



江 蘇 大 學

JIANGSU UNIVERSITY

2017-2018 学年第 1 学期

《信息论与编码》课程设计

学院名称: 计算机科学与通信工程学院

专业班级: 信息安全 1501

学生姓名: 沈鑫楠

学生学号: 3150604028

教师姓名: 朱小龙

完成日期: 2018 年 1 月 6 日

目录

1.LZW 编码.....	3
1.1 任务说明.....	3
1.2 设计思路.....	3
1.2.1 问题分析.....	3
1.2.2 实现原理与流程图.....	4
1.3 实现源码.....	5
1.4 运行结果.....	9
2.信道容量迭代计算.....	10
2.1 任务说明.....	10
2.2 设计思路.....	10
2.2.1 问题分析.....	10
2.2.2 实现原理与流程图.....	11
2.3 实现源码.....	11
2.4 运行结果.....	15
3.综合编码.....	16
3.1 任务说明.....	16
3.2 设计思路.....	17
3.2.1 问题分析.....	17
3.2.2 实现原理与流程图.....	18
3.3 实现源码.....	19
3.4 运行结果.....	30
4.设计体会.....	30

1.LZW 编码

1.1 任务说明

输入：由集合{a,b,c,d}内字符构成的输入串，输入序列长度 $L \leq 100$

处理：先编码，再对编码结果译码

输出：编码结果，译码结果

输入文件:in2.txt，含至少两组输入，每组包含满足要求的串

输出文件:out2.txt，对每组输入的编码和译码结果

1.2 设计思路

1.2.1 问题分析

LZW 编码的原理：先建立初始化字典，然后将待编码的输入数据流分解成短语词条。对于每个短语词条，检查是否在字典中已经存在，如果存在就输入下一个字符形成一个新的短语词条。当 I 在字典内，而 Ix 不在字典内时，输出字典内 I 的位置，并将 Ix 存入字典中，并为其确立顺序号，然后把 x 给 I ，当作新的短语词条的首字符。重复上述过程，直到输入流全部处理完为止。

