МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9 НА ТЕМУ:**

**Сжатие/распаковка данных на основе статистических методов**

Ф.И.О.

Трофимчук Михаил Витальевич

Минск 2022

**Цель:** приобретение практических навыков использования статистических методов Шеннона − Фано и Хаффмана (ShannonFano and Huffman coding) для сжатия/распаковки данных.

**Теоретические сведения**

До появления уже упоминавшихся работ К. Шеннона кодирование символов алфавита при передаче сообщения по каналам связи осуществлялось одинаковым количеством битов, получаемым по формуле Хартли (см. формулу (2.2)). Позднее начали появляться способы, кодирующие символы разным числом битов в зависимости от вероятности появления их в тексте, подтверждение чему мы получили при выполнении лабораторной работы № 2. Таким образом, за счет использования для каждого значения байта кодов ASCII (символа алфавита) кода различной длины в соответствии с частостью (вероятностью появления этого символа в сообщении) можно значительно уменьшить общий размер данных.

Эта базовая идея лежит в основе алгоритмов статистических (вероятностных) методов сжатия: Шеннона − Фано и Хаффмана.

Статистические алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся символов алфавита или конкретного сообщения. В первом случае метод считается статическим статистическим, во втором – динамическим статистическим: вероятностные свойства символов подсчитываются для конкретного сообщения или потока данных.

Частота или вероятность появления того или иного символа алфавита в произвольном сообщении, лежащая в основе алгоритмов, дали название этим алгоритмам и соответствующим методам.

Иногда эти методы называют также префиксными.

Таким образом, использование описываемых методов предусматривает создание кодовой таблицы (подобно кодам ASCII или base64). Формально процедура сжатия (прямое преобразование) состоит в подстановке соответствующего бинарного кода вместо символа исходного алфавита и наоборот – при обратном.

Методы относятся к классу «сжатие без потерь». Различие между двумя рассматриваемыми методами состоит лишь в особенностях формирования таблицы бинарных кодов. При формировании этой таблицы для обоих методов можно воспользоваться статистическими свойствами алфавитов, полученными при выполнении лабораторной работы № 2. преобразовании.

**Ход работы**

Разработаем авторское приложение в соответствии с целью  
лабораторной работы.

Выполнить прямое и обратное  
преобразования сообщения, состоящего из собственных имени и  
фамилии.

На основе данных, полученных в лабораторной работе № 2:

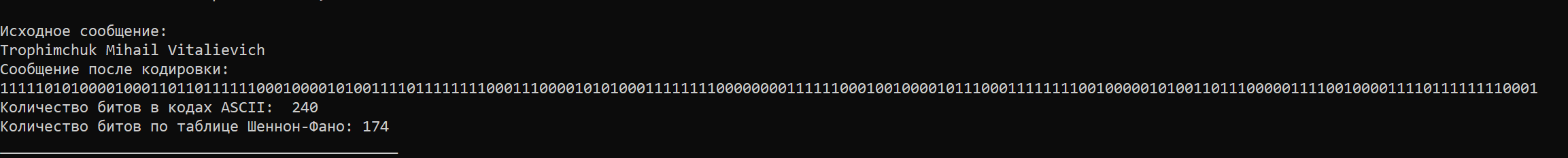
****

Рисунок 1 – Сжатие на основе метода Шеннона − Фано

Динамически, на основе анализа сжимаемого сообщения:

