МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ МАШИН СИСТЕМ И СЕТЕЙ

Лабораторная работа №7
по курсу «Методы и средства передачи информации»

Тема: «Порождающая и проверочная матрица»

Выполнил:
Балашов С.А., А-08-19
Проверил:
доц. Оцоков Ш.А.

Москва 2021

Задание 1.

Вариант 2

а) Доказать, что для кода с проверочной матрицей

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

порождающая матрица равна

$$G = \begin{matrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$$

Доказать можно с помощью проверки условия: $G * H^T = 0$

Решение приведено в приложении 1

б) Составить программу, которая выводит все кодовые слова и определяет кодовое расстояние

Табл. Решение задания 1

H	G	H^T	$G*H^T$	Кодовые слова	Кодовое расстояние
[[0 0 0 1 1 1 1] [0 1 1 0 0 1 1] [1 0 1 0 1 0 1]]	[[1 1 1 0 0 0 0] [1 0 0 1 1 0 0] [0 1 0 1 0 1 0] [1 1 0 1 0 0 1]]	[[0 0 1] [0 1 0] [0 1 1] [1 0 0] [1 0 1] [1 1 0] [1 1 1]]	[[0 0 0] [0 0 0] [0 0 0] [0 0 0]]	[[0 0 0 0 0 0 0] [1 1 0 1 0 0 1] [0 1 0 1 0 1 0] [1 0 0 0 0 1 1] [1 0 0 1 1 0 0] [0 1 0 0 1 0 1] [1 1 0 0 1 1 0] [0 0 0 1 1 1 1] [1 1 1 0 0 0 0] [0 0 1 1 0 0 1] [1 0 1 1 0 1 0] [0 1 1 0 0 1 1] [0 1 1 1 1 0 0] [1 0 1 0 1 0 1] [0 0 1 0 1 1 0] [1 1 1 1 1 1 1]]	3

Задание 2.

Показать, что для систематического кода с порождающей матрицей

```
        1 … 0 … 0      p11 … p1m
        0 … 1 … 0      p21 … p2m
G =           .        .        .
        0     0     1      pk1     pkm
```

в качестве проверочной можно взять следующую матрицу:

```
        -p11   -p21   …      -pki    1      0      …      0
        -p12   -p22   …      -pk2    0      1      …      0
H =            ………………………………………………
        -p1m  -p2m  …      -pkm  0      0             1
```

Напишу программу, в которой будут случайно создаваться порождающие матрицы G для систематического кода. Проведу тесты, перемножив несколько таких матриц с проверочными матрицами H заданного типа.

Табл. 2 Тесты для задания 2

G	H	H^T	$G*H^T$	Определитель $G*H^T$
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0.] [0. 1. 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. 0. 1.] [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 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. 0. 0. 1. 0. 1. 0.] [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1.] [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.] [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1.] [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1.]]	[[1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 1. 1. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.] [1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.] [0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.] [0. 1. 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.] [1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.] [1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.] [0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.] [1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.] [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]	[[1. 0. 1. 0. 0. 1. 1. 0. 1. 1.] [0. 1. 0. 1. 1. 0. 1. 1. 1. 1.] [0. 1. 1. 1. 1. 1. 1. 1. 0. 1.] [0. 1. 1. 0. 1. 0. 1. 1. 1. 0.] [1. 1. 0. 1. 1. 1. 0. 0. 0. 0.] [1. 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. 0. 0.] [0. 0. 0. 1. 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. 0. 0. 0. 0. 0. 1. 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. 0. 0. 0. 0. 0. 1.]]	[[1. 0. 1. 0. 0. 1. 0. 1. 0. 1.] [0. 1. 0. 1. 1. 1. 1. 1. 1. 1.] [0. 1. 1. 1. 1. 1. 0. 1. 0. 0.] [0. 1. 1. 0. 1. 1. 1. 1. 0. 0.] [1. 1. 0. 1. 1. 0. 0. 0. 0. 0.] [1. 0. 0. 0. 0. 0. 1. 0. 1. 0.] [0. 1. 0. 0. 0. 0. 0. 0. 0. 1.] [0. 0. 1. 0. 0. 1. 0. 0. 0. 1.] [0. 0. 0. 1. 0. 0. 0. 1. 0. 1.] [0. 0. 0. 0. 1. 0. 0. 0. 1. 1.]]	0
[[1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 1.] [0. 1. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 1.] [0. 0. 1. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0.] [0. 0. 0. 1. 0. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1.] [0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 1. 1. 0. 1. 0.]]	[[1. 0. 0. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 0. 0.] [0. 1. 0. 1. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0.] [0. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0.] [1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0.] [1. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1.]]	[[1. 0. 0. 1. 1.] [0. 1. 0. 0. 0.] [0. 0. 1. 1. 1.] [0. 1. 0. 0. 1.] [1. 1. 1. 0. 0.] [1. 0. 1. 0. 0.] [1. 1. 1. 1. 0.] [1. 1. 0. 0. 1.] [1. 0. 0. 0. 0.] [0. 0. 1. 0. 1.] [1. 0. 0. 0. 0.] [0. 1. 0. 0. 0.] [0. 0. 1. 0. 0.] [0. 0. 0. 1. 0.] [0. 0. 0. 0. 1.]]	[[0. 0. 1. 0. 1.] [1. 0. 1. 1. 0.] [1. 1. 0. 1. 1.] [1. 0. 1. 0. 1.] [1. 1. 1. 0. 1.]]	0