1.2.2 实现原理与流程图

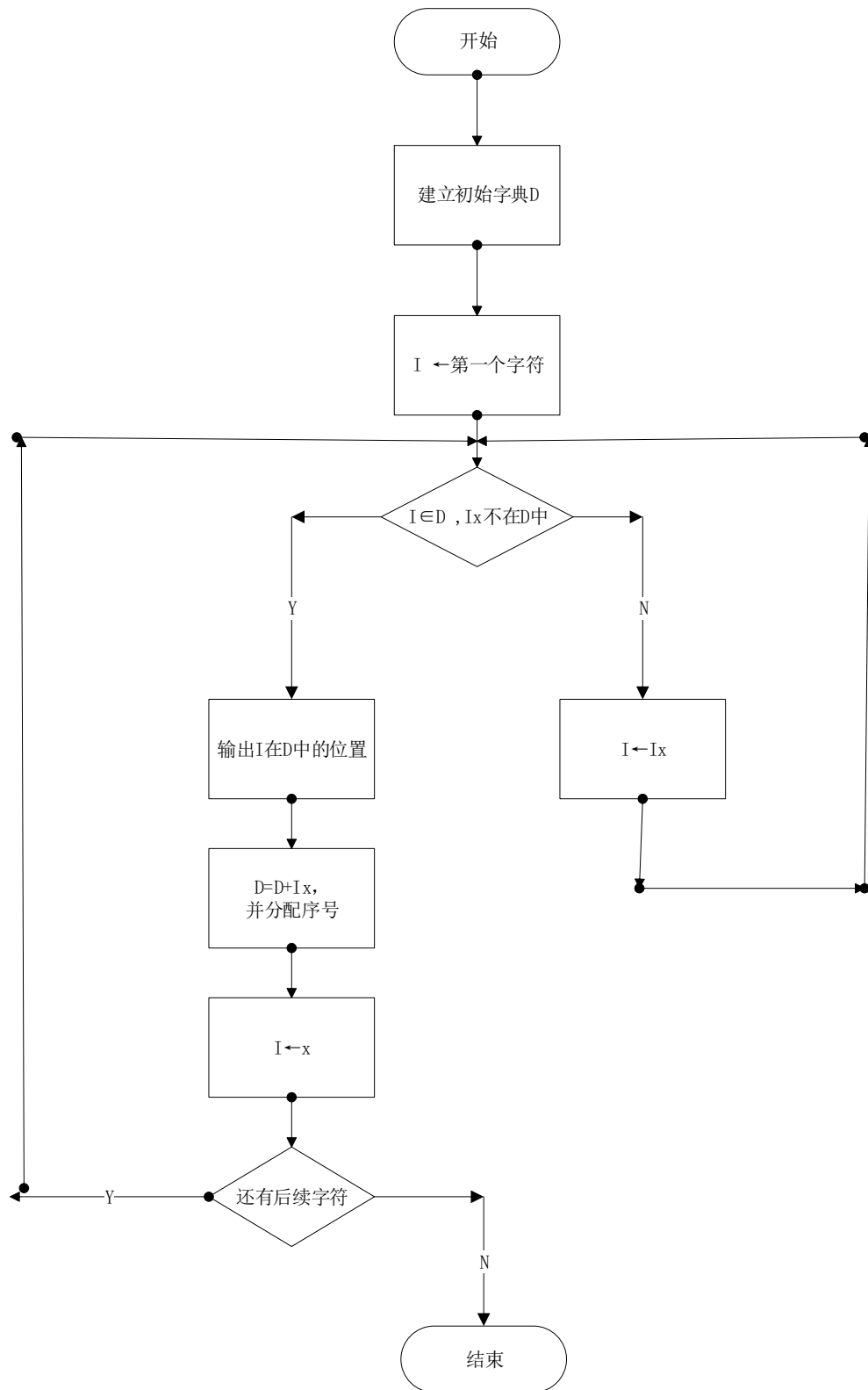


图 1 LZW 编码流程图

1.3 实现源码

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
namespace ConsoleApplication1
{
    class Program
    {
        public static int[] result_array ;//编码结果
        public static string[] temparr;//字典
        public static string ori;//译码结果
        static void EnCode(string w)//编码
        {
            char []w_array=w.ToCharArray();
            int len = w_array.Length;
            string result = "";
            //建立初始化字典
            char ca ;
            //string temp1 = "";
            temparr = new string[len];
            int flag = 0;
            for (int i = 0; i < 26;i++ )//遍历 a 到 z
            {
                ca = (char)('a' + i);
                if(w.IndexOf(ca)!=-1)//如果字符串中存在该字母
                {
                    temparr[flag++] = ca.ToString();//加入到字典中
                }
            }
            result_array = new int[len];
            int k = 0;
```

```

int f2 = 0;
Stack<int> s1 = new Stack<int>();
//逐位编码
for (int i = 0; i < len; )
{
    string cur = "";
    cur = cur + w_array[i];
    for(int j=i+1;;j++)
    {
        for (k = 0; k < temparr.Length; k++)
        if (temparr[k]!=null&&temparr[k] == cur) break;//查找字典中有无此字符串
        if (k==temparr.Length)//字典中没有这个字符串
        {
            k = s1.Pop();//导出前一个字典中存在的字符串的索引
            result_array[f2++] = k+1;//导出到输出码中
            temparr[flag++] = cur;//字典中增添该项
            break;//进行下一项的编码
        }
        else//字典中有这个字符串
            s1.Push(k);//记录下词条的索引
        if(j>=len)//范围越界
        {
            for (k = 0; k < temparr.Length; k++)
            if (temparr[k] != null && temparr[k] == cur) break;//查找当前字符串在字典中是否存在
            if(k==temparr.Length)//不存在
            {
                k = s1.Pop();//导出前一个索引值
                result_array[f2++] = k + 1;//记录到输出码中
                temparr[flag++] = cur;//字典中增添该项
            }
            else//存在
            {
                result_array[f2++] = k + 1;//索引值导出到输出码中
            }
            result_array[f2++] = -1;//结束标记 eof
        }
    }
}

```

```

        i = len;//改变 i 值，以便退出外层循环
        break;//退出内层循环
    }
    else//范围不越界
        cur = cur + w_array[j];//形成新词条
    i++;//改变指向下一个待编码字符的指针
}
}
return;
}
static void DeCode(int[]c)//解码
{
    int len = c.Length;
    ori = "";
    for (int i = 0; i < len; i++)//遍历码字
    {
        if (c[i] == -1) break;//结束标志，退出循环
        ori = ori + temparr[c[i] - 1];//在字典中查找该编码对应的字符串
    }
    return;
}
static void Main(string[] args)
{
    StreamReader sr = new StreamReader(@"G:\subject\csharp\ConsoleApplication1\in2.txt");//打开
    文件以读入
    StreamWriter sw = new StreamWriter(@"G:\subject\csharp\ConsoleApplication1\out2.txt", true);//
    打开文件以写入

    char[,] wordlist=new char[20,100];//二维数组，存储输入序列
    int[,] codelist = new int[20, 100];//二维数组，存储编码序列
    char[,] relist = new char[20, 100];//二维数组，存储译码序列
    string line = sr.ReadLine();
    int i = 0;
    int j = 0;
    int len1 = 0;
    while(line!=null&&line!="")

```

```

{
    string[] temp1 = line.Split(',');//逗号分割
    j=0;
    len1 = temp1.Length;
    string temp2 = "";
    for (j = 0; j < len1; j++)
    {
        wordlist[i, j] = Char.Parse(temp1[j]);//字符串转化为 int 数组
        temp2 = temp2 + wordlist[i, j]//拼接字符串
    }
    string temp3="";
    EnCode(temp2);
    for(j=0;j<len1;j++)//遍历字符
    {
        if (result_array[j] == -1)
        {
            temp3 = temp3 + "EOF";
            break;//遇到结束标志退出循环
        }
        codelist[i, j] = result_array[j]//保存编码结果
        temp3 = temp3 + result_array[j] + ",";
    }
    string temp4="";
    DeCode(result_array);
    char []t=ori.ToCharArray();

    for (j = 0; j < len1;j++ )//遍历字符
    {
        relist[i, j] = t[j]//保存译码结果
        temp4 = temp4 + t[j];
    }
    sw.Write("第" + (i + 1) + "组编码结果: ");
    sw.Write(temp3);
    sw.Write("\t 译码结果: ");
    sw.WriteLine(temp4);
}

```



