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 № 3 по курсу «Кодирование и сжатие данных»

Вариант №13

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	ги появления букв	
ант	источника A	$P_1(A)$	$P_2(A)$	Алгоритм
13	$\{z,w,x\}$	{0.01,0.9,0.09}	$\left\{2^{-1},2^{-2},2^{-2}\right\}$	Гильберта- Мура

Описание метода

Ниже приведен алгоритм кодирования по методу Гильберта-Мура:

Инициализация. Сопоставим каждой букве кумулятивную вероятность

$$q_{\scriptscriptstyle m} = \sum_{\scriptscriptstyle i=1}^{\scriptscriptstyle m-1} p_{\scriptscriptstyle i}$$
 и вычислим $\sigma_{\scriptscriptstyle m} = q_{\scriptscriptstyle m} + \frac{p_{\scriptscriptstyle m}}{2}$.

 $I_m = \left\lceil -\log_2 \frac{p_m}{2} \right\rceil$ разрядов после запятой в двоичной записи числа σ_m . Средняя длина кодового слова: $\bar{I}(Y) \leq H(X) + 2$.

Пример. Рассмотрим тот же пример. Только для наглядности изменим последовательность в ансамбле.

Построим код Гильбера-Мура:

Буква x_i	$P(x_i)$	\boldsymbol{q}_i	σ_{i}	I_i	$\left[\sigma_i^{}\right]_2^{}$	Код
e	0.1	0.0	0.05	$\lceil 4.3 \rceil = 5$	0.0000110011001100	00001
c	0.15	0.1	0.17 5	$\lceil 3.7 \rceil = 4$	0.0010110011001100	0010
b	0.2	0.2 5	0.35	$\lceil 3.3 \rceil = 4$	0.0101100110011001	0101
d	0.1	0.4 5	0.5	$\lceil 4.3 \rceil = 5$	0.1000000000000000000000000000000000000	10000
a	0.35	0.5 5	0.72 5	$\lceil 2.5 \rceil = 3$	0.1011100110011001	101
f	0.1	0.9	0.95	$\lceil 4.3 \rceil = 5$	0.1111001100110011	11110

Средняя длина кодового слова $\bar{l}(Y) = 3.95$, и избыточность данного неравномерного кода: $r = \bar{l} - H(X) = 1.55$. Кроме того, неравенство Крафта имеет вид строгого неравенства. Это означает, что в целом данный код хуже всех рассмотренных для данного примера.