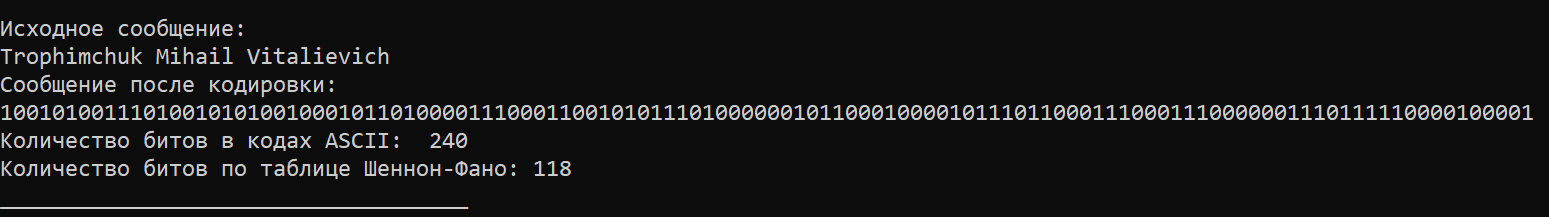
****

Рисунок 2 – Сжатие на основе метода Шеннона − Фано

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.IO;  namespace NineLab  {  public class ShannonFanoSymbol  {  public char symbol;  public int count;  public double viorite;  public string code;  public ShannonFanoSymbol(char sim, int count, double vior, string code)  {  this.viorite = vior;  this.symbol = sim;  this.count = count;  this.code = code;  }  public static List<ShannonFanoSymbol> AddSymbols(List<ShannonFanoSymbol> symbols, string line)  {  foreach (var character in line)  {  if (symbols.Find(x => x.symbol == character) == null)  {  symbols.Add(new ShannonFanoSymbol(character, 1, 0.0, ""));  }  else  {  symbols.Where(x => x.symbol == character).ToList().ForEach(x => x.count++);  }  }  return symbols;  }  public static void Show(List<ShannonFanoSymbol> symbols)  {  foreach (var symbol in symbols)  {  Console.Write("Символ: {0} Кол-во: {1} ", symbol.symbol, symbol.count);  if (symbol.viorite != 0) {  Console.Write("Вероятность: {0}", symbol.viorite);  }  if (symbol.code != "") {  Console.Write(" Код: {0}", symbol.code);  }  Console.WriteLine();  }  Console.WriteLine();  }    }  class Program  {    static void Main(string[] args)  {  List<ShannonFanoSymbol> symbols = new List<ShannonFanoSymbol>();  using (StreamReader stream = new StreamReader(@"C:\Users\Helen\Desktop\Защита\SecondLab\Litovski.txt", Encoding.Default))  {  string messagef;  while ((messagef = stream.ReadLine()) != null)  {  symbols = ShannonFanoSymbol.AddSymbols(symbols, messagef);  }  }  Console.WriteLine();  Console.WriteLine("На основе данных, полученных в лабораторной работе № 2");  Console.WriteLine("Таблица символов: ");  Console.WriteLine();  ShannonFanoSymbol.Show(symbols);  double symbolssum = symbols.Sum(x => x.count);  Console.WriteLine("Сумма всех символов текста на латинском языке: " + symbolssum);  for (int i = 0; i < symbols.Count; i++)  {  symbols[i].viorite = symbols[i].count / symbolssum;  }  Console.WriteLine("Сумма вероятностей всех символов таблицы: " + (symbols.Sum(x => x.viorite)));  Console.WriteLine();  symbols = symbols.OrderByDescending(x => x.viorite).ToList();  ShannonFanoSymbol.Show(symbols);  Console.WriteLine();  Console.WriteLine("Таблица с кодом для каждого символа: " );  Console.WriteLine();  symbols = AddCodes(symbols);  foreach (var symbol in symbols)  {  symbol.code = symbol.code.Remove(symbol.code.Length - 1, 1);  }  ShannonFanoSymbol.Show(symbols);  string blockofFIO = "Trophimchuk Mihail Vitalievich";  string decodingOfFIO = "";  foreach (var charFIO in blockofFIO)  {  decodingOfFIO += (symbols.Where(x => x.symbol == charFIO).FirstOrDefault()).code;  }  Console.WriteLine("Исходное сообщение: ");  Console.WriteLine(blockofFIO);  Console.WriteLine("Сообщение после кодировки: ");  Console.WriteLine(decodingOfFIO);  Console.WriteLine("Количество битов в кодах ASCII: " + blockofFIO.Count() \* 8);  Console.WriteLine("Количество битов по таблице Шеннон-Фано: " + decodingOfFIO.Count());  Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  Console.WriteLine();  Console.WriteLine("Декодирование: ");  Console.WriteLine();  string Encoded = "";  string FIOdecoded = "";  for (int i = 0; i < decodingOfFIO.Count(); i++)  {  Encoded += decodingOfFIO[i];  if (symbols.Find(x => x.code == Encoded) != null)  {  FIOdecoded += symbols.Find(x => x.code == Encoded).symbol;  Encoded = "";  }  }  Console.WriteLine(FIOdecoded);  Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  Console.WriteLine();  Console.WriteLine("Динамически, на основе анализа сжимаемого сообщения: ");  Console.WriteLine();  symbols.Clear();  string message = "Trophimchuk Mihail Vitalievich";  symbols = ShannonFanoSymbol.AddSymbols(symbols, message);  ShannonFanoSymbol.Show(symbols);  symbolssum = symbols.Sum(x => x.count);  Console.WriteLine("Сумма всех символов текста на латинском языке: " + symbolssum);  for (int i = 0; i < symbols.Count; i++)  {  symbols[i].viorite = symbols[i].count / symbolssum;  }  Console.WriteLine("Сумма вероятностей всех символов таблицы: " + (symbols.Sum(x => x.viorite)));  Console.WriteLine();  symbols = symbols.OrderByDescending(x => x.viorite).ToList();  ShannonFanoSymbol.Show(symbols);  Console.WriteLine();  Console.WriteLine("Таблица с кодом для каждого символа: ");  Console.WriteLine();  symbols = AddCodes(symbols);  foreach (var symbol in symbols)  {  symbol.code = symbol.code.Remove(symbol.code.Length - 1, 1);  }  ShannonFanoSymbol.Show(symbols);  blockofFIO = "Trophimchuk Mihail Vitalievich";  decodingOfFIO = "";  foreach (var charFIO in blockofFIO)  {  decodingOfFIO += (symbols.Where(x => x.symbol == charFIO).First()).code;  }  Console.WriteLine("Исходное сообщение: ");  Console.WriteLine(blockofFIO);  Console.WriteLine("Сообщение после кодировки: ");  Console.WriteLine(decodingOfFIO);  Console.WriteLine("Количество битов в кодах ASCII: " + blockofFIO.Count() \* 8);  Console.WriteLine("Количество битов по таблице Шеннон-Фано: " + decodingOfFIO.Count());  Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  Console.WriteLine();  Console.WriteLine("Декодирование");    Encoded = "";  FIOdecoded = "";  for (int i = 0; i < decodingOfFIO.Count(); i++)  {  Encoded += decodingOfFIO[i];  if (symbols.Find(x => x.code == Encoded) != null)  {  FIOdecoded += symbols.Find(x => x.code == Encoded).symbol;  Encoded = "";  }  }  Console.WriteLine(FIOdecoded);  Console.ReadLine();  }  public static List<ShannonFanoSymbol> AddCodes(List<ShannonFanoSymbol> symbols)  {  int counter = 0;  double probability = 0.0;  List<ShannonFanoSymbol> first = new List<ShannonFanoSymbol>();  List<ShannonFanoSymbol> second = new List<ShannonFanoSymbol>();  while (probability < (symbols.Sum(x => x.viorite) / 2))  {  probability += symbols[counter].viorite;  counter++;  }  for (int i = 0; i < counter; i++)  {  symbols[i].code += "0";  first.Add(symbols[i]);  }  for (int i = counter; i < symbols.Count; i++)  {  symbols[i].code += "1";  second.Add(symbols[i]);  }  if (symbols.Count > 1)  {  first = AddCodes(first);  second = AddCodes(second);  first.AddRange(second);  symbols = first;  }  return symbols;  }  }  } |

Листинг 1 – Код приложения

**Вывод**: в данной работе был рассмотрен способ сжатия данных на основе статистических данных – метод Шеннона – Фано. Метод не является оптимальным так как при равном распределении вероятностей он может выдавать несколько различных вариантов результатов. Однако метод дает оптимальное распределение вероятностей и максимальное сжатие.