```

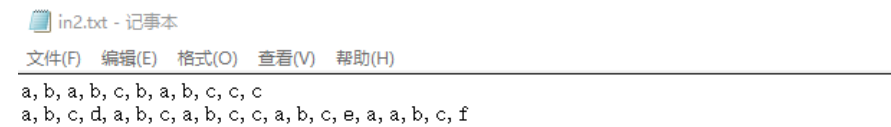
        line = sr.ReadLine();

        i++;
    }
    sr.Close();
    sw.Close();
}
}
}

```

1.4 运行结果

输入两组数据，如图 2 所示



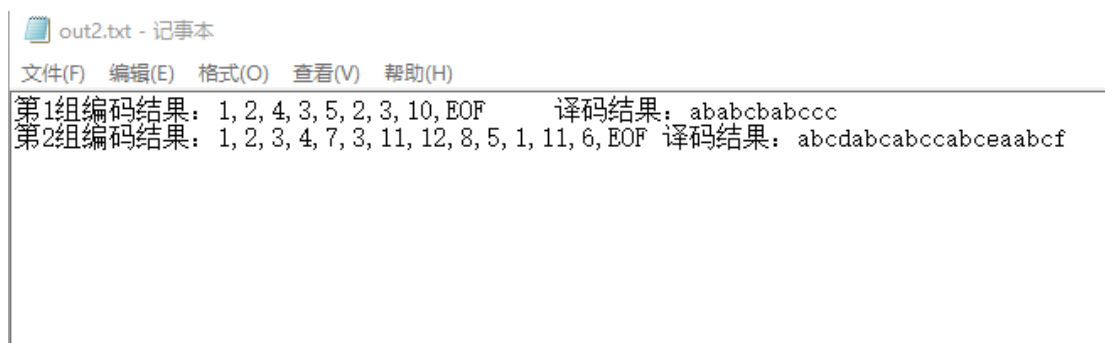
in2.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

a, b, a, b, c, b, a, b, c, c, c
a, b, c, d, a, b, c, a, b, c, c, a, b, c, e, a, a, b, c, f

图 2 LZW 编码输入数据

输出结果如图 3 所示



out2.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

第1组编码结果: 1, 2, 4, 3, 5, 2, 3, 10, EOF 译码结果: ababcbabccc
第2组编码结果: 1, 2, 3, 4, 7, 3, 11, 12, 8, 5, 1, 11, 6, EOF 译码结果: abcdabcbabccabceaaabcf

图 3 LZW 编码输出结果

结果分析：经过计算，编码结果正确，并能够正确地进行译码。

2.信道容量迭代计算

2.1 任务说明

输入：信道传递矩阵

输出：信道容量的近似值。

输入文件:in3.txt，含至少两组输入

输出文件:out3.txt，对每组输入的编码结果

2.2 设计思路

2.2.1 问题分析

首先确定一个允许的误差 δ ($0 < \delta < 1$)，然后进行迭代计算

①初始化信源分布 $P^{(0)} = (P_1, P_2, \dots, P_t)$ (一般初始化为均匀分布)，置迭代计数器 $k=0$;

$$\textcircled{2} \text{ 计算 } \varphi_{ji}^{(k)} = \frac{P_{ij} P_i^{(k)}}{\sum_i P_{ij} P_i^{(k)}} \quad i = 1, 2, \dots, s$$

$$\textcircled{3} \text{ 迭代计算 } P_i^{(k+1)} = \frac{\exp\left[\sum_j P_{ij} \ln \varphi_{ji}^{(k)}\right]}{\sum_i \left\{ \exp\left[\sum_j P_{ij} \ln \varphi_{ji}^{(k)}\right] \right\}} \quad i = 1, 2, \dots, r$$

$$\textcircled{4} \text{ 计算信道容量 } C^{(k+1)} = \ln \left\{ \sum_i \exp\left[\sum_j P_{ij} \ln \varphi_{ji}^{(k)}\right] \right\}$$

$$\textcircled{5} \text{ 如果 } \frac{|C^{(k+1)} - C^{(k)}|}{C^{(k+1)}} \leq \delta, \text{ 转向第}\textcircled{7}\text{步, 否则转向第}\textcircled{6}\text{步}$$

⑥置 $k + 1 \rightarrow k$ ，并转向第②步进行下一轮迭代

⑦输出 $C^{(k+1)}$ 的结果

2.2.2 实现原理与流程图

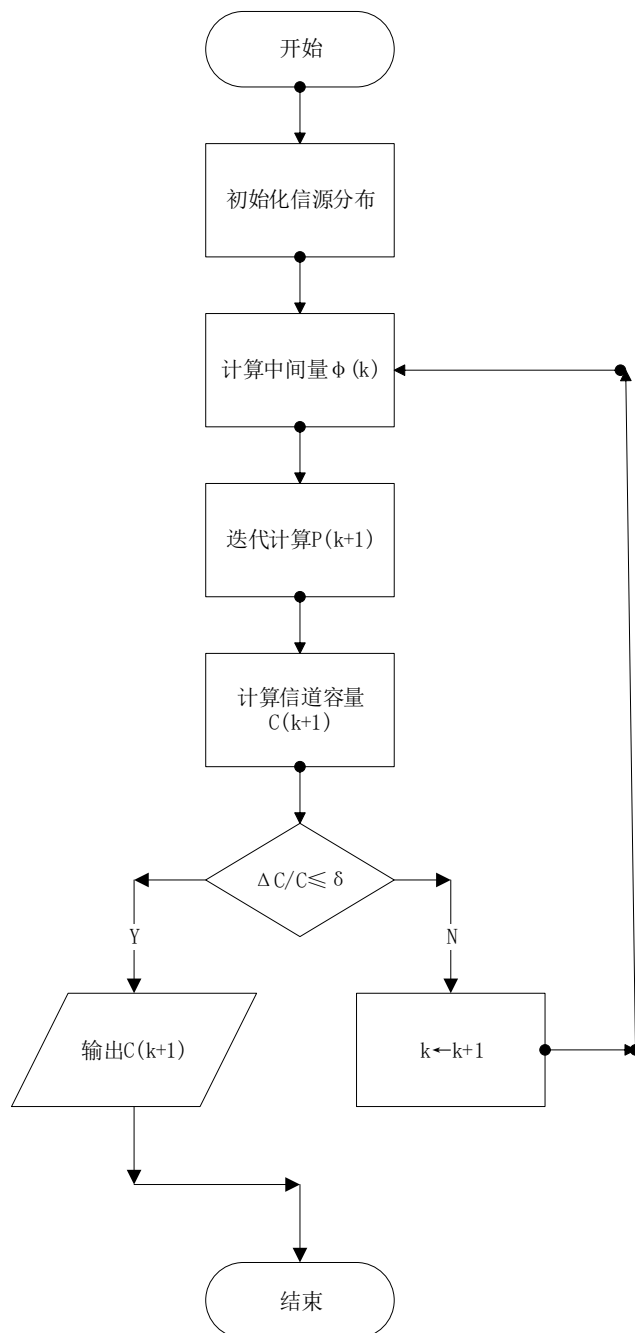


图 4 信道容量迭代流程图

2.3 实现源码

