信息论与编码实验

实验一　二维随机变量信息熵的计算

一、实验目的

掌握二变量多种信息量的计算方法。

二、实验要求

1．熟悉二变量多种信息量的计算方法，设计实验的数据结构和算法；

2．编写计算二维随机变量信息量的书面程序代码。

三、实验仪器、设备

1．计算机－系统最低配置 256M 内存、P4 CPU。

2．C++ 编程软件－ Visual C++ 7.0 （Microsoft Visual Studio 2003） Visual C++ 8.0（Microsoft Visual Studio 2005）

四、实验内容

离散二维随机变换熵的计算

说明：

（1）利用 random 函数和归一化方法构造一个二维离散随机变量（X，Y）；

（2）分别计算 X 与 Y 的熵、联合熵、条件熵：H（X）、H（Y）、H（X,Y）H（X|Y）、I（X|Y）；

（3）对测试通过的程序进行规范和优化；

（4）编写本次实验的实验报告。

五、实验原理

实验过程中涉及的各种熵的主要公式（或定义式）：

1、离散信源熵(平均不确定度/平均信息量/平均自信息量)

$$H(X)=\sum p(x_i)I(x_i)=-\sum p(x_i)\log p(x_i)$$

2、在给定某个 y_j 条件下，x_i的条件自信息量为 $I(x_i/y_j)$，X 集合的条件熵 H（X/y_j）为

$$H(X/y_j)=\sum_i p(x_i/y_j)I(x_i/y_j)$$

在给定Y条件下，X集合的条件熵H(X/Y)为：

$$H(X/Y)=\sum_j p(y_j)H(X/y_j)=\sum_{i,j} p(y_j)p(x_i/y_j)I(x_i/y_j)$$

$$=\sum_{i,j} p(x_iy_j)I(x_i/y_j)$$

相应地，在给定 X（即各个 x_i）的条件下，Y 集合的条件熵 H(Y/X)定义为:

$$H(Y/X)=\sum_{i,j} p(x_iy_j)I(y_j/x_i)=-\sum_{i,j} p(x_iy_j)\log p(y_j/x_i)$$

3、联合熵是联合符号集合 XY 上的每个元素对 x_iy_j 的自信息量的概率加权统计平均值，表示 X 和 Y 同时发生的不确定度。

$$H(X\quad Y)=\sum_{i,j}p(x_iy_j)I(x_iy_j)=-\sum_{i,j}p(x_iy_j)\log p(x_iy_j)$$

六、实验报告及总结

1、根据实验的内容，写出程序流程图。

2、分析并总结离散信源熵、条件熵、联合熵与信源统计分布或条件分布、联合分布之间的关系。

附：计算信源熵、联合熵和条件熵的程序代码清单：

```
#include<stdio.h>
#include<cmath>
#include<iomanip>
#include<time.h>
#include<iostream>
using namespace std;
void main()
{
    int k,n,t=0;
    double a[4][4],b=0,c=0;
    srand((unsigned)time(NULL));
    for(k=0;k<4;k++)
    {
        for(n=0;n<4;n++)
        {
            a[k][n]=rand()%100;
            t+=a[k][n];
        }
    }
    cout<<"从0到100间随机取得行列的random函数："<<endl;
    for(k=0;k<4;k++)
    {
        for(n=0;n<4;n++)
        {
            cout<<setw(5)<<a[k][n];
        }
         cout<<endl;
    }
    cout<<"函数归一化："<<endl;
    for(k=0;k<4;k++)
    {
        for(n=0;n<4;n++)
        {
            cout<<setw(12)<<a[k][n]/t;
        }
```

```
        cout<<endl;
    }
    cout<<"H(Y)计算："<<setw(20)<<"H(X)计算："<<endl;
    int e=1;
    for(k=0;k<4;k++)
    {
        double i=0,g=0;
        for(n=0;n<4;n++)
        {
            i+=(a[k][n]/t);
            g+=(a[n][k]/t);
        }
        cout<<"P(Y"<<k+1<<")："<<i<<setw(8)<<"P(X"<<e<<")："<<g<<endl;
        ++e;
        b-=(i*log(i)/log(2.0));
        c-=(g*log(g)/log(2.0));
    }
    cout<<"H(Y)=-∑p(Y)log p(Y)="<<b<<endl;
    cout<<"H(X)=-∑p(X)log p(X)="<<c<<endl;
    cout<<"联合熵H（X,Y）计算："<<endl;
    b=0;
    int r,u,h=0;
    for(k=0;k<4;k++)
    {
        for(n=0;n<4;n++)
        {
            if(a[k][n]!=0)
            {
                b-=((a[k][n]/t)*log(a[k][n]/t)/log(2.0));
            }
                else
                {
                    r=k,u=n;
                    h=1;
                    break;
                }
        }
    }
    if(h==0)
    cout<<"H(X,Y)=-∑∑p(X,Y)log p(X,Y)="<<b<<endl;
    else cout<<"P("<<r+1<<","<<u+1<<")为零，中断,无值"<<endl;
    cout<<"条件熵H（X|Y）计算："<<endl;
    b=0,h=0;
    for(k=0;k<4;k++)
    {
        double i=0;
```

```
            for(n=0;n<4;n++)
            {
                i+=(a[k][n]/t);
            }
            for(n=0;n<4;n++)
            {
                if(a[k][n]!=0)
                {
                    b-=((a[k][n]/t)*log((a[k][n]/t)/i)/log(2.0));
                }
                else {h=1;break;}
            }
        }
        if (h==0){cout<<"H(X|Y)=-∑∑P(X,Y)log(P(X,Y)/P(Y))="<<b<<endl;}
        else cout<<"P("<<r+1<<","<<u+1<<")为零，中断,无值"<<endl;
        cout<<"I（X|Y）计算："<<endl;
        if(h==0)cout<<"I（X|Y）=H(X)-H(X|Y)="<<c-b<<endl;
        else cout<<"P("<<r+1<<","<<u+1<<")为零，中断,无值"<<endl;
    }
```

