Sprawozdanie Lab08

Autor: Kamil Szóstak

```
import numpy as np
import matplotlib.pyplot as plt
import cmath
import math
import pylab
from scipy.interpolate import interp1d

def Binary(string):
    bin = ''.join(format(i, 'b') for i in bytearray(string, encoding ='utf-8'))
    bin = list(map(int, bin))
    #print('variant=littleEndian\n','conversion of ', string,' to binary is equal to [ ', bin,' ]')
    return bin
word = Binary('a')
#Test
word = [0,1,0,0,0,0,0,1]

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]])
hamm = np.array([[1,0,1,0,1,0,1], [0,1,1,0,0,1,1], [0,0,0,1,1,1,1]])

G_SECDED = np.array([[1,1,1,0,0,0,0,1], [1,0,0,1,1,0,0,1], [0,1,0,1,0,1,0,1], [1,1,0,1,0,0,1,0]])
hamm_SECDED = np.array([[1,0,1,0,1,0,1,0], [0,1,1,0,0,1,1,0], [0,0,0,1,1,1,1,0,], [1,1,1,1,1,1,1,1]])

print('Zadanie nr.1:')

print('Slowo zapisane binarnie - ', word)

word1 = np.array(word[0:4])
word2 = np.array(word[4:8])
```

```
print('pierwszy bit - ', word1)
print('drugi bit - ', word2)

h_word1 = (np.dot(G, word1)).transpose() % 2
print('Wektor nr.1 - ', h_word1)
h_word2 = (np.dot(G, word2)).transpose() % 2
print('Wektor nr.2 - ', h_word2)
Wynik = np.concatenate((h_word1,h_word2))
print('Hamming74 - ',Wynik)

print('\n')
print('Zadanie nr.2:')

negacja=np.logical_not(word).astype(int)
print('Negacja - ',negacja)
print('\n')

print('Zadanie nr.3:')

h = np.dot(hamm, h_word1.T) % 2
n_0 = h[0] ; n_1 = h[1] ; n_2 = h[2]

if n_0 == 0 | n_1 == 0 | n_2 == 0:
    p1 = (word[0] + word[1] + word[3] + word[4] + word[6]) % 2
    p2 = (word[0] + word[2] + word[3] + word[5] + word[6]) % 2
    p3 = (word[1] + word[2] + word[3] + word[7]) % 2
    p4 = (word[4] + word[5] + word[6] + word[7]) % 2
print('wektor danych - ', p1, p2, p3, p4)
print('Hamming - ', p1, p2, word[0], p3, word[1], word[2], word[3], p4, word[4], word[5],word[6],
word[7])
```

```
print('\n')

print('Zadanie nr.4:')


h_word1_e = (np.dot(G_SECDED.T, word1)).transpose() % 2

print('Wektor nr.1 (SECDED) - ', h_word1_e)

h_word2_e = (np.dot(G_SECDED.T, word2)).transpose() % 2

print('Wektor nr.2 (SECDED) - ', h_word2_e)

Wynik = np.concatenate((h_word1_e,h_word2_e))

print('Hamming74: ',Wynik)


hamm_SECDED = np.dot(hamm_SECDED,h_word1_e.T) % 2

n0 = hamm_SECDED[0] ; n2 = hamm_SECDED[2] ; n1 = hamm_SECDED[1]


if n0 == 0 | n1 == 0 | n2 == 0:

    p1_e = (word[0] + word[1] + word[3] + word[4] + word[6]) % 2

    p2_e = (word[0] + word[2] + word[3] + word[5] + word[6]) % 2

    p3_e = (word[1] + word[2] + word[3] + word[7]) % 2

    p4_e = (word[4] + word[5] + word[6] + word[7]) % 2

    c = (p1_e + p2_e + word[0] + p3_e + word[1] + word[2] + word[3] + p4_e + word[4] + word[5] +
word[6] + word[7]) % 2

    neg_c = np.logical_not(c).astype(int)


print('wektor danych - ', p1_e, p2_e, p3_e, p4_e, neg_c)

print('Hamming - ', p1_e, p2_e, word[0], p3_e, word[1], word[2], word[3], p4_e, word[4],
word[5],word[6], word[7], neg_c)
```

```
Zadanie nr.1:
Slowo zapisane binarnie -  [0, 1, 0, 0, 0, 0, 0, 1]
pierwszy bit -  [0 1 0 0]
drugi bit -  [0 0 0 1]
Wektor nr.1 -  [1 0 0 1 1 0 0]
Wektor nr.2 -  [1 1 0 1 0 0 1]
Hamming74 -  [1 0 0 1 1 0 0 1 1 0 1 0 0 1]


Zadanie nr.2:
Negacja -  [1 0 1 1 1 1 1 0]


Zadanie nr.3:
wektor danych -  1 0 0 1
Hamming -  1 0 0 0 1 0 0 1 0 0 0 1


Zadanie nr.4:
Wektor nr.1 (SECDED) -  [1 0 0 1 1 0 0 1]
Wektor nr.2 (SECDED) -  [1 1 0 1 0 0 1 0]
Hamming74:  [1 0 0 1 1 0 0 1 1 1 0 1 0 0 1 0]
wektor danych -  1 0 0 1 1
Hamming -  1 0 0 0 1 0 0 1 0 0 0 1 1
[Finished in 0.64s]
```