```
using System;  
using System.Collections.Generic;  
using System.Linq;
```

```

using System.Text;
using System.Threading.Tasks;
using System.IO;
namespace ConsoleApplication2
{
    class Program
    {
        public static double[] ps;//信源符号概率分布
        public static double[,] p;//传递矩阵
        public static double[,] phi;//中间量
        public static int r;//信源数目
        public static int s;//信宿数目
        public static double C;//信道容量
        public static double C_last;//上次计算出的信道容量
        public static double dieta = 0.01;//允许的误差
        static void Calculate()
        {
            //初始化数据
            for (int i = 0; i < r; i++)
                ps[i] = 1.0 / (double)r;//赋值
            //迭代计算
            bool flag = true;//标志
            phi = new double[20, 20];
            while(flag)
            {
                //计算 phi(j,i)
                double temp;
                for(int j=0;j<s;j++)
                {
                    temp = 0.0;
                    for (int i = 0; i < r; i++)
                        temp = temp + ps[i] * p[i, j];//分母
                    for (int i = 0; i < r; i++)
                        phi[j,i] = (ps[i] * p[i, j]) / temp;//计算结果
                }
            }
        }
    }
}

```

```

//计算 ps(i)
double pup = 0.0;//分子
double pdown = 0.0;//分母
//下面计算分母
bool flag_1 = true;
for(int i=0;i<r;i++)
{
    temp = 0.0;
    for (int j = 0; j < s; j++)
        if (phi[j, i] != 0)
            temp = temp + p[i, j] * (Math.Log(phi[j, i])/Math.Log(Math.E));
        else if (p[i, j] == 0 && phi[j, i] == 0)
            temp = temp + 0.0;
        else if(p[i,j]!=0&&phi[j,i]==0)
            flag_1 = false;
    if (flag_1)
        pdown = pdown + Math.Exp(temp);
    else
        pdown = pdown + 0.0;
}
//下面计算分子
for(int i=0;i<r;i++)
{
    bool flag_2 = true;
    temp = 0.0;
    for(int j=0;j<s;j++)
    {
        if (phi[j, i] != 0)
            temp = temp + p[i, j] * (Math.Log(phi[j, i])/Math.Log(Math.E));
        else if (p[i, j] == 0 && phi[j, i] == 0)
            temp = temp + 0.0;
        else if (p[i, j] != 0 && phi[j, i] == 0)
            flag_2 = false;
    }
    if (flag_2)

```

```

        pup = Math.Exp(temp) ;
    else
        pup = 0.0;
    ps[i] = pup / pdown;//最终结果
}

C = Math.Log(pdown)/Math.Log(2);//信道容量迭代结果
if (Math.Abs(C - C_last) / C <= dieta) flag = false;//小于允许误差 退出迭代计算
    C_last = C;//更新上次计算的信道容量的值
}
}

static void Main(string[] args)
{
    StreamReader sr = new StreamReader(@"G:\subject\csharp\ConsoleApplication2\in3.txt");
    StreamWriter sw = new StreamWriter(@"G:\subject\csharp\ConsoleApplication2\out3.txt", true);

    ps = new double[20];
    p = new double[20, 20];
    string line;
    int i = 0;
    int j = 0;
    r = 0;
    s = 0;
    while(true)
    {
        line = sr.ReadLine();
        if (line==null)
        {
            r = i;
            Calculate();
            sw.WriteLine("信道容量为:" + C);
            break;
        }
        if (line == "")
        {
            r = i;
            Calculate();

```

