

Министерство образования и науки РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

Тульский государственный университет

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

## **АРИФМЕТИЧЕСКОЕ КОДИРОВАНИЕ**

Лабораторная работа № 4  
по курсу «Кодирование и сжатие данных»

Вариант №4

Выполнил:	студент группы 220601	_____	Белым А.А.
		(подпись)	
Проверил:	к. т. н., доцент каф. ИБ	_____	Гетманец В.М.
		(подпись)	

Тула 2014

## Цель работы

Целью работы является освоить алгоритм арифметического кодирования.

## Задание

Реализовать приложение для кодирования с помощью алгоритма арифметического кодирования.

## Описание метода

Пусть имеется некий алфавит, а также данные о частотности использования символов (опционально). Тогда рассмотрим на координатной прямой отрезок от 0 до 1.

Назовём этот отрезок рабочим. Расположим на нём точки таким образом, что длины образованных отрезков будут равны частоте использования символа, и каждый такой отрезок будет соответствовать одному символу.

Теперь возьмём символ из потока и найдём для него отрезок среди только что сформированных, теперь отрезок для этого символа стал рабочим. Разобьём его таким же образом, как разбили отрезок от 0 до 1. Выполним эту операцию для некоторого числа последовательных символов. Затем выберем любое число из рабочего отрезка. Биты этого числа вместе с длиной его битовой записи и есть результат арифметического кодирования использованных символов потока.

## Текст программы

Далее представлен текст программы, выполняющей кодирование и декодирование по методу арифметического кодирования.

```
#ifndef DICTIONARY_H
#define DICTIONARY_H
#include <QString>
#include <QDebug>
#include <QTextStream>
#include <QVector>
#include <QAbstractTableModel>

struct Dict{
    QChar sym;
    double prob, start, stop;
    Dict(QChar sym, double prob) :
        sym(sym), prob(prob), start(0), stop(prob)
    {}
};
```

```

    Dict():
        sym('\0'),prob(0),start(0),stop(prob)
    {}

};

class Dictionary:public QVector<Dict>,public QAbstractTableModel
{
public:
    Dictionary();
    int rowCount ( const QModelIndex & parent) const ;
    int columnCount ( const QModelIndex & parent ) const;
    QVariant data ( const QModelIndex & index, int role ) const;
    QVariant headerData ( int section, Qt::Orientation orientation, int role )
const;
    Qt::ItemFlags flags ( const QModelIndex & index ) const;

    friend QTextStream& operator >>(QTextStream& is,Dictionary& d);
};

bool operator ==(const Dict& d1,const QChar& c);

QTextStream& operator >>(QTextStream& is,Dict& d);

QTextStream& operator >>(QTextStream& is,Dictionary& d);

double acode(const QString& in,QVector<Dict> &d,QString &res);
QString adecode(const double a,const int n,QVector<Dict> &d);
#endif // DICTIONARY_H
#include "dictionary.h"
#include <cmath>
Dictionary::Dictionary()
{
}

int Dictionary::rowCount ( const QModelIndex & parent = QModelIndex() ) const {
    return 2;
}

int Dictionary::columnCount ( const QModelIndex & parent = QModelIndex() ) const{
    return size();
}

QVariant Dictionary::data ( const QModelIndex & index, int role = Qt::DisplayRole
) const{
    switch(role){
        case Qt::DisplayRole:
            if(index.column()<size()){
                switch(index.row()){
                    case 0:
                        return QVariant(this->at(index.column()).sym);
                    case 1:
                        return QVariant(this->at(index.column()).prob);
                    default:
                        return QVariant(QVariant::Invalid);
                }
            } else
                return QVariant(QVariant::Invalid);
        case Qt::TextAlignmentRole:
            return QVariant(Qt::AlignRight|Qt::AlignVCenter);
        default:
            return QVariant(QVariant::Invalid);
    }
}

