«Дискретная математика для программистов»
Лабораторная работа № 3. (весна 2024, МО-2)
Вариант 5
Предметом исследования являются методы кодирования.

1.Код Хемминга. Строку «abstract» перевести в двоичный ANSI код, разбить на два блока по 32 бита, добавить контрольные биты, имитировать ошибки в 5 бите первого блока и 21 бите второго блока, восстановить исходную информацию.

```python
str = "abstract"
binary_str = ""
for char in str:
        ascii_value = ord(char)
        binary_value = bin(ascii_value)[2:].zfill(8) # Преобразование в 8-битную двоичную строку
        binary_str += binary_value

print("Сама строка:",  binary_str)
print("Первый блок:", binary_str[:32])
print("Второй блок:", binary_str[32:])
```

Run test

```
C:\Users\User\AppData\Local\Programs\Python\Python310\python.exe "C:\Programming\Github\labs_term_04
Сама строка: 01100001011000100111001101110100011100100110000101100011011101 00
Первый блок: 01100001011000100111001101110100
Второй блок: 01110010011000010110001101110100
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	1	0	0	1	1	0	1	1	1	0	1	0	0
0	1	1	1	0	0	1	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	1	1	0	1	0	0

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	1	1	1	0	0	1	1	0	1	0	1	1	0	1	0	0
0	0	0	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0	1	1	0	1	0	0

1	2	4	8	16	32
7	6	11	8	7	3
9	8	11	7	6	3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
1	0	0	1	1	1	0	0	1	0	0	1	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	1	1	0	1	1	1	1	0	1	0	0
1	0	0	1	1	1	1	1	0	0	1	0	0	1	1	0	0	0	0	0	1	0	1	1	0	1	0	1	1	0	1	1	1	1	0	1	0	0

1	2	4	8	16	32
9	6	12	9	8	4
10	9	12	9	7	4

Ошибка в позиции
1 + 8 = 9
2 + 8 + 16 = 26

2.Расстояние Хемминга. Для набора букв «нопрстфх» придумать двоичные коды, с расстоянием не менее 2 и продемонстрировать поиск ошибки. Придумать двоичные коды, с расстоянием не менее 3 и продемонстрировать поиск и исправление ошибки.

Двоичные коды с расстоянием не менее 2:

н	00000	п	11100	с	10011	ф	01110	х	11001
о	00111	р	10101	т	11111				

Код для подсчета расстояний:

```
m = ["*", 'н', 'о', 'п', 'р', 'с', 'т', 'ф', 'х']
n = ['00000', '00111', '01110', '11100', '10101', '11001', '10011', '11111']
table = [m]
for i in range(0, len(n)):
    line = [m[i + 1]]
    for j in range(0, i):
        line.append(" ")
    line.append("-")
    for k in range(i + 1, len(n)):
        num = 0
        for l in range(0, len(n[i])):
            if n[i][l] != n[k][l]:
                num += 1
        line.append(str(num))
    table.append(line)

for i in range(0, len(n) + 1):
    print(table[i])
```

Результат работы кода:

```
C:\Users\User\AppData\Local\Programs\Python\Py
['*', 'н', 'о', 'п', 'р', 'с', 'т', 'ф', 'х']
['н', '-', '3', '3', '3', '3', '3', '3', '5']
['о', ' ', '-', '2', '4', '2', '4', '2', '2']
['п', ' ', ' ', '-', '2', '4', '4', '4', '2']
['р', ' ', ' ', ' ', '-', '2', '2', '4', '2']
['с', ' ', ' ', ' ', ' ', '-', '2', '2', '2']
['т', ' ', ' ', ' ', ' ', ' ', '-', '2', '2']
['ф', ' ', ' ', ' ', ' ', ' ', ' ', '-', '2']
['х', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '-']