```

        sw.WriteLine("信道容量为:" + C);
        i = 0;
        j = 0;
        p = new double[20, 20];
        continue;
    }
    string[] temp1 = line.Split(',');
    s = temp1.Length;
    for (j = 0; j < s; j++)
        p[i, j] = double.Parse(temp1[j]);
    i++;
}
sr.Close();
sw.Close();
}
}
}

```

2.4 运行结果

输入数据如图 5 所示

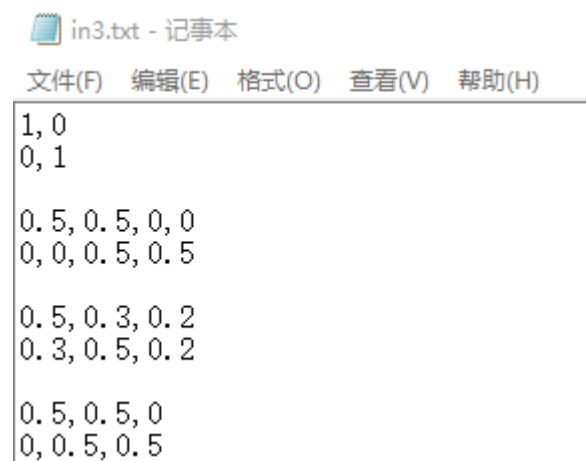
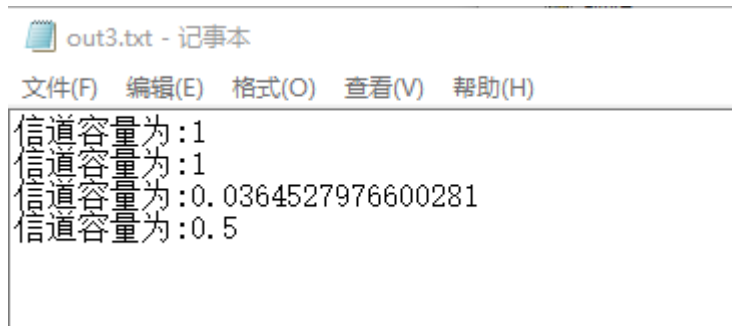


图 5 信道容量迭代输入

输出结果如图 6 所示



```
out3.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
信道容量为:1
信道容量为:1
信道容量为:0.0364527976600281
信道容量为:0.5
```

图 6 信道容量迭代输出结果

结果分析：经过检查，信道容量计算结果正确。

3.综合编码

3.1 任务说明

游程编码+HUFFMAN 编码+加密编码+信道编码及对应的译码

一个二元无记忆信源，0 符号的出现概率为 1/4, 1 符号的出现概率为 3/4。现要求对该信源连续出现的 n 个符号序列，进行游程编码/对游程编码结果进行 Huffman 编码/使用加密算法进行加密/进行信道编码；然后模拟信道的传输过程，并对收到的信息串进行信道译码/解密译码/Huffman 译码/游程译码。

假定，连续出现的 0 或 1 序列的长度不超过 16， n 不小于 256。

其中加密编码/解密译码可自主选择对称加密算法或非对称算法；

信道编码要求采用(7,4)系统循环码,其中, $g(x)=x^3+x+1$ ，译码采用简化的译码表；

信道为 BSC 信道， $p=10^{-1}$

输入：长为 n 的 0/1 串

输出：1. 游程编码结果，2. Huffman 编码结果，3. 加密编码结果 4. 信道编码结果

5. 模拟接收串，6. 信道译码结果，7. 解密编码结果 8. Huffman 译码结果

9. 游程译码结果

输入文件:in5.txt，含至少两组输入

输出文件:out5.txt，对每组输入的处理结果

3.2 设计思路

3.2.1 问题分析

对输入的比特串先进行游程编码，然后霍夫曼编码，然后加密编码，然后采用循环码进行信道编码，编码完成后进入模拟信道传输，进行循环码的解码，解密译码，霍夫曼译码，最后游程译码。

对于游程编码，把游程长度用自然数来标记，比特串就变成交替出现的表示游程长度的自然数序列；对于霍夫曼编码，首先计算信源符号的概率，按递减的次序排列，然后用 0 码和 1 码分配给概率最小的两个符号，并合并成一个新符号，将两个信源符号的概率之和作为新符号的概率，调整顺序使概率仍为递减排列，依次进行上述步骤，直到最后只剩两个信源符号为止，然后从最后一级缩减信源开始，依编码路径由后向前返回得到码字；加密编码采用 DES 加密的方式，加密前要先设置加密密钥；模拟信道传输采用随机数的方式，产生一个 10 以内的随机数，在任意选定一个数，如果随机数恰好等于这个数，则传输出现错误，对字符串对应位置取反，否则就是这一位传输无错误；译码的过程就是上述编码过程的逆运算。

3.2.2 实现原理与流程图

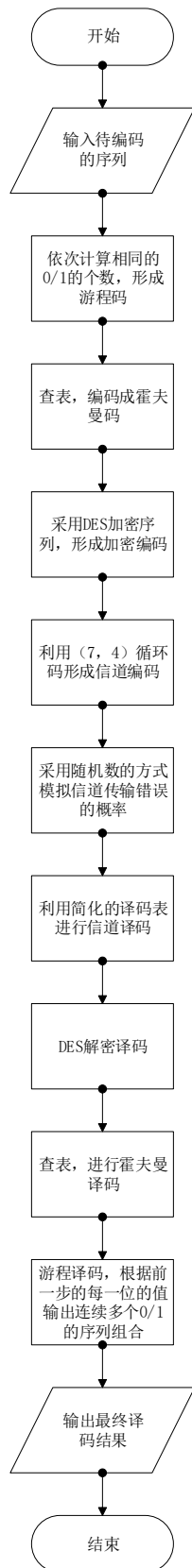


图7 综合编码流程图

3.3 实现源码

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Security.Cryptography;
using System.Web;
namespace ConsoleApplication3
{
    class Program
    {
        public static string str;
        public static int[] code_1;
        public static string[] code_2;
        public static string code_3;
        public static string code_4;
        public static string recv_0;
        public static string recv_1;
        public static string recv_2;
        public static int[] recv_3;
        public static string final_str;
        public static int maxlen ;

        public static string[] TAB0 = new string[] { "0", "11", "101", "1001", "10001", "100001",
"1000001", "10000001", "100000001", "1000000001", "10000000001", "100000000001",
"1000000000001",    "10000000000001",    "100000000000001",    "1000000000000001",
"1000000000000000" };//0 游程编码表

        public static string[] TAB1 = new string[] { "00", "10", "010", "110", "0110", "1110",
"01110", "11110", "011110", "111110", "011111", "111111", "01111101", "01111100",
"11111100", "111111011", "111111010" };//1 游程编码表

        public static string[] S_TAB = new string[] { "000", "001", "010", "011", "100", "101",
"110", "111" };//简化译码表（第一行）

        public static string[] E_TAB = new string[] { "0000000", "1000000", "0000010",
```

