МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Кафедра комп’ютерних систем, мереж та кібербезпеки

**Лабораторна робота №3**

РОЗРАХУНОК ОПТИМАЛЬНОГО З ВИКОРИСТАННЯМ МЕТОДУ ШЕННОНА-ФАНО

«Теорія інформації та кодування»

(назва дисципліни)

ХАІ.503.525СТ2.20О

Виконали: студенти 2 курсу групи № 000

напряму підготовки (спеціальності)

123-«Комп’ютерні системи та мережі»

(шифр і назва напряму підготовки (спеціальності))

студент\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали студента)

студент\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали студента)

студент\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали студента)

студент\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали студента)

Прийняв: к.т.н., доцент\_

\_\_\_\_\_\_\_\_\_\_\_\_Колісник М.О.\_\_\_

(посада, науковий ступінь, прізвище й ініціали)

Дата сдачи:

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2020

**Мета роботи:** освоїти методику побудови оптимального коду з використанням методу Шеннона-Фано.

**Завдання:**

Завдання 1. Визначити кількість інформації на символ повідомлення, складеного з цього алфавіту:

а) якщо символи алфавіту зустрічаються з рівними можливостями;

б) якщо ймовірності появи символів відповідають підкоряються наступним законом



Завдання 2: Визначити, наскільки недовантажені символи в другому випадку.

1.2. Число символів алфавіту k = (номер за списком) + 10. Тривалості символів  Вірогідність появи символів взяти з пункту 1.1.

Завдання 3: знайти швидкість передачі повідомлень, складених з таких символів?

1.3 Побудувати оптимальний код повідомлення з використанням методу Шеннона-Фано, описаного вище.

**Математична постановка задачі:** освоєння методики побудови оптимального коду з використанням методу Шеннона-Фано.

1. Поставити джерело дискретної інформації, описавши його в стандартній формі і охарактеризувавши його значенням ентропії Н (Х), що обчислюється в силу визначення ;

2. Скласти таблицю формування ефективного коду символів ИДИ, що має (n + l) рядок і не більше (n + 3) стовпців;

3. Виписати в порядку убування значень ймовірностей символи xi розмістивши їх у другому стовпці, а значення ймовірностей їх появи на виході ИДИ - в третьому;

4. Провести перший крок процедури ефективного кодування, для чого символи розділити на дві групи з можливо рівними сумами ймовірностей символів груп, після чого всім символам верхньої групи в старший розряд коду в четвертому стовпці вписати одиницю (1), а нижній групи - нуль (0 );

5. Провести другий крок процедури ефективного кодування, для чого символи кожної з двох груп, отриманих в п.4 алгоритму, розділити на дві підгрупи з можливо рівними сумами ймовірностей символів суміжних підгруп, після чого всім символам верхніх підгруп в розряд коду, наступний за старшим , в п'ятому стовпці вписати одиницю (1), а нижніх підгруп - нуль (0);

6. Провести наступні кроки процедури ефективного кодування за схемою, описаною в п.п.4 і 5 алгоритму, вписуючи щоразу в наступний розряд коду символів верхніх подподгрупп одиницю (1), а нижніх подподгрупп - нуль (0), причому процедуру проводити до тих пір, поки не будуть закодовані всі символи алфавіту, число ітерацій якої в залежності від числа п кодованих символів ИДИ і структури розподілу ймовірностей p (xi) по символам може становити від одиниці до n-1;

7. Виписати в останній правий стовпець таблиці значення чисел розрядів сформованих ефективних кодів, що представляють собою довжини l (K (xi) цих кодів з тим, щоб обчислити середнє на символ число двійкових розрядів коду за допомогою співвідношення



8. Оцінити ступінь близькості до ентропії Н (Х) ИДИ шляхом контролю виконання нерівності



Якщо нерівність (4.10) виконується, то здійснити перехід до п. 14 алгоритму, якщо нерівність (4.10) не виконується, то здійснити перехід до п.9 алгоритму;

9. Модифікувати джерело шляхом введення агрегованих символів характеризуються вірогідністю p (xl) появи на виході модифікованого ИДИ, рівними ;

