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 № 1

по дисциплине: Теория информации

тема: «Исследование кодирования по методу Хаффмана. Оценка эффективности кода»

Выполнил: ст. группы ПВ-223
Игнатьев Артур Олегович

Проверил:
Твердохлеб Виталий Викторович

Белгород 2024г.

Лабораторная работа №1

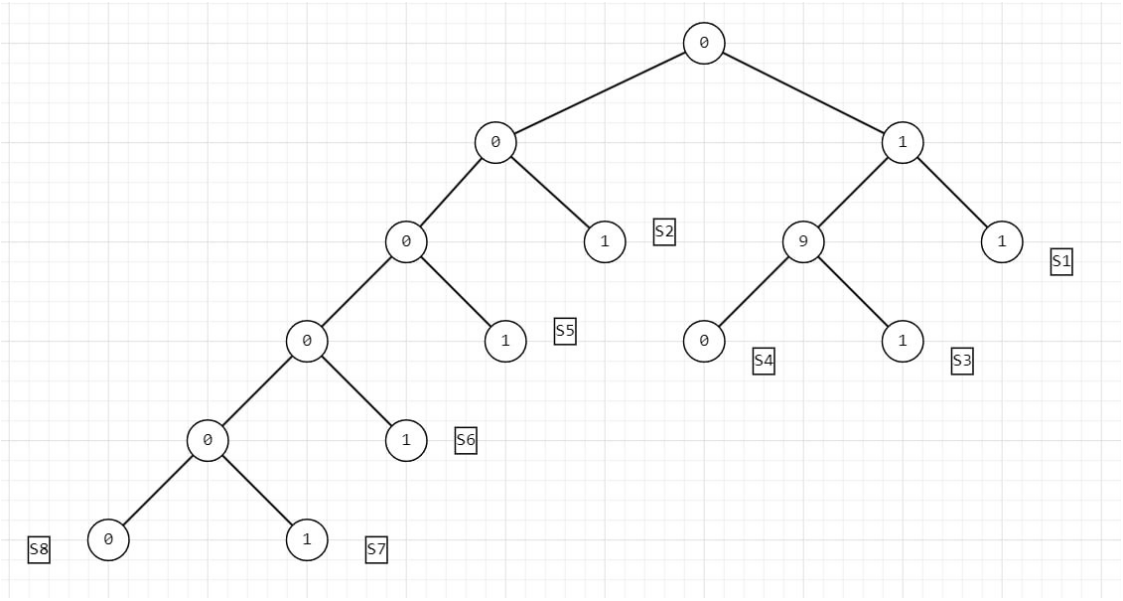
«Исследование кодирования по методу Хаффмана. Оценка эффективности кода»

Цель работы: Исследование кодирования по методу Хаффмана. Оценка эффективности кода.

Решение задач:

Вариант 3

1. Построить кодовое представление сообщения, вероятности появления символов в пределах алфавита которого приведены в табл.1.