"0100000", "0000001", "0000100", "0001000", "0010000" };//简化译码表（第二行）

```
static void Encode1()//游程编码
{
    char[] str_arr = str.ToCharArray();
    code_1 = new int[str_arr.Length];
    int temp = 0;
    int count=0;
    int i = 0;
    if (str_arr[0] != '0')//保证开始时为 0 游程，如果不是补 0
        code_1[i++] = 0;
    while(temp<str_arr.Length-1)
    {
        if(str_arr[temp]!=str_arr[temp+1])//不相同
        {
            code_1[i++] = count + 1;//统计相同的数字的个数
            count = 0;//计数器清 0
            temp++;
            continue;
        }
        temp++;
        count++;
    }
    if(count>0)
        code_1[i++] = count+1;
    code_1[i++] = -1;//EOF
}
```

```
static void Encode2()//霍夫曼编码
{
    int len = code_1.Length;
    code_2 = new string[len];
    for (int i = 0; i < len; i++)//计算实际长度
        if (code_1[i] == -1)
        {
            len = i;
        }
    }
```

```

        break;
    }
    for (int i = 0; i < len; i++)
        if (i % 2 == 0)
            code_2[i] = TAB0[code_1[i]]; //查 0 游程编码表
        else
            code_2[i] = TAB1[code_1[i]]; //查 1 游程编码表
    }
    static void Encode3() //加密编码
    {
        string str = "";
        int i = 0;
        while (code_2[i] != null && code_2[i] != "")
            str = str + code_2[i++]; //转换成字符串
        string key = "abcabc"; //设置密钥
        DESCryptoServiceProvider des = new DESCryptoServiceProvider(); //des 加密
        byte[] inputByteArray;
        inputByteArray = Encoding.Default.GetBytes(str);
        des.Key =
        ASCIIEncoding.ASCII.GetBytes(System.Web.Security.FormsAuthentication.HashPasswordForSt
oringInConfigFile(key, "md5").Substring(0, 8));
        des.IV =
        ASCIIEncoding.ASCII.GetBytes(System.Web.Security.FormsAuthentication.HashPasswordForSt
oringInConfigFile(key, "md5").Substring(0, 8));
        System.IO.MemoryStream ms = new System.IO.MemoryStream();
        CryptoStream cs = new CryptoStream(ms, des.CreateEncryptor(),
        CryptoStreamMode.Write);
        cs.Write(inputByteArray, 0, inputByteArray.Length);
        StringBuilder ret = new StringBuilder();
        foreach (byte b in ms.ToArray())
        {
            ret.AppendFormat("{0:X2}", b);
        }
        code_3 = ret.ToString(); //以字符串形式存储
    }

```

```

}
static void Encode4()//信道编码
{
    code_4 = "";
    char[] lastcode = code_3.ToCharArray();
    int[] m = new int[] { 0, 0, 0, 0 };
    int[] c = new int[] { 0, 0, 0, 0, 0, 0 };
    for(int i=0;i<lastcode.Length;i++)
    {
        //16 进制转换成 2 进制
        if (lastcode[i] >= '0' && lastcode[i] <= '9')
        {
            int temp = lastcode[i] - '0';
            m[3] = temp % 2;
            m[2] = temp / 2 % 2;
            m[1] = temp / 4 % 2;
            m[0] = temp / 8 % 2;
        }
        else if (lastcode[i] >= 'A' && lastcode[i] <= 'F')
        {
            int temp = lastcode[i] - 'A' + 10;
            m[3] = temp % 2;
            m[2] = temp / 2 % 2;
            m[1] = temp / 4 % 2;
            m[0] = temp / 8 % 2;
        }
        else continue;
        //c=m*G,计算出结果
        c[0] = m[0];
        c[1] = m[1];
        c[2] = (m[0] + m[2]) % 2;
        c[3] = (m[0] + m[1] + m[3]) % 2;
        c[4] = (m[1] + m[2]) % 2;
        c[5] = (m[2] + m[3]) % 2;
    }
}