10. Здійснити ефективне кодування агрегованих символів модифікованого ИДИ за схемою, представленої п.п.2 - 7 алгоритму;

11. Оцінити за допомогою співвідношення (4.9) середню довжину коду на агрегований символ, з наступним перерахунком її шляхом ділення одержаної довжини на число елементів в агрегованому символі - блоці;

12. Виконати перевірку виконання нерівності (4.10). Якщо нерівність (4.10) виконується, то здійснити перехід до п. 14 алгоритму, якщо немає, то здійснити перехід до п. 13 алгоритму;

13. Здійснити чергову модифікацію джерела дискретної інформації, збільшивши число вихідних символів в блоках на одиницю, і виконати п.п.2 - 7 і п.п. 10 - 12 алгоритму;

14. Передати таблиці отриманих ефективних кодів символів (в загальному випадку блоків символів) в алгоритмічну і апаратне середовище ефективного кодування.

**Хід роботи**

Команда №3

**Лістинг програми:**

**Form1.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_4\_TIC

{

public partial class Form1 : Form

{

public Dictionary<char, double> SDI;

Dictionary<char, string> codingTable;

char[] arrSortedKeys;

double[] arrDefaultVer;

double entropyInput;

double avarageCountBitsPerSymbol;

double closeCoef;

bool needModef;

public Form1()

{

InitializeComponent();

}

private void ButtCode\_Click(object sender, EventArgs e)

{

InitializeFieldAfterInputText();

CalculateVeroutnostiPoavleniy();

GlobalScript();

CodingText();

SetDGVSDI();

}

void SetDGVSDI()

{

dgvSDI.ColumnCount = 2;

dgvSDI.RowCount = SDI.Keys.Count();

char[] arrOfKeys = SDI.Keys.ToArray();

double[] arrOfValues = SDI.Values.ToArray();

dgvSDI.Columns[0].HeaderText = "Символ";

dgvSDI.Columns[1].HeaderText = "Вероятность";

dgvSDI.Columns[0].Width = 25;

for (int i = 0; i < dgvSDI.RowCount; i++)

{

if (needModef)

{

dgvSDI[0, i].Value = "X" + arrOfKeys[i];

}

else

{

dgvSDI[0, i].Value = arrOfKeys[i];

}

dgvSDI[1, i].Value = arrOfValues[i];

}

}

void CodingText()

{

string res = "";

for(int i = 0; i < tbForCoding.Text.Length; i++)

{

res += codingTable[tbForCoding.Text[i]];

}

tbForSending.Text = res;

}

void InitializeFieldAfterInputText()

{

SDI = new Dictionary<char, double>(tbForCoding.Text.Length);

codingTable = new Dictionary<char, string>(SDI.Keys.Count);

}

void GlobalScript()

{

SortSDIAndCalcEntropy();

if(needModef == false)

arrDefaultVer = SDI.Values.ToArray();

// MessageBox.Show(DivideForHalfs(0, SDI.Count).ToString());

arrSortedKeys = SDI.Keys.ToArray();

ShenonFanoMainFunc(0, SDI.Keys.Count);

//MessageBox.Show("After shenon fano");

CalcAvarageCountBitsPerSymb();

AnalizGettedCountAndInputEntropy();

}

void ShenonFanoMainFunc(int L, int R)

{

int half = DivideForHalfs(L, R);

// MessageBox.Show("half:" + half.ToString());

for(int j = L; j < half; j++)

{

codingTable[arrSortedKeys[j]] += "1";

// MessageBox.Show("for 1:" + arrSortedKeys[j] + "=" + codingTable[arrSortedKeys[j]]);

}

for (int j = half; j < R; j++)

{

codingTable[arrSortedKeys[j]] += "0";

// MessageBox.Show("for 1:" + arrSortedKeys[j] + "=" + codingTable[arrSortedKeys[j]]);

}

// MessageBox.Show("R - L = " + (R-L).ToString());

if (half - L > 1)

{

ShenonFanoMainFunc(L, half);

