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 №2

по дисциплине: Теория информации тема:
«Исследование кодов Шеннона-Фано»

Выполнил: ст. группы ПВ-233
Мороз Роман Алексеевич

Проверил:
Твердохлеб Виталий Викторович

Белгород 2025 г.

Цель работы: исследовать коды Шеннона-Фано.

Задания

1. Построить код для сообщения, содержащего строку панграммы «в чащах юга жил бы цитрус? да но фальшивый экземпляр!». Для полученного кода рассчитать показатели коэффициента сжатия и дисперсии.

Символ	*Вероятность*	*Этапы*						*Код*
		I	*II*	*III*	*IV*	*V*	*VI*	
‘ ‘	*9/53*	*0*	*0*	*0*				*000*
а	*5/53*			*1*				*001*
и	*3/53*		*1*	*0*	*0*			*0100*
л	*3/53*				*1*			*0101*
в	*2/53*			*1*	*0*	*0*		*01100*
р	*2/53*					*1*		*01101*
ы	*2/53*				*1*	*0*		*01110*
б	*1/53*					*1*		*01111*
г	*1/53*	*1*	*0*	*0*	*0*	*0*		*10000*
д	*1/53*					*1*	*0*	*100010*
е	*1/53*						*1*	*100011*
ж	*1/53*				*1*	*0*	*0*	*100100*
з	*1/53*						*1*	*100101*
й	*1/53*					*1*	*0*	*100110*
к	*1/53*						*1*	*100111*
м	*1/53*			*1*	*0*	*0*		*10100*
н	*1/53*					*1*	*0*	*101010*
о	*1/53*						*1*	*101011*
п	*1/53*				*1*	*0*		*10110*
с	*1/53*					*1*	*0*	*101110*
т	*1/53*						*1*	*101111*
у	*1/53*		*1*	*0*	*0*	*0*		*11000*
ф	*1/53*					*1*	*0*	*110010*
х	*1/53*						*1*	*110011*
ц	*1/53*				*1*	*0*	*0*	*110100*
ч	*1/53*						*1*	*110101*
ш	*1/53*					*1*	*0*	*110110*
щ	*1/53*						*1*	*110111*
ь	*1/53*			*1*	*0*	*0*		*11100*
э	*1/53*					*1*	*0*	*111010*
ю	*1/53*						*1*	*111011*
я	*1/53*				*1*	*0*		*11110*

?	*1/53*					*1*	*0*	*111110*
!	*1/53*						*1*	*111111*

n = 53 - количество символов в сообщении

Всего в алфавите 34 символа. По формуле $N = 2^I$, N - мощность алфавита, I - кол-во бит, необходимо для кодирования символа алфавита с помощью двоичного кода.

$I = log_2 N;$ $I = log_2 34 \approx 6$

$B = n \times \eta$, *где* n *- количество символов в сообщении,* η *- необходимый объем бит для представления одного символа, в данном случае* $\eta = I = 6$, $n = 53$ $B = 53 \cdot 6 = 318$

$B' = 254$

$$K_{comp} = \frac{B}{B'} = \frac{318}{254} \approx 1{,}25$$

$$l_{cp.} = \sum p_i \cdot l_i = 4.73$$

$$\delta = \sum p_i \cdot \left(l_i - l_{cp.}\right)^2 = 1{,}44$$

2. Построить код для сообщения, содержащего строку «victoria nulla est, Quam quae confessos animo quoque subjugat hostes» Для полученного кода рассчитать показатели коэффициента сжатия и дисперсии.