Process finished with exit code 0
```

Декодер исправляет все ошибки, кратность которых не превышает

$$l \leq INT\left[\frac{d_{min} - 1}{2}\right]$$

В нашем случае $d_{min} = 2$, тогда код может исправить ноль ошибок, обнаружить одну ошибку

Поиск ошибки:

Допустим, что пришло сообщение 00110. Оно не совпадает ни с одним из кодов нашей таблицы. Минимальное расстояние, равное одному, соответствует двум символам:

Ф = 01110

О = 00111

Найдена одна ошибка, исправить нельзя.

Двоичные коды с расстоянием не менее 3:

н	00000000	р	11000001	ф	10100100
о	00000111	с	01010101	х	11111111
п	00011001	т	10110010		

Код:

```
m = ["*", 'н', 'о', 'п', 'р', 'с', 'т', 'ф', 'х']
n = ['00000000', '00000111', '00011001', '11000001', '01010101', '10110010', '10100100', '11111111']
table = [m]
for i in range(0, len(n)):
    line = [m[i + 1]]
    for j in range(0, i):
        line.append(" ")
    line.append("-")
    for k in range(i + 1, len(n)):
        num = 0
        for l in range(0, len(n[i])):
            if n[i][l] != n[k][l]:
                num += 1
        line.append(str(num))
    table.append(line)