实验二 简单信源编码方法实现----香农编码

一、实验目的

掌握香农编码方法。

二、实验要求

1．熟悉离散信源的编码方法，设计香农编码的数据结构和算法；

2．编写香农编码的书面程序代码。

三、实验仪器、设备

1．计算机－系统最低配置 256M 内存、P4 CPU。

2．C++ 编程软件－ Visual C++ 7.0 （Microsoft Visual Studio 2003） Visual C++ 8.0（Microsoft Visual Studio 2005）

四、实验内容

离散信源的香农编码方法

说明：

（1）根据香农编码算法实现香农编码；

（2）编写本次实验的实验报告。

五、实验原理

香农编码算法及步骤如下：

1、将信源发出的 N 个消息符号按其概率的递减次序依次排列。

$$p_1 \geq p_2 \geq \cdots p_N$$

2、按下式计算第 i 个消息的二进制代码组的码长，并取整。

$$-\log p(s_i) \leq l_i \leq -\log p(s_i) + 1$$

3、为了编成唯一可译码，首先计算第 i 个消息的累加概率

$$P_i = \sum_{k=1}^{i-1} p(s_k)$$

4、将累加概率 Pi （为小数）变成二进制数

5、除去小数点，并根据码长 li ，取小数点后 li 位数作为第 i 个消息的码字。

六、实验报告及总结

1、根据实验的内容，写出程序流程图。

2、总结香农编码的基本方法和过程，计算平均码长和编码效率。

附：香农编码过程的程序代码清单：

myRand 产生信源随机概率，myLength 计算码长，myQuickSort 对信源概率排序，主函数调用。

(1). myRand

```
// .h
#ifndef MYRAND_H_INCLUDED
#define MYRAND_H_INCLUDED

extern void myRand(double*, int);

#endif // MYRAND_H_INCLUDED
//.cpp
#include <time.h>
#include <stdlib.h>
void myRand(double *p, int n)
{
	int i=0;
	double sum =0;
	srand(time(NULL));
	for(i=0; i<n; i++)
	{

		*(p+i) = rand() % 100 + 1;
		sum += *(p+i);
	}
	for(i=0; i<n; i++)
	{
		*(p+i) /= sum;
	}

}
```

(2)myLength

```
//.h
#ifndef MYLENGTH_H_INCLUDED
#define MYLENGTH_H_INCLUDED

extern int myLength(double);

#endif // MYLENGTH_H_INCLUDED
//.cpp
#include <math.h>

int myLength(double p)
```

```
{
	int length;
	double result;
	if(p<=0)
		length = 1;
	result = log(1/p)/log(2);
	length = (int)result;
	if(result - length > 0)
		length += 1;
	return length;
}
```

(3)myQuickSort

```
//.h
#ifndef MYQUICKSORT_H_INCLUDED
#define MYQUICKSORT_H_INCLUDED

extern void myQuickSort(double *, int, int);

#endif // MYQUICKSORT_H_INCLUDED
//.cpp
int mySortBase(double *p, int left, int right)
{
	double m_flag;
	double tmp;
	m_flag = *(p + left);
	while(left < right)
	{
		while( m_flag > *(p+right) )
		{
			right--;
		}
		if(m_flag > *(p+right) );
		{
			tmp = *(p+right);
			*(p+right) = *(p+left);
			*(p+left) = tmp;
		}
		while( m_flag < p[left])
		{
			left++;
		}
		if(m_flag < p[left])
		{
			tmp = *(p+right);
			*(p+right) = *(p+left);
			*(p+left) = tmp;
```

```
            }
        }
        *(p+left) = m_flag;
        return left;
    }

    void myQuickSort(double *p, int left, int right)
    {
        int mid;

        if(left<right)
     {
         mid = mySortBase(p,left,right);
            myQuickSort(p,left,mid-1);

            myQuickSort(p,mid+1,right);
     }

    }
```

