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

Тульский государственный университет

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ОСНОВНЫЕ МЕТОДЫ ПОСИМВОЛЬНОГО КОДИРОВАНИЯ

Лабораторная работа № 2 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)	

Цель работы

Целью работы является освоить основные алгоритмы посимвольного кодирования.

Задание

- 1. Реализовать приложение для кодирования с помощью заданного в варианте алгоритма:
- вероятности появления символов алфавита должны храниться в одном файле, а последовательность, подлежащая кодированию, в другом;
 - закодированный текст должен сохраняться в файл;

Приложение должно:

- выводить полученные кодовые слова для всех символов алфавита;
- вычислять среднюю длину кодового слова;
- вычислять избыточность;
- проверять неравенство Крафта.
- 2. Реализовать приложение для декодирования с помощью заданного в варианте алгоритма:
- вероятности появления символов алфавита должны храниться в одном файле, а закодированная последовательность в другом;
 - раскодированная последовательность должна сохраняться в файл.
- 3. Рассмотреть 3 распределения вероятностей символов алфавита: равномерное, $P_1(A)$ и $P_2(A)$. Для каждого распределения получить кодовые слова, вычислить среднюю длину кодового слова, избыточность и проверить неравенство Крафта.
- 4. С помощью реализованных приложений исследовать зависимость получаемых кодовых слов от распределения вероятностей символов алфавита:
- смоделировать последовательность, символы которой подчиняются равномерному распределению. Закодировать эту последовательность при равномерном, $P_1(A)$ и $P_2(A)$ распределениях и вычислить длину каждой из трёх закодированных последовательностей;
- смоделировать последовательность, символы которой подчиняются распределению $P_1(A)$. Закодировать эту последовательность при равномерном,

- $P_1(A)$ и $P_2(A)$ распределениях и вычислить длину каждой из трёх закодированных последовательностей;
- смоделировать последовательность, символы которой подчиняются распределению $P_2(A)$. Закодировать эту последовательность при равномерном, $P_1(A)$ и $P_2(A)$ распределениях и вычислить длину каждой из трёх закодированных последовательностей;
- сделать выводы о связи распределения символов последовательности, вероятностях символов, используемых при кодировании, и длины получившегося кода.

Вари-	Алфавит	Вероятности	A	
ант	источника А	$P_1(A)$	$P_2(A)$	Алгоритм
4	$\{a,b,c,d\}$	{0.01,0.1,0.09,0.8}	$\left\{2^{-2},2^{-3},2^{-1},2^{-3}\right\}$	Шеннона- Фано

Описание метода

Ниже приведен алгоритм кодирования по методу Шеннона-Фано:

Инициализации. Все буквы алфавита записываются в порядке убывания вероятностей.

Цикл. Всю совокупность букв делят на две примерно равные по сумме вероятностей группы. Одной из них присваивают двоичный ноль, другой – двоичную единицу. Далее каждую группу вновь делят на две и повторяют цикл до тех пор, пока в группе не останется одна буква.

*Пример*Построим код Шеннона-Фано:

Буква $^{\mathcal{X}_{i}}$	$P(x_i)$					Код
a	0.35	1	1	stop		11
b	0.2	1	0	stop		10
c	0.15		1	1	stop	011
d	0.1	0	1	0	stop	010
e	0.1	0	0	1	stop	001
f	0.1		0	0	stop	000
		1-й шаг	2-й шаг	3-й шаг		

Заметим, что полученный код аналогичен коду Хаффмана для этого примера. В общем случае это не так.

Декодирование полученного кода аналогично декодированию, приведенному в предыдущих пунктах.

При построении кода Шеннона и Шеннона-Фано требуется упорядочить сообщения по убыванию вероятностей. Это, пожалуй, является наиболее трудоемкой частью алгоритмов.