}

if (R - half > 1)

{

ShenonFanoMainFunc(half, R);

}

}

void AnalizGettedCountAndInputEntropy()

{

closeCoef = (avarageCountBitsPerSymbol - entropyInput) / entropyInput;

labelCloseCoef.Text = closeCoef.ToString("G4");

if(closeCoef > 0.15)

{

needModef = true;

ModificateSDI();

ChangeCodingTable();

SetDGVSDI();

GlobalScript();

}

}

void ChangeCodingTable()

{

Dictionary<char, string> tmpTable = new Dictionary<char, string>(0);

for (int i=0; i < SDI.Keys.Count;i++)

{

tmpTable.Add(char.Parse(i.ToString()), "");

}

codingTable = tmpTable;

}

void ModificateSDI()

{

Dictionary<char, double> tmpDict = new Dictionary<char, double>(SDI.Count \* 2);

double[] modArrVer = CalcModifiedVer();

MessageBox.Show("length:" + modArrVer.Length.ToString());

for (int i = 0; i < SDI.Count \* 2; i++)

{

tmpDict[char.Parse(i.ToString())] = modArrVer[i];

}

SDI = tmpDict;

}

double[] CalcModifiedVer()

{

double[] arrNewVer = new double[SDI.Count \* 2];

double[] arrOfValues = SDI.Values.ToArray();

for(int i =0; i < SDI.Count\*2;i++)

{

arrNewVer[i] = arrOfValues[i % SDI.Keys.Count];

}

for (int i = 0; i <= 1; i++)

{ // Math.Ceiling(Math.Log(SDI.Keys.Count, 2))

for (int j = 0; j < SDI.Count; j++)

{

MessageBox.Show("arrOfValues" + i.ToString() + "||" + arrDefaultVer[i]);

arrNewVer[j + i \* SDI.Count] \*= arrDefaultVer[i];

MessageBox.Show("arrNewVer " + (j + i \* SDI.Count).ToString() + "|" + arrNewVer[j + i \* SDI.Count].ToString() + "/n\n");

}

}

return arrNewVer;

}

void CalcAvarageCountBitsPerSymb()

{

avarageCountBitsPerSymbol = 0;

foreach(var ch in codingTable.Keys)

{

avarageCountBitsPerSymbol += SDI[ch] \* codingTable[ch].Length;

}

labelAvarageCountBitsPerSymbol.Text = avarageCountBitsPerSymbol.ToString("G4");

}

void CalculateVeroutnostiPoavleniy()

{

// calculate veroutnosti

foreach (char el in tbForCoding.Text)

{

if (!SDI.Keys.Contains(el))

{

SDI.Add(el, tbForCoding.Text.Count(charI => charI == el) / (double)tbForCoding.Text.Length);

}

if(!codingTable.Keys.Contains(el))

{

codingTable.Add(el, "");

}

}

}

void SortSDIAndCalcEntropy()

{

// calculate entropy

entropyInput = 0;

SDI = SDI.OrderByDescending(x => x.Value).ToDictionary(x => x.Key, x => x.Value);

foreach (var veroytnost in SDI.Values)

{

entropyInput += veroytnost \* Math.Log(veroytnost, 2);

}

entropyInput = 0 - entropyInput;

labelInputEntropy.Text = entropyInput.ToString("G4");

}

int DivideForHalfs(int L, int R)

{

double[] arrSDIValues = SDI.Values.ToArray();

double halfValue = 0;

int i = L;

int m = i;

for(int j =L; j< R;j++)

{

halfValue += arrSDIValues[j];

}

halfValue /= (double)2;

// MessageBox.Show(halfValue.ToString("G4"));

double counterValue = 0;

while ((counterValue + arrSDIValues[i] < halfValue) && (i < R))

{

counterValue += arrSDIValues[i];

i++;

}

double deltaFromTopSide = Math.Abs(halfValue - counterValue);

// MessageBox.Show("Top" + deltaFromTopSide.ToString("G4"));

double deltaFromBottomSide = Math.Abs((counterValue + arrSDIValues[i]) - halfValue);

