序设计。
(2) 通过设置不同的错误模式，考察检错能力
(3) 考察错误模式，得出有关错误检测概率的结论。
(4) 程序设计的推广：对于任意循环码（任意的生成多项式），设计相应的编码程序和译码程序。

三、实验环境

1. 个人计算机，任意可以完成实验的平台，如 Java 平台、Python 语言、R 语言、Matlab 平台、Magma 平台等。
2. 对于信息与计算科学专业的学生，建议选择 Java、Python、R 等平台。
3. 对于非信息与计算科学专业的学生，建议选择 Matlab、Magma 等平台。

四、实验记录与实验结果分析

（注意记录实验中遇到的问题。实验报告的评分依据之一是实验记录的细致程度、实验过程的真实性、实验结果的解释和分析。**如果涉及实验结果截屏，应选择白底黑字。**）

1. 循环码编码原理：

有信息码构成信息多项式 $m(x)=m_{k-1}x^{k-1}+\cdots+m_0$，其中最高幂次为 k-1；用 x^{n-k} 乘以信息多项式 m(x)，得到的 $x^{n-k}m(x)$，最高幂次为 n-1，该过程相当于把信息码（m_{k-1}，m_{k-2}，……，m_1，m_0）移位到了码字前 k 个信息位，其后是 r 个全为零的监督位；

用 g(x)除 $x^{n-k}m(x)$ 得到余式 r(x)，其次数必小于 g(x)的次数，即小于（n-k），将此 r(x)加于信息位后做监督位，即将 r(x)于 $x^{n-k}m(x)$ 相加，得到的多项式必为一码多项式。

循环码纠错原理：

纠错码的译码是该编码能否得到实际应用的关键所在。译码器往往比编码较难实现，对于纠错能力强的纠错码更复杂。根据不同的纠错或检错目的，循环码译码器可分为用于纠错目的和用于检错目的的循环码译码器。

通常，将接收到的循环码组进行除法运算，如果除尽，则说明正确传输；如果未除尽，则在寄存器中的内容就是错误图样，根据错误图样可以确定一种逻辑，来确定差错的位置，从而达到纠错的目的。用于纠错目的的循环码的译码算法比较复杂，感兴趣的话可以参考一些参考书。而用于检错目的循环码，一般使用 ARQ 通信方式。检测过程也是将接受到的码组进行除法运算，如果除尽，则说明传输无误；如果未除尽，则表明传输出现差错，要求发送端重发。用于这种目的的循环码经常被成为循环冗余校验码，即 CRC 校验码。CRC 校验码由于编码电路、检错电路简单且易于实现，因此得到广泛的应用。在通过 MODEM 传输文件的协议如 ZMODEM、XMODEM 协议中均用到了 CRC 校验技术。在磁盘、光盘介质存储技术中也使用该方法。

当码字 c 通过噪声信道传送时，会受到干扰而产生错误。如果信道产生的错误图样是 e，译码器收到的 n 重接受矢量是 y,则表示为：

$$y = c + e$$

上式也可以写成多项式形式：

$$y(x) = c(x) + e(x)$$

译码器的任务就是从 y(x)中得到$\hat{e}(x)$，然后求的估值码字

$$\hat{c}(x) = y(x) + \hat{e}(x)$$

并从中得到信息组$\hat{m}(x)$。

循环码译码可按以下三个步骤进行：

（1）有接收到的 y(x)计算伴随式 s(x)；

（2）根据伴随式 s(x)找出对应的估值错误图样$\hat{e}(x)$；

（3）计算$\hat{c}(x) = y(x) + \hat{e}(x)$，得到估计码字$\hat{c}(x)$。若$\hat{c}(x) = c(x)$,则译码正确，否则，若$\hat{c}(x) \neq c(x)$，则译码错误。

由于 g(x) 的次数为 n - k 次，g(x) 除 E(x) 后得余式（即伴随式）的最高次数为 n-k-1 次，故 S(x) 共有 2^{n-k} 个可能的表达式，每一个表达式对应一个错误格式。可以知道(7,4）循环码的 S(x) 共有 $2^{(7-4)}=8$ 个可能的表达式，可根据错误图样表来纠正(7,4）循环码中的一位

