MSSV: 23520315

Họ và tên: Tào Minh Đức

Link code của các bài dưới: https://github.com/Duck4705/Crypto.git

BÁO CÁO BÀI TẬP TUẦN 1, 2

Bài 1: Cryptanalysis Affine cipher (brute force a,b). Provise your python code

- Phần code:

```python
def affine_decrypt(text, a, b):
    """
    Decrypts the input text using the Affine cipher with keys a and b.
    Computes the modular inverse of a and applies the decryption formula:
      D(y) = a_inv * (y - b) mod 26.
    """
    a_inv = mod_inverse(a, m: 26)
    if a_inv is None:
        raise ValueError(f"No modular inverse for a = {a}. Please choose a value of 'a' that is coprime with 26.")

    alphabets = string.ascii_uppercase
    result = []
    for char in text:
        if char.isupper():
            y = ord(char) - ord('A')
            x = (a_inv * (y - b)) % 26
            result.append(alphabets[x])
        elif char.islower():
            y = ord(char.upper()) - ord('A')
            x = (a_inv * (y - b)) % 26
            result.append(alphabets[x].lower())
        else:
            result.append(char)
    return ''.join(result)

1 usage
```

```
#Đoạn code này dùng để brute force mà khi không biết trước key với điều kiện biết vùng giới hạn của a và b
#Giới hạn của a là số không phải ước chung với 26 nằm trong đoạn từ [1,26]
#Giới hạn của b là số nằm trong đoạn [1,25]
#Tổng số vòng loop tối đa là 300 lần nếu sử dụng brute force
n = 0 # Dùng để đếm số lần thứ n brute force
for a_term in range(1, 26):
    for b_term in range(1, 25):
        #Dòng này kiểm tra xem a có phải là ước chung của 26 không nếu có bỏ qua không thực hiện gì cả
        if mod_inverse(a_term, m: 26) == None:
            continue
        n+=1
        #Dòng dưới đây là giải mã với a, b được chọn ra trong đoạn nhắc ở trên
        input(f"\nPress Enter to continue to brute force with a = {a_term}, b = {b_term} n = {n}...")
        output = affine_decrypt(encrypted_text, a_term, b_term)
        print("\nDecrypted text:")
        print(output)
```

- Giải thích:

Đoạn code trên brute force key a và b để tìm ra được được đoạn văn bản trước khi mã hóa

Điều kiện a không phải là ước chung của 26 và a nằm trong khoảng [1,26]

B nằm trong khoảng [1,25]

Cụ thể:

a = {1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}

b = [1,25]

với số trường hợp a và b như vậy thì cần brute force tối đa khoảng 12 * 25 = 300 lần

Ví dụ: khi mã hóa với a = 5, b = 7

Chuỗi sau khi được mã hóa là :

```
Enter the text to encrypt: Putin dang noi chuyen voi ong Trump ne Zelensky oi

Encrypted text:
Edyvu whul uzv rqdxbu izv zul Yodpe ub Cbkbutfx zv
```