Текст программы

Далее представлен текст программы, выполняющей кодирование и декодирование по методу Шеннона-Фанно.

```
#ifndef DICTIONARY H
#define DICTIONARY
#include <QString>
#include <QDebug>
#include <QTextStream>
#include <QVector>
#include <QAbstractTableModel>
struct Dict{
    QChar sym;
    double prob;
    QString code;
    Dict (QChar sym, double prob):
        sym(sym),prob(prob),code("")
    {}
    Dict():
        sym('\0'),prob(0),code("")
    {}
};
void shennon fano(QVector<Dict> &v,unsigned start,unsigned end);
void shennon fano(QVector<Dict> &v);
QString code (const QString &src, QVector < Dict > &v);
QString decode (const QString &src, QVector < Dict > &v);
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 )
    Qt::ItemFlags flags ( const QModelIndex & index ) const;
    friend QTextStream& operator >>(QTextStream& is,Dictionary& d);
};
QDebug operator <<(QDebug dbg,const Dict& d);
bool operator <(const Dict& d1,const Dict& d2);
bool operator == (const Dict& d1,const QChar& c);
bool operator ==(const Dict& d1,const QString& code);
QTextStream& operator >>(QTextStream& is,Dict& d);
QTextStream& operator >> (QTextStream& is, Dictionary& d);
#endif // DICTIONARY H
#include "dictionary.h"
Dictionary::Dictionary()
{
}
int Dictionary::rowCount ( const QModelIndex & parent = QModelIndex() ) const {
```

```
return 3;
}
int Dictionary::columnCount ( const QModelIndex & parent = QModelIndex() ) const{
    return size();
1
QVariant Dictionary::data ( const QModelIndex & index, int role = Qt::DisplayRole
) const{
    switch(role) {
        case Qt::DisplayRole:
            if(index.column() < size()) {</pre>
                switch(index.row()){
                case 0:
                     return QVariant(this->at(index.column()).sym);
                     return QVariant(this->at(index.column()).prob);
                     return QVariant(this->at(index.column()).code);
                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"));
                     return QVariant(tr("Вероятность"));
                     return QVariant(tr("Код"));
                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;
}
QDebug operator << (QDebug dbg,const Dict& d) {
    dbg.space() << "char: " << d.sym << "code: " << d.code;</pre>
    return dbg.space();
```

```
bool operator <(const Dict& d1,const Dict& d2) {</pre>
    return d1.prob>d2.prob||(d1.prob==d2.prob&&d1.sym<d2.sym);
}
bool operator == (const Dict& d1, const QChar& c) {
    return d1.sym==c;
}
bool operator ==(const Dict& d1,const QString& code){
    return d1.code==code;
}
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) {
    if(d.size()){
        d.beginRemoveColumns(QModelIndex(),0,d.size()-1);
        d.clear();
        d.endRemoveColumns();
    while(!is.atEnd()){
        is>>t>>ws;
        d.beginInsertColumns(QModelIndex(),d.size(),d.size());
