Домашнее задание 2

Кокорин Илья, М3439

Вариант № 64

1 Начальные условия

Проверочная матрица $H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$

Порождающая матрица $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$

$r = 4$ (количество строк проверночной матрицы)

$n = 10$ (количество столбцов проверочной и порождающей матриц)

$k = 6$ (количество строк порождающей матрицы)

2 Указать информационную совокупность

Найдём список из $k = 6$ номеров линейно независимых столбцов матрицы G. Это и будет информационная совокупность.

Заметим, что первые 6 стобцов матрицы G образуют единичную подматрицу I_6, и поэтому они, очевидно, линейно независимы. Тогда информационная совокупность равна $(1, 2, 3, 4, 5, 6)$

3 Построить таблицу синдромного декодирования

Всего $2^r = 2^4 = 16$ синдромов s. $s \in \{0, 1\}^r$

Для каждого s найдём такое e, что $e \cdot H^T = s$, и вес e минимален.

$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$

$H^T = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

Будем действовать по следующему алгоритму: если s является i-ой H^T, то ставим 1 в i-ую компоненту e, а остальные компоненты забиваем нулями. Иначе ищем две такие строки i-ую и j-ую, которые бы в результате

операции $\oplus$ давали необходимую s, тогда ставим 1 в i-ую и j-ую компоненты e, а остальные забиваем нулями. И так далее.

s:	e
0000:	0000000000
0001:	0000010000
0010:	0100000000
0011:	0100010000
0100:	0010000000
0101:	1000000000
0110:	0000001000
0111:	0001000000
1000:	0010100000
1001:	0010000010
1010:	0000000100
1011:	0000010100
1100:	0000100000
1101:	0000000010
1110:	0000000001
1111:	0000010001

Напишем программу, проверяющую наши построения

```
def transpose(matrix):
        n = len(matrix)
        m = len(matrix[0])
        transposed = [[None for _ in range(n)] for _ in range(m)]

        for i in range(n):
                for j in range(m):
                        transposed[j][i] = matrix[i][j]
        return transposed


h = [
        [0, 0, 0, 0, 1, 0, 0, 1, 1, 1],
        [1, 0, 1, 1, 1, 0, 1, 0, 1, 1],
        [0, 1, 0, 1, 0, 0, 1, 1, 0, 1],
        [1, 0, 0, 1, 0, 1, 0, 0, 1, 0]
]

h_t = transpose(h)


def multiply(vect, mat):
        n = 1
        m = 10
        k = 4
        result = [0 for _ in range(k)]

        for j in range(k):
                for z in range(m):
                        result[j] += vect[z] * mat[z][j]
                        result[j] %= 2
        return result


ss = [
```

```
        [0, 0, 0, 0],
        [0, 0, 0, 1],
        [0, 0, 1, 0],
        [0, 0, 1, 1],
        [0, 1, 0, 0],
        [0, 1, 0, 1],
        [0, 1, 1, 0],
        [0, 1, 1, 1],
        [1, 0, 0, 0],
        [1, 0, 0, 1],
        [1, 0, 1, 0],
        [1, 0, 1, 1],
        [1, 1, 0, 0],
        [1, 1, 0, 1],
        [1, 1, 1, 0],
        [1, 1, 1, 1]
]

es = [
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 1, 0, 1, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
        [0, 0, 0, 0, 0, 1, 0, 0, 0, 1]
]

right = 0
for cur_s, cur_e in zip(ss, es):
        if multiply(cur_e, h_t) != cur_s:
                print(cur_s, cur_e, multiply(cur_e, h_t))
        else:
                right += 1

print(right) # 16
```