// MessageBox.Show("bottom"+deltaFromBottomSide.ToString("G4"));

if ((deltaFromTopSide > deltaFromBottomSide) &&(i<R))

{

// MessageBox.Show("return i++");

return (++i);

}

else

{

// MessageBox.Show("return i");

return i;

}

}

private void ButtSetVeroatnosti\_Click(object sender, EventArgs e)

{

InitializeFieldAfterInputText();

CalculateVeroutnostiPoavleniy();

FormSetVeroatnosti f2 = new FormSetVeroatnosti(this);

f2.ShowDialog();

GlobalScript();

SetDGVSDI();

}

private void ButtSend\_Click(object sender, EventArgs e)

{

tbForReceivingCode.Text = tbForSending.Text;

}

private void ButtDecode\_Click(object sender, EventArgs e)

{

string decodedStr = "";

int pointerEl = 0, i = 0;

while (pointerEl < tbForReceivingCode.Text.Length)

{

for(int j =1; j < tbForReceivingCode.Text.Length; j++)

{

i = IsExistCode(tbForReceivingCode.Text.Substring(pointerEl, j));

//MessageBox.Show("substr:" + tbForReceivingCode.Text.Substring(pointerEl, j) + "| i =" + i.ToString());

if ( i >= 0 )

{

decodedStr += codingTable.Keys.ToArray()[i];

// MessageBox.Show("keys:" + codingTable.Keys.ToArray() );

pointerEl += j;

break;

}

}

//MessageBox.Show("decoded str:" + decodedStr);

}

tbForDecoding.Text = decodedStr;

}

int IsExistCode(string code)

{

string[] arrCodes = codingTable.Values.ToArray();

for(int i =0; i < codingTable.Values.Count; i++)

{

if(arrCodes[i] == code)

{

return i;

}

}

return -1;

}

}

}

**FormSetVeroatnosti.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_4\_TIC

{

public partial class FormSetVeroatnosti : Form

{

Dictionary<char, double> forSetVeroatnosi;

Form1 f1;

public FormSetVeroatnosti(Form1 form1)

{

InitializeComponent();

f1 = form1;

dgvSet.ColumnCount = 2;

dgvSet.RowCount = f1.SDI.Keys.Count();

char[] arrOfKeys = f1.SDI.Keys.ToArray();

for (int i = 0; i < dgvSet.RowCount; i++)

{

dgvSet[0, i].Value = arrOfKeys[i];

}

dgvSet.Columns[0].HeaderText = "Символ";

dgvSet.Columns[1].HeaderText = "Вероятность";

}

private void Button1\_Click(object sender, EventArgs e)

{

forSetVeroatnosi = new Dictionary<char, double>(0);

for(int i = 0; i< f1.SDI.Keys.Count; i++)

{

forSetVeroatnosi.Add(char.Parse( dgvSet[0, i].Value.ToString()), double.Parse(dgvSet[1, i].Value.ToString()));

}

f1.SDI = forSetVeroatnosi;

this.Close();