Текст программы

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

```
#ifndef DICTIONARY H
#define DICTIONARY H
#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 hilbert moore(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 )
const;
    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);</pre>
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);
                case 1:
                     return QVariant(this->at(index.column()).prob);
                case 2:
                     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"));
                case 2:
                     return QVariant(tr("Код"));
                default:
                     return QVariant(QVariant::Invalid);
                1
            } 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);</pre>
}
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) {
    Dict t;
    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();
    hilbert moore(d);
    return is;
}
void hilbert_moore(QVector<Dict> &v) {
    double sum=0,sigma=0;
    int l=0, code;
    for(auto &i:v){
        sigma=sum+i.prob/2;
        qDebug()<<sigma;</pre>
        l=int(ceil(-log2(i.prob)))+1;
        gDebug()<<1;</pre>
        sigma*=1<<1;
        qDebug()<<sigma;</pre>
        code=int(sigma);
        {
             QTextStream t(&i.code);
            t<<br/>bin<<code;
        for(int j=i.code.length();j<1;i.code='0'+i.code,j++);</pre>
        sum+=i.prob;
    }
}
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);
            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());
            qDebug()<<ui->srctext->toPlainText();
            if(ui->code->isChecked())
                ui->dsttext->setPlainText(code(ui->srctext->toPlainText(),d));
                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 .

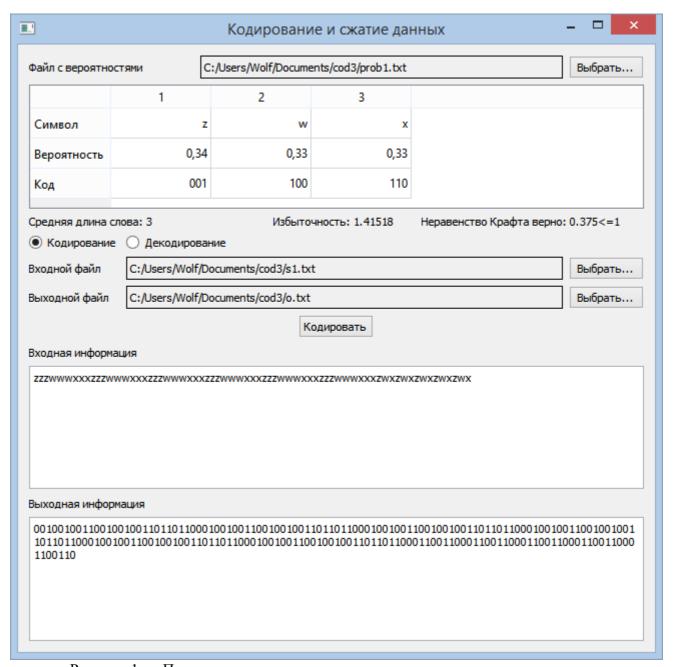


Рисунок 1 — Последовательность равномерная и распределение равномерное

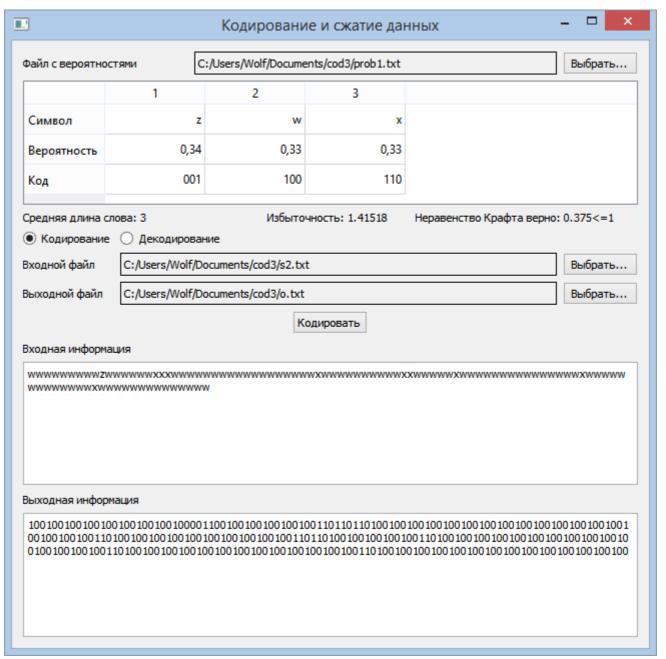


Рисунок 2 — Последовательность P_{I} и распределение равномерное

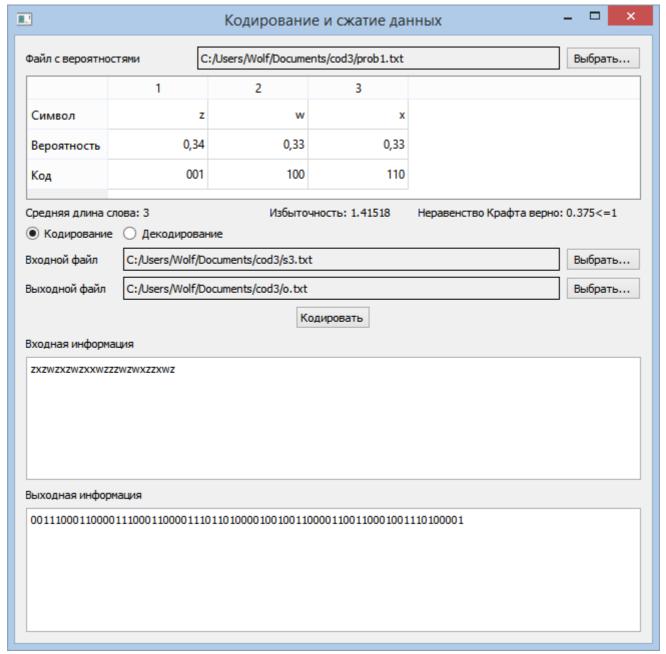


Рисунок 3 — Последовательность P_2 и распределение равномерное