Символ	*Вероятность*	*Этапы*						*Код*
		I	*II*	*III*	*IV*	*V*	*VI*	
‘ ‘	*9/68*	*0*	*0*	*0*				*000*
s	*7/68*			*1*				*001*
u	*7/68*		*1*	*0*				*010*
a	*6/68*			*1*	*0*			*0110*
o	*6/68*				*1*			*0111*
e	*5/68*	*1*	*0*	*0*	*0*			*1000*
q	*4/68*				*1*			*1001*
t	*4/68*			*1*	*0*			*1010*
i	*3/68*				*1*			*1011*
n	*3/68*		*1*	*0*	*0*	*0*		*11000*
c	*2/68*					*1*		*11001*
l	*2/68*				*1*	*0*		*11010*
m	*2/68*					*1*		*11011*
b	*1/68*			*1*	*0*	*0*	*0*	*111000*
f	*1/68*						*1*	*111001*
g	*1/68*					*1*	*0*	*111010*

h	*1/68*						*1*	*111011*
j	*1/68*				*1*	*0*	*0*	*111100*
r	*1/68*						*1*	*111101*
v	*1/68*					*1*	*0*	*111110*
,	*1/68*						*1*	*111111*

n = 68 - количество символов в сообщении

Всего в алфавите 21 символ. По формуле $N = 2^I$, N - мощность алфавита, I - кол-во бит, необходимо для кодирования символа алфавита с помощью двоичного кода.

$I = log_2 N;$ $I = log_2 21 \approx 5$

$B = n \times \eta$ *, где* n *- количество символов в сообщении,* η *- необходимый объем бит для представления одного символа, в данном случае* $\eta = I = 5$, $n = 68$

$B = 68 \cdot 5 = 340$

$B' = 282$

$$K_{comp} = \frac{B}{B'} = \frac{340}{278} \approx 1{,}21$$

$$l_{cp.} = \sum p_i \cdot l_i = 4{,}08$$

$$\delta = \sum p_i \cdot (l_i - l_{cp.})^2 = 0{,}86$$

3. Построить консольное приложение, реализующее процесс кодирования по методу Шеннона-Фано (с возможностью расчета коэффициента сжатия и дисперсии).

```
import math
  class Node:     def __init__(self, symbol,
probability):
        self.symbol = symbol            # Символ
        self.probability = probability  # Вероятность символа
        self.code = ''                  # Код Шеннона-Фано для символа
  def
shannon_fano(node_list):
    # Если список узлов содержит только один элемент, выходим из функции
if len(node_list) == 1:
        return

    # Вычисляем сумму вероятностей всех символов
    total_probability = sum(node.probability for node in node_list)
    # Вычисляем половину суммы вероятностей
half_probability = total_probability / 2

    cumulative_probability = 0  # Инициализируем накопленную вероятность
index = 0                   # Индекс, по которому разделить список на две
                                # части
    # Находим индекс для разделения списка
for i, node in enumerate(node_list):
        cumulative_probability += node.probability
if cumulative_probability >= half_probability:
            index = i
break

    # Присваиваем коды узлам     for i,
node in enumerate(node_list):
if i <= index:
            node.code += '0'  # Добавляем '0' для левой части
```

```
        else:
            node.code += '1'  # Добавляем '1' для правой части
    # Рекурсивно разделяем и присваиваем коды для левой и правой
частей     shannon_fano(node_list[:index + 1])
shannon_fano(node_list[index + 1:])
  def
main():
    message = input("Введите ваше сообщение: ")

    # Создаем словарь для подсчета встречаемости каждого символа
symbol_counts = {}     for symbol in message:         if symbol
in symbol_counts:             symbol_counts[symbol] += 1
else:
            symbol_counts[symbol] = 1

    # Создаем узлы для каждого символа
nodes = [Node(symbol, count / len(message))
             for symbol, count in symbol_counts.items()]
    # Если вероятности равны, сортируем символы в лексикографическом порядке
nodes.sort(key=lambda x: (x.probability, x.symbol), reverse=True)
    # Применяем кодирование методом Шеннона-Фано
shannon_fano(nodes)

    # Выводим коды для каждого символа
print("Символ\tВероятность\tКод")
encoded_message_length = 0
    mean_code_length = 0  # Средняя длина кодового слова
for node in nodes:
        print(f"{node.symbol}\t{node.probability}\t\t{node.code}")
        encoded_message_length += len(node.code) * symbol_counts[node.symbol]
mean_code_length += len(node.code) * node.probability
    # Вычисляем общий объем бит по формуле
    bit_volume = len(message) * round(math.log(len(set(message)), 2) + 1)
    # Вычисляем коэффициент сжатия
    compression_ratio = bit_volume / encoded_message_length
    # Рассчитываем дисперсию
    variance = sum(node.probability * (len(node.code) -
mean_code_length) ** 2 for node in nodes)

    print(f"Общий объем бит: {bit_volume}")
    print(f"Количество бит после кодирования: {encoded_message_length}")
print(f"Коэффициент сжатия: {compression_ratio}")     print(f"Дисперсия:
{variance}")
  if __name__ ==
"__main__":
    main()
```