（4） **main**

```
#include <stdio.h>
#include "myQuickSort.h"
#include "myRand.h"
#include "myLength.h"

#define N 7



int main()
{
    double p[N] = {0.20, 0.19, 0.18, 0.17, 0.15, 0.10, 0.01};
    double q[N];
    int i,j;
    int k;          //编码 长度
    double b_c; // 二 进制 数
    // 生成 随机数
    puts("Rand:\n");
    myRand(p, N);
    for(i=0; i<N; i++)
    {
        printf("%4.2lf\t", p[i]);
    }
    puts("\n");
    // 排序
    puts("Sort:\n");
```

```
    myQuickSort(p, 0, N-1);
    for(i=0; i<N; i++)
    {
        printf("%4.2lf\t", p[i]);
    }
    puts("\n");
    //累加 概率
    puts("Q:");
    q[0] = 0;

    for(i=1; i<N; i++)
    {
        q[i] = q[i-1] + p[i-1];
    }
    for(i=0; i<N; i++)
    {
        printf("%4.2lf\t", q[i]);
    }
    puts("\n");
    //输出 编码
    for(i=0; i<N; i++)
    {
        printf("P[%d]: %4.2lf\t", i+1, p[i]);
        printf("Q[%d]: %4.2lf\t", i+1, q[i]);
        k = myLength(p[i]);
        b_c = 2*q[i];

        for(j=0; j<k; j++)
        {

            if(b_c >= 1)
            {
                putchar('1');
                b_c -= 1;
            }
            else
            {
                putchar('0');
            }
            b_c *= 2;
        }
        puts("\n");
    }

    return 0;
}
```

实验三 简单信源编码方法实现---- Huffman 编码

一、实验目的

掌握 Huffman 编码方法。

二、实验要求

1．熟悉离散信源的编码方法，重点是 Huffman 编码方法，设计 Huffman 编码的数据结构和算法；

2．编写 Huffman 编码的书面程序代码。

三、实验仪器、设备

1．计算机－系统最低配置 256M 内存、P4 CPU。

2．C++ 编程软件－ Visual C++ 7.0 （Microsoft Visual Studio 2003） Visual C++ 8.0（Microsoft Visual Studio 2005）

四、实验内容

离散信源的 Huffman 编、译码方法

说明：

（1）利用 random 函数构造一个一维离散随机变量分布 P（X）；

（2）构造离散随机变量的概率压缩表；

（3）根据概率压缩表构造 Huffman 编码表，并实现 Huffman 编码；

（4）完成 Huffman 译码；

（4）编写本次实验的实验报告。

五、实验原理

Huffman 编码算法及步骤如下：

① 将信源消息按照概率大小顺序排队。

② 按照一定的规则，从最小概率的两个消息开始编码。

③ 将经过编码的两个消息的概率合并，并重新按照概率大小排序，重复步骤②。

④ 重复上面步骤③，一直到合并的概率达到 1 时停止。这样便可以得到编码树状图。

⑤ 按照后出先编码的方式编程，即从数的根部开始，将 0 和 1 分别放到合并成同一节点的任意两个支路上，这样就产生了这组 Huffman 码。

哈夫曼编码的计算例子：

给定离散信源概率分布如下：

$$\begin{bmatrix} U \\ p(u) \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \\ 0.20 & 0.19 & 0.18 & 0.17 & 0.15 & 0.10 & 0.01 \end{bmatrix}$$

$$\begin{aligned} H(U) = & -0.2\log 0.2 - 0.19\log 0.19 - 0.18\log 0.18 - 0.17\log 0.17 \\ & -0.15\log 0.15 - 0.10\log 0.10 - 0.01\log 0.01 \quad = 2.61 \end{aligned}$$

