МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9 НА ТЕМУ:**

**Сжатие/распаковка данных на основе статистических методов**

Выполнила студентка 3 курса 1 группы

Халалеенко Андрей Николаевич

Минск 2024

**Задание 1:** Разработать авторское приложение в соответствии с целью лабораторной работы. Входной блок данных может иметь произвольную длину.

**Задание 2:** С помощью приложения выполнить прямое и обратное преобразования сообщения, состоящего из собственных имени и фамилии.

Можно использовать любой из известных методов сортировки символов массива. Метод кодировки (Шеннона − Фано, Хаффмана) использовать по указанию преподавателя.

Для кодирования методом Шеннона − Фано был использован следующий код на языке *python*:

# разбиваем словарь на 2 части и возвращаем их

def divide\_list(list):

    if len(list) == 2:

        return [list[0]],[list[1]]

    else:

        n = 0

        for i in list:

            n += i[1]

        x = 0

        distance = abs(2\*x - n)

        j = 0

        for i in range(len(list)):

            x += list[i][1]

            if distance < abs(2\*x - n):

                j = i

    return list[0:j+1], list[j+1:]

c ={}

# в коды первой часть словаря добавляется 1, в коды второй 0

def label\_list(list):

    list1,list2 = divide\_list(list)

    for i in list1:

        i[2] += '0'

        c[i[0]] = i[2]

    for i in list2:

        i[2] += '1'

        c[i[0]] = i[2]

    if len(list1)==1 and len(list2)==1:

        return

    # рекурсия

    label\_list(list2)

    return c

# составляем сжатое слово из кодов

def compress\_shannon\_fano(c, seq):

    encoded = ''

    for i in seq:

        encoded += c[i[0]]

    return encoded

# находим исходное сообщение по словарю из сжатого сообщения

def decompress\_shannon\_fano(c, seq):

    decompressed\_string = ''

    code = ''

    letter\_binary = []

    for key, value in c.items():

        letter\_binary.append([key,value])

    for digit in seq:

        code += digit

        pos = 0

        for letter in letter\_binary:

            if code == letter[1]:

                decompressed\_string += letter\_binary[pos] [0]

                code= ''

            pos += 1

    return decompressed\_string

Листинг 1 – метод Шеннона – Фано

Для кодирования методом Хаффмана был использован следующий код на языке *python*:

def combine\_nodes(nodes, huffman\_tree):

    pos = 0

    newnode = []

    if len(nodes) > 1:

        nodes.sort()

        nodes[pos].append("1")

        nodes[pos+1].append("0")

        combined\_node1 = (nodes[pos] [0] + nodes[pos+1] [0])

        combined\_node2 = (nodes[pos] [1] + nodes[pos+1] [1])

        newnode.append(combined\_node1)

        newnode.append(combined\_node2)

        newnodes=[]

        newnodes.append(newnode)

        newnodes = newnodes + nodes[2:]

        nodes = newnodes

        huffman\_tree.append(nodes)

        combine\_nodes(nodes, huffman\_tree)

    return huffman\_tree

def print\_huffman\_tree(huffman\_tree):

    huffman\_tree.sort(reverse = True)

    print('Дерево Хаффмана:')

    checklist = []

    for level in huffman\_tree:

        for node in level:

            if node not in checklist:

                checklist.append(node)

            else:

                level.remove(node)

    count = 0

    for level in huffman\_tree:

        print('Level', count,':',level)

        count+=1

    print()

    return checklist

def bin\_list(seq, huffman\_tree):

    letter\_binary = []

    only\_letters = get\_letters(seq)

    checklist = print\_huffman\_tree(huffman\_tree)

    for letter in only\_letters:

        code = ''

        for node in checklist:

            if len(node) > 2 and letter in node[0]:

                code = code + node[2]

        lettercode = [letter, code]

        letter\_binary.append(lettercode)

    print('Binary code generated:')

    for letter in letter\_binary:

        print(letter[0], letter[1])

    return letter\_binary

def compress\_huffman(letter\_binary, seq):

    encoded = ''

    for character in seq:

        for item in letter\_binary:

            if character in item:

                encoded = encoded + item[1]

    return encoded

Листинг 2 – метод Хаффмана

При этом таблица отсортированных символов строится:

а) на основе данных, полученных в лабораторной работе № 2;

б) динамически, на основе анализа сжимаемого сообщения.