Результат работы программы:

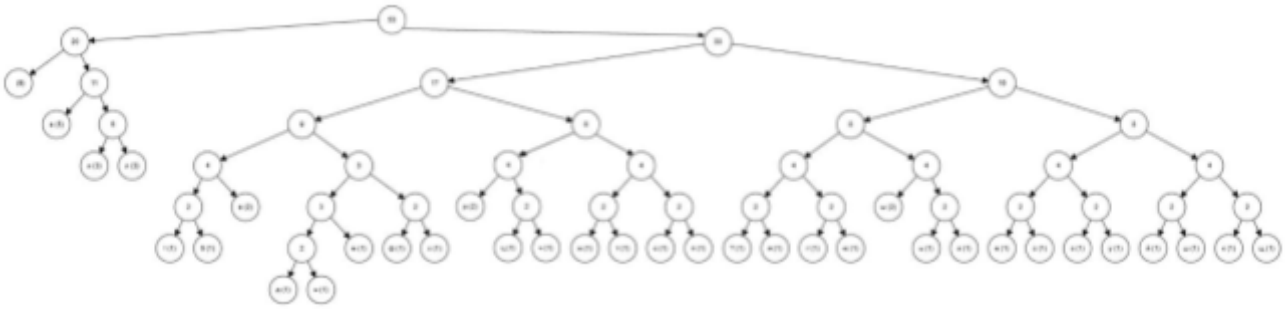
```
Введите ваше сообщение: в чащах юга жил бы цитрус? да но фальшивый экземпляр!
Символ  Вероятность     Код
        0.16981132075471697             000
а       0.09433962264150944             001
л       0.05660377358490566             01000
и       0.05660377358490566             01001
ы       0.03773584905660377             0101
р       0.03773584905660377             01100
в       0.03773584905660377             01101
я       0.018867924528301886            0111
ю       0.018867924528301886            100000
э       0.018867924528301886            100001
ь       0.018867924528301886            100010
щ       0.018867924528301886            100011
ш       0.018867924528301886            100100
ч       0.018867924528301886            100101
ц       0.018867924528301886            10011
х       0.018867924528301886            101000
ф       0.018867924528301886            101001
у       0.018867924528301886            10101
т       0.018867924528301886            101100
с       0.018867924528301886            101101
п       0.018867924528301886            10111
о       0.018867924528301886            110000
н       0.018867924528301886            110001
м       0.018867924528301886            110010
к       0.018867924528301886            110011
й       0.018867924528301886            110100
з       0.018867924528301886            110101
ж       0.018867924528301886            11011
е       0.018867924528301886            111000
д       0.018867924528301886            111001
г       0.018867924528301886            11101
б       0.018867924528301886            111100
?       0.018867924528301886            111101
!       0.018867924528301886            11111
Общий объем бит: 318
Количество бит после кодирования: 254
Коэффициент сжатия: 1.2519685039370079
Дисперсия: 1.4474902100391587
PS D:\VS Code\TEST>
```

```
Введите ваше сообщение: victoria nulla est, quam quae confessos animo quoque subjugat hostes
Символ  Вероятность     Код
        0.1323529411764706              0000
u       0.10294117647058823             0001
s       0.10294117647058823             001
o       0.08823529411764706             010
a       0.08823529411764706             011
e       0.07352941176470588             10000
t       0.058823529411764705            10001
q       0.058823529411764705            1001
n       0.04411764705882353             1010
i       0.04411764705882353             1011
m       0.029411764705882353            11000
l       0.029411764705882353            11001
c       0.029411764705882353            11010
v       0.014705882352941176            110110
r       0.014705882352941176            110111
j       0.014705882352941176            111000
h       0.014705882352941176            111001
g       0.014705882352941176            111010
f       0.014705882352941176            111011
b       0.014705882352941176            11110
,       0.014705882352941176            11111
Общий объем бит: 340
Количество бит после кодирования: 282
Коэффициент сжатия: 1.2056737588652482
Дисперсия: 0.8607266435986156
PS D:\VS Code\TEST>
```

4. Получить кодовые представления сообщений из пунктов 1 и 2 задания по методу Хаффмана. Сравнить полученные результаты с методом Шеннона-Фано по показателям сжатия и дисперсии. Сделать соответствующие выводы.