错误，其伴随式如表所示。

BCH（7，4）循环码错误图样表：

错误图样	错误图样码字	伴随式S(x)	伴随式
$E_6(x)=x_6$	1000000	x^2	100
$E_5(x)=x_5$	0100000	x^2+x	110
$E_4(x)=x_4$	0010000	x^2+x+1	111
$E_3(x)=x_3$	0001000	$x+1$	011
$E_2(x)=x_2$	0000100	x^2+1	101
$E_1(x)=x_1$	0000010	x	010
$E_0(x)=x_0$	0000001	1	001
$E(x)=0$	0000000	0	000

上式指出了系统循环码的译码方法：将收到的码字$R(x)$ 用$g(x)$ 去除，如果除尽则无错；否则有错。如果有错，可由余式$S(x)$ 一一找出对应图样，然后将错误图样$E(x)$ 与$R(x)$ 模2 和，即为所求码字$C(x)$ ，从而实现纠错目的。

Complement.java

```
package IT8;


/*补位操作*/
public class Complement {
	public static int[] Generate(int[] P,int n) {
		int len = P.length;
		int[] PP = new int[len+n-1];
		for(int i = len;i<len+n-1;i++) {
			PP[i] = 0;
		}
		for(int i = 0;i<len;i++) {
			PP[i] = P[i];
		}
		return PP;
	}
```

```
}
```

CreatePol.java

```
package IT8;

public class CreatePol {

	//生成被除式
	public static int[] genDividend(int n) {
		int[] dividend = new int[n+1];
		dividend[0] = 1;
		dividend[n] = 1;
		for(int i = 1;i<n;i++) {
			dividend[i] = 0;
		}
		return dividend;
	}

	//生成除式
	public static int[] genDivisor(int[] div,int n) {
		int len = div.length;
		int [] divisor = new int[len+n];
		for(int i = 0;i<len;i++) {
			divisor[i] = div[i];
		}
	   for(int i = len;i<len+n;i++) {
		divisor[i] = 0;
	   }
		return divisor;

	}

}
```

Divide.java

```
package IT8;


import java.lang.reflect.Field;
import java.util.Arrays;

public class Divide {

	public static int[][] divide(int[] divisor, int[] dividend) {
```

```
int len = dividend.length;
int[] divisor2 = new int[len];
int[][] result = new int[2][len];

for(int i = 0;i<len;i++) {
	divisor2[i] = divisor[i];
}

int[] factor= new int[len];
int[] remainder = new int[len];

// 初始商为0
for (int i = 0; i < len; i++) {
	factor[i] = 0;
}

for(;;){
	int flag1 = 0, flag2 = 0;
	if (Arrays.equals(dividend, divisor2)) {
		factor[0] = 1;
		for(int i = 0;i<len;i++) {
			remainder[i] = 0;
		}
		break;
	}
	else {
		System.out.println();
		System.out.println("除法开始");

		// 找出右边第一个不为1的下标
		for(int i = 0;i<len;i++) {
			divisor[i] = divisor2[i];
		}
		for (int j = len - 1; j >= 0; j--) {
			if (dividend[j] == 1) {
				flag1 = j;
				break;
			}
		}
		for (int j = len - 1; j >= 0; j--) {
			if (divisor[j] == 1) {
				flag2 = j;
				break;
			}
		}

		if (flag1 > flag2) {
			int re = flag1 - flag2;
			factor[re] = 1;
```

```
                        divisor = Moving.move(divisor, re);

                        for (int k = 0; k < len; k++) {
                            dividend[k] = dividend[k] ^ divisor[k];
                        }
                    }
                    for(int i = 0;i<len;i++) {
                        remainder[i] = dividend[i];
                    }

                    if(flag1 == flag2) {
                        factor[0] = 1;
                        if(flag1 == flag2 &&  Arrays.equals(dividend,
divisor2)) {
                            factor[0] = 1;
                            for(int i = 0;i<len;i++) {
                                remainder[i] = 0;
                            }
                            break;
                        }
                        break;
                    }

                    if(flag1<flag2) {
                        break;
                    }
                }
            }
            for(int i = 0;i<len;i++) {
                result[0][i] = factor[i];
            }
            for(int i = 0;i<len;i++) {
                result[1][i] = remainder[i];
            }
            System.out.println("商和余数为: ");
            for(int i = 0;i<len;i++) {
                System.out.print(result[0][i]+" ");
            }
            System.out.print("\n");

            for(int i = 0;i<len;i++) {
                System.out.print(result[1][i]+" ");
            }
            System.out.print("\n");



            return result;
        }
```

```
}
```