Với trường hợp a = 5 và b = 7 phải brute force mất tận lần thứ 55 mới ra kết quả

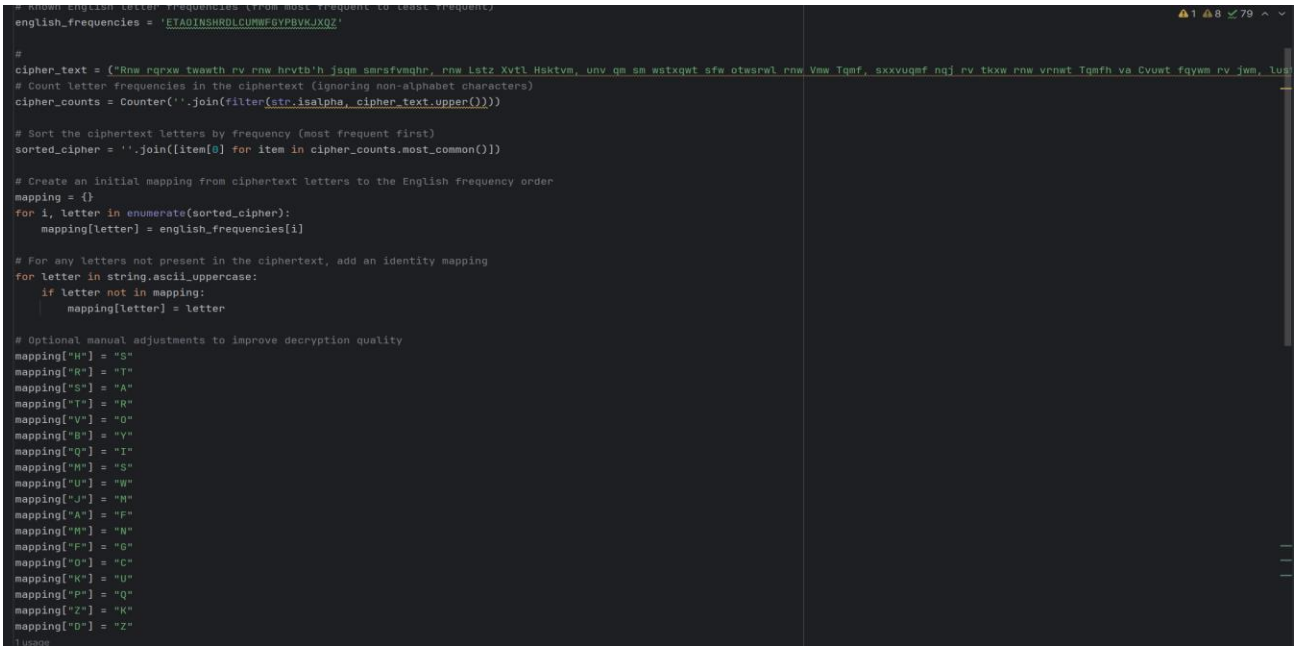
```
Press Enter to continue to brute force with a = 5, b = 5 n = 53...

Decrypted text:
Fkjyd tqdw dey sxkoud ley edw Jhkcf du Pubudiao ey

Press Enter to continue to brute force with a = 5, b = 6 n = 54...

Decrypted text:
Kpodi yvib ijd xcptzi qjd jib Omphk iz Uzgzinft jd

Press Enter to continue to brute force with a = 5, b = 7 n = 55...

Decrypted text:
Putin dang noi chuyen voi ong Trump ne Zelensky oi
```

Những trường hợp a lớn hơn có thể mất nhiều lần brute force hơn

Bài 2: Cryptanalysis SimpleSubstitutionCipher

- Phần code:

Phần code dựa vào mẫu code đã cho sẵn việc cần làm là xử lý ánh xạ 1 : 1 dựa vào tần suất xuất hiện các từ phổ biến được thống kê và tần suất xuất hiện của các từ trong encrypted text, sau đó khi chỉnh sửa ánh xạ



```
# Known English letter frequencies (from most frequent to least frequent)
english_frequencies = 'ETAOINSHRDLCUMWFGYPBVKJXQZ'

#
cipher_text = {"Rnw rqrxw twawth rv rnw hrvtb'h jsqm smrsfvmqhr, rnw Lstz Xvtl Hsktvm, unv qm sm wstxqwt sfw otwsrwl rnw Vmw Tqmf, sxxvuqmf nqj rv tkxw rnw vrnwt Tqmfh va Cvuwt fqywm rv jwm, lus
# Count letter frequencies in the ciphertext (ignoring non-alphabet characters)
cipher_counts = Counter(''.join(filter(str.isalpha, cipher_text.upper())))

# Sort the ciphertext letters by frequency (most frequent first)
sorted_cipher = ''.join([item[0] for item in cipher_counts.most_common()])

# Create an initial mapping from ciphertext letters to the English frequency order
mapping = {}
for i, letter in enumerate(sorted_cipher):
    mapping[letter] = english_frequencies[i]

# For any letters not present in the ciphertext, add an identity mapping
for letter in string.ascii_uppercase:
    if letter not in mapping:
        mapping[letter] = letter

# Optional manual adjustments to improve decryption quality
mapping["H"] = "S"
mapping["R"] = "T"
mapping["S"] = "A"
mapping["T"] = "R"
mapping["V"] = "O"
mapping["B"] = "Y"
mapping["Q"] = "I"
mapping["M"] = "S"
mapping["U"] = "W"
mapping["J"] = "M"
mapping["A"] = "F"
mapping["M"] = "N"
mapping["F"] = "G"
mapping["O"] = "C"
mapping["K"] = "U"
mapping["P"] = "Q"
mapping["Z"] = "K"
mapping["D"] = "Z"
```

Đây là một phần đoạn văn đã được mã hóa và phần ánh xạ lại sau khi thực hiện ánh xạ theo tần suất xuất hiện nhiều nhất của các chữ cái trong tiếng anh

