Łukasz Turowski, 45136, TD_20A

LAB_08

Zadanie 1.

Funkcja z poprzednich laboratoriów (String 2 Bits):

```
def S2BS(s):
    result = []
    for c in s:
        b = bin(ord(c))[2:]
        if len(b) < 8:
            b = '0' + b
        result.extend([int(x) for x in b])
    return result
```

Funkcja kodująca kod Hamminga (7,4):

```
def Hamming_encoder(d):
    shape = d.shape[0]
    enc = np.zeros((shape, 7))
    for i in range(shape):
        enc[i] = np.dot(G, d[i]) % 2
    return enc
```

Zadanie 2.

Funkcja negująca bity:

```
def flipbit(b):
    if (b == 0):
        return 1
    else:
        return 0
```

```
for i in range(int(len(x))):
    x[i, 3] = flipbit(x[i, 3])
```

Negujemy bity na pozycjach

```
# 0 4
# 1 4
# 2 4
# 3 4
```

Zadanie 3.

Funkcja dekodująca kod Hamminga:

```
def Hamming_decoder(d):
    shape = d.shape[0]
    dec = np.zeros((shape, 4))
    for i in range(shape):
        p = np.dot(H, d[i].T) % 2
        h = int(p[0] * 2 ** 0 + p[1] * 2 ** 1 + p[2] * 2 ** 2)
        if (h > 0):
            d[i, h - 1] = flipbit(d[i, h - 1])
            print(i, h)
        dec[i] = [d[i, 2], d[i, 4], d[i, 5], d[i, 6]]
    return dec
```

Wyniki:

```
Macierz G = np.array([[1, 1, 0, 1], [1, 0, 1, 1], [1, 0, 0, 0], [0, 1, 1, 1], [0,
1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])

Macierz H = np.array([[1, 0, 1, 0, 1, 0, 1], [0, 1, 1, 0, 0, 1, 1], [0, 0, 0, 1,
1, 1, 1]])

x = S2BS('hi')
# w całości: [0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1]
x = split(x)
# podzielone: [[0 1 1 0] [1 0 0 0] [0 1 1 0] [1 0 0 1]]
x = Hamming_encoder(x)
# zakodowane:[[1. 1. 0. 0. 1. 1. 0.] [1. 1. 1. 0. 0. 0. 0.] [1. 1. 0. 0. 1. 1. 0.]
[0. 0. 1. 1. 0. 0. 1.]]
# negujemy bity na pozycjach:
# 0 4
# 1 4
# 2 4
# 3 4
x = Hamming_decoder(x)
# odkodowane: [[0. 1. 1. 0.] [1. 0. 0. 0.] [0. 1. 1. 0.] [1. 0. 0. 1.]]
```