S8 – 00000

S7 – 10000

S6 – 1000

S5 – 100

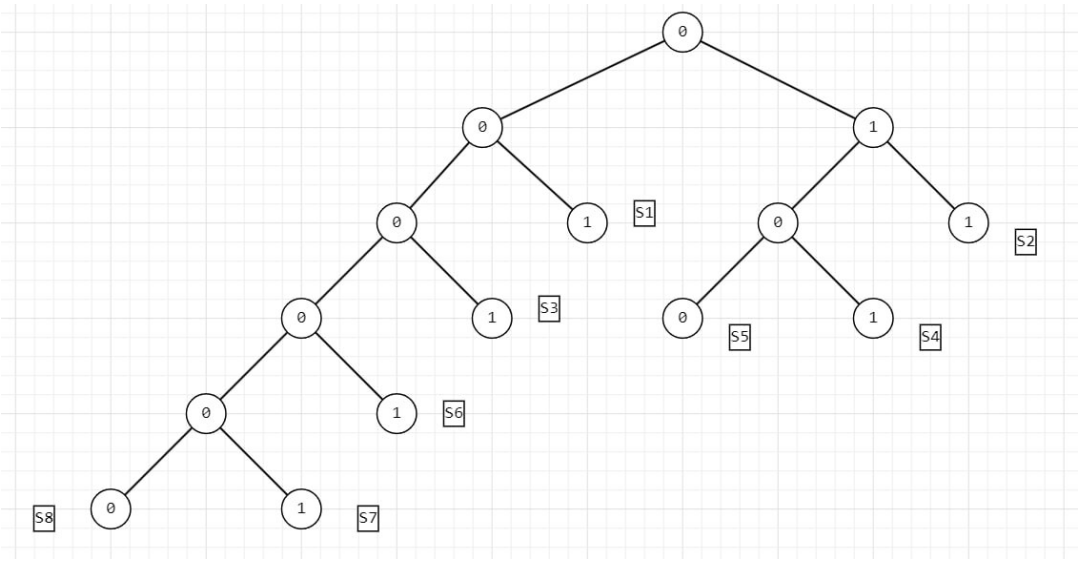
S4 – 001

S3 – 101

S2 – 10

S1 – 11

2. Построить кодовое представление сообщения, вероятности появления символов в пределах алфавита которого приведены в табл.2.



S8 – 00000

S7 – 10000

S6 – 1000

S5 – 001

S4 – 101

S3 – 100

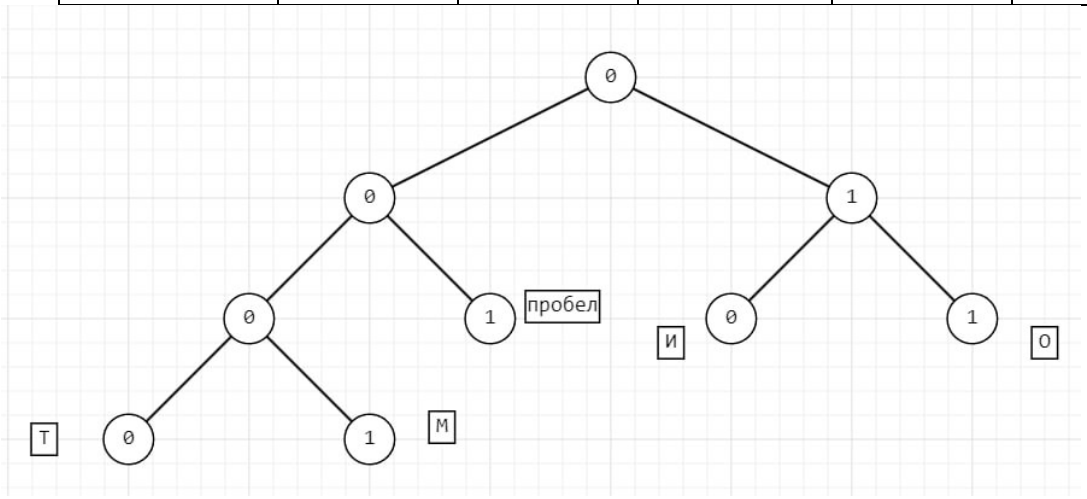
S2 – 11

S1 – 10

3. Построить кодовое представление сообщения:

оитомии о ими ооoитми о о о ооиимтомиимотоим оои тоо и и м оио иомтоо тоимо т и

Символ	0	и	пробел	м	Т
Вероятность	0.31	0.24	0.22	0.13	0.10



Т – 000

М – 100

Пробел – 10

И – 01

О – 11

Кодовое представление сообщения:

111001011011101000110010011100011111110010011100011001100110011111010011010110111010011110101110011001111100001011110010001000011001110110010001101101011110001011100111100010001

4. Для условий, приведенных в заданиях 1 и 2 и 3, выявить возможность построения альтернативных кодовых моделей сообщения. В случае обнаружения таковых, выявить наиболее эффективные из них по критериям.