- Giải thích:

Đây là mã hóa văn bản bằng cách tạo khóa random

Enter the text to encrypt: The title refers to the story's main antagonist, the Dark Lord Sauron, who in an earlier age created the One Ring, allowing him to rule the other Rings of Power given to men, dwarves, and elves, in his campaign to conquer all of Middle-earth. From homely beginnings in the Shire, a hobbit land reminiscent of the English countryside, the story ranges across Middle-earth, following the quest to destroy the One Ring, seen mainly through the eyes of the hobbits Frodo, Sam, Merry, and Pippin. Aiding the hobbits are the wizard Gandalf, the men Aragorn and Boromir, the elf Legolas, and the dwarf Gimli, who unite as the Company of the Ring in order to rally the Free Peoples of Middle-earth against Sauron's armies and give Frodo a chance to destroy the One Ring in the fires of Mount Doom.

Encrypted text:

Rnw rqrxw twawth rv rnw hrvtb'h jsqm smrsfvmqhr, rnw Lstz Xvtl Hsktvm, unv qm sm wstxqwt sfw otwsrwl rnw Vmw Tqmf, sxxvuqmf nqj rv tkxw rnw vrnwt Tqmfh va Cvuwt fqywm rv jwm, lustywh, sml wxywh, qm nqh osjcsqfm rv ovmpkwt sxx va Jqllxw-wstrn. Atvj nvjwxb ewfqmmqmfh qm rnw Hnqtw, s nveeqr xsml twjqmqhowmr va rnw Wmfxqhn ovkmrtbhqlw, rnw hrvtb tsmfwh sotvhh Jqllxw-wstrn, avxxvuqmf rnw pkwhr rv lwhrtvb rnw Vmw Tqmf, hwwm jsqmxb rntvkfn rnw wbwh va rnw nveeqrh Atvlv, Hsj, Jwttb, sml Cqccqm. Sqlqmf rnw nveeqrh stw rnw uqdstl Fsmlsxa, rnw jwm Stsfvtm sml Evtvjqt, rnw

wxa Xwfvxsh, sml rnw lusta Fqjxq, unv kmqrw sh rnw Ovjcsmb va rnw Tqmf qm vtlwt rv tsxxb rnw Atww Cwvcxwh va Jqllxw-wstrn sfsqmhr Hsktvm'h stjqwh sml fqyw Atvlv s onsmow rv lwhrtvb rnw Vmw Tqmf qm rnw aqtwh va Jvkmr Lvvj.

- Các bước giải mã:

```python
from collections import Counter
import string

# Known English letter frequencies (from most frequent to least frequent)
english_frequencies = 'ETAOINSHRDLCUMWFGYPBVKJXQZ'

#
cipher_text = ("Rnw rqrxw twawth rv rnw hrvtb'h jsqm smrsfvmqhr, rnw Lstz Xvtl Hsktvm, unv qm sm wstxqwt sfw otwsrwl rnw Vmw Tqmf, sxxvuqmf nqj rv tkxw
# Count letter frequencies in the ciphertext (ignoring non-alphabet characters)
cipher_counts = Counter(''.join(filter(str.isalpha, cipher_text.upper())))

# Sort the ciphertext letters by frequency (most frequent first)
sorted_cipher = ''.join([item[0] for item in cipher_counts.most_common()])

# Create an initial mapping from ciphertext letters to the English frequency order
mapping = {}
for i, letter in enumerate(sorted_cipher):
    mapping[letter] = english_frequencies[i]

# For any letters not present in the ciphertext, add an identity mapping
for letter in string.ascii_uppercase:
    if letter not in mapping:
        mapping[letter] = letter

# Optional manual adjustments to improve decryption quality
#mapping["Z"] = "H"
```

```
D:\Project_Pycharm\Mat_ma_hoc\.venv\Scripts\python.exe D:\Project_Pycharm\Mat_ma_hoc\HackSubs
A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
 --+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--
M  W  P  X  B  C  G  R  I  U  F  D  O  H  G  K  N  A  S  I  Y  T  E  L  V  J

Decrypted Text:
Ahe anale iemeir at ahe ratiw'r usno soasctonra, ahe Dsij Ltid Rsfito, yht no so esilnei sce

Process finished with exit code 0
```

Decrypted Text:

