Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Защита информации и надёжность информационных систем»

**Лабораторная работа №10**

**Тема «СЖАТИЕ/РАСПАКОВКА ДАННЫХ АРИФМЕТИЧЕСКИМ МЕТОДОМ»**

Выполнил:

Студент 3 курса 7 группы ФИТ

Тимошенко Д. В.   
 Проверила:   
 асс. Николайчук А. Н.

Минск 2024

**Цель работы:** приобретение практических навыков использования арифметических методов сжатия/распаковки данных.

**Задание на лабораторную работу**

1. Разработать авторское приложение в соответствии с целью лабораторной работы.

2. С помощью приложения выполнить прямое и обратное преобразования сообщений в соответствии с таблицей.

Каждый студент выполняет задание, состоящее из двух частей. Первая часть предусматривает кодирование/декодирование сообщения, указанного в 2-м столбце, вторая часть – составного сообщения, полученного конкатенацией последовательностей из 2-го столбца, указанных в 3-м столбце. Например, для варианта № 1 такой конкатенацией будет последовательность «летоисчислениевремяпрепровождение».

3. Дать оценку возможности переполнения при выполнении вычислений.

4. Сравнить характеристики арифметического сжатия с вероятностными алгоритмами.

5. Результаты оформить в виде отчета по установленным правилам.



Вариант 11

Как мы установили, вероятностные (префиксные) методы являются достаточно простыми и эффективными: они основаны на использовании кодов переменной длины и для вероятностей появления символов алфавита, кратных степеням числа 2 (1/2, 1/4, 1/8 и т. п.), дают наилучшие результаты. При других значениях вероятностей, как правило, самый короткий код получается большим, чем двоичный логарифм этой вероятности (взятый с отрицательным знаком). Например, при р(аi) = 0,4 получим −log 2 0,15 = 1,32. Понятно, что мы не можем закодировать этот символ только 2 битами (либо одним), т. е. решение не всегда является оптимальным. Анализируемого недостатка лишены арифметические методы.

Пpи аpифметическом сжатии (кодиpовании) текст пpедставляется вещественными числами в интеpвале от 0 до 1. По меpе анализа текста отобpажающий его интеpвал уменьшается, а количество битов для его пpедставления возpастает. Очеpедные символы текста сокpащают величину интеpвала, исходя из значений соответствующих веpоятностей.

Основная идея арифметического метода сжатия заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям.

Таким образом, как и во всех энтропийных алгоритмах, исходной является информация о частоте встречаемости каждого символа алфавита.

Алгоритмы прямого и обратного преобразований базируются на операциях с «рабочим отрезком».

Рабочим отрезком называется интервал [a; b] с расположенными на нем точками. Причем точки расположены таким образом, что длины образованных ими отрезков пропорциональны (или равны) частоте (вероятности) появления соответствующих символов.

Приложение, для выполнения задания, написано на языке программирования C#.NET. Ниже представлены листинг функций, используемых во всех задания, листинг 1.1-1.2.

|  |
| --- |
| public class Node  {  public char Symbol { get; set; }  public decimal High { get; set; }  public decimal Low { get; set; }  public override string ToString()  {  return string.Format("Low: {0} | High: {1}", Low.ToString(), High.ToString());  }  }  public class InfoString  {  public static StringBuilder Sb { get; set; }  }  } |