Символ	*Вероятность*	*Символ*	*Вероятность*	*Символ*	*Вероятность*
‘ ‘	*9/53*	*з*	*1/53*	*ц*	*1/53*
а	*5/53*	*й*	*1/53*	*ч*	*1/53*
и	*3/53*	*к*	*1/53*	*ш*	*1/53*
л	*3/53*	*м*	*1/53*	*щ*	*1/53*
в	*2/53*	*н*	*1/53*	*ь*	*1/53*
р	*2/53*	*о*	*1/53*	*э*	*1/53*
ы	*2/53*	*п*	*1/53*	*ю*	*1/53*
б	*1/53*	*с*	*1/53*	*я*	*1/53*
г	*1/53*	*т*	*1/53*	*?*	*1/53*
д	*1/53*	*у*	*1/53*	*!*	*1/53*
е	*1/53*	*ф*	*1/53*		
ж	*1/53*	*х*	*1/53*		

Символ	*Код*	*Символ*	*Код*	*Символ*	*Код*
‘ ‘	*00*	*з*	*111010*	*ц*	*101110*
а	*010*	*й*	*111100*	*ч*	*101111*
и	*0110*	*к*	*111110*	*ш*	*111101*
л	*0111*	*м*	*100101*	*щ*	*111111*
в	*10001*	*н*	*1001001*	*ь*	*110110*
р	*10110*	*о*	*101010*	*э*	*110111*
ы	*11010*	*п*	*101011*	*ю*	*110011*
б	*100001*	*с*	*111001*	*я*	*110001*
г	*110010*	*т*	*101001*	*?*	*110000*
д	*1001000*	*у*	*111011*	*!*	*100000*
е	*111000*	*ф*	*100110*		
ж	*101000*	*х*	*100111*		



n = 53 – количество символов в сообщении.

Всего в алфавите 34 символов. По формуле $N = 2^I$, N - мощность алфавита, I - кол-во бит, необходимо для кодирования символа алфавита с помощью двоичного кода.

$I = \log_2 N;$ $\quad I = \log_2 34 \approx 6$

$B = n \times \eta$, *где* n *- количество символов в сообщении,* η *- необходимый объем бит для представления одного символа, в данном случае* $\eta = I = 6$, $n = 53$.