Вычисление коэффициента сжатия и дисперсии

Задание 3:

Т – 000, М – 100, пробел – 10, И – 01, О – 11.

n=79; η=3;

B=79*3=237;

B'=176;

Коэффициент сжатия :

$$K_{comp} = \frac{B}{B'} = \frac{237}{176} = 1{,}346$$

Средняя длинна кода L.

$$L = \sum_{i=1}^{5} p_i l_i = 0.1 * 3 + 0.13 * 3 + 0.22 * 2 + 0.24 * 2 + 0.31 * 2 = 2.23$$

Значение дисперсии:

$$\delta = \sum_{i=1}^{5} p_i \ (l_i - L)^2$$

$$= 0.1 * (3 - 2.23)^2 + 0.13 * (3 - 2.23)^2 + 0.22 * (2 - 2.23)^2$$

$$+ 0.24 * (2 - 2.23)^2 + 0.31 * (2 - 2.23)^2 = 0.1771$$

5. Алгоритм Хаффмана на языке Python

```
import heapq
from collections import defaultdict

def buildHuffmanTree(text):
    # Подсчет частоты символов текста в словаре
    symbols_freq = defaultdict(int)
    for symbol in text:
        symbols_freq[symbol] += 1

    # Строим список [<"вес" символа>, [<символ>, <место для кода>]] из
словаря
    heap = [[weight, [symbol, ""]] for symbol, weight in
symbols_freq.items()]
    # Преобразуем список в кучу
    heapq.heapify(heap)

    while len(heap) > 1:
        # Удаляем два наименьших элемента из кучи
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)

        # Формируем коды для двух значений
        for pair in left[1:]:
            pair[1] = '0' + pair[1]

        for pair in right[1:]:
            pair[1] = '1' + pair[1]

        # Добавляем результирующий "вес" в кучу
        heapq.heappush(heap, [left[0] + right[0]] + left[1:] + right[1:])

    return heap[0]

# Перевод получившегося результата из кучи в список
def buildHuffmanCodes(tree):
    huff_codes = {}

    for pair in tree[1:]:
        symbol, code = pair
        huff_codes[symbol] = code

    return huff_codes
```

Работа на примере задачи 3:

```
text = "оитомии о ими оооитми о о о ооиимтомиимотоим оои тоо и и м оио иомтоо
тоимо т и"

# Построение дерева Хаффмана и генерация кодов
huffmanTree = buildHuffmanTree(text)
huffmanCodes = buildHuffmanCodes(huffmanTree)

print(f"{text}")
print(f"{huffmanCodes}")
```

```
оитомии о ими оооитми о о о ооиимтомиимотоим оои тоо и и м оио иомтоо тоимо т и
{' ': '00', 'т': '010', 'м': '011', 'и': '10', 'о': '11'}

Process finished with exit code 0
```

Функция нахождения коэффициента сжатия и избыточности кода

```
def printCompressionRatioAndRedundacy(text):
    different_symbols_num = symbolsNum(text)
    all_symbols_num = sum([value for value in different_symbols_num.val-
ues()])

    eta = 1
    while eta * eta < len(different_symbols_num):
        eta += 1

    b = eta * all_symbols_num

    huffmanTree = buildHuffmanTree(text)
    huffmanCodes = buildHuffmanCodes(huffmanTree)

    bst = 0
    for key in different_symbols_num:
        if key in huffmanCodes:
            bst += different_symbols_num[key] * len(huffmanCodes[key])

    k = b / bst

    for num in different_symbols_num:
        different_symbols_num[num] = different_symbols_num[num] / all_sym-
bols_num

    l = 0
    for key in different_symbols_num:
        l += different_symbols_num[key] * len(huffmanCodes[key])

    delta = 0
    for key in different_symbols_num:
        delta += different_symbols_num[key] * math.pow(len(huffmanCodes[key])
- l, 2)

    print(f"Compression: {k}")
    print(f"Redundancy: {delta}")
```

Работа на примере задачи 3:

```
Compression: 1.3465909090909092
Redundancy: 0.17593334401538216
```

Вывод: в ходе выполнения лабораторной работы исследовано кодирование по методу Хаффмана. Оценена эффективность кода.