}

}

}

**Результати роботи програми**

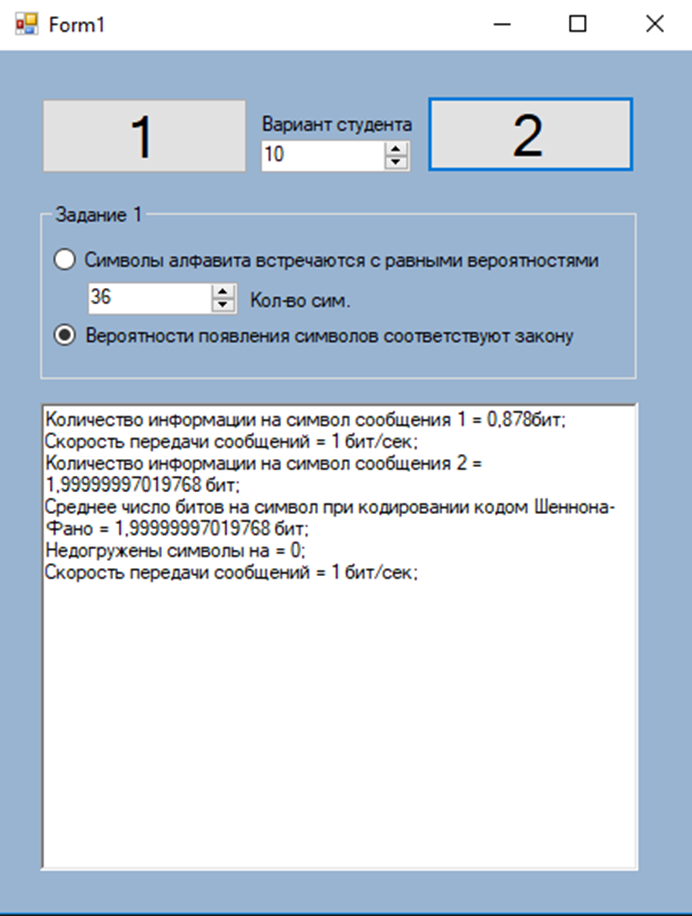


Рисунок 1 – Виконання першої задачі

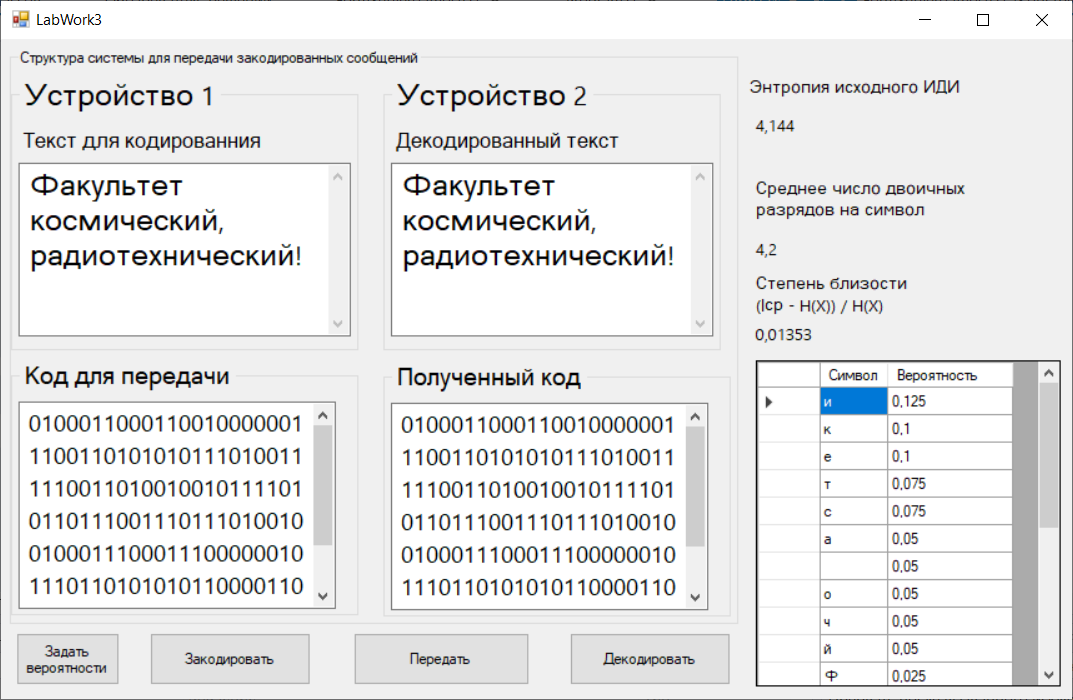


Рисунок 2 – Виконання другої задачі

Рисунок 3 – Графік

**Висновки:**

В ході лабораторної роботи було розроблено додаток на C# з використанням технології Windows Forms. Ми навчилися будувати оптимальний код з використанням методу Шенона-Фано та визначати кількості інформації на символ в повідомленні.

Також ми відточили навички застосування формул Шеннона і Хартлі на практиці й відточили навички програмування на мові C # і розробки презентацій.