ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Доцент, канд. техн. наук		В.А. Миклуш
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

Методы кодирования. Коды Шеннона-Фано, Хаффмана

по курсу: Теория информации, данные, знания

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1. Цель работы:

Изучение методов статистического кодирования, алгоритмов Шеннона-Фано, Хаффмана.

2. Задание:

В соответствии с вариантом:

1. Построить дерево;
2. Представить алгоритм (блок-схему);
3. Написать программу, реализующую заданный метод кодирования;
4. Провести ручную трассировку;
5. Сравнить полученные результаты между собой;
6. Рассчитать среднее число элементарных сигналов (длина кода)

3. Исходные данные

Исходный текст выбран под вариантом 17 и представлен на итальянском языке: «Si mangia per vivere, non si vive per mangiare».

Заданный метод кодирования: Шеннона-Фано.

4. Теоретические сведения:

Алгоритм Шеннона-Фано:

1. Все кодируемые символы располагаются в порядке убывания вероятностей;
2. Разбиваем сортированный алфавит на две составляющие, вероятности символов в которых являются максимально близкими друг к другу;
3. В префиксный код первой части символов добавляем – 0, в префиксный код второй части добавляем – 1;
4. Для каждой составляющей, имеющей не менее двух символов, рекурсивно исполняем шаги от 2 до 4.

При своей сравнительной простоте, алгоритм Шеннона-Фано имеет недостатки, т.к. хотя разбиение на каждом шаге можно считать оптимальным, тем не менее, алгоритм не может гарантировать оптимальный итоговый результат в целом.

Пример:

Буква	A	B	C	D	E	F
Частота появления	50	39	18	49	35	24

Решение:

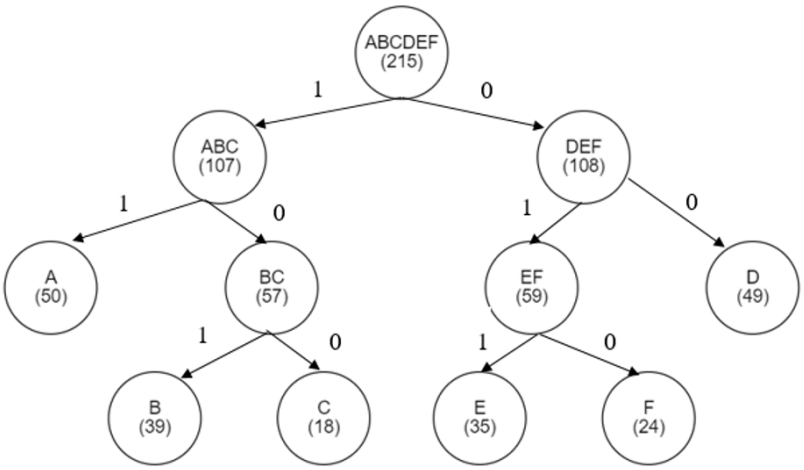


Рисунок 1.1 – Пример построения дерева по алгоритму Шеннона-Фано

Расчет среднего числа элементарных сигналов:

Среднее число элементарных сигналов, также известное как средняя длина кодового слова, рассчитывается по следующей формуле:

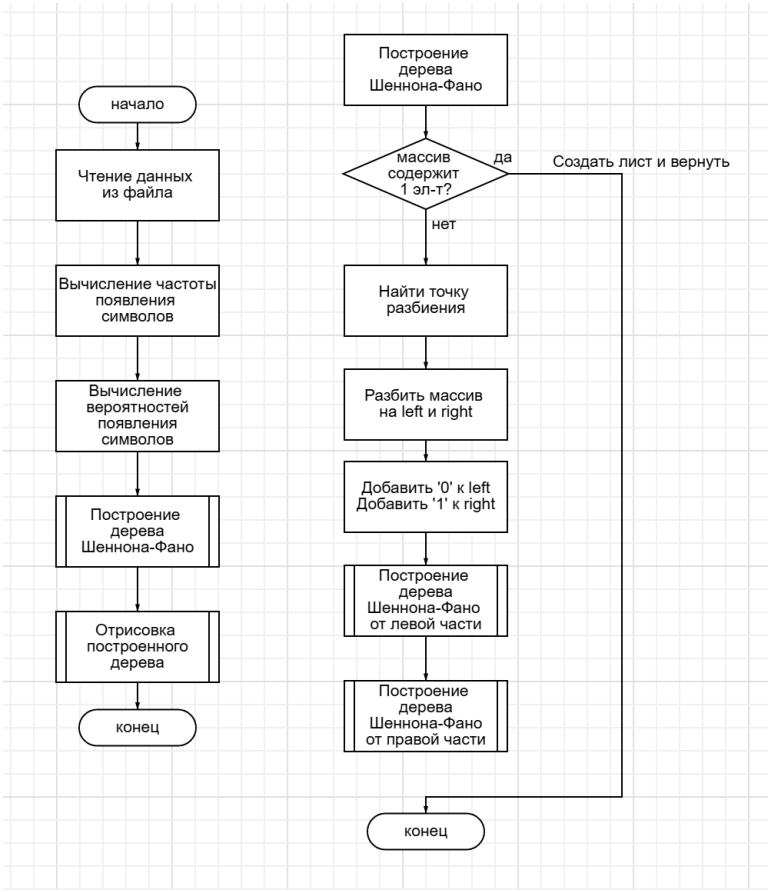
$$L_{\text{ср}} = \sum_{i=1}^{N} p_i * l_i, \quad (1)$$

где: $L_{\text{ср}}$ – средняя длина кодового слова. N – количество символов в алфавите. p_i – вероятность появления i-го символа. l_i – длина кодового слова для i-го символа.

4. Ход работы:

Для выполнения работы был выбран язык программирования высокого уровня Python. Была разработана программа (Приложение А), кодирующая символы из поданной на вход последовательности по алгоритму Шеннона-Фано.

Для наглядности работы алгоритма была сделана блок схема (рисунок 2):



а) б)

Рисунок 2 – Блок схема алгоритма Шеннон-Фано (а – основная программа, б – подпрограмма построение дерева)

Результат построения дерева программой представлен на рисунке 3.

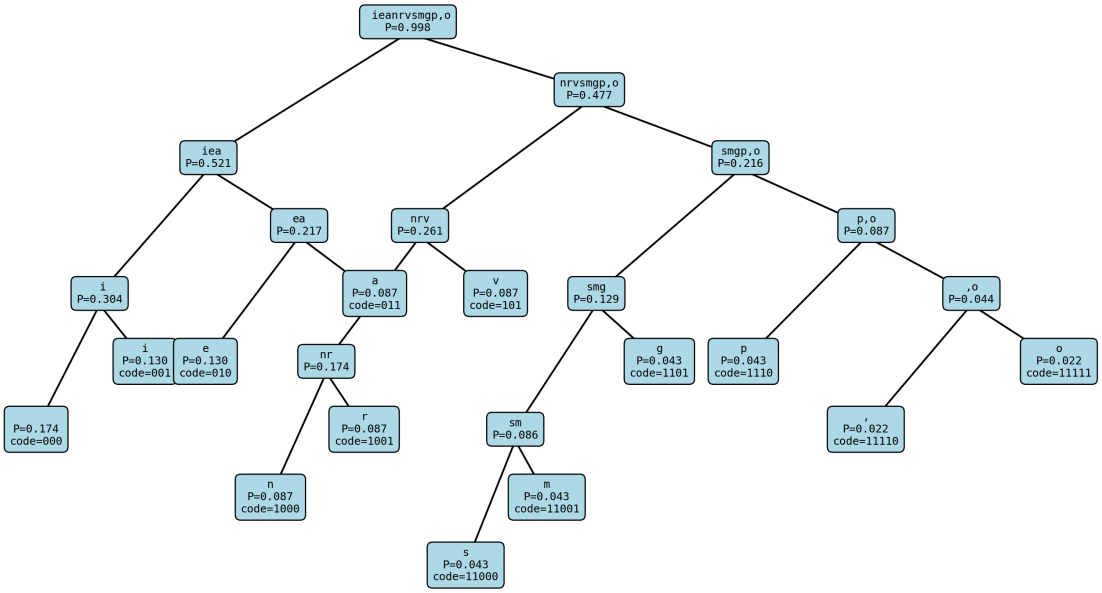


Рисунок 3 – Построенное дерево для заданной фразы

Для проверки работы написанной программы и правильности результата была выполнена ручная трассировка, представленная на рисунке 4.1, 4.2.