GetCode.java

```
package IT8;

/*生成编码*/
public class GetCode {
	public static int[] genCode(int[] dividend,int[] divisor,int n) {
		//计算 X^r*P(X)
		int len = dividend.length;
		int[] dividend2 = new int[len];
		for(int i = 0;i<len;i++) {
			dividend2[i] = dividend[i];
		}
		int[] R = new int[len];
		int[] code = new int[len];
		//计算 R(x)
		int[][] R1 = Divide.divide(divisor, dividend);
		for(int i = 0;i<len;i++) {
			R[i] = R1[1][i];
		}
		for(int i = 0;i<len;i++) {
			code[i] = R[i] | dividend2[i];
		}
		return code;



	}

}
```

Moving.java

```
package IT8;

public class Moving {
	public static int[] move(int[] a, int k) {
		int n = a.length;
		int m = Math.min(k, n - k);
		int i, j;
```

```
        if (n % m != 0) {
            int start = 0;
            // Integer now = start;
            int to = (start + k) % n;
            int getdata;
            int setdata = a[start];
            while (to != start) {
                getdata = a[to];
                a[to] = setdata;
                to = (to + k) % n;
                setdata = getdata;
            }
            a[to] = setdata;
        } else {
            int start = 0;
            int now = start;
            int to = (now + k) % n;
            int getdata;
            int setdata = a[start];
            for (i = 0; i < m; i++) {
                start = i;
                setdata = a[start];
                to = (start + k) % n;
                now = to;
                while (now != start) {
                    getdata = a[to];
                    a[to] = setdata;
                    now = to;
                    to = (to + k) % n;
                    setdata = getdata;

                }
            }
        }
        return a;
    }


    public static void main(String[] args) {

    int[] data = { 1,0,0,0,0,1,0,0,0};
    int[] b = move(data,1);
    for(int i = 0;i<data.length;i++) {
         System.out.print(data[i]);
    }
    System.out.println();
    for(int i = 0;i<data.length;i++) {
         System.out.print(b[i]);
    }
```