Листинги 1.1 – Класс для узла

|  |
| --- |
| public class Compressor  {  public List<Node> Nodes { get; set; }  public Dictionary<char, decimal> Frequencies { get; set; }  public Node ResultNode { get; set; }  public void Build(string source)  {  Nodes = new List<Node>();  decimal inc = 1 / (decimal)source.Length;  Frequencies = new Dictionary<char, decimal>();  for (int i = 0; i < source.Length; i++)  {  if (!Frequencies.ContainsKey(source[i]))  {  Frequencies.Add(source[i], 0);  }  Frequencies[source[i]] += inc;  }  Frequencies = Frequencies.OrderBy(x => x.Value).ToDictionary(x => x.Key, y => y.Value);  decimal low = 0;  foreach (var item in Frequencies)  {  Nodes.Add(new Node { Symbol = item.Key, Low = Math.Round(low, 5), High = Math.Round(low + item.Value, 5) });  low += item.Value;  }  }  public decimal Compress(string source)  {  InfoString.Sb = new StringBuilder();  ResultNode = new Node { Symbol = '\*', High = 1, Low = 0 };  foreach (var item in source)  {  decimal oldHigh = ResultNode.High;  decimal oldLow = ResultNode.Low;  InfoString.Sb.Append(ResultNode.ToString()).Append(Environment.NewLine);  ResultNode.Symbol = '\*';  ResultNode.High = oldLow + (oldHigh - oldLow) \* Nodes.Find(x => x.Symbol == item).High;  ResultNode.Low = oldLow + (oldHigh - oldLow) \* Nodes.Find(x => x.Symbol == item).Low;  }  InfoString.Sb.Append(ResultNode.ToString()).Append(Environment.NewLine);  return ResultNode.Low;  }  public string Decompress(decimal compress, int leng, int t)  {  StringBuilder sb = new StringBuilder();  InfoString.Sb = new StringBuilder();  for (int i = 0; i < leng; i++)  {  char symbol = Nodes.Find(x => Math.Round(compress, t) >= x.Low && Math.Round(compress, t) < x.High).Symbol;  InfoString.Sb.Append(compress.ToString() + $"\t-- {symbol}").Append(Environment.NewLine);  sb.Append(symbol);  Node tempNode = Nodes.Find(x => x.Symbol == symbol);  compress = (compress - tempNode.Low) / (tempNode.High - tempNode.Low);  }  return sb.ToString();  }  } |

Листинг 1.2 – Класс для компрессора

Код выполнения самих заданий лабораторной работы, листинг 1.3.

|  |
| --- |
| static void Main(string[] args)  {  {  string word = "сорокадневный";  int wordLength = word.Length;  Compressor wordCompressor = new Compressor();  wordCompressor.Build(word);  Console.WriteLine("Intervals:");  foreach (var i in wordCompressor.Nodes)  {  Console.WriteLine($"p({i.Symbol}) = {i.High - i.Low}\n");  }  foreach (var i in wordCompressor.Nodes)  {  Console.WriteLine($"{i.Symbol} {i.Low} - {i.High}\n");  }  var compressResult = wordCompressor.Compress(word);  Console.WriteLine("Compress info:");  Console.WriteLine(InfoString.Sb.ToString());  Console.WriteLine($"Result: {compressResult}\n");  var decompressResult = wordCompressor.Decompress(compressResult, wordLength, wordLength / 2 + 1);  Console.WriteLine("Decompress info:");  Console.WriteLine(InfoString.Sb.ToString());  Console.WriteLine($"Result: {decompressResult}");  }  Console.WriteLine("\n\n");  {  string word = "сорокадневныйтелегаммааппарат";  int wordLength = word.Length;  Compressor wordCompressor = new Compressor();  wordCompressor.Build(word);  Console.WriteLine("Intervals:");  foreach (var i in wordCompressor.Nodes)  {  Console.WriteLine($"p({i.Symbol}) = {i.High - i.Low}\n");  }  foreach (var i in wordCompressor.Nodes)  {  Console.WriteLine($"{i.Symbol} {i.Low} - {i.High}\n");  }  var compressResult = wordCompressor.Compress(word);  Console.WriteLine("Compress info:");  Console.WriteLine(InfoString.Sb.ToString());  Console.WriteLine($"Result: {compressResult}\n");  var decompressResult = wordCompressor.Decompress(compressResult, wordLength, wordLength / 2 + 1);  Console.WriteLine("Decompress info:");  Console.WriteLine(InfoString.Sb.ToString());  Console.WriteLine($"Result: {decompressResult}");  }  } |

Листинг 1.4 – выполнение задания лабораторной работы

Вывод представленного кода, рисунок 1.1.

|  |  |
| --- | --- |
| a | б |

Рисунок 1.1 – Вывод кода: а – кодирование; б - декодирование

**Вывод:** в ходе выполнения лабораторной работы были изучены основные варианты алгоритма арифметического метода сжатия/распаковки. Были приобретены практические навыки работы с ними. Также было выяснено, что существует необходимость в использование дополнительных средств кодирования, так как при большом количестве символов теряется точность итоговой числовой последовательности.