2.18 Consider the following linear recurrence over $\mathbb{Z}_2$ of degree four:

$$z_{i+4} = (z_i + z_{i+1} + z_{i+2} + z_{i+3}) \bmod 2,$$

$i \geq 0$. For each of the 16 possible initialization vectors $(z_0, z_1, z_2, z_3) \in (\mathbb{Z}_2)^4$, determine the period of the resulting keystream.

Answer:

```python
from itertools import product

# 定义 LFSR 函数
def lfsr(initial_state):
state = initial_state[:] # 复制初始状态，避免修改原始输入
seen_states = {tuple(state)} # 使用集合存储已见状态
count = 0

while True:
# 应用线性递归关系
new_bit = (state[0] + state[1] + state[2] + state[3]) % 2
state = state[1:] + [new_bit] # 移位并添加新状态
count += 1
# 如果新状态已经见过，结束循环
if tuple(state) in seen_states:
break
seen_states.add(tuple(state))
return count

# 遍历所有可能的初始向量（IVs），因为 Z2 的范围是 [0, 1]
ivs = list(product(range(2), repeat=4))

# 计算每个 IV 的周期
periods = {iv: lfsr(list(iv)) for iv in ivs}

# 打印周期
for iv, period in periods.items():
print(f"IV: {iv}, Period: {period}")
```

```
● (base) wangyidan@wangyidandeMacBook-Pro 密码学 % /usr/local/bin/python3 "/Users/wangyidan/Deskto
  p/密码学/hw1/HW1(2.13).py"
  IV: (0, 0, 0, 0), Period: 1
  IV: (0, 0, 0, 1), Period: 5
  IV: (0, 0, 1, 0), Period: 5
  IV: (0, 0, 1, 1), Period: 5
  IV: (0, 1, 0, 0), Period: 5
  IV: (0, 1, 0, 1), Period: 5
  IV: (0, 1, 1, 0), Period: 5
  IV: (0, 1, 1, 1), Period: 5
  IV: (1, 0, 0, 0), Period: 5
  IV: (1, 0, 0, 1), Period: 5
  IV: (1, 0, 1, 0), Period: 5
  IV: (1, 0, 1, 1), Period: 5
  IV: (1, 1, 0, 0), Period: 5
  IV: (1, 1, 0, 1), Period: 5
  IV: (1, 1, 1, 0), Period: 5
  IV: (1, 1, 1, 1), Period: 5
```

2.23 Suppose we are told that the plaintext

breathtaking

yields the ciphertext

RUPOTENTOIFV

where the *Hill Cipher* is used (but m is not specified). Determine the encryption matrix.

Answer:

```
from sympy import Matrix, mod_inverse

# 将字母转换为数字
def letters_to_numbers(letters):
return [ord(letter) - ord('A') for letter in letters.upper()]

# 将文本转换为数字，假设它已经是大写且没有空格或非字母字符
def text_to_numeric(text):
return [letters_to_numbers(text[i:i+3]) for i in range(0,
len(text), 3)]

# 使用明文和密文求解加密矩阵
def solve_hill_cipher(plaintext, ciphertext):
# 将明文和密文分割成大小为 3 的块，并转换为数字
plaintext_blocks = text_to_numeric(plaintext)
ciphertext_blocks = text_to_numeric(ciphertext)

# 使用第一个块来确定加密矩阵
P = Matrix(plaintext_blocks[0:3])
C = Matrix(ciphertext_blocks[0:3])

# 求加密矩阵 A，使 P * A = C mod 26
# P 求模 26 的倒数解 A
try:
P_inv = P.inv_mod(26)
except ValueError as e:
return str(e), None # If the inverse doesn't exist, return
the error message

A = P_inv * C % 26
return None, A

# 给定明文和密文
plaintext = "breathtaking"
ciphertext = "RUPOTENTOIFV"
```

```
# 假设 n = 3，求解加密矩阵
error, encryption_matrix = solve_hill_cipher(plaintext,
ciphertext)

# 检查矩阵是否找到并打印出来
if encryption_matrix:
# Format and print the matrix
matrix_as_list = encryption_matrix.tolist()
formatted_matrix = '\n'.join(['\t'.join(map(str, row)) for
row in matrix_as_list])
print("Encryption matrix:\n", formatted_matrix)
else:
# If there was an error, print it
print("Error:", error)
```