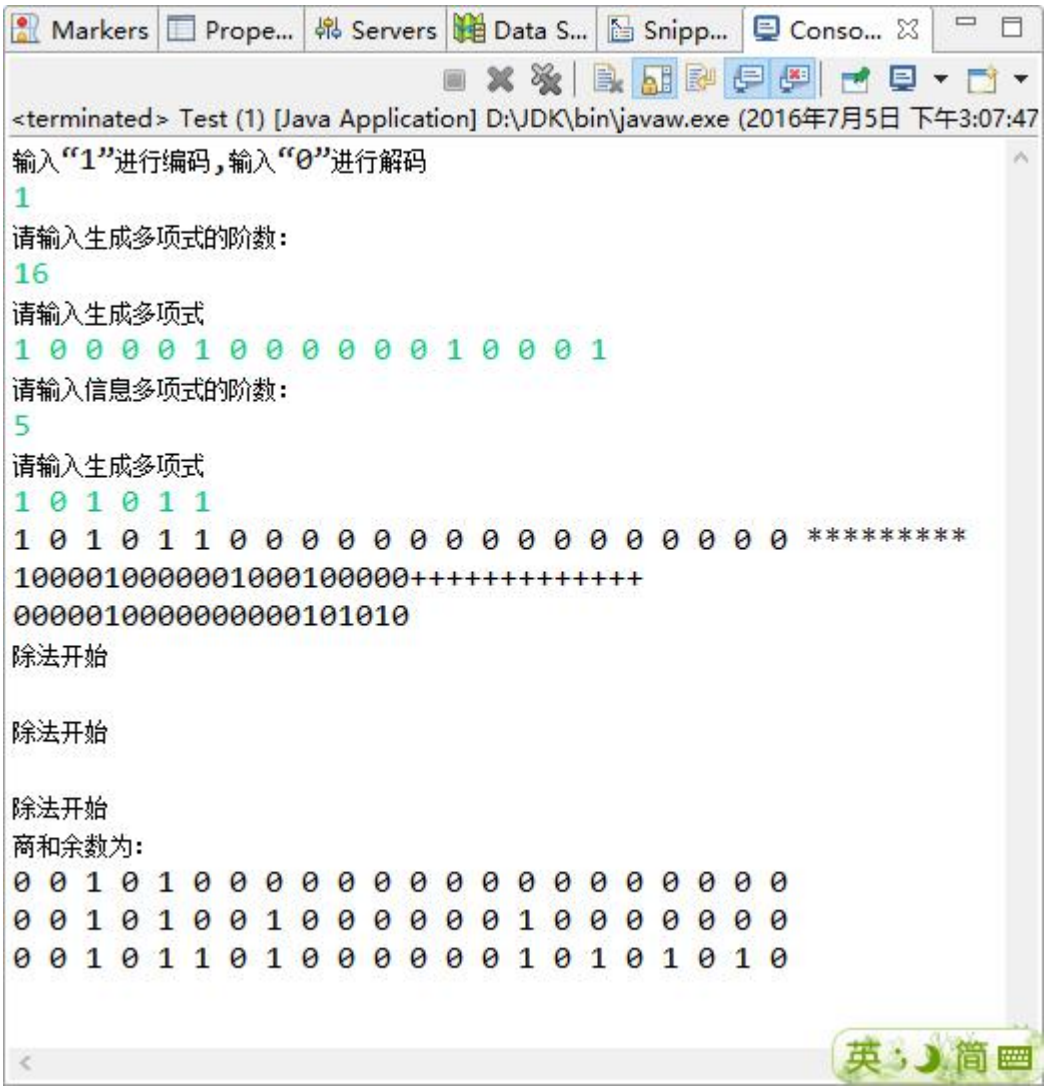
```
	}

}
```

Test.java

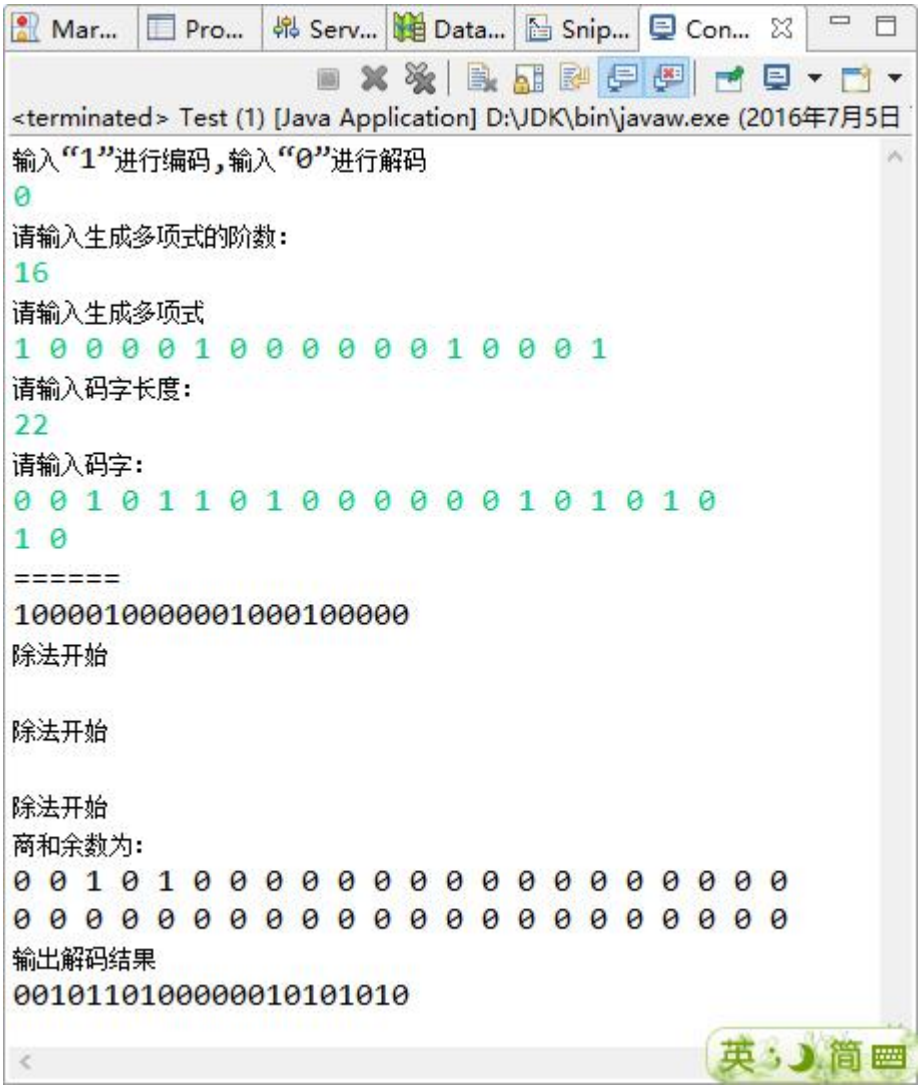
```
package IT8;

import java.util.Scanner;


public class Test {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("输入“1”进行编码,输入“0”进行解码");
		int temp = sc.nextInt();
		sc.reset();
		System.out.println("请输入生成多项式的阶数：");
		int k = sc.nextInt();
		sc.reset();
		System.out.println("请输入生成多项式");
		int[] gener = new int[k+1];
		for(int i = 0;i<k+1;i++) {
			gener[i] = sc.nextInt();
		}
		if(temp == 1) {
			//信息多项式 info
			sc.reset();
			System.out.println("请输入信息多项式的阶数：");
			int d = sc.nextInt();
			sc.reset();
			System.out.println("请输入生成多项式");
			int[] info = new int[d+1];
			for(int i = 0;i<d+1;i++) {
				info[i] = sc.nextInt();
			}

			int len = gener.length;
			int len1 = info.length;
			//计算 x^r*m(x),m(x)为信息多项式
			int[] t = Complement.Generate(info, len);
			for(int i = 0;i<t.length;i++) {
				System.out.print( t[i]+ " ");
			}
			Moving.move(t, len-1);
```

```
                //处理生成多项式
                int [] divisor = CreatePol.genDivisor(gener, len1-1);
                System.out.println("*********");
                for(int i = 0;i<divisor.length;i++) {
                      System.out.print(divisor[i]);
                }

                System.out.println("++++++++++++");
                for(int i = 0;i<divisor.length;i++) {
                      System.out.print(t[i]);
                }
                //生成循环码
                int[] code  = GetCode.genCode(t, divisor, len);