        d<<t;
        d.endInsertColumns();
    shennon fano(d);
    return is;
}
void shennon fano(QVector<Dict> &v,unsigned start,unsigned end){
    double p=0, s=0;
    int n=start;
    for(unsigned i=start;i<end;i++) {</pre>
        p+=v[i].prob;
    for(unsigned i=start;i<end&&s<p/2;i++) {</pre>
        v[i].code+='1';
        s+=v[i].prob;
        n++;
    }
    for(unsigned i=n;i<end;i++){</pre>
        v[i].code+='0';
    }
    if(n-start>1)
        shennon_fano(v,start,n);
    if(end-n>1)
        shennon_fano(v,n,end);
}
void shennon fano(QVector<Dict> &v){
    qStableSort(v);
    shennon fano(v,0,v.length());
QString code (const QString &src, QVector < Dict > &v) {
    QString s;
    for(auto &i:src){
        auto p=qFind(v,i);
```

```
s+=p->code;
    return s;
QString decode (const QString &src, QVector < Dict > &v) {
    QString s,t;
    for(auto &i:src){
        t+=i;
        auto p=qFind(v,t);
        if(p!=v.end()){
            s+=p->sym;
            t="";
        }
    }
    return s;
#ifndef 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 code toggled(bool checked);
    void on pushButton 2 clicked();
    void on decode toggled(bool checked);
    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);
            double s1=0, s2=0, s3=0;
            for(auto &i:d){
                s1+=i.code.length()*i.prob;
                s2+=pow(0.5,i.code.length());
                s3+=i.prob*log2(i.prob);
            }
            s3+=s1;
            ui->avg->setText(tr("Средняя длина слова: %1").arg(s1));
            ui->kraft->setText(tr("Неравенство Крафта %1:
2<=1").arg(s2<=1?"верно":"неверно").arg(s2));
            ui->overhead->setText(tr("Избыточность: %1").arg(s3));
        }
    }
}
void MainWindow::on code toggled(bool checked)
    if (checked) {
        QString s;
        s=ui->src path->text();
        ui->src path->setText(ui->dst path->text());
        ui->dst path->setText(s);
        ui->srctext->setPlainText("");
        ui->dsttext->setPlainText("");
    }
}
void MainWindow::on pushButton 2 clicked()
{
    ui->src path->setText(QFileDialog::getOpenFileName(this));
}
void MainWindow::on decode toggled(bool checked)
    if(checked){
        QString s;
        s=ui->src path->text();
        ui->src path->setText(ui->dst path->text());
        ui->dst path->setText(s);
        ui->srctext->setPlainText("");
        ui->dsttext->setPlainText("");
}
```

```
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());
            gDebug()<<ui->srctext->toPlainText();
            if(ui->code->isChecked())
                ui->dsttext->setPlainText(code(ui->srctext->toPlainText(),d));
            else
                ui->dsttext->setPlainText(decode(ui->srctext->toPlainText(),d));
            qDebug() << decode (ui->srctext->toPlainText(),d);
            out<<ui->dsttext->toPlainText();
        }
   }
}
```