Для выполнения этого задания был использован следующий код на языке *python*:

# возвращает словарь {символ:количество потворений этого символа}

def get\_letters\_amount(seq):

    letters\_dictionary = {}

    for i in seq:

        if i not in letters\_dictionary:

            letters\_dictionary[i] = 0

        letters\_dictionary[i] += 1

    return letters\_dictionary

# возвращает словарь {символ:вероятность этого символа}

def get\_letters\_probabilities(seq):

    probs = {}

    letters\_dictionary = get\_letters\_amount(seq)

    total\_amount = sum(letters\_dictionary.values())

    for i in letters\_dictionary.keys():

        probs[i] = letters\_dictionary[i]/total\_amount

    return probs

# возвращает словарь вероятностей алфавита для символов сообщения

def get\_statistically\_seq\_probabilities(seq, statistically\_probs):

    seq\_probabilities = {}

    for i in seq:

        if i in statistically\_probs:

            seq\_probabilities[i] = statistically\_probs[i]

    return seq\_probabilities

# сортирует словарь и приводит к виду [буква, частота, код]

def create\_list(data):

    list = dict(collections.Counter(data))

    list\_sorted = sorted(iter(list.items()), key = lambda k\_v:(k\_v[1],k\_v[0]),reverse=True)

    final\_list = []

    for key,value in list\_sorted:

        final\_list.append([key,value,''])

    return final\_list

Листинг 3 – построение таблицы отсортированных символов

**Задание 3:** Определить эффективность (в сравнении с кодами ASCII) сжатия сообщения.

Для выполнения этого задания код, разработанный в предыдущем задании был дополнен следующим кодом:

print('Длина сообщения методом Шеннона-Фано:', len(compressed))

print('Длина в ASCII:', len(''.join(format(ord(ch), '08b') for ch in seq)))

Листинг 4 – задание 3

Результат работы итогового кода:

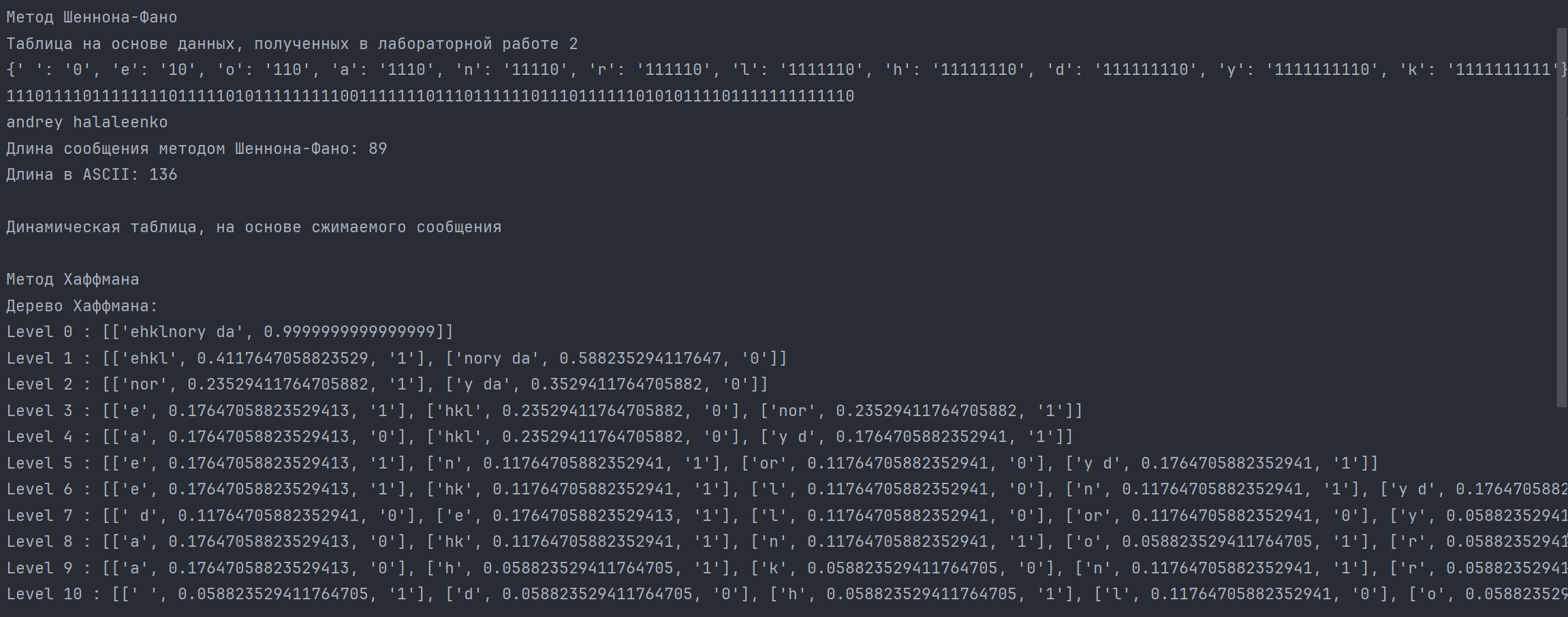


Рисунок 1 ­­­­– Результат работы программы

**Вывод:** в ходе лабораторной работы были приобретены практические навыки использования статистических методов Шеннона − Фано и Хаффмана (ShannonFano and Huffman coding) для сжатия/распаковки данных.