密碼工程 Quiz4

學號:112550090 姓名:曾士珍

Problem1

a) Is $x^8 + x^4 + x^3 + x^2 + 1$ a primitive polynomial?

能在 GF(2)可以生成最大週期序列的多項式為 primitive polynomial

用以下程式碼來實作該多項式的 LFSR：

```python
arr = [0, 0, 0, 0, 0, 0, 0, 1]
for i in range(1, 257):
    print(f"{i}: {arr}")
    chk = arr[0]
    for j in range(7):
        arr[j] = arr[j+1]
    arr[7] = 0
    if chk == 1:
        arr[3] ^= 1
        arr[4] ^= 1
        arr[5] ^= 1
        arr[7] ^= 1
```

```
1: [0, 0, 0, 0, 0, 0, 0, 1]
2: [0, 0, 0, 0, 0, 0, 1, 0]
3: [0, 0, 0, 0, 0, 1, 0, 0]
4: [0, 0, 0, 0, 1, 0, 0, 0]
5: [0, 0, 0, 1, 0, 0, 0, 0]
6: [0, 0, 1, 0, 0, 0, 0, 0]
7: [0, 1, 0, 0, 0, 0, 0, 0]
8: [1, 0, 0, 0, 0, 0, 0, 0]
9: [0, 0, 0, 1, 1, 1, 0, 1]
10: [0, 0, 1, 1, 1, 0, 1, 0]
11: [0, 1, 1, 1, 0, 1, 0, 0]
12: [1, 1, 1, 0, 1, 0, 0, 0]
13: [1, 1, 0, 0, 1, 1, 0, 1]
14: [1, 0, 0, 0, 0, 1, 1, 1]
15: [0, 0, 0, 1, 0, 0, 1, 1]
16: [0, 0, 1, 0, 0, 1, 1, 0]
17: [0, 1, 0, 0, 1, 1, 0, 0]
```

```
240: [0, 0, 0, 1, 0, 1, 1, 0]
241: [0, 0, 1, 0, 1, 1, 0, 0]
242: [0, 1, 0, 1, 1, 0, 0, 0]
243: [1, 0, 1, 1, 0, 0, 0, 0]
244: [0, 1, 1, 1, 1, 1, 0, 1]
245: [1, 1, 1, 1, 1, 0, 1, 0]
246: [1, 1, 1, 0, 1, 0, 0, 1]
247: [1, 1, 0, 0, 1, 1, 1, 1]
248: [1, 0, 0, 0, 0, 0, 1, 1]
249: [0, 0, 0, 1, 1, 0, 1, 1]
250: [0, 0, 1, 1, 0, 1, 1, 0]
251: [0, 1, 1, 0, 1, 1, 0, 0]
252: [1, 1, 0, 1, 1, 0, 0, 0]
253: [1, 0, 1, 0, 1, 1, 0, 1]
254: [0, 1, 0, 0, 0, 1, 1, 1]
255: [1, 0, 0, 0, 1, 1, 1, 0]
256: [0, 0, 0, 0, 0, 0, 0, 1]
```

由結果可以看出週期為 255(為 8 個位元時可以產生的最大週期數)，在第 256 組時回到跟第 1 組一樣的狀況

由執行結果可以得出$x^8 + x^4 + x^3 + x^2 + 1$為 primitive polynomial

b) What is the maximum cycle length generated by $x^8 + x^4 + x^3 + x^2 + 1$?

多項式的最高次方項係數是 8(代表 LFSR 的寄存器有 8 個位元)且是 primitive polynomial，因此可以生成的序列的最大週期長度為$2^8 - 1 = 255$

c) Are all irreducible polynomials primitive polynomials?

不是，所有的 primitive polynomials 都是不可約的，但反之不一定成立。

反例：$x^5 + x + 1$不可被因式分解

```python
arr = [0, 0, 0, 0, 1]
for i in range(1, 32):
    print(f'{i}: {arr}')
    chk = arr[0]
    for j in range(4):
        arr[j] = arr[j+1]
    arr[4] = 0
    if chk == 1:
        arr[3] ^= 1
        arr[4] ^= 1
```

```
1: [0, 0, 0, 0, 1]
2: [0, 0, 0, 1, 0]
3: [0, 0, 1, 0, 0]
4: [0, 1, 0, 0, 0]
5: [1, 0, 0, 0, 0]
6: [0, 0, 0, 1, 1]
7: [0, 0, 1, 1, 0]
8: [0, 1, 1, 0, 0]
9: [1, 1, 0, 0, 0]
10: [1, 0, 0, 1, 1]
11: [0, 0, 1, 0, 1]
12: [0, 1, 0, 1, 0]
13: [1, 0, 1, 0, 0]
14: [0, 1, 0, 1, 1]
15: [1, 0, 1, 1, 0]
16: [0, 1, 1, 1, 1]
17: [1, 1, 1, 1, 0]
18: [1, 1, 1, 1, 1]
19: [1, 1, 1, 0, 1]
20: [1, 1, 0, 0, 1]
21: [1, 0, 0, 0, 1]
22: [0, 0, 0, 0, 1]
```