$B = 53 \cdot 6 = 318$

$B' = 251$

$K_{comp} = \frac{B}{B'} = \frac{318}{251} = 1{,}26$

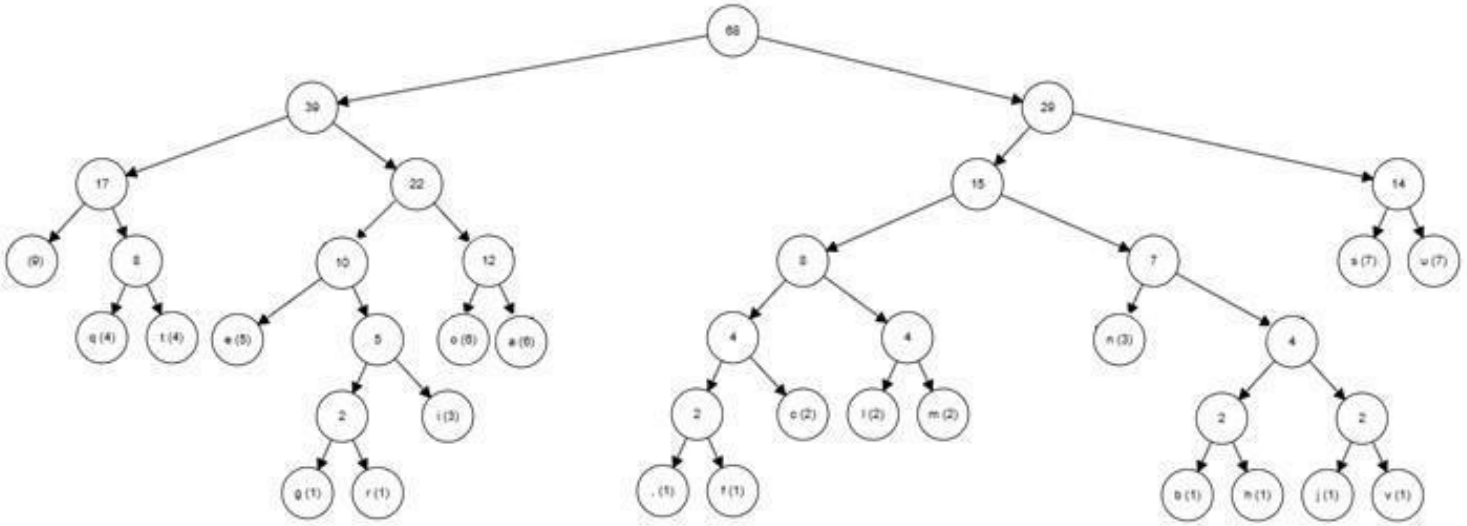
$l_{cp.} = \sum l_i \cdot p_i = 4{,}74$

$\delta = \sum p_i \cdot \left(l_i - l_{cp.}\right)^2 = 1{,}48$

Символ	*Вероятность*	*Символ*	*Вероятность*	*Символ*	*Вероятность*
‘ ‘	*9/68*	*t*	*4/68*	*f*	*1/68*
s	*7/68*	*i*	*3/68*	*g*	*1/68*
u	*7/68*	*n*	*3/68*	*h*	*1/68*
a	*6/68*	*c*	*2/68*	*j*	*1/68*
o	*6/68*	*l*	*2/68*	*r*	*1/68*
e	*5/68*	*m*	*2/68*	*v*	*1/68*
q	*4/68*	*b*	*1/68*	*,*	*1/68*

Символ	*Код*	*Символ*	*Код*	*Символ*	*Код*
‘ ‘	*000*	*t*	*0011*	*f*	*100001*
s	*110*	*i*	*01111*	*g*	*011100*
u	*111*	*n*	*1011*	*h*	*101001*
a	*0100*	*c*	*10001*	*j*	*101010*
o	*0101*	*l*	*10010*	*r*	*011101*
e	*0110*	*m*	*10011*	*v*	*101011*

q	*0010*	*b*	*101000*	*,*	*100000*



n = 68 - количество символов в сообщении

Всего в алфавите 21 символ. По формуле $N = 2^I$, N - мощность алфавита, I - кол-во бит, необходимо для кодирования символа алфавита с помощью двоичного кода.

$I = log_2 N;$ $I = log_2 21 \approx 5$

$B = n \times \eta$ *, где* n *- количество символов в сообщении,* η *- необходимый объем бит для представления одного символа, в данном случае* $\eta = I = 5$, $n = 68$

$B = 68 \cdot 5 = 340$

$B' = 278$

$$K_{comp} = \frac{B}{B'} = \frac{340}{278} = 1{,}22$$

$$l_{cp.} = \sum l_i \cdot p_i = 4{,}09$$

$$\delta = \sum p_i \cdot \left(l_i - l_{cp.}\right)^2 = 0{,}99$$

Оба этих алгоритма используют коды переменной длины: часто встречающийся символ кодируется двоичным кодом меньшей длины, редко встречающийся - кодом большей длины. Коды Шеннона-Фано и Хаффмана – префиксные, что позволяет однозначно декодировать любую последовательность кодовых слов. В отличие от алгоритма Шеннона-Фано, алгоритм Хаффмана обеспечивает минимальную длину кодовой последовательности при побайтном кодировании.

Вывод: в ходе выполнения лабораторной работы были исследованы коды Шеннона-Фано.