for i in range(0, len(n) + 1):
    print(table[i])
```

```
for i in range(0, len(n)) › for k in range(i + 1, len(n)) › for l in range(0, len(n[i])) › if n[i][l] != n[k][l]
Run   test
C:\Users\User\AppData\Local\Programs\Python\Python310\python.exe "C:\Programming\Github\labs_term_04\Au
['*', 'н', 'о', 'п', 'р', 'с', 'т', 'ф', 'х']
['н', '-', '3', '3', '3', '4', '4', '3', '8']
['о', ' ', '-', '4', '4', '3', '5', '4', '5']
['п', ' ', ' ', '-', '4', '3', '5', '6', '5']
['р', ' ', ' ', ' ', '-', '3', '5', '4', '5']
['с', ' ', ' ', ' ', ' ', '-', '6', '5', '4']
['т', ' ', ' ', ' ', ' ', ' ', '-', '3', '4']
['ф', ' ', ' ', ' ', ' ', ' ', ' ', '-', '5']
['х', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '-']
```

Если минимальное расстояние равно трем, то это значит, что код может обнаружить 2 ошибки и исправить одну.

Поиск ошибки:

Допустим, что пришло сообщение 00001111. Оно не совпадает ни с одним из кодов нашей таблицы.

н: $d(00000000, 00001111) = 4$, с: $d(00000111, 00001111) = 1$

о: $d(00011001, 00001111) = 3$, т: $d(11000001, 00001111) = 5$

п: $d(01010101, 00001111) = 4$, ф: $d(10110010, 00001111) = 6$

р: $d(10100100, 00001111) = 5$, х: $d(11111111, 00001111) = 4$

Минимальное расстояние, равное одному, соответствует букве «с».

Найдена одна ошибка и одна исправлена.

3.Сжатие способом кодирования серий (RLE). Первый байт указывает сколько раз нужно повторить следующий байт. Если первый байт равен 00_{16}, то затем идет счетчик, показывающий сколько за ним следует неповторяющихся данных.