LFSR 過程程式碼	週期僅為 21

5 個位元時可以產生的最大週期數應為$2^5 - 1 = 31$，故 irreducible polynomials 不一定是 primitive polynomials

Problem2

a) Please use $x^8 + x^4 + x^3 + x^2 + 1$ as a characteristic polynomial to write a Python program to encrypt the following plaintext message with the initial key 00000001, then decrypt it to see if your encryption is correct.

Encrypted text:

0100000001010110010010100101000101010011011101010001011111000101010111000110100000110001101110111001100111010101010110100111000000000010111010110011010100000111011111011001101111010111110001000110010000010100010000110100110101001100011001010010111010001001110010110110001000011100110011111011011000001111011001101011000010010001010010101111

0001001000111011110010000000110100010111000000000101110010010100101101001110010001110110110000011001111100110001111011110011001110011001
1110111100000010000110011111111100110000011011111010001010001001010011010000011010110001101111101111001100011000010111011010010000000101
0010011010100011110110100100101101011010001010011010111101010010100010000100111101000011111010101101101010001001000111100100000101111110
1111001100101010110110010100111111110000111111000011010010010111001000010110110001110100111101010110011100010010010110000011111010010010
1111100011111101010101001011011101100011100110101111101000101001010111011111011111110011000010101101100001101101001001100000001100011000
1001110100100101111001100010000100001111101100100010100101001111110100000110110100000111010000000111110100000101101001101000000100011110
1111000110001101001101101101000101111110000000010111111100110001110111011010011111000011101111010000010010000111101100010011110101110100100001110101100110100100000111101001001111110001100001011110110100110
1001111111111010001111101010000010110100101011001010110010010110111011100001111111011110011110000010110010000011110110000110001000101101
1001010011100010000000101111010110000000011010110010111000110000000110000100001111100101001000010010110100101010001111101001000101011010010010101111001100001110111110001000101001110000100011000111111000011
0000010111100101111011010011010010101110100100110111100000011111011010000100111000110010100000101001101100010000111001101010101101111001110111011100010111001101000111011101110111010101111111001110011001010
0101100001000010011001000001110111111001001110011011111110101000100110111110010110101101001100010001000001001110111101001000100111100000101010101010010100100000101001011010000100110010100000001110101000101
0100011101010011101010100111100010111100100001000110011101000001100111010110011001000000110011110001001001111011100110000000110110110101
1010010011110100001001001100011110000010010110000110110101000111010100011101110010000111010100000110110101010111011000101111

Decrypted text:

ATNYCUWEARESTRIVINGTOBEAGREATUNIVERSITYTHATTRANSCENDSDISCIPLINARYDIVIDESTOSOLVETHEINCREASINGLYCOMPLEXPROBLEMSTHATTHEWORLDFACESWEWILLCONTINUETOBEGUIDEDBYTHEIDEATHATWECANACHIEVESOMETHINGMUCHGREATERTOGETHERTHANWECANINDIVIDUALLYAFTERALLTHATWASTHEIDEATHATLEDTOTHECREATIONOFOURUNIVERSITYINTHEFIRSTPLACE

b) Due to the property of ASCII coding the ASCII A to Z, the MSB of each byte will be zero (left most bit); therefore, every 8 bits will reveal 1 bit of random number (i.e.

keystream); if it is possible to find out the characteristic polynomial of a system by solving of linear equations?

可以，以 4-stage 的 LFSR，我們可以列出以下等式：

$$
\begin{aligned}
a_n &= (a_{n+1}C_3 + a_{n+2}C_2 + a_{n+3}C_1 + a_{n+4}C_0) mod\ 2 \\
a_{n+1} &= (a_{n+2}C_3 + a_{n+3}C_2 + a_{n+4}C_1 + a_{n+5}C_0)\ mod\ 2 \\
a_{n+2} &= (a_{n+3}C_3 + a_{n+4}C_2 + a_{n+5}C_1 + a_{n+6}C_0)\ mod\ 2 \\
a_{n+3} &= (a_{n+4}C_3 + a_{n+5}C_2 + a_{n+6}C_1 + a_{n+7}C_0)\ mod\ 2
\end{aligned}
$$

若可以知道一組$a_n, a_{n+1} \ldots, a_{n+7}$的值，便可以計算出$C_0, C_1, C_2, C_3$的值

更一般化的來說，若能知到 2n 個 output bits 的值，就可以解出 n-stage 的 LFSR

c) **Extra credit:** Write a linear equations program solving program to find the charac teristic polynomial for this encryption with initial 00000001.

