Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных машин

Отчет

По лабораторной работе №3

«Инъективное преобразование букв открытого сообщения в подмножество

символов шифротекста меньшей мощности»

|  |  |  |
| --- | --- | --- |
| Выполнил:  Ст. гр. 050501  Тускенис Д. С. |  | Проверил:  Кобяк И. П. |

Минск 2014

1. **ПОСТАНОВКА ЗАДАЧИ**

Выполнить инъективное преобразование произвольного текста и оценить коэффициент сжатия информации.

1. **АЛГОРИТМ РЕШЕНИЯ**
2. Выбрать код символа-разделителя (например, 01)
3. Определить частоты появления различных символов в заданном тексте
4. С помощью счетчика Джексона сформировать коды неравномерной длины, не содержащие код символа-разделителя
5. На основании полученных частот присвоить каждому символу определенный код: символы с наибольшей частотой кодируются последовательностью наименьшей длины.
6. Заменить каждый символ в исходном тексте соответствующей последовательностью из пункта 4.
7. **ЛИСТИНГ ПРОГРАММЫ**

static List<String> GetCodesFromJacksonCounter(int count)

{

List<String> codes = new List<string>();

int currentCodeLength = 1;

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

{

String code = new String('0', currentCodeLength);

for (int j = 0; j < code.Length + 1; j++)

{

if ((code.Length < divider.Length || code.Substring(0, divider.Length) != divider) && codes.Count < count)

{

codes.Add(code);

code = shiftString(code);

}

else

{

break;

}

}

if (!(codes.Count < count))

{

break;

}

currentCodeLength++;

}

return codes;

}

static String shiftString(String s)

{

String result = "1";

for (int i = 1; i < s.Length; i++)

{

result += s[i - 1];

}

return result;

}

static String EncodeStringWithCodingTable(String s, Dictionary<Char, String> codingTable)

{

String result = "";

if (s.Length == 0)

{

s = "\n";

}

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

{

result += codingTable[s[i]];

if (i != s.Length - 1)

{

result += divider;

}

}

return result;

}

static String DecodeStringWithCodingTable(String s, Dictionary<Char, String> codingTable)

{

String result = "";

String[] encodedCharacters = Regex.Split(s, divider);

foreach (String encodedCharacter in encodedCharacters)

{

if (codingTable.ContainsValue(encodedCharacter))

{

foreach (Char c in codingTable.Keys.ToArray())

{

if (codingTable[c] == encodedCharacter)

{

result += c;

break;

}

}

}

}

return result;

}

static Dictionary<Char, String> GetCodingTableFromFrequencies(Dictionary<Char, Double> frequencies)

{

Dictionary<Char, String> codingTable = new Dictionary<Char, String>();

List<String> codes = GetCodesFromJacksonCounter(frequencies.Keys.Count);

List<KeyValuePair<Char, Double>> sortedFrequencies = (from entry in frequencies orderby entry.Value descending select entry).ToList();

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

{

codingTable[sortedFrequencies[i].Key] = codes[i];

}

return codingTable;

}

static Dictionary<Char, Double> GetCharFrequenciesFromFile(String path)

{

Dictionary<Char, Double> frequencies = new Dictionary<Char, Double>();

StreamReader stream = File.OpenText(path);

Int32 streamLength = 0;

while (!stream.EndOfStream)

{

String line = stream.ReadLine();

if (line.Length == 0)

{

line = "\n";

}

streamLength += line.Length;

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

{

Char c = line[i];

if (frequencies.ContainsKey(c))

{

frequencies[c] = frequencies[c] + 1;

}

else

{

frequencies[c] = 1;

}

}

}

stream.Close();

Array keys = frequencies.Keys.ToArray();

foreach (Char c in keys)

{

frequencies[c] = frequencies[c] / (Double)streamLength;

}

return frequencies;

}

1. **РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

