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

Дмитриев Андрей Александрович

Проверил:

Твердохлеб Виталий Викторович

Белгород 2024 г.

**Цель работы**: исследовать кодирования по методу Хаффмана. Научиться оценивать эффективность кода.

**Задания лабораторной работы**

**Задание №1.** Построить кодовое представление сообщения, вероятности появления символов в пределах алфавита которого приведены в табл.1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 |
| Вероятность | 0.23 | 0.19 | 0.16 | 0.16 | 0.10 | 0.10 | 0.05 | 0.01 |

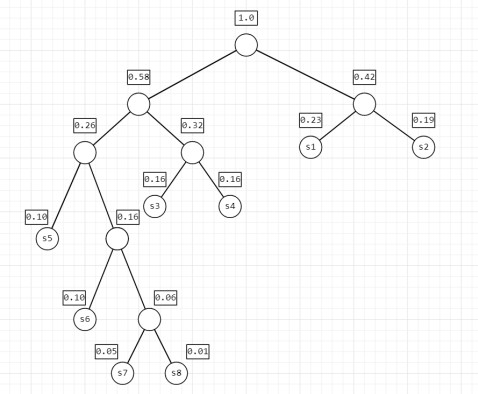
Кодовое представление сообщения мы можем построить благодаря методу деревьев Хаффмана.

1. Создать таблицу частотности символов в заданной строке или файле.
2. Создать лес узлов-деревьев, в котором каждый узел состоит из одного символа и его частоты в таблице.
3. Объединять два узла дерева с наименьшими частотами в новый узел, который будет иметь сумму их частот.
4. Удалить объединенные узлы из леса и добавить новый узел в лес. 5. Повторять шаги 3-4 до тех пор, пока не останется один узел в лесу - корень дерева.
5. Присвоить коды символам, используя левое направление (0) для листьев, а правое направление (1) для узлов-родителей.
6. Создать закодированную версию строки, используя полученные коды символов.
7. Создать таблицу декодирования символов, используя коды символов.
8. Декодировать закодированную строку, используя таблицу декодирования символов.

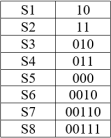
Ниже представлено полученное дерево Хаффмана.

В таблице есть символы с одинаковой вероятностью, значит, могут быть построены различные деревья, однако их эффективность останется неизменной.

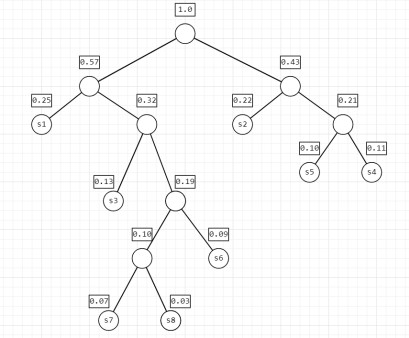
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 |
| Вероятность | 0.25 | 0.22 | 0.13 | 0.11 | 0.1 | 0.09 | 0.07 | 0.03 |

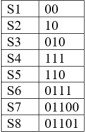


Коды символов

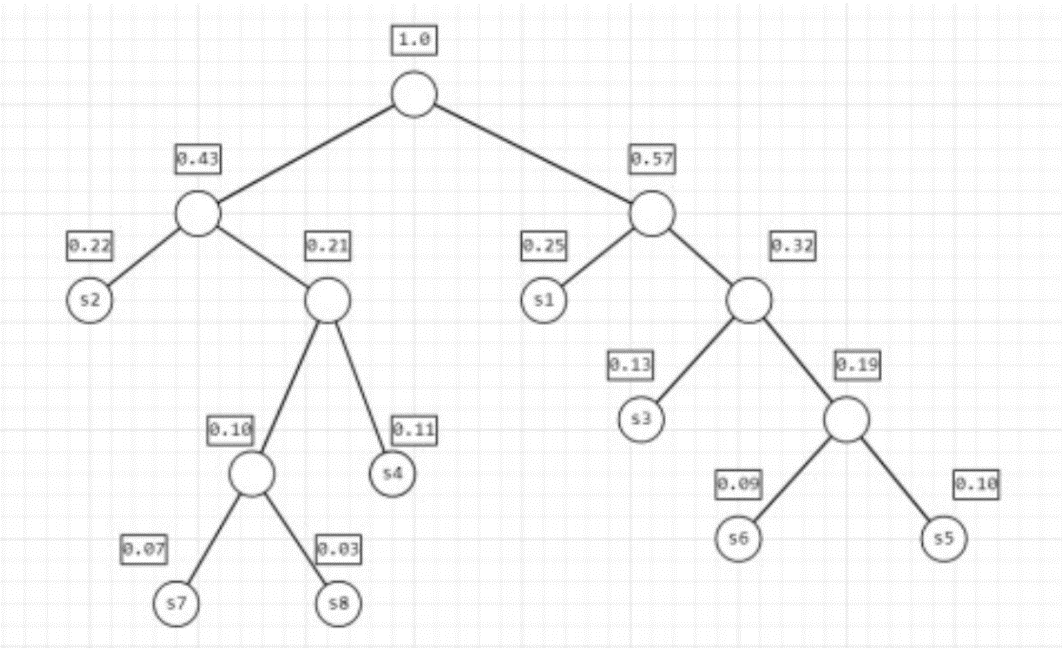
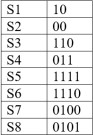


**Задание №2.** Построить кодовое представление сообщения, вероятности появления символов в пределах алфавита которого приведены в табл.2.