[[1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0.] [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1.] [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0.] [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0.] [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.] [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1.] [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 1.] [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 1. 0. 0. 0.] [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 0.]]	[[0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.] [1. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.] [0. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.] [0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.] [0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.] [0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.] [0. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.] [1. 1. 0. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.] [0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]	[[1. 0. 0. 0. 1. 0. 1. 1. 0. 0.] [0. 1. 0. 1. 1. 1. 0. 1. 0. 0.] [0. 0. 1. 0. 1. 1. 1. 0. 0. 0.] [0. 1. 0. 1. 0. 1. 1. 0. 1. 1.] [1. 1. 1. 0. 1. 0. 0. 0. 1. 0.] [0. 1. 1. 1. 0. 1. 0. 1. 0. 0.] [1. 0. 1. 1. 0. 0. 1. 0. 0. 1.] [1. 1. 0. 0. 0. 1. 0. 1. 0. 0.] [0. 0. 0. 1. 1. 0. 0. 0. 1. 1.] [0. 0. 0. 1. 0. 0. 1. 0. 1. 1.]]	0
[[1. 0. 0. 0. 0. 1. 1. 1. 0. 1. 0. 0. 1.] [0. 1. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0.] [0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 1. 1.]]	[[1. 1. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0.] [0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0.] [1. 1. 0. 1. 1. 1. 0. 1. 0. 0. 0. 0. 1.]]	[[1. 0. 1.] [1. 1. 1.] [0. 1. 0.] [0. 0. 1.] [0. 0. 1.] [1. 0. 1.] [0. 1. 0.] [1. 1. 1.] [1. 0. 0.] [0. 1. 0.] [1. 0. 0.] [0. 1. 0.] [0. 0. 1.]]	[[1. 1. 0.] [1. 0. 1.] [1. 1. 0.]]	0

Приложение 1. Код программы для решения заданий 1 и 2.

```
import numpy as np

def first():
    h = np.array([[0, 0, 0, 1, 1, 1, 1], [0, 1, 1, 0, 0, 1, 1], [1, 0, 1, 0, 1, 0, 1]])
    print('h = \n', h)
    g = np.array([[1, 1, 1, 0, 0, 0, 0], [1, 0, 0, 1, 1, 0, 0], [0, 1, 0, 1, 0, 1, 0], [1, 1, 0,
1, 0, 0, 1]])
    print('g = \n', g)
    ht = h.transpose()
    print('Transposed h = \n', ht)
    result = g.dot(ht) % 2
    print('g*ht = \n', result)

    keys = np.array([[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]], bool)

    codeWords = keys.dot(g)
    print('Code words:\n', codeWords % 2)

    def hammingDistance(a, b):
        distance = 0
        for i in range(len(a)):
            if a[i] != b[i]:
                distance += 1
        return distance

    def minHammingDistance(code):
        minHammingDistance = len(code[0])
        for a in code:
            for b in code:
                comparison = a == b
                equal = comparison.all()
                if not equal:
                    tmp = hammingDistance(a, b)
                    if tmp < minHammingDistance:
                        minHammingDistance = tmp
        return minHammingDistance

    print("min Hamming distance: ", minHammingDistance(codeWords))


def second():
    num_rows = np.random.randint(2, 11)
    num_cols = np.random.randint(2, 11)
    q = np.random.randint(0, 2, (num_rows, num_cols))
    i = np.eye(num_rows)
    g = np.hstack((i, q))
    h = np.hstack((((q + 1) % 2), i))
    ht = h.transpose()
    check = g.dot(ht) % 2
    print('g = \n', g)
    print('h = \n', h)
    print('Transposed h = \n', ht)
    print('g*ht = \n', check)
    print('det(g*ht) = ', (int)(np.linalg.det(check) % 2))

first()
second()
```