```

```

        c[6] = m[3];
        for (int j = 0; j < 7; j++)
            code_4 = code_4 + c[j]; //以字符串形式存储
    }
}

static void SendString() //模拟信道
{
    char[] str_temp = code_4.ToCharArray();
    int iSeed = 10;
    Random ro = new Random(iSeed); //产生随机种子
    long tick = DateTime.Now.Ticks;
    Random ran = new Random((int)(tick & 0xffffffffL) | (int)(tick >> 32));
    int len = str_temp.Length;
    for (int i = 0; i < len; i++)
    {

        int num = ran.Next() % 10; //生成 10 以内的随机数
        if (num == 2) //如果随机数等于某个数（比如 2），表明产生了错误，这样错误概率正好是 1/10
        {
            //产生错误，0 变成 1，1 变成 0
            if (str_temp[i] == '0') str_temp[i] = '1';
            else if (str_temp[i] == '1') str_temp[i] = '0';
        }
    }
    recv_0 = new string(str_temp); //按字符串形式存储
}

static void Decode1() //信道译码
{
    int []c = new int[7];
    int [] s = new int[3];
    char[] m = recv_0.ToCharArray();
    int[] o = new int[4];
    int len = m.Length;
    string s1 = "";

```

```

recv_1="";
for(int i=0;i<len;i=i+7)
{
    for (int j = i; j < i + 7; j++)
        c[j - i] = m[j] - '0';
    //计算伴随式
    s[0] = (c[2] + c[3] + c[4] + c[6]) % 2;
    s[1] = (c[1] + c[2] + c[3] + c[5]) % 2;
    s[2] = (c[0] + c[1] + c[2] + c[4]) % 2;
    string temp = "";
    temp = temp + s[0];
    temp = temp + s[1];
    temp = temp + s[2];
    for(int j=0;j<=7;j++)
        if(temp==S_TAB[j])
        {
            char[] t1 = E_TAB[j].ToCharArray();//查表，纠正错误
            int[] t1_1 = new int[7];
            for (int k = 0; k < 7; k++)
                t1_1[k] = t1[k] - '0';
            for (int k = 0; k < 7; k++)
                c[k] = (c[k] + t1_1[k]) % 2;
            break;
        }
    //恢复 4 位信息元
    o[0] = c[0];
    o[1] = c[1];
    o[2] = (c[0] + c[2]) % 2;
    o[3] = c[6];
    for (int j = 0; j < 4; j++)
        str = str + o[j];
}
//转换成 16 进制
char[] str_arr = str.ToCharArray();
int f=0;

```



```

for (int i = 0; i < str_arr.Length; i = i + 4)//4 位分组
{
    if (i + 1 >= str_arr.Length) f = (str_arr[i] - '0') * 8;
    else if (i + 2 >= str_arr.Length) f = (str_arr[i] - '0') * 8 + (str_arr[i + 1] - '0') * 4;
else if (i + 3 >= str_arr.Length) f = (str_arr[i] - '0') * 8 + (str_arr[i + 1] - '0') * 4 + (str_arr[i + 2] -
'0') * 2;

    else
f = (str_arr[i] - '0') * 8 + (str_arr[i + 1] - '0') * 4 + (str_arr[i + 2] - '0') * 2 + (str_arr[i + 3] - '0') * 1;
    if (f < 10)
        recv_1 = recv_1 + f;
    else
    {
        char cc=' ';
        switch(f)
        {
            case 10: cc = 'A'; break;
            case 11: cc = 'B'; break;
            case 12: cc = 'C'; break;
            case 13: cc = 'D'; break;
            case 14: cc = 'E'; break;
            case 15: cc = 'F'; break;
        }
        recv_1 = recv_1 + cc;
    }
}

static void Decode2()//解密译码
{

    DESCryptoServiceProvider des = new DESCryptoServiceProvider();//des 解密
    int len;
    string key = "abcabc";//设置密钥
    len = recv_1.Length / 2;
    byte[] inputByteArray = new byte[len];
    int x, i;

```

```

        for (x = 0; x < len; x++)
        {
            i = Convert.ToInt32(recv_1.Substring(x * 2, 2), 16);
            inputByteArray[x] = (byte)i;
        }

        des.Key =
        ASCIIEncoding.ASCII.GetBytes(System.Web.Security.FormsAuthentication.HashPasswordForSt
oringInConfigFile(key, "md5").Substring(0, 8));

        des.IV =
        ASCIIEncoding.ASCII.GetBytes(System.Web.Security.FormsAuthentication.HashPasswordForSt
oringInConfigFile(key, "md5").Substring(0, 8));

        System.IO.MemoryStream ms = new System.IO.MemoryStream();
        CryptoStream cs = new CryptoStream(ms, des.CreateDecryptor(),
        CryptoStreamMode.Write);

        cs.Write(inputByteArray, 0, inputByteArray.Length);
        string temp = Encoding.Default.GetString(ms.ToArray());
        char []tt=temp.ToCharArray();
        len = tt.Length;
        //转换成 2 进制
        for(i=0;i<len;i++)
        {
            string s1 = "";
            int t1 = (int)tt[i];
            while(t1>0)
            {
                int t2 = t1 % 2;
                t1 = t1 / 2;
                s1 = t2 + s1;
            }
            recv_2 = recv_2 + s1;//以字符串形式存储
        }
    }

    static void Decode3()//霍夫曼译码
    {
        recv_3 = new int[recv_2.Length];
    }