Тестовый пример

На рисунках 1, 2 и 3 приведены примеры работы программы для последовательности, символы которой подчиняются равномерному распределению, для равномерного распределения вероятностей, распределений P_1 и P_2 .

■ Кодирование и сжатие данных – □ ×						
Файл с вероятностями C:/Users/Wolf/Documents/cod2/prob1.txt Выбрать						
	1	1 2 3 4				
Символ	a	b	с	d		
Вероятность	0,25	0,25	0,25	0,25		
Код	11	10	01	00		
		14-E				
Средняя длина сКодирование			чность: 0	Неравенство Краф	рта верно: 1<=1	
Входной файл		ocuments/cod2/s1.tx	t		Выбрать	
Выходной файл	C:/Users/Wolf/Do	ocuments/cod2/o.txt			Выбрать	
		К	одировать			
Входная информ	ация					
aaabbbcccdddaa	aabbbcccddd					
Выходная информация						
11111110101001010100000011111111010100101						

Рисунок 1 — Последовательность равномерная и распределение равномерное

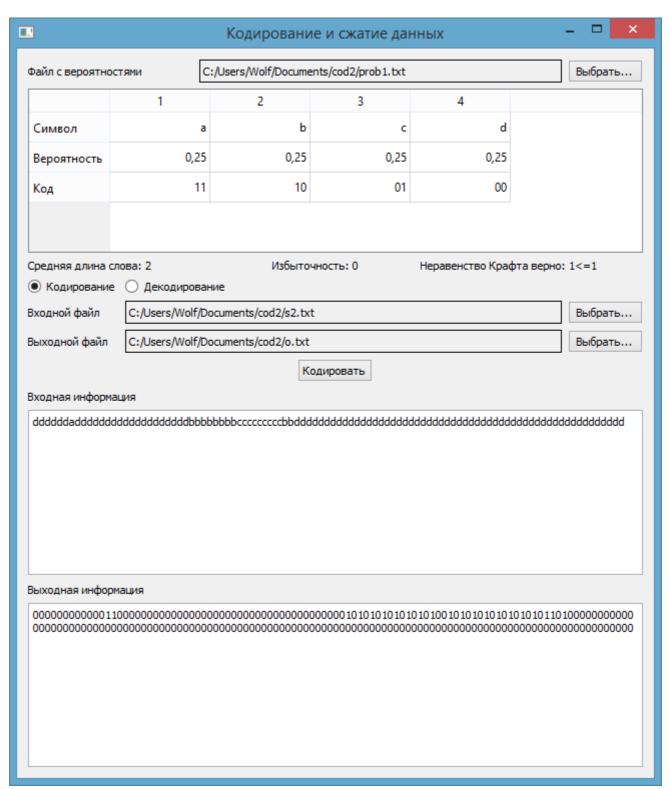


Рисунок 2 — Последовательность P_{I} и распределение равномерное

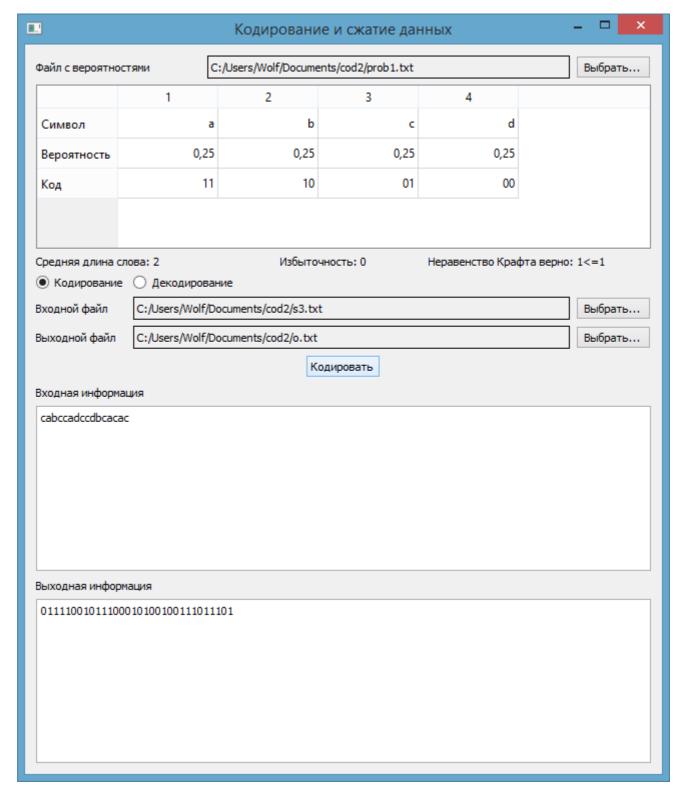


Рисунок 3 — Последовательность P_2 и распределение равномерное

На рисунках 4, 5 и 6 приведены примеры работы программы для последовательности, символы которой подчиняются распределению P_1 , для равномерного распределения вероятностей, распределений P_1 и P_2 .

■ Кодирование и сжатие данных – □ ×						
Файл с вероятностями C:/Users/Wolf/Documents/cod2/prob2.txt Выбрать						
	1	1 2 3 4				
Символ	d	b	с	a		
Вероятность	0,8	0,1	0,09	0,01		
Код	1	01	001	000		
Средняя длина	cnopa: 13	Избыто	чность: 0.331172	Неравенство Крас	hta Benuo: 1<=1	
Кодирование			4H0C1B1 01331172	Перавенство Крас	рта верно. 1<-1	
Входной файл	C:/Users/Wolf/Do	cuments/cod2/s1.tx	t		Выбрать	
Выходной файл	C:/Users/Wolf/Do	cuments/cod2/o.txt			Выбрать	
		К	одировать			
Входная информ	ация					
aaabbbcccdddaa	aabbbcccddd					
Выходная информация						
00000000001010100100100111100000000000101						