Пример альтернативного дерева (другая эффективность)



**Задача №3.** Построить кодовое представление сообщения:

оитомии о ими оооитми о о о ооиимтомиимотоим оои тоо и и м оио и омтоо тоимо т и

Решение:

Построим таблицу появления символов в сообщении.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Символ | о | и | Т | м | ‘ ‘ |
| Кол-во появлений | 25 | 19 | 8 | 10 | 18 |

Построим закодированное сообщение:

оитомии о ими оооитми о о о ооиимтомиимотоим оои тоо и и м оио и омтоо тоимо т и =

1110010110111010001100100111000111111100100111000110011001100111110100110101101110100111101011100110011111000010111100100010000110011101100100011011010111100010111001111000100010

**Задание №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 𝐾𝑐𝑜𝑚𝑝 и 𝛿.

Решение:

Возьмём для расчёта следующий вариант кодирования:

𝑠1 = 01, 𝑠2 = 00, 𝑠3 = 101, 𝑠4 = 110, 𝑠5 = 1111, 𝑠6 = 100, 𝑠7 = 11101, 𝑠8 = 11100.

Вычислим среднюю длину кода (𝐿) и энтропию (𝐻) для кодирования:

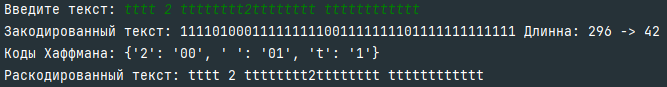
Получим ответ:

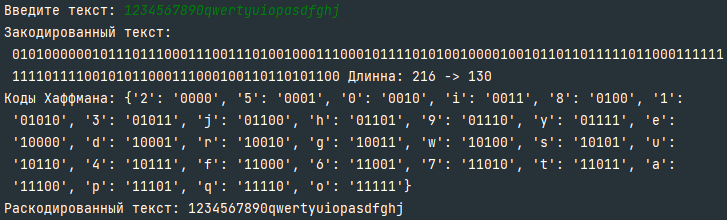
|  |  |
| --- | --- |
|  |  |

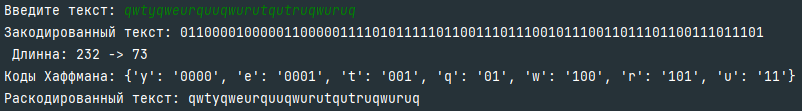
**Кодовая реализация.**

import heapq  
from collections import defaultdict  
  
  
class Node:  
 def \_\_init\_\_(self, char, freq):  
 self.char = char  
 self.freq = freq  
 self.left = None  
 self.right = None  
  
 def \_\_lt\_\_(self, other):  
 return self.freq < other.freq  
  
  
def build\_huffman\_tree(text):  
 *# Подсчет частоты символов в тексте* freq\_map = defaultdict(int)  
 for char in text:  
 freq\_map[char] += 1  
  
 *# Создание очереди с приоритетом из узлов-листьев* priority\_queue = []  
 for char, freq in freq\_map.items():  
 heapq.heappush(priority\_queue, Node(char, freq))  
  
 *# Строим дерево Хаффмана* while len(priority\_queue) > 1:  
 left\_node = heapq.heappop(priority\_queue)  
 right\_node = heapq.heappop(priority\_queue)  
 merged\_node = Node(None, left\_node.freq + right\_node.freq)  
 merged\_node.left = left\_node  
 merged\_node.right = right\_node  
 heapq.heappush(priority\_queue, merged\_node)  
  
 return priority\_queue[0]  
  
  
def build\_huffman\_codes(node, current\_code, huffman\_codes):  
 if node is None:  
 return  
  
 *# Если узел - лист, то добавляем его код в словарь кодов Хаффмана* if node.char is not None:  
 huffman\_codes[node.char] = current\_code  
 return  
  
 *# Рекурсивно строим коды для левого и правого поддеревьев* build\_huffman\_codes(node.left, current\_code + "0", huffman\_codes)  
 build\_huffman\_codes(node.right, current\_code + "1", huffman\_codes)  
  
  
def huffman\_encoding(text):  
 root = build\_huffman\_tree(text)  
 huffman\_codes = {}  
 build\_huffman\_codes(root, "", huffman\_codes)  
  
 encoded\_text = ""  
 for char in text:  
 encoded\_text += huffman\_codes[char]  
  
 return encoded\_text, huffman\_codes  
  
  
def huffman\_decoding(encoded\_text, huffman\_codes):  
 decoded\_text = ""  
 shift = 0  
 while shift < len(encoded\_text):  
 for val, code in huffman\_codes.items():  
 if code in encoded\_text[shift:len(code)+shift]:  
 decoded\_text += val  
 shift += len(code)  
  
 return decoded\_text  
  
  
*# Пример использования*text = input("Введите текст: ")  
encoded\_text, huffman\_codes = huffman\_encoding(text)  
print("Закодированный текст:", encoded\_text, "Длинна:", len(text), "->", len(encoded\_text))  
print("Коды Хаффмана:", huffman\_codes)  
decoded\_text = huffman\_decoding(encoded\_text, huffman\_codes)  
print("Раскодированный текст:", decoded\_text)

**Результаты работы алгоритма:**

****

****

****

**Вывод**: в ходе выполнения лабораторной работы были изучен алгоритм кодирования по методу Хаффмана, и применён метод оценки эффективности кода Хаффмана.