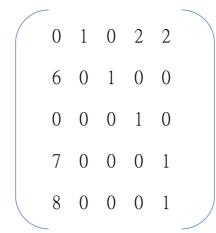
107062208 邱靖豪 Assignment 1

Q1:



My Hill cipher matrix(A)

如何求 A?

我的學號是 107062208, 我的想法是利用 first row 或 first column 來做 cofactor expansion,我們可以找出學號中,相差數為 1 的數,把他們放在 first row 或 first column,且緊鄰的位置,接著,再讓他們能分出來的 M,對角線都是 1,接著其他沒用到的位置都補上 0,這樣根據我們求 det 的方法,就能確定得出的行列式值一定是 1 或-1。以我的學號做說明,我挑出 7 跟 8,並把他們放在 a[4][1]跟 a[5][1]的位置,所以,我要讓 a[1][1]*A[1][1], a[2][1]*A[2][1], a[3][1]*A[3][1]都變成 0,然後讓 7,8 都乘以一個單位矩陣,就可以得 出 det(A)為 1 的矩陣。

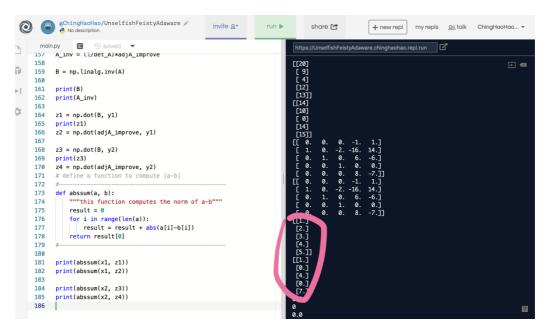
如何求 A-1?

```
M41 = np.array(
                                         M52 = np.array(
       [[1, 0, 2, 2],
[0, 1, 0, 0],
[0, 0, 1, 0],
[0, 0, 0, 1]])
                                                 [[0, 0, 2, 2], [6, 1, 0, 0],
                                                  [0, 0, 1, 0],
[7, 0, 0, 1]])
M51 = np.array(
                                         M23 = np.array(
       [[1, 0, 2, 2], [0, 1, 0, 0],
                                                 [[0, 1, 2, 2], [0, 0, 1, 0],
        [0, 0, 1, 0],
[0, 0, 0, 1]])
                                                  [7, 0, 0, 1],
[8, 0, 0, 1]])
M12 = np.array(
                                         M43 = np.array(
       [[6, 1, 0, 0],
                                                 [[0, 1, 2, 2],
        [0, 0, 1, 0],
[7, 0, 0, 1],
[8, 0, 0, 1]])
                                                  [6, 0, 0, 0],
                                                  [0, 0, 1, 0],
[8, 0, 0, 1]])
                                                                                 M45 = np.array(
M32 = np.array(
                                         M53 = np.array(
                                                                                        [[0, 1, 0, 2],
[6, 0, 1, 0],
[0, 0, 0, 1],
[8, 0, 0, 1]])
       [[0, 0, 2, 2],
                                                 [[0, 1, 2, 2],
        [6, 1, 0, 0],
[7, 0, 0, 1],
[8, 0, 0, 1]])
                                                  [6, 0, 0, 0],
[0, 0, 1, 0],
[7, 0, 0, 1]])
                                                                                 M55 = np.array(
M42 = np.array(
                                         M34 = np.array(
       [[0, 0, 2, 2],
[6, 1, 0, 0],
[0, 0, 1, 0],
                                                 [[0, 1, 0, 2],
[6, 0, 1, 0],
[7, 0, 0, 1],
[8, 0, 0, 1]])
                                                                                         [[0, 1, 0, 2],
                                                                                          [6, 0, 1, 0],
[0, 0, 0, 1],
         [8, 0, 0, 1]])
                                                                                           [7, 0, 0, 1]])
```

我覺得我的想法非常直觀,我是利用講義中給的求 adjoint A 公式,先 找出 cofactor Aij,而要找出 cofactor Aij,要先求出 det(Mij),Mij 的求 法我是直接去從我原本的矩陣去消去而來,接著利用教授提供的 function,用遞迴找出 det。在求 M 時,我已經先考慮到之後在進行求 A-1 時,會變成 0 項的元素了,因此就不列舉出來。接著就可以算 det(A) 的值。然後,我根據 cofactor Aij 的公式讓 Aij 依序被算出,就可以算 出 adjoint A。

```
det_A = (A[3][0]*(((-1)**(4+1)) * mydet(M41))) + (A[4][0]*(((-1)**(5+1)) * mydet(M51)))
                                                  mydet(M41)
                    ((-1)**(4+1))
((-1)**(5+1))
((-1)**(1+2))
((-1)**(3+2))
((-1)**(4+2))
((-1)**(5+2))
((-1)**(2+3))
((-1)**(4+3))
((-1)**(5+3))
((-1)**(4+5))
((-1)**(5+5))
         A51
                                                  mydet (M51)
         A12
                                                  mydet (M12)
         A32
                                                  mydet (M32)
         A42
A52
A23
A43
                                                  mydet (M42)
                                                  mydet (M52)
                                                  mydet (M23)
                                                  mydet(M43)
         A53
                                                  mydet(M53)
         A34
                                                  mydet(M34)
138
139
         A45
                                                  mydet(M45)
         A55
                                                  mydet(M55)
140
141
        adjA_improve = np.array(
    [[0, 0, 0, A41, A51],
    [A12, 0, A32, A42, A52],
    [0, A23, 0, A43, A53],
    [0, 0, A34, 0, 0],
    [0, 0, 0, A45, A55]])
        A_{inv} = (1/det_A)*adjA_{improve}
         B = np.linalg.inv(A)
        print(B)
print(A)
```

還有經過確認後,正確的加密以及解密結果:



Q3:

Compare original and the decoded message

我們可以發現,兩個做法都以 x=[1,2,3,4,5]T 去做 encode 和 decode, 出來的解果都是對的,在最後 abssum 的地方,用

np.linalg.inv(A) 也沒有出現誤差。但是我發現其他同學都有得出誤差值,於是我也去探究原因,原因是因為電腦是用[AII]這種高斯消去法去做 inverse, 在做 row operations 時 ,舉例若同乘1/3,電腦會當作乘0.33333333...,所以在最後才會產生誤差,而我們用 adjoint 這題都是整數在做運算,因此不會有這些誤差。