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 .

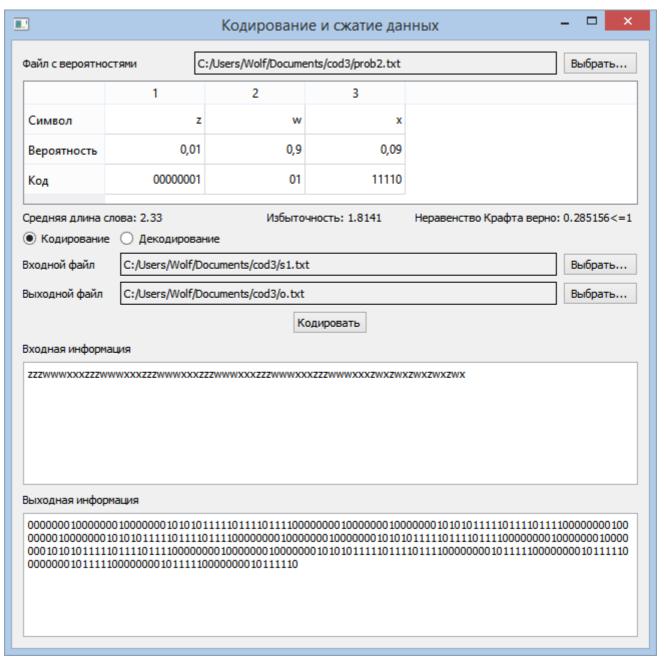


Рисунок 4 — Последовательность равномерная и распределение P_1

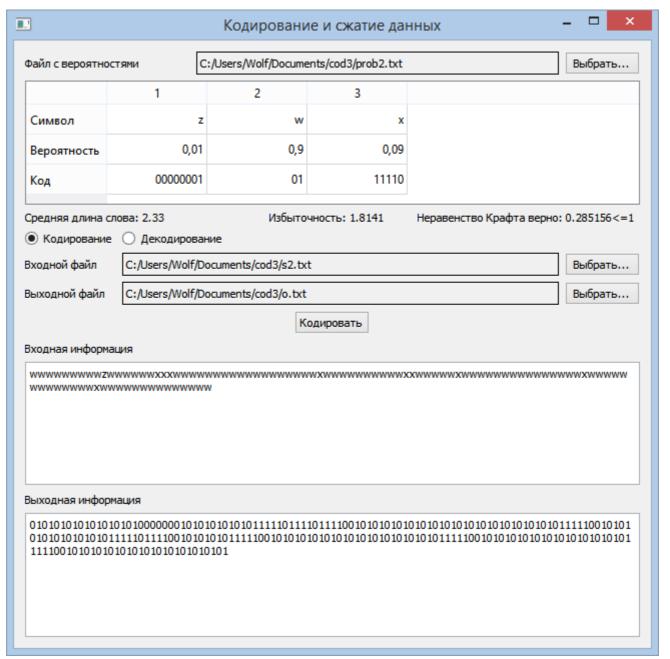


Рисунок 5 — Последовательность P_1 и распределение P_1

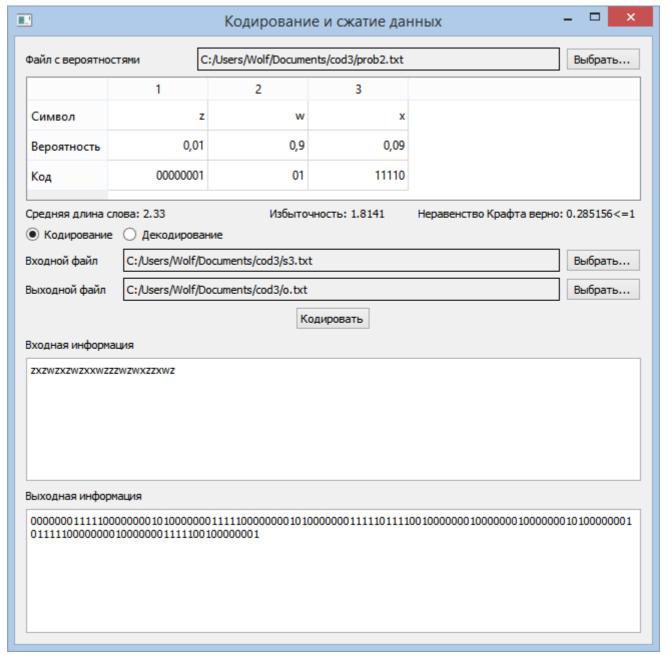


Рисунок 6 — Последовательность P_2 и распределение P_1

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 .

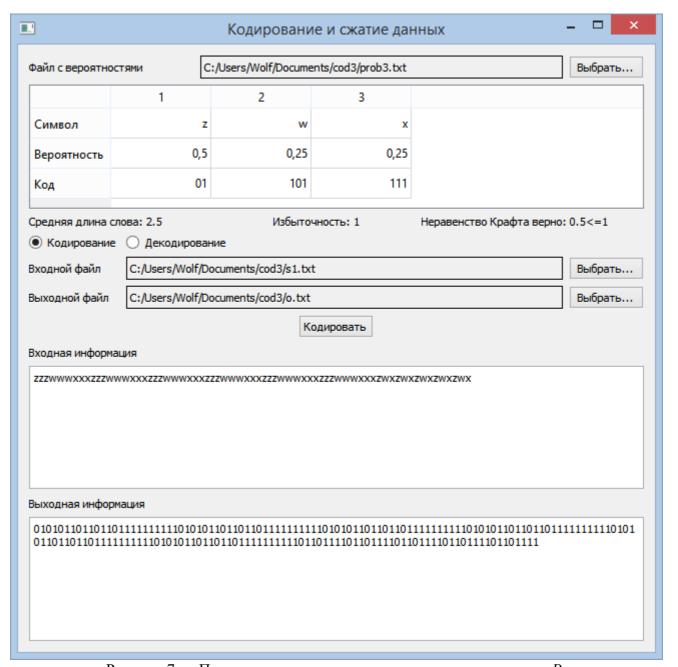


Рисунок 7 — Последовательность равномерная и распределение P_2

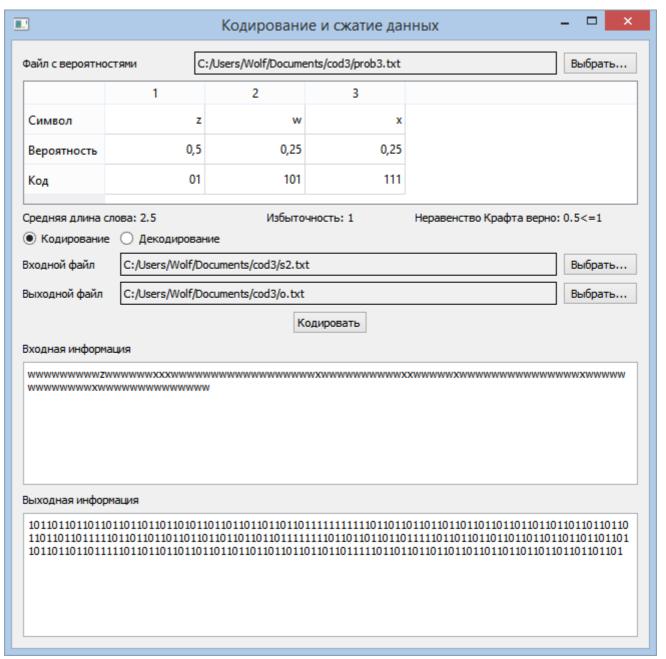


Рисунок 8 — Последовательность P_1 и распределение P_2

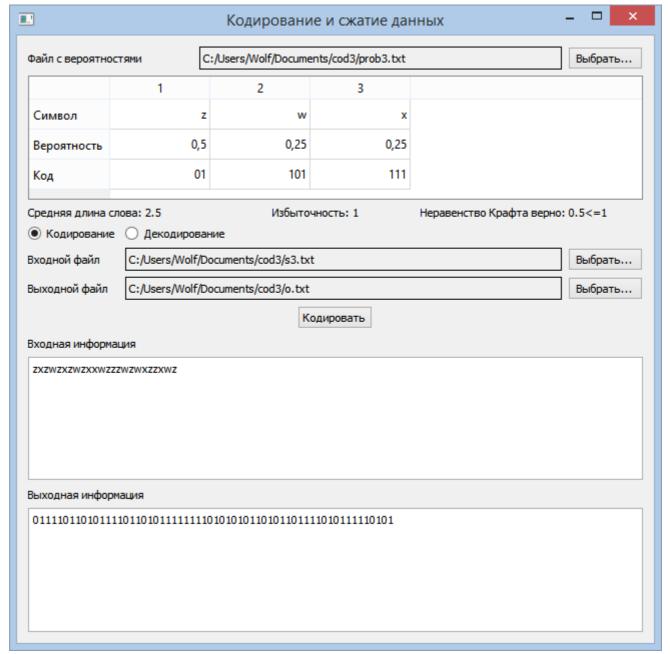


Рисунок 9 — Последовательность P_2 и распределение P_2

Вывод

В данной работе рассмотрен код Гильберта-Мура. Этот код является префиксным посимвольным. Недостатком посимвольных кодов является необходимость знать вероятность появление букв в тексте, иначе код оказывается неэффективным. Код Гильберта-Мура является одним их наименее эффективных префиксных кодов. Была написана программа, реализующая кодирование и декодирование кодом Гильберта-Мура.