Рисунок 4 — Последовательность равномерная и распределение P_1

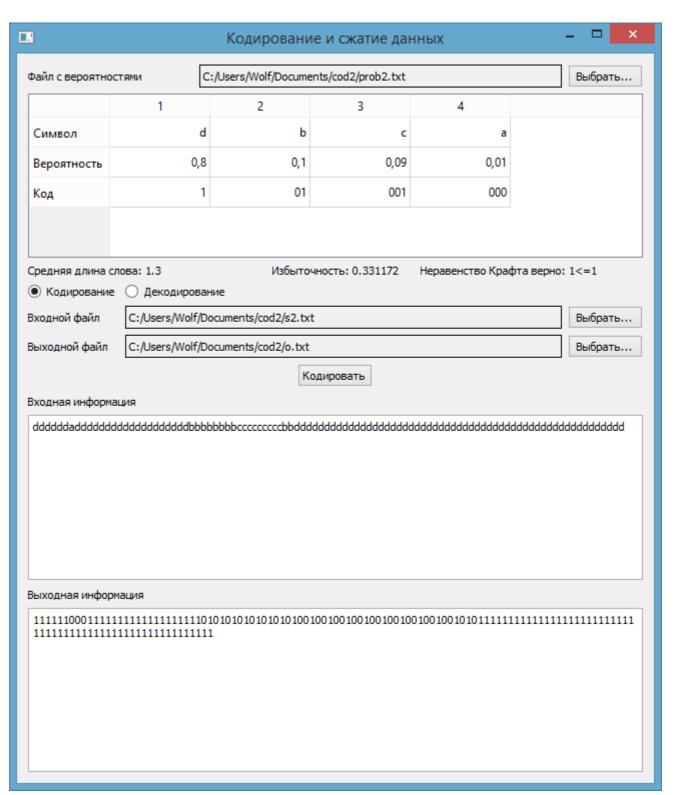


Рисунок 5 — Последовательность P_1 и распределение P_1

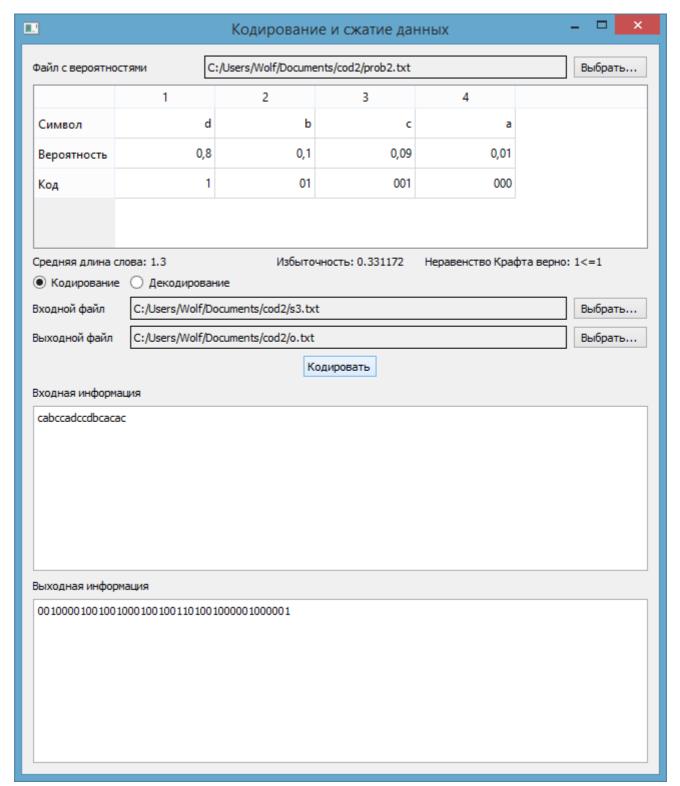


Рисунок 6 — Последовательность P_2 и распределение P_1

На рисунках 7, 8 и 9 приведены примеры работы программы для последовательности, символы которой подчиняются распределению P_2 , для равномерного распределения вероятностей, распределений P_1 и P_2 .

■ Кодирование и сжатие данных – □ ×						
Файл с вероятностями С:/Users/Wolf/Documents/cod2/prob3.txt Выбрать						
	1	2 3 4				
Символ	с	a	b	d		
Вероятность	0,5	0,25	0,125	0,125		
Код	1	01	001	000		
	,					
Средняя длина	cnoppy 1 75	14264.170	чность: 0	Неравенство Крас	http://popular.id=1	
Средняя длинаКодировани			AHOCIP. O	перавенство крас	рта верно: 1<=1	
Входной файл	C:/Users/Wolf/Do	cuments/cod2/s1.tx	t		Выбрать	
Выходной файл	C:/Users/Wolf/Do	cuments/cod2/o.txt			Выбрать	
		К	одировать			
Входная информ	пация					
aaabbbcccdddaa	aabbbcccddd					
Выходная информация						
010101001001001111000000000010101001001						