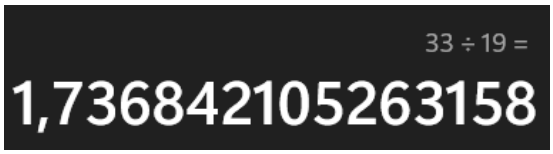
Строка для сжатия:

aaaaadgggggggggggggggghtyiklooooop

До сжатия: aaaaadgggggggggggggggghtyiklooooop (33 байт)

После сжатия: 5a1d15g06htyikl5o1p (19 байт)

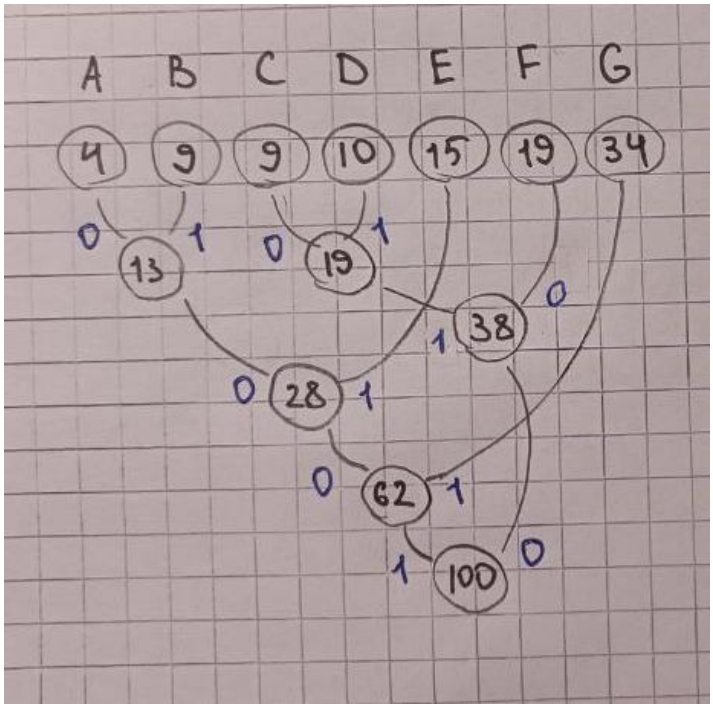
Степень сжатия



Коэффициент сжатия

4.Алгоритм Хаффмана. Пусть при подсчете вхождения каждого из символов в файл получили следующее:
(A 4), (B 9), (C 9), (D 10), (E 15), (F 19), (G 34)

Построить коды символов и продемонстрировать на примерах кодирование и раскодирование.



A 1000

B 1001

C 010

D 011

E 101

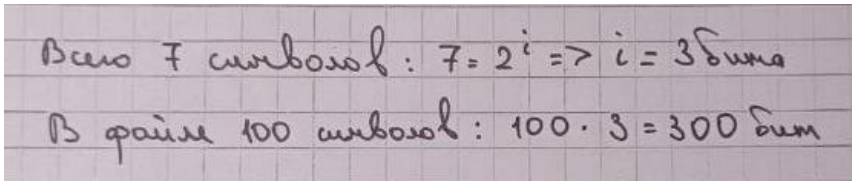
F 00

G 11

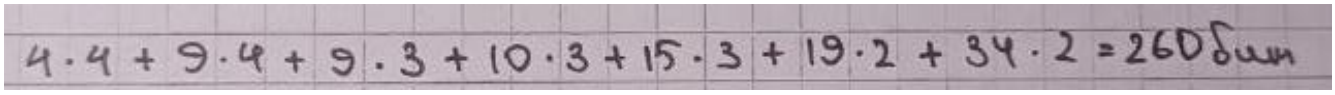
Кодирование слова BADGE:
1001 1000 011 11 101

Декодируем с помощью дерева

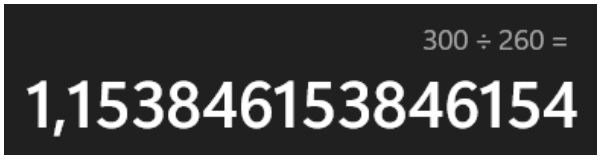
До сжатия:



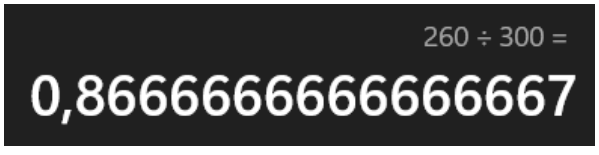
После сжатия:



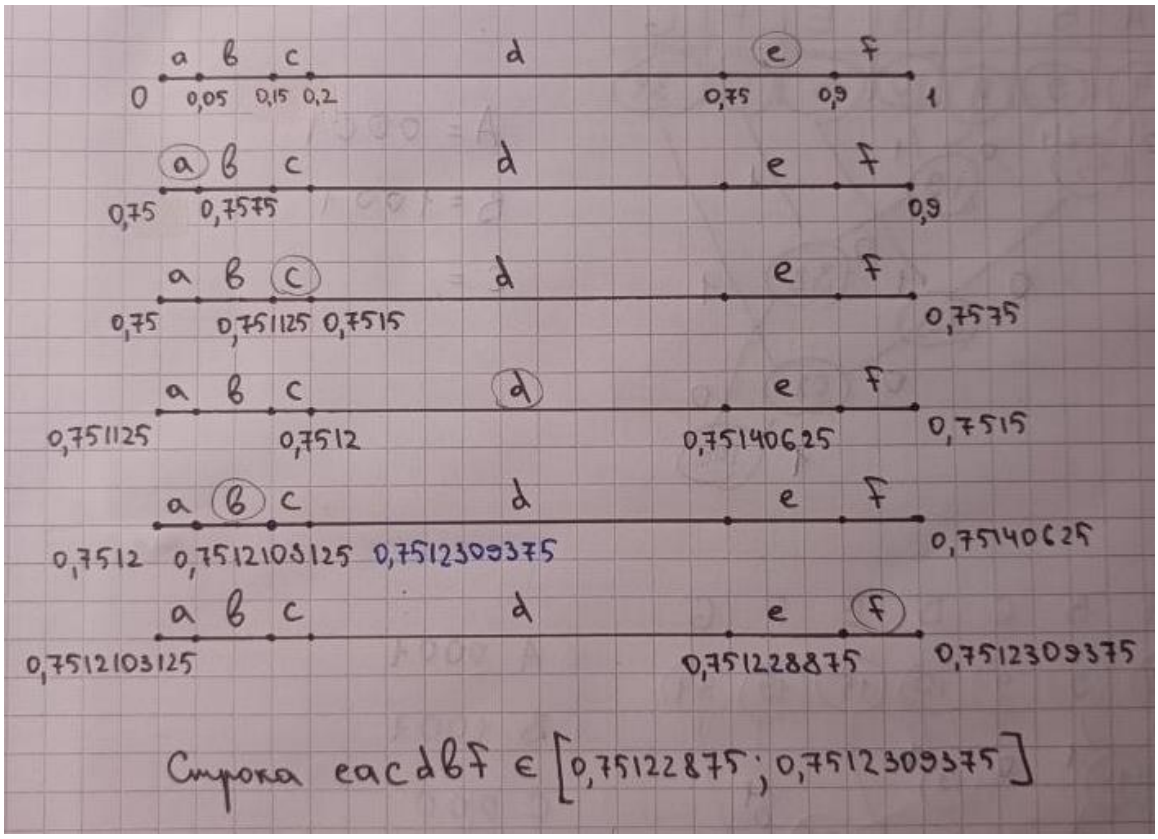
Степень сжатия



Коэффициент сжатия



5.Арифметическое кодирование. Пусть алфавит состоит из символов: a b c d e f
с вероятностями соответственно 0.05 0.10 0.05 0.55 0.15 0.10
Построить код для строки: eacdbf
Ответ дать в виде двоичной строки.

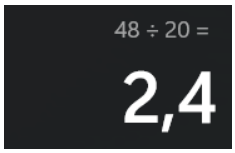


Пусть строка будет закодирована $751229_{10} = 10110111011001111101_2$

До сжатия: eacdbf (6 байт = 48 бит)

После сжатия: 10110111011001111101 (20 бит)

Степень сжатия



Коэффициент сжатия