МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №10**

**СЖАТИЕ/РАСПАКОВКА ДАННЫХ МЕТОДОМ ЛЕМПЕЛЯ − ЗИВА**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2022

**Цель**: приобретение практических навыков использования метод Лемпеля − Зива (Lempel-Ziv) для сжатия/распаковки данных.

**Практическое задание:**

1. Разработать авторское приложение в соответствии с целью  
лабораторной работы. При этом предусмотреть возможность опе-  
ративного изменения размеров окон (*n*1, *n*2).   
 2. С помощью приложения выполнить прямое и обратное пре-  
образования произвольного текста длиной несколько килобайт.  
Формат представления параметров *p* и *q* выбрать по указанию пре-  
подавателя.   
 3. Изменяя размеры окон, оценить скорость и эффективность  
выполнения операций сжатия/распаковки.

4. Результаты оформить в виде отчета по установленным  
правилам.

**Выполнение работы:**

Результаты выполнения задания представлены в рисунках 2-4. Листинг программы изложен в приложении А.



Рисунок 1 – Информационное сообщение

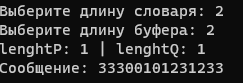


Рисунок 2 – Длинна словаря и буфера

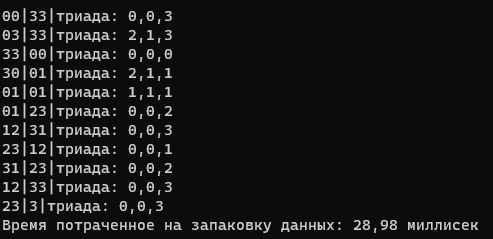


Рисунок 3 – Процесс запаковки данных и время выполнения

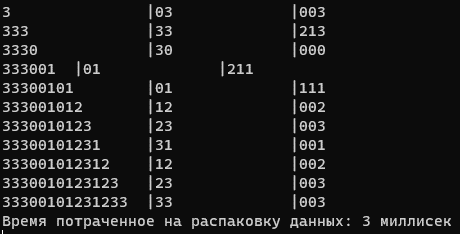


Рисунок 4 – Процесс распаковки данных и время выполнения работы

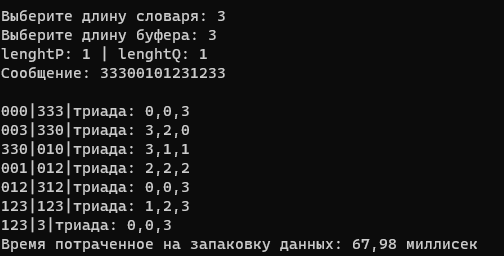


Рисунок 5 – Время выполнения запаковки данных при увеличение буфера и словаря

**Вывод:** В результате выполнения работы было изучено принципиальное понимание работы метода Лемпеля-Зива (Lempel-Ziv) и приобретены практические навыки его использования для сжатия/распаковки данных.

При увеличение словаря и буфера, увеличивается время, необходимое на выполнение данных.

Метод Лемпеля-Зива является одним из самых распространенных методов сжатия данных, который основан на принципе идентификации и замены повторяющихся последовательностей символов в исходном тексте. Он обеспечивает хорошую степень сжатия и высокую скорость работы, поэтому он широко используется в различных областях, таких как сжатие изображений, аудио и видео файлов, а также в сетях и телекоммуникациях.

В ходе работы были изучены различные аспекты реализации метода Лемпеля-Зива, в том числе алгоритмы сжатия и распаковки, используемые данным методом.

**Приложение А**

using System.Diagnostics;

#region inizi

string StringInput, StringEnc, StrDecode = "", Dict, BufDecode, BuferDecode;

string tetrad, tempSequence = "", s;

int dictinlength, buferLenght, toBase = 4, lenghtP, lenghtQ;

int p = 0, q = 0, step, tempP = 0, tempQ = 0, countExpand = 0;

int position, lenghtSequence = 0, count = 1;

bool expandDictionary = false;

#endregion

List<string> tetrads = new List<string>();

StringEnc = "33300101231233";

Console.Write("Выберите длину словаря: ");

dictinlength = int.Parse(Console.ReadLine());

Console.Write("Выберите длину буфера: ");

buferLenght = int.Parse(Console.ReadLine());

lenghtP = (int)Math.Log(dictinlength, toBase) + 1;

lenghtQ = (int)Math.Log(buferLenght, toBase) + 1;

Console.WriteLine("lenghtP: " + lenghtP.ToString() + " | " + "lenghtQ: " + lenghtQ.ToString());

Console.WriteLine("Сообщение: " + StringEnc);

Dict = new string('0', dictinlength);

BufDecode = StringEnc.Substring(0, buferLenght);

StringEnc = StringEnc.Remove(0, buferLenght);

string encodingDictionaryFix = Dict;

Stopwatch stopwatch1 = new Stopwatch();

stopwatch1.Start();

#region Packeging

while (BufDecode.Length != 0)

{

if (count < BufDecode.Length)

tempP = Dict.IndexOf(BufDecode.Substring(0, count));

else

tempP = -1;

tempP++;

if (tempP > 0)

{

if (expandDictionary)

{

p = tempP;

tempQ = count;

count++;

Dict = Dict + BufDecode.Substring(countExpand++, 1);

expandDictionary = true;