Рисунок 7 — Последовательность равномерная и распределение P_2

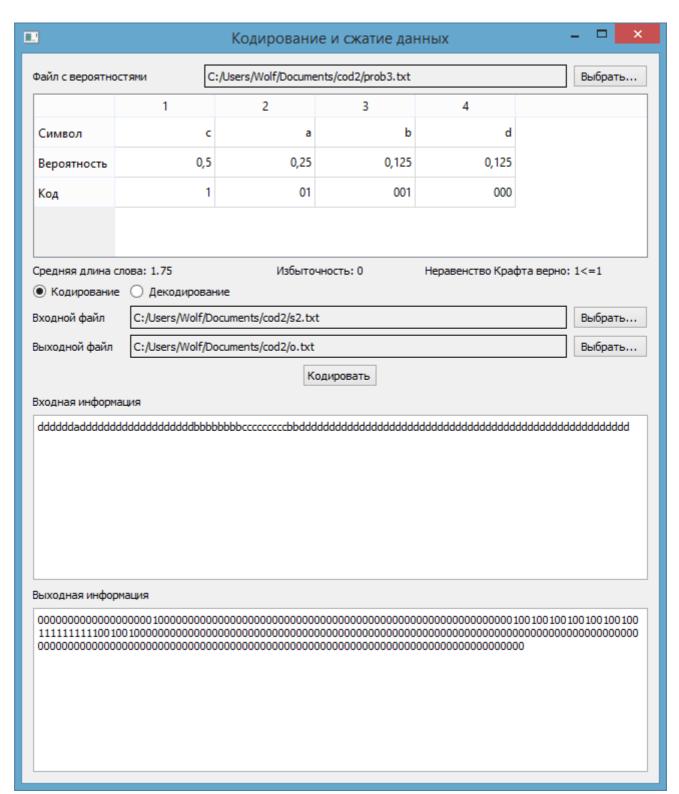


Рисунок 8 — Последовательность P_1 и распределение P_2

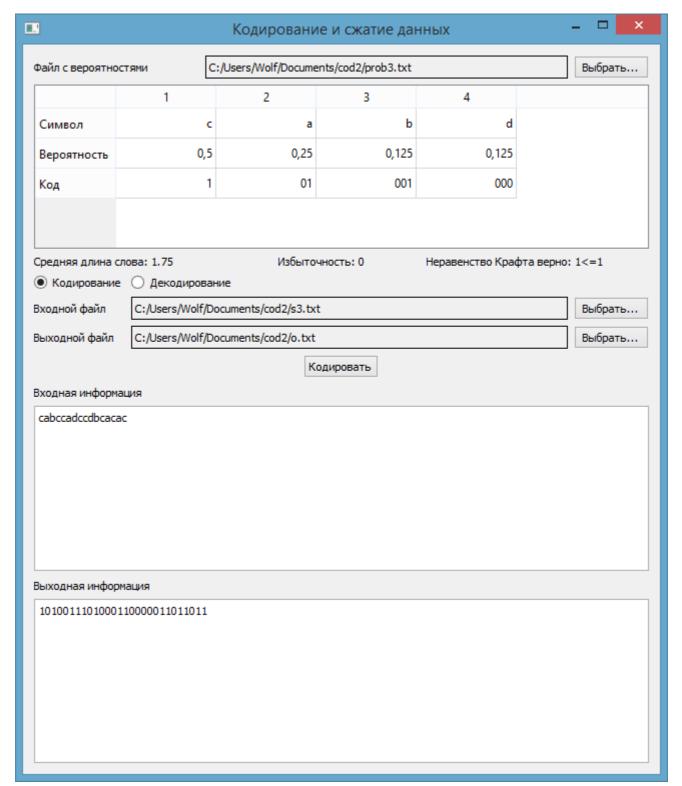


Рисунок 9 — Последовательность P_2 и распределение P_2

Вывод

В данной работе рассмотрен код Шеннона-Фано. Этот код является префиксным посимвольным. Недостатком посимвольных кодов является необходимость знать вероятность появление букв в тексте, иначе код оказывается неэффективным. Кроме того, код Шеннона-Фано менее эффективен, чем код

Хаффмана. Была написана программа, реализующая кодирование и декодирование кодом Шеннона-Фано.