```

```

int cur_sta = 0;//当前游程
int pos = 0;
int i, j;
char[] s_arr = recv_2.ToCharArray();
for(i=0;i<s_arr.Length;)
{
    string s_cur = "";
    for(j=i;j<s_arr.Length;j++)
    {
        s_cur = s_cur + s_arr[j];
        bool flag=false;
        if (cur_sta == 0)
        {
            for (int k = 0; k < TAB0.Length; k++)//查表
            if (s_cur == TAB0[k])//找到了对应的码字
            {
                recv_3[pos++] = k;//记录
                flag = true;
                cur_sta = 1;//下一个是 1 游程
            }
        }
        else
        {
            for (int k = 0; k < TAB1.Length; k++)//查表
            if (s_cur == TAB1[k])//找到了对应的码字
            {
                recv_3[pos++] = k;//记录
                flag = true;
                cur_sta = 0;//下一个是 0 游程
            }
        }
        if(flag)
        {
            i = j + 1;//改变位置
            break;
        }
    }
}

```

```

        }
    }
    if (j >= s_arr.Length) break; //超出范围 退出循环
}
recv_3[pos++] = -1; //EOF
}

static void Decode4() //游程译码
{
    char cur; //当前字符
    char[] temp = new char[maxlen];
    int pos = 0;
    int loc = 0;
    int len = recv_3.Length;
    for (pos = 0; pos < len; pos++)
    {
        if (pos % 2 == 0) cur = '0'; //0 游程
        else cur = '1'; //1 游程
        for (int i = 0; i < recv_3[pos]; i++)
        {
            if (loc >= maxlen)
            {
                final_str = new string(temp);
                return;
            }
            temp[loc++] = cur; //按照个数加入字符串
        }
    }
    final_str = new string(temp);
}

static void Main(string[] args)
{
    StreamReader sr = new StreamReader(@"G:\subject\csharp\ConsoleApplication3\in5.txt");
    StreamWriter sw = new StreamWriter(@"G:\subject\csharp\ConsoleApplication3\out5.txt", true);
    string line = sr.ReadLine(); //读入数据

```

```

while(line!=null&&line!="")
{
    str = line;
    maxlen = str.Length;
    Encode1();//游程编码
    sw.Write("游程编码结果: ");
    for (int i = 0; i < code_1.Length; i++)
    {
        if (code_1[i] == -1) break;
        sw.Write(code_1[i]+" ");
    }
    Encode2();//霍夫曼编码
    sw.Write(" 霍夫曼编码结果:");
    for (int i = 0; i < code_2.Length;i++ )
    {
        if (code_2[i] == null || code_2[i] == "") break;
        sw.Write(code_2[i] + " ");
    }
    Encode3();//加密编码
    sw.Write(" 加密编码结果: " + code_3);
    Encode4();//循环码信道编码
    sw.Write(" 信道编码结果: " + code_4);
    SendString();//模拟信道传输
    sw.Write(" 模拟接收串: " + recv_0);
    Decode1();//循环码信道译码
    sw.Write(" 信道译码结果: " + recv_1);
    Decode2();//解密译码
    sw.Write(" 解密译码结果: " + recv_2);
    Decode3();//霍夫曼译码
    sw.Write(" 霍夫曼译码结果: ");
    for (int i = 0; i < recv_3.Length;i++ )
    {
        if (recv_3[i] == -1) break;
        sw.Write(recv_3[i]+" ");
    }
}

```

```

        Decode4();//游程译码
        sw.WriteLine(" 游程译码结果: " + final_str);
        line = sr.ReadLine();//读入下一个数据
    }
    sr.Close();
    sw.Close();
}
}
}

```

3.4 运行结果

输入数据:

```

11011111111111111011001101111111101111010101001111011000100111101001110110111
01111111111101111111101110100111101111010111011111011111011111111101010011
11110111101111111101110111110001011101110001011111110101111110111111101111110
10111111111111110110 (256 位)

```

输出结果:

最终译码结果:

```

01100010011111111111111111101000000111101111111110011111111110010111111111111
11011111111111111101101111001001111110111111011111111111101111111111101111110011101
111011100000011111001011010000011111111110111111111111111111111111111110111111
1111111111111111101

```

结果分析:

由于信道有 $p=10^{-1}$ 的出错率, 因此译码结果会与一开始输入的数据有所不同, 产生错误是正常现象。

4.设计体会

通过这次课程设计, 我对基本的一些编码和译码方法有了一个更熟练的掌握, 能够把学习到的理论知识更好地运用到实际的操作中来。此外, 这次课程设计也让我的逻辑思维能力得到了一个提升, 我能够更好地更有条理地去处理一系列的问题。在分析问题时, 我能够仔细深入地分析要解决的问题, 并选择一个最佳的解决方案。

在课程设计中, 我也难免会遇到各种各样的问题, 一开始是程序抛出异常, 后来程序运行结果并不正确。面对这些问题, 我合理地利用了编译环境的调试功能, 耐心仔细地把程序一步

一步调试出来，最终获得了满意的结果。这次课程设计也锻炼了我发现问题和解决问题的能力，一开始我的输入序列比较短，输出结果是正常的，当我把输入序列长度变长后，输出结果出现了难以预料的错误，于是我仔细分析调试，发现是加密和解密模块做的有问题，改进后对于一个较长的输入序列，输出结果也能得到满意的结果。

通过一个星期的课程设计，虽然过程不是一帆风顺的，但是最终程序的运行结果都能达到预期的效果，我懂得了要把书本上的理论知识运用到实际的动手操作中并在设计中发现和解决问题，这样才能有所提高。