因为明文有 12 个字符所以加密矩阵的类型可以是 2*2,3*3,4*4 和 6*6 在代码中我尝试使用 3*3 的加密矩阵并输出了结果。

```
(base) wangyidan@wangyidandeMacBook-Pro 密码学 % /usr/local/bin/python3 "/Users/wangyidan/Desktop/
密码学/hw1/HW1(2.23).py"
Encryption matrix:
 3      21      20
4       15      23
6       14      5
```

2.30 We describe another stream cipher, which incorporates one of the ideas from the *Enigma* machime used by Germany in World War II. Suppose that π is a fixed permutation of $\mathbb{Z}_{26}$. The key is an element $K \in \mathbb{Z}_{26}$. For all integers $i \geq 1$, the keystream element $z_i \in \mathbb{Z}_{26}$ is defined according to the rule $z_i = (K + i - 1) \bmod 26$. Encryption and decryption are performed using the permutations π and π^{-1}, respectively, as follows:

$$e_z(x) = \pi(x) + z \bmod 26$$

and

$$d_z(y) = \pi^{-1}(y - z \bmod 26),$$

where $z \in \mathbb{Z}_{26}$.

Suppose that π is the following permutation of $\mathbb{Z}_{26}$:

x	0	1	2	3	4	5	6	7	8	9	10	11	12
$\pi(x)$	23	13	24	0	7	15	14	6	25	16	22	1	19

x	13	14	15	16	17	18	19	20	21	22	23	24	25
$\pi(x)$	18	5	11	17	2	21	12	20	4	10	9	3	8

The following ciphertext has been encrypted using this stream cipher; use exhaustive key search to decrypt it:

```
WRTCNRLDSAFARWKXFTXCZRNHNYPDTZUUKMPLUSOXNEUDO
KLXRMCBKGRCCURR
```