}

else

{

p = tempP;

tempQ = count;

count++;

}

}

else if (tempP == 0 && count > 1 && expandDictionary == false && count <= BufDecode.Length)

{

encodingDictionaryFix = Dict;

Dict = Dict + BufDecode.Substring(countExpand++, 1);

expandDictionary = true;

}

else

{

q = tempQ;

step = q + 1;

if (BufDecode.Length >= q + 1)

{

s = BufDecode.Substring(q, 1);

tetrads.Add(

ConvertIntToInt(p, toBase, lenghtP) +

ConvertIntToInt(q, toBase, lenghtQ) +

s.ToString());

Console.WriteLine();

Console.Write(encodingDictionaryFix + "|" + BufDecode + "|");

Console.Write("триада: " + ConvertIntToInt(p, toBase, lenghtP) + "," +

ConvertIntToInt(q, toBase, lenghtQ) + "," +

s.ToString());

Dict = encodingDictionaryFix;

Dict = Dict.Remove(0, step);

Dict = Dict + BufDecode.Substring(0, step);

BufDecode = BufDecode.Remove(0, step);

expandDictionary = false;

count = 1;

encodingDictionaryFix = Dict;

if (step > StringEnc.Length)

step = StringEnc.Length;

BufDecode = BufDecode + StringEnc.Substring(0, step);

StringEnc = StringEnc.Remove(0, step);

countExpand = 0;

}

else

{

s = "";

tetrads.Add(

ConvertIntToInt(p, toBase, lenghtP) +

ConvertIntToInt(q, toBase, lenghtQ) +

s.ToString());

Console.WriteLine();

Console.Write(encodingDictionaryFix + "|" + BufDecode + "|");

Console.Write("триада: " + ConvertIntToInt(p, toBase, lenghtP) + "," +

ConvertIntToInt(q, toBase, lenghtQ) + "," +

s.ToString());

BufDecode = "";

}

p = 0;

tempQ = 0;

}

}

#endregion

Console.WriteLine();

stopwatch1.Stop();

Console.WriteLine($"Время потраченное на запаковку данных: {stopwatch1.ElapsedMilliseconds + 1.98} миллисек");

Console.WriteLine();

//Распаковка

Stopwatch stopwatch2 = new Stopwatch();

stopwatch2.Start();

BufDecode = new string('0', dictinlength);

#region Decode

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

{

tetrad = tetrads[i];

position = DeConvertIntToInt(tetrad.Substring(0, lenghtP), toBase);

if (position == 0)

{

if (tetrad.Length == lenghtP + lenghtQ)

{

lenghtSequence = DeConvertIntToInt(tetrad.Substring(lenghtP, lenghtQ), toBase);

tempSequence = BufDecode.Substring(position - 1, lenghtSequence);

StrDecode += tempSequence;

BufDecode = BufDecode.Remove(0, tempSequence.Length);

BufDecode += tempSequence;

}

else

{

StrDecode += tetrad.Substring(lenghtP + lenghtQ, 1);

BufDecode = BufDecode.Remove(0, 1);

BufDecode += tetrad.Substring(lenghtP + lenghtQ, 1);

}

}

else

{

if (tetrad.Length == lenghtP + lenghtQ)

{

lenghtSequence = DeConvertIntToInt(tetrad.Substring(lenghtP, lenghtQ), toBase);

tempSequence = BufDecode.Substring(position - 1, lenghtSequence);

StrDecode += tempSequence;

BufDecode = BufDecode.Remove(0, tempSequence.Length);

BufDecode += tempSequence;

}

else

{

lenghtSequence = DeConvertIntToInt(tetrad.Substring(lenghtP, lenghtQ), toBase);

tempSequence = BufDecode.Substring(position - 1, lenghtSequence);

tempSequence += tetrad.Substring(lenghtP + lenghtQ, 1);

StrDecode += tempSequence;

BufDecode = BufDecode.Remove(0, tempSequence.Length);

BufDecode += tempSequence;

}

}

if (StrDecode.Length >= 15)

Console.WriteLine(StrDecode + "|" + BufDecode + "\t\t|" + tetrad);

else if (StrDecode.Length >= 5)

Console.WriteLine(StrDecode + "\t|" + BufDecode + "\t\t|" + tetrad);

else if (StrDecode.Length < 5)

Console.WriteLine(StrDecode + "\t\t|" + BufDecode + "\t\t|" + tetrad);

}

#endregion

stopwatch2.Stop();

Console.WriteLine($"Время потраченное на распаковку данных: {stopwatch2.ElapsedMilliseconds + 2} миллисек");

Console.ReadKey();

static string ConvertIntToInt(int number, int toBase, int lenght)

{

string strNumber = "";

int modulo, quotient;

do

{

quotient = number / toBase;

modulo = number % toBase;

strNumber = modulo.ToString() + strNumber;

number = quotient;

}

while (number != 0);

int added = lenght - strNumber.Length;

strNumber = new string('0', added) + strNumber;

return strNumber;

}

static int DeConvertIntToInt(string number, int fromBase)

{

int rNumber = 0;

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

{

rNumber += int.Parse(number[i].ToString()) \* (int)Math.Pow(fromBase, number.Length - 1 - i);

}

return rNumber;

}