Ahe anale iemeir at ahe ratiw'r usno soasctonra, ahe Dsij Ltid Rsfito, yht no so esilnei sce giesaed ahe Toe Inoc, slltynoc hnu at ifle ahe tahei Inocr tm Ptyei cnveo at ueo, dysiver, sod elver, no hnr gsupsnco at gtokfei sll tm Unddle-esiah. Mitu htuelw becnoonocr no ahe Rhnie, s htbbna lsod ieunonrgeoa tm ahe Eoclnrh gtfoaiwrnde, ahe ratiw isocer sgitrr

Unddle-esiah, mtlltynoc ahe kfera at deraitw ahe Toe Inoc, reeo usnolw ahitfch ahe ewer tm ahe htbbnar Mitdt, Rsu, Ueiiw, sod Pnppno. Sndnoc ahe htbbnar sie ahe ynxsid Csodslm, ahe ueo Sisctio sod Btituni, ahe elm Lectlsr, sod ahe dysim Cnuln, yht fonae sr ahe Gtupsow tm ahe Inoc no tidei at isllw ahe Miee Petpler tm Unddle-esiah scsnora Rsfito'r siuner sod cnve Mitdt s ghsoge at deraitw ahe Toe Inoc no ahe mnier tm Utfoa Dttu.

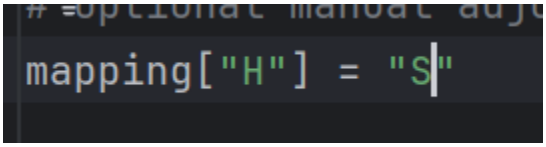
Đây là đoạn văn bản sau khi ánh xạ dựa trên tần suất xuất hiện các từ trong tiếng anh bằng cách ánh xạ nó với các từ tương ứng xuất hiện nhiều nhất trong đoạn văn sắp xếp theo tuần xuất giảm dần

Sau khi xem đoạn văn trên thì ta vẫn chưa thể hiểu đoạn văn trên nói gì và nó vẫn chưa có chữ tiếng Anh nào cụ thể cả. Nhưng chúng ta có thể dựa vào vị trí của mạo từ a an the, các sỡ hưu cách như ‘s để đoán nó được ánh xạ từ chữ nào và thay đổi cho hợp lý hơn

Ví dụ: “ratiw'r” chúng ta có thể thấy “ ‘r ” ở đây chỉ có thể là “ ‘s ” sở hữu cách trong tiếng Anh

Chữ R được ánh xạ từ chữ H

Nên ta có thể sửa lại như sau:



Decrypted Text:

Ahe anale iemeis at ahe satiw's usno soasctonsa, ahe Dsij Ltid Ssfito, yht no so esilnei sce giesaed ahe Toe Inoc, slltynoc hnu at ifle ahe tahei Inocs tm Ptyei cnveo at ueo, dysives, sod elves, no hns gsupsnco at gtokfei sll tm Unddle-esiah. Mitu htuelw becnoonocs no ahe Shnie, s htbbna lsod ieunonsgeoa tm ahe Eoclnsh gtfoaiwsnde, ahe satiw isoces sgitss Unddle-esiah, mtlltynoc ahe kfesa at desaitw ahe Toe Inoc, seeo usnolw ahitfch ahe ewes tm ahe htbbnas Mitdt, Ssu, Ueiiw, sod Pnppno. Sndnoc ahe htbbnas sie ahe ynxsid Csodslm, ahe ueo Sisctio sod Btituni, ahe elm Lectlss, sod ahe dysim Cnuln, yht fonae ss ahe Gtupsow tm ahe Inoc no tidei at isllw ahe Miee Petples tm Unddle-esiah scsnosa Ssfito's siunes sod cnve Mitdt s ghsoge at desaitw ahe Toe Inoc no ahe mnies tm Utfoa Dttu.

Đây là sau khi ánh xạ lại chữ H sang S

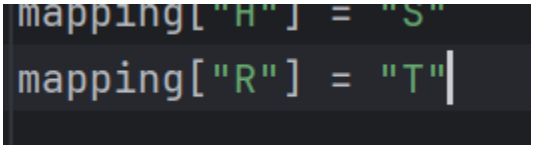
Tiếp theo ta có thể thấy các từ lặp lại như “Ahe” khả năng cao nó là từ she hoặc là từ the

Nhưng vì ở đoạn này “Ahe anale iemeis at ahe satiw's usno soasctonsa,”

Ta có thể khẳng định ngay nó là chữ “the” bởi vì từ she chỉ có thể đứng đầu câu thôi không phải ở giữa câu như đoạn này

Và chữ A được ánh xạ từ R