Answer:

```python
def decrypt(ciphertext, K, pi):
# 反置换字典
pi_inverse = {v: k for k, v in pi.items()}
# 解密函数
def dz(y, z):
return pi_inverse[(y - z) % 26]
plaintext = ''
for i, c in enumerate(ciphertext):
z = (K + i) % 26
y = ord(c) - ord('A')
x = dz(y, z)
plaintext += chr(x + ord('A'))
return plaintext
# 置换π
pi = {0: 23, 1: 13, 2: 24, 3: 0, 4: 7, 5: 15, 6: 14, 7: 6,
8: 25, 9: 16, 10: 22, 11: 1, 12: 19,
13: 18, 14: 5, 15: 11, 16: 17, 17: 2, 18: 21, 19: 12, 20: 20,
21: 4, 22: 10, 23: 9, 24: 3, 25: 8}
ciphertext =
"WRTCNRLDSAFARWKXFTXCZRNHNYPDTZUUKMPLUSOXNEUDOKLXRMCBKGRC
CURR"
# 尝试每个可能的密钥 K
for K in range(26):
plaintext = decrypt(ciphertext, K, pi)
print(f"K={K}: {plaintext}")
```

```
(base) wangyidan@wangyidandeMacBook-Pro 密码学 % /usr/local/bin/python3 "/Users/wangyidan/Desktop/
密码学/hw1/HW1(2.30).py"
K=0: KJQIXTOKWQSFOXKZFROXOKQWFIFRQKJFVOERWERWIFVTKQQRSFVRWOFICFPW
K=1: SFJCZPVSXJUGVZSEGLVZVSJXGCGLJSFGYVHLXHLXCGYPSJJLUGYLXVGCAGWX
K=2: UGFAEWYUZFMBYEUHBDYEYUFZBABDFUGBRYODZODZABRWUFFDMBRDZYBAKBXZ
K=3: MBGKHXRMEGNTRHMOTIRHRMGETKTIGMBTLRVIEVIEKTLXMGGINTLIERTKSTZE
K=4: NTBSOZLNHBQPLONVPCLOLNBHPSPCBNTPDLYCHYCHSPDZNBBCQPDCHLPSUPEH
K=5: QPTUVEDQOTJWDVQYWADVDQTOWUWATQPWIDRAORAOUWIEQTTAJWIAODWUMWHO
K=6: JWPMYHIJVPFXIYJRXKIYIJPVXMXKPJWXCILKVLKVMXCHJPPKFXCKVIXMNXOV
K=7: FXWNROCFYWGZCRFLZSCRCFWYZNZSWFXZACDSYDSYNZAOFWWSGZASYCZNQZVY
K=8: GZXQLVAGRXBEALGDEUALAGXREQEUXGZEKAIURIURQEKVGXXUBEKURAEQJEYR
K=9: BEZJDYKBLZTHKDBIHMKDKBZLHJHMZBEHSKCMLCMLJHSYBZZMTHSMLKHJFHRL
K=10: THEFIRSTDEPOSITCONSISTEDOFONETHOUSANDANDFOURTEENPOUNDSOFGOLD
K=11: POHGCLUPIHWVUCPAVQUCUPHIVGVQHPOVMUKQIKQIGVMLPHHQWVMQIUVGBVDI
K=12: WVOBADMWCOXYMAWKYJMAMWOCYBYJOWVYNMSJCSJCBYNDWOOJXYNJCMYBTYIC
K=13: XYVTKINXAVZRNKXSRFNKNXVARTRFVXYRQNUFAUFATRQIXVVFZRQFANRTPRCA
K=14: ZRYPSCQZKYELQSZULGQSQZYKLPLGYZRLJQMGKMGKPLJCZYYGELJGKQLPWLAK
K=15: ELRWUAJESRHDJUEMDBJUJERSDWDBRELDFJNBSNBSWDFAERRBHDFBSJDWXDKS
K=16: HDLXMKFHULOIFMHNITFMFHLUIXITLHDIGFQTUQTUXIGKHLLTOIGTUFIXZISU
K=17: OIDZNSGOMDVCGNOQCPGNGODMCZCPDOICBGJPMJPMZCBSODDPVCBPMGCZECUM
K=18: VCIEQUBVNIYABQVJAWBQBVINAEAWIVCATBFWNFWNEATUVIIWYATWNBAEHAMN
K=19: YACHJMTYQCRKTJYFKXTJTYCQKHKXCYAKPTGXQGXQHKPMYCCXRKPXQTKHOKNQ
K=20: RKAOFNPRJALSPFRGSZPFPRAJSOSZARKSWPBZJBZJOSWNRAAZLSWZJPSOVSQJ
K=21: LSKVGQWLFKDUWGLBUEWGWLKFUVUEKLSUXWTEFTEFVUXQLKKEDUXEFWUVYUJF
K=22: DUSYBJXDGSIMXBDTMHXBXDSGMYMHSDUMZXPHGPHGYMZJDSSHIMZHGXMYRMFG
K=23: IMURTFZIBUCNZTIPNOZTZIUBNRNOUIMNEZWOBWOBRNEFIUUOCNEOBZNRLNGB
K=24: CNMLPGECTMAQEPCWQVEPECMTQLQVMCNQHEXVTXVTLQHGCMMVAQHVTEQLDQBT
K=25: AQNDWBHAPNKJHWAXJYHWHANPJDJYNAQJOHZYPZYPDJOBANNYKJOYPHJDIJTP
```