枚舉所有可能的$C_0, C_1, C_2 \ldots C_7$，再將已知的$a_0, a_1 \ldots, a_{15}$代入 b)的方程組確認，程式碼如下圖：

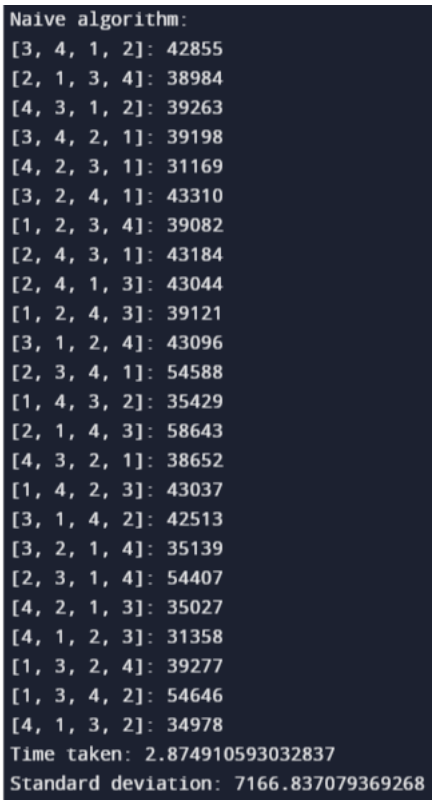
程式碼	output

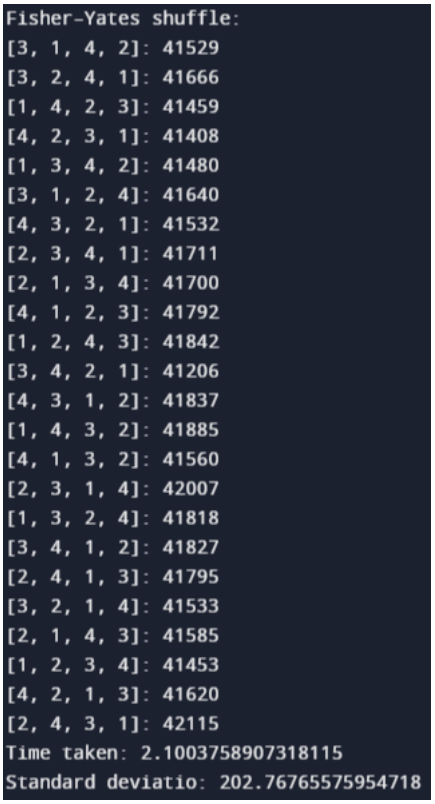
```python
a = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0]
for i in range(2 ** 8):
    rec = 0
    s = format(i, f'0{8}b')
    for j in range(8):
        now = 0
        for k in range(8):
            now += a[j+1+k] * int(s[7-k])
            now %= 2
        if a[j] != now:
            rec = 1
            break
    if rec == 0:
        for i in range (len(s)):
            print(f'C{i}:{s[i]}')
        break
```

```
C0:1
C1:0
C2:0
C3:0
C4:1
C5:1
C6:1
C7:0
```

Problem3

a) Please write a Python program to simulate two algorithms with a set of 4 cards, shuffling each a million times. Collect the count of all combinations and output.





b) Based on your analysis, which one is better, why?

Fisher-Yates Shuffle Algorithm 較好。

1. 由 standard deviation 可以看出，其相較於 Naïve Shuffle Algorithm 有更均勻分布的洗牌结果。
2. 由上圖程式跑 1000000 次的執行時間可以看出，其相較於 Naïve Shuffle Algorithm 更有執行效率較高。

c) What are the drawbacks of the other one, and what causes these drawbacks?

Naïve Shuffle Algorithm 在選擇位置時沒有進行有效的隨機化，僅僅是在每次迭代中隨機選擇一個位置，而未考慮已經選擇過的位置或者已經移動過的元素，這可能導致某些元素被過度集中或遺漏，從而影響洗牌結果的均勻性和隨機性。