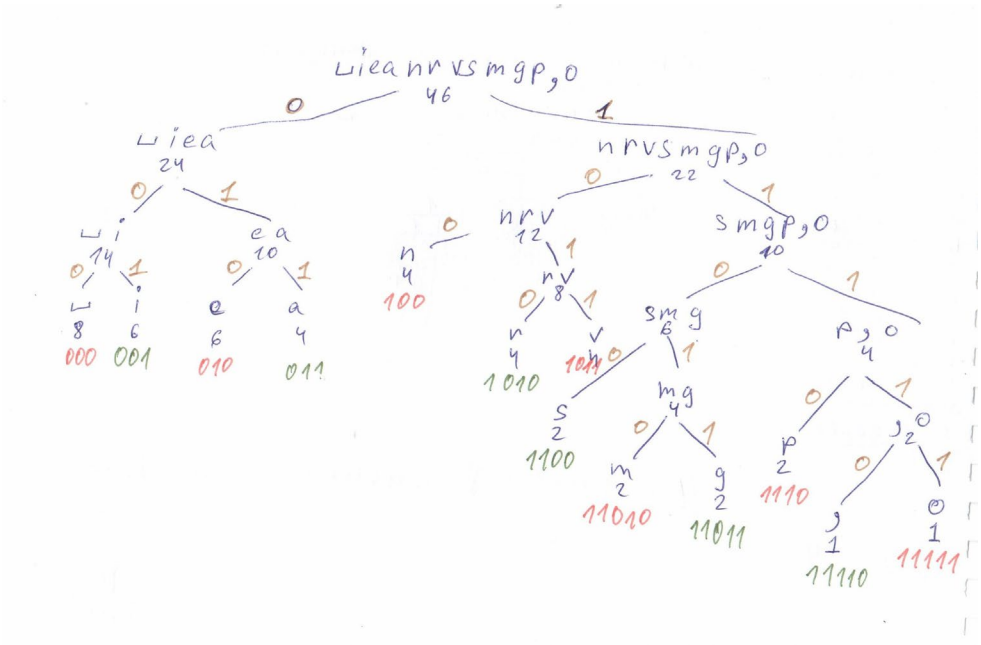


Рисунок 4.1 – Ручная трассировка – дерево Шеннона - Фано

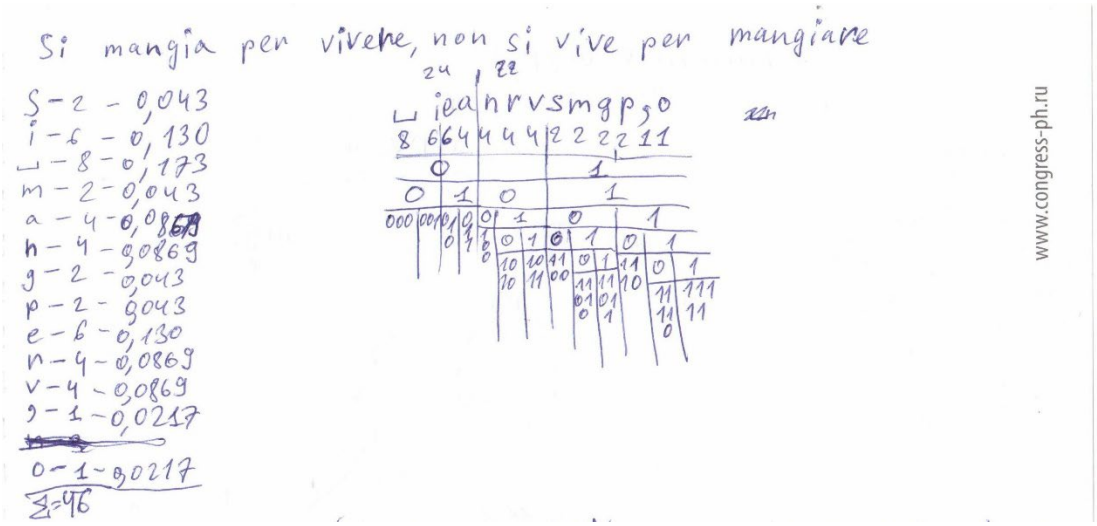


Рисунок 4.2 – Ручная трассировка, кодирование

Как видно, расхождения в результатах минимально и различается только в тех символах, где поделить ровно было невозможно и выбирался произвольный символ – “nrv”: в ручном варианте поделено на “n” и “rv”, а в программном варианте на “v” и “nr”, при том, что символы “n”, “r”, “v” имеют равную частоту и/или вероятность появления.