                for(int i = 0;i<t.length;i++) {
                      System.out.print( code[i]+ " ");
                }

          }
          if(temp == 0) {
                sc.reset();
                System.out.println("请输入码字长度: ");
                int L = sc.nextInt();
                sc.reset();
                System.out.println("请输入码字: ");
                int[] codeword = new int[L];
                for(int i = 0;i<L;i++) {
                      codeword[i] = sc.nextInt();
                }
                //处理生成多项式使得与码字有相同长度
                int codeLen = codeword.length;
                int[] gener2 = Complement.Generate(gener, codeLen-k);
                System.out.println("======");
                for(int i = 0;i<gener2.length;i++) {
                      System.out.print(gener2[i]);
                }


                GetCode CRCdecode=new GetCode();
                int[] word = CRCdecode.genCode(codeword, gener2, k);
                System.out.println("输出解码结果");
                for(int i = 0;i<word.length;i++) {
                      System.out.print(word[i]);
                }
          }
     }

}
```

测试结果：



最后一行为编码结果。



最后一行为解码结果。

五、实验体会（请认真填写自己的真实体会）

循环码是一类重要的线性码，由于它具有严谨的代数结构，因此，其性能易于分析，有较强的检错和纠错能力。它有许多特殊的代数性质，它使计算机通信以一种以数据通信形式出现，实现了在计算机与计算机之间或计算机与终端设备之间进行有效的与正确地信息传递，它使得现代通信的可靠性与有效性实现了质的飞跃循环码还有易于实现的特点，很容易用带反馈的移位寄存器实现其硬件。

六、参考文献

1. Thomas M. Cover, Joy A. Thomas. Elements of Information Theory (2nd Edition) [M]. John Wiley & Sons, Inc. Chapter 7

2.（如有其它参考文献，请列出）