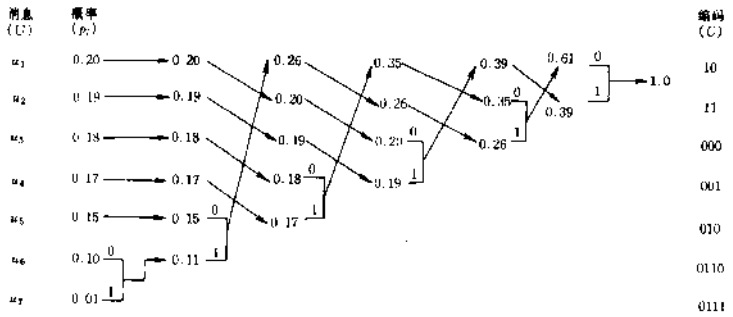
平均码长：

$$\overline{K} = \sum_{i=1}^{7} p(u_i)K_i = 0.2\times 2 + 0.19\times 2 + 0.18\times 3 + 0.17\times 3 + 0.15\times 3 + 0.10\times 4 + 0.01\times 4 = 2.72$$

编码效率

$$\eta = \frac{H(U)}{\overline{K}} = \frac{2.61}{2.72} = 95.96\%$$

对本信源进行了哈夫曼编码，可以得到如下的一种可能编码：



六、实验报告及总结

1、根据实验的内容，写出程序流程图。

2、总结哈夫曼编码的基本方法和过程，计算平均码长和编码效率。

附：哈夫曼编码过程的程序代码清单：

```
#include<stdio.h>
#include<cstdio>
#include<iomanip>
#include<time.h>
#include<cassert>
#include<string>
#include<iostream>
using namespace std;
double *a;
string *c;
struct elem
{
    double a2;
    double a3;
};
class stack
```

```
{
    int size;
    int top;
    elem *list;
public:
    stack(const int sz=0){size=sz;top=0;list=new elem[sz];}
    ~stack(){delete []list;}
    void clear(){top=0;}
    void push(const elem& item){assert(top<size);list[top++]=item;}
    elem pop(){assert(!isEmpty());return list[--top];}
    elem topValue() const{assert(!isEmpty());return list[top-1];}
    bool isEmpty() const{return top==0;}
};
void aa(int n)
{
    double w=0;
    a=new double[n];
    srand((unsigned)time(NULL));
    cout<<"随机生成归一化一维离散变量："<<endl;
    for(int i=0;i<n;i++)
    {
        a[i]=rand()%50;
        w+=a[i];
    }
    for(int i=0;i<n;i++)
    {
        a[i]=a[i]/w;
    }
    double p;
    for(int i=0;i<n-1;i++)
    {
        for(int j=n-2;j>=i;j--)
        {
            if(a[j]<a[j+1])
            {
                p=a[j+1];
                a[j+1]=a[j];
                a[j]=p;
            }
        }
    }
    cout<<"P(X):";
    for(int i=0;i<n;i++)
    {
        cout.precision(3);
        cout<<a[i]<<setw(8);
```

```
    }
}
void huffman(double *a,string *c,int n)
{
    elem mp;stack s(n);
    double *b;b=new double[n];for(int i=0;i<n;i++){b[i]=a[i];}
    double *d;d=new double[n];for(int i=0;i<n;i++){d[i]=i;}
    double *e;e=new double[n];for(int i=0;i<n;i++){e[i]=i;}
    string t;
    for(int m=n;m>=2;m--)
    {
        b[m-2]+=b[m-1];
        mp.a2=d[m-2];mp.a3=d[m-1];
        s.push(mp);
        double mp,mp1;
        for(int i=0;i<n-1;i++)
        {
            for(int j=n-2;j>=i;j--)
            {
                if(b[j]<b[j+1])
                {
                    mp=b[j+1]; mp1=d[j+1];
                    b[j+1]=b[j]; d[j+1]=d[j];
                    b[j]=mp;d[j]=mp1;
                }
            }
        }
        cout<<left<<setw(6)<<"\nP(X):";
        for(int i=0;i<m-1;i++)
        {
            cout.precision(3);
            cout<<setw(8)<<b[i];
        }
    }
    while(!s.isEmpty())
        {
            mp=s.pop();
            for(int i=0;i<n;i++)
            {
                if(mp.a2==e[i])
                {
                    t=c[i];
                }
            }
            for(int i=0;i<n;i++)
            {
```

```
                if(mp.a2==e[i])
                {
                    c[i]=t;
                    c[i]+="0";
                }
                else if(mp.a3==e[i])
                {
                    c[i]=t;
                    c[i]+="1";
                }
            }
        }
}
void main()
{
    int n;
    cout<<"输入N: ";
    cin>>n;
    c=new string[n];
    aa(n);
    huffman(a,c,n);
    cout<<endl;
    cout<<setw(6)<<"各项Huffman编码为:"<<endl;
    for(int i=0;i<n;i++)
    {
        cout.precision(3);
        cout<<setw(8)<<a[i];
        cout.precision(0);
        cout<<setw(n+4)<<c[i];
        cout<<endl;
    }
}
```