Среднее число элементарных сигналов (длина кода) посчитаем по формуле (1):

$$L_{\text{ср}} = \frac{5*4+4*4+3*5}{13} = \frac{51}{13} \approx 3,923\ldots \approx 4 \text{ бит}$$

5. Вывод:

В ходе выполнения лабораторной работы был изучен метод кодирования текстовой информации алгоритмом Шеннона-Фано. Была написана программа, строящая дерево по этому алгоритму, а также для проверки её работы была проведена ручная трассировка и сравнение результатов: почти идентичны. Также было рассчитано среднее число единичных сигналов = 4.

ПРИЛОЖЕНИЕ А

Листинг программы

```
import matplotlib.pyplot as plt

# === Чтение текста и подсчет частот ===
Code = []
with open("text.txt", encoding='utf8') as file:
    text = file.read().lower()

d = {}
for i in text:
    d[i] = d.get(i, 0) + 1

# переводим в вероятности
sum_sim = sum(d.values())
for i in d:
    d[i] = round(d[i] / sum_sim, 3)

# сортируем по убыванию вероятности
d = sorted(d.items(), key=lambda x: x[1], reverse=True)

# создаем массив для кодов
arr = []
for i in d:
    arr.append([i[0], i[1], ''])

# === Рекурсивная функция построения дерева Шеннона-Фано ===
def build_tree(arr):
    if len(arr) == 1:
        sym, prob, code = arr[0]
        return {
            'symbols': [sym],
            'prob_sum': prob,
            'code': code,
            'left': None,
            'right': None
        }

    total = sum(x[1] for x in arr)
    sum1 = 0
    index = 0
    for i, j in enumerate(arr):
        sum1 += j[1]
        if sum1 >= total / 2:
            # выбираем индекс так, чтобы разбиение было ближе к половине
            if i > 0 and abs((sum1 - j[1]) - total / 2) < abs(sum1 - total /
2):
                index = i
            else:
                index = i + 1
            break

    left_arr = arr[:index]
    right_arr = arr[index:]

    # добавляем коды
    for i in left_arr:
        i[2] += '0'
    for i in right_arr:
        i[2] += '1'

    left_node = build_tree(left_arr) if left_arr else None
```

```
    right_node = build_tree(right_arr) if right_arr else None

    return {
        'symbols': [s[0] for s in arr],
        'prob_sum': sum(x[1] for x in arr),
        'code': '',
        'left': left_node,
        'right': right_node
    }

tree_root = build_tree(arr)

# === Рекурсивная функция для отрисовки дерева ===
def plot_tree(node, x=0, y=0, dx=5, ax=None):
    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 8))
        ax.axis('off')
        plot_tree(node, x, y, dx, ax=ax)
        plt.show()
        return

    # подпись узла
    symbols_str = ''.join(node['symbols'])
    prob_str = f"{node['prob_sum']:.3f}"
    code_str = node.get('code', '')
    label = f"{symbols_str}\nP={prob_str}"
    if code_str:
        label += f"\ncode={code_str}"

    ax.text(x, -y, label, ha='center', va='center',
            bbox=dict(facecolor='lightblue', edgecolor='black',
boxstyle='round,pad=0.5'),
            fontsize=9, fontfamily='monospace')

    # соединяем с потомками
    if node['left']:
        ax.plot([x, x - dx-0.5], [-y - 0.25, -y - 3], color='black')
        plot_tree(node['left'], x - dx-0.5, y + 3, dx / 2, ax=ax)
    if node['right']:
        ax.plot([x, x + dx], [-y - 0.25, -y - 1.5], color='black')
        plot_tree(node['right'], x + dx, y + 1.5, dx / 1.2, ax=ax)


plot_tree(tree_root)
```