```

```

}

QVariant Dictionary::headerData ( int section, Qt::Orientation orientation, int
role = Qt::DisplayRole ) const{
    switch(role){
        case Qt::DisplayRole:
            if(orientation==Qt::Horizontal){
                return QVariant(section+1);
            } else if (orientation==Qt::Vertical){
                switch(section){
                    case 0:
                        return QVariant(tr("Символ"));
                    case 1:
                        return QVariant(tr("Вероятность"));
                    default:
                        return QVariant(QVariant::Invalid);
                }
            } else
                return QVariant(QVariant::Invalid);
        default:
            return QVariant(QVariant::Invalid);
    }
}

}

Qt::ItemFlags Dictionary::flags ( const QModelIndex & index ) const{
    return Qt::ItemIsSelectable|Qt::ItemIsEnabled;
}

bool operator ==(const Dict& d1,const QChar& c){
    return d1.sym==c;
}

QTextStream& operator >>(QTextStream& is,Dict& d){
    QChar c; double p;
    is>>c; d.sym=c;
    ws(is);
    is>>p; d.prob=p;
    return is;
}

QTextStream& operator >>(QTextStream& is,Dictionary& d){
    Dict t;double s=0;
    if(d.size()){
        d.beginRemoveColumns(QModelIndex(),0,d.size()-1);
        d.clear();
        d.endRemoveColumns();
    }
    while(!is.atEnd()){
        is>>t>>ws;
        t.start=s;t.stop=s+t.prob;s+=t.prob;
        d.beginInsertColumns(QModelIndex(),d.size(),d.size());
        d<<t;
        d.endInsertColumns();
    }
    return is;
}

double acode(const QString& in, QVector<Dict> &d, QString &res){
    double start=0,stop=1;
    QTextStream out(&res);
    for(auto &i:in){
        auto p=qFind(d,i);

```

```

        double t;
        t=start+(stop-start)*p->start;
        stop=start+(stop-start)*p->stop;
        start=t;
        out<<"\"<p>sym<<"\" - [\"<start<<, \"<stop<<"]\n";
    }
    out<<"Entire text: \"<start<<\"\n";
    return start;
}

QString adecode(const double a, const int n, QVector<Dict> &d){
    double gstart=0,gstop=1;
    QString res,t;
    QTextStream out(&res);
    for (int N=0;N<n;N++){
        for(auto &p:d){
            double start=gstart,stop=gstop;
            double tt;
            tt=start+(stop-start)*p.start;
            stop=start+(stop-start)*p.stop;
            start=tt;
            qDebug()<<gstart<<" "<<gstop;
            if((a>=start)&&(a<stop)){
                t.append(p.sym);
                out<<"[\"<start<<, \"<stop<<] - \"<p>.sym<<\"\"<n";
                gstart=start;gstop=stop;
                break;
            }
        }
    }
    out<<t<<\"\n";
    return res;
}

#ifdef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "dictionary.h"
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

    void on_pushButton_4_clicked();

private:
    Ui::MainWindow *ui;
    Dictionary d;
};

#endif // MAINWINDOW_H

```

```

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QVector>
#include <QChar>
#include <QFile>
#include <QFileDialog>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent) ,
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->tableView->setModel(&d);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    QString path=QFileDialog::getOpenFileName(this);
    if(path!=""){
        QFile f(path);
        if(f.open(QIODevice::ReadOnly|QIODevice::Text)){
            QTextStream in(&f);
            in>>d;
            ui->probab_path->setText(path);
        }
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    ui->src_path->setText(QFileDialog::getOpenFileName(this));
}

void MainWindow::on_pushButton_3_clicked()
{
    ui->dst_path->setText(QFileDialog::getSaveFileName(this));
}

void MainWindow::on_pushButton_4_clicked()
{
    if(d.size()>0){
        QFile fin(ui->src_path->text()),fout(ui->dst_path->text());
        if(fin.open(QIODevice::ReadOnly|
QIODevice::Text)&&fout.open(QIODevice::WriteOnly|QIODevice::Text)){
            QTextStream in(&fin),out(&fout);
            ui->srctext->setPlainText(in.readAll());
            QString tmp;
            double code=acode(ui->srctext->toPlainText(),d,tmp);
            ui->dsttext->setPlainText(tmp);
            ui->dcddtext->setPlainText(adeocode(code,ui->srctext-
>toPlainText().length(),d));
            out<<ui->dsttext->toPlainText();
            out<<ui->dcddtext->toPlainText();
        }
    }
}

```

}  
}

## Тестовый пример

На рисунке 1 представлен результат кодирования при использовании возрастающего распределения символов, а на рисунке 2 — при использовании убывающего.

The screenshot shows a window titled "Кодирование и сжатие данных" (Encoding and compression of data). It contains several sections for configuring the encoding process.

**File with probabilities:** A text box shows the path "C:/Users/Wolf/Documents/cod4/prob1.txt" with a "Выбрать..." (Choose...) button next to it.

**Symbol and Probability Table:**

	1	2	3	4	5
Символ	s	i	w	m	-
Вероятность	0,5	0,2	0,1	0,1	0,1

**Input/Output Files:**

Входной файл: C:/Users/Wolf/Documents/cod4/s1.txt (Выбрать...)  
Выходной файл: C:/Users/Wolf/Documents/cod4/o.txt (Выбрать...)

**Encoding Button:** A button labeled "Кодировать" (Encode).

**Input Information:** A text box containing "swiss\_miss".

**Кодирование (Encoding):** A list of binary codes and their corresponding probability intervals:

- "i" - [0.375, 0.385]
- "s" - [0.375, 0.38]
- "s" - [0.375, 0.3775]
- "-" - [0.37725, 0.3775]
- "m" - [0.37745, 0.377475]
- "i" - [0.377463, 0.377468]
- "s" - [0.377463, 0.377465]
- "s" - [0.377463, 0.377464]
- Entire text: 0.377463

**Декодирование (Decoding):** A list of the same binary codes and intervals, followed by the reconstructed text:

- [0.375, 0.385] - "i"
- [0.375, 0.38] - "s"
- [0.375, 0.3775] - "s"
- [0.37725, 0.3775] - "-"
- [0.37745, 0.377475] - "m"
- [0.377463, 0.377468] - "i"
- [0.377463, 0.377465] - "s"
- [0.377463, 0.377464] - "s"
- swiss\_miss

Рисунок 1 — Результат при возрастающем распределении

Кодирование и сжатие данных

Файл с вероятностями

C:/Users/Wolf/Documents/cod4/prob1.txt

Выбрать...

	1	2	3	4	5
Символ	-	m	w	i	s
Вероятность	0,1	0,1	0,1	0,2	0,5

Входной файл

C:/Users/Wolf/Documents/cod4/s1.txt

Выбрать...

Выходной файл

C:/Users/Wolf/Documents/cod4/o.txt

Выбрать...

Кодировать

Входная информация

swiss\_miss

Кодирование

"i" - [0.615, 0.625]

"s" - [0.62, 0.625]

"s" - [0.6225, 0.625]

"-" - [0.6225, 0.62275]

"m" - [0.622525, 0.62255]

"i" - [0.622533, 0.622537]

"s" - [0.622535, 0.622537]

"s" - [0.622536, 0.622537]

Entire text: 0.622536

Декодирование

[0.615, 0.625] - "i"

[0.62, 0.625] - "s"

[0.6225, 0.625] - "s"

[0.6225, 0.62275] - "-"

[0.622525, 0.62255] - "m"

[0.622533, 0.622537] - "i"

[0.622535, 0.622537] - "s"

[0.622536, 0.622537] - "s"

swiss\_miss

Рисунок 2 — Результат при убывающем распределении

## Вывод

В данной работе рассмотрено арифметическое кодирование. Этот метод обеспечивает почти оптимальную степень сжатия с точки зрения энтропийной оценки кодирования Шеннона.

В отличие от алгоритма Хаффмана, метод арифметического кодирования показывает высокую эффективность для дробных неравномерных интервалов



распределения вероятностей кодируемых символов. Однако в случае равновероятного распределения символов метод арифметического кодирования приближается к префиксному коду Хаффмана и даже может занимать на один бит больше.

Была написана программа, реализующая кодирование и декодирование арифметическим кодированием.