МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

по дисциплине: «Теория информации»

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверил:

Твердохлеб В.В.

Белгород 2023 г.

Тема: «Исследование возможности применения методов энтропийного кодирования для обработки двоичных последовательностей»

Вариант 12

Содержание отчета

1. Аналитика касательно построения кодов для исходной двоичной последовательности

2. Примеры кодовой реализации п.3, п.3, п.4 и п.6.

3. Результаты обработки кодов, полученных в п.5.

4. Текстовая последовательность, восстановленная к читаемому виду.

5. Общие выводы

Ход работы:

1 Построить обработчик, выполняющий компрессию по алгоритму Гилберта-Мура.

```
std::vector<bool> GetFloatNBits(
        float number,
        int n
) {
  std::vector<bool> res(n);
  for (int i = 0; i < n; i++) {
    number *= 2;
    res[i] = int(number);
    number -= int(number);
  }

  return res;
}

std::vector<FanoCode> GetMooreHilbertCode(const std::unordered_map<char, int> &counters) {
  if (counters.empty()) {
    return {};
  }

  if (counters.size() == 1) {
    return {{counters.begin()->first, 1, std::vector{true}}};
  }

  std::vector<FanoCode> codes;
  std::vector<float> d;
  std::vector<float> sigma;
  size_t total = 0;

  for (auto &i: counters) {
    codes.push_back({i.first, i.second});
    total += i.second;
  }

  for (int i = 0; i < codes.size(); i++) {
    double p = static_cast<float>(codes[i].count) / total;

    if (i == 0) {
```

```
      d.push_back(0);
    } else {
      double prev_p = static_cast<float>(codes[i - 1].count) / total;
      d.push_back(d[i - 1] + prev_p);
    }

    sigma.push_back(d[i] + p / 2);
    int length = ceil(-std::log2(p)) + 1;
    codes[i].code = GetFloatNBits(sigma[i], length);
  }

  return codes;
}
```

2 Создать генераторы данных, работающих как источники Хартли и Бернулли (в двоичном алфавите).

```
std::string HartliGenerator(
        int n,
        int n_bits
) {
  std::string t;
  std::random_device rd;
  std::mt19937 gen(rd());
  std::uniform_int_distribution<unsigned char> d(0, std::min((1 << n_bits) - 1, 127));
  for (int i = 0; i < n; i++) {
    t.push_back(d(gen));
  }

  return t;
}

std::string BernoulliGenerator(
        int n,
        float p
) {
  std::string t;
  std::random_device rd;
  std::mt19937 gen(rd());
  std::bernoulli_distribution d(p);
  for (int i = 0; i < n; i++) {
    t.push_back(d(gen));
  }

  return t;
}
```

3. Построить коды Гилберта-Мура для каждой из последовательностей, для чего предварительно сегментировать каждую цепочку по 8 символов.

```
Сообщение сгенерированное источником Хартли:
"&j@"s")~7#"=aBMKR,"|"H*\zymC~m 5]V!╗   ^Fm NSf"e
<f> |  1 |       0000001
<S> |  1 |       0000011
<N> |  1 |       0000110
<F> |  1 |       0001000
<       > |  1 |       0001011
<V> |  1 |       0001110
<]> |  1 |       0010000
<5> |  1 |       0010011
< > |  2 |       001011
<m> |  3 |       001110
<"> |  2 |       010001
<╗> |  1 |       0100111
<=> |  1 |       0101010
<^> |  1 |       0101100
<#> |  1 |       0101111
<@> |  1 |       0110001
<z> |  1 |       0110100
<s> |  1 |       0110111
<)> |  1 |       0111001
<"> |  1 |       0111100
<j> |  1 |       0111110
<M> |  1 |       1000001
<a> |  1 |       1000011
<&> |  1 |       1000110
<"> |  1 |       1001000
<B> |  1 |       1001011
<C> |  1 |       1001110
<~> |  2 |       101000
> |  1 |       1010101
<|> |  1 |       1011000
<K> |  1 |       1011010
<"> |  1 |       1011101
<R> |  1 |       1011111
<7> |  1 |       1100010
```

```
<7> |  1 |       1100010
<r> |  1 |       1100101
<[illegible]> |  2 |       110100
<,> |  1 |       1101100
<[illegible]> |  1 |       1101111
<H> |  1 |       1110001
<e> |  1 |       1110100
<*> |  1 |       1110111
<!> |  1 |       1111001
<\> |  1 |       1111100
<y> |  1 |       1111110
100100010001100111110011000101000101101110111100011100110100011000100101111101110101010101000011100101110000011011010101
111111001011101001010101110110011011111011000110100111000111101111111100011010011111100011101001110101000001110001011001
001100100000001110111100101001110001011010110000010000011100010110000110000001100000010100011110100

Сообщение сгенерированное источником Бернулли:
   ╓                          ╓         ╓
<╓> |  3 |       000001
< > | 47 |       10
10101000000110101010101010101010101010101010101010101010101010100000001101010101010101010000001101010101010101010
```

```
Сообщение:
Love is too young to know what conscience is,Yet who knows not conscience is born of love?
<l> |  1 |       00000001
<r> |  1 |       00000100
<b> |  1 |       00000111
<f> |  1 |       00001001
<,> |  1 |       00001100
<c> |  6 |       00010
<a> |  1 |       00100000
<h> |  2 |       0010010
<s> |  6 |       00110
<o> | 14 |       0100
<Y> |  1 |       01100010
<v> |  2 |       0110011
<L> |  1 |       01101010
<i> |  5 |       011100
<e> |  7 |       10000
<w> |  4 |       100100
< > | 16 |       1011
<n> |  9 |       11010
<t> |  5 |       111001
<g> |  1 |       11110000
<?> |  1 |       11110011
<y> |  1 |       11110110
<u> |  1 |       11111000
<k> |  2 |       1111110
011010100100011001110000101101110000110101111100101000100101111110110010011111000110101111000010111110010100101111111101
101001001001001011100100001001000100000111001101100010010011010001100001001110010000110100001010000101101110000110000011
000110001010000111001101110010000100100100101111111101101001001000011010111101001001110011011000100100110100011000010
011100100001101000010100001011011100001101011000001110100000000100110101011010000000100110110000000010100011001110000111100
11
```

4. Вычислить полученные коэффициенты сжатия и величину дисперсии для каждой последовательности.

```
Алгоритм Гильберта-Мура для источника Хартли:
Длина исходного сообщения в битах: 400
Длина закодированного сообщения в битах: 339
Коэффициент сжатия: 1.1799411
Средняя длина кодового слова: 6.7800002
Дисперсия: 0.4099

Алгоритм Гильберта-Мура для источника Бернулли:
Длина исходного сообщения в битах: 400
Длина закодированного сообщения в битах: 118
Коэффициент сжатия: 3.3898306
Средняя длина кодового слова: 2.3599999
Дисперсия: 0.4656

Алгоритм Гильберта-Мура для сообщения:
Длина исходного сообщения в битах: 720
Длина закодированного сообщения в битах: 482
Коэффициент сжатия: 1.493776
Средняя длина кодового слова: 5.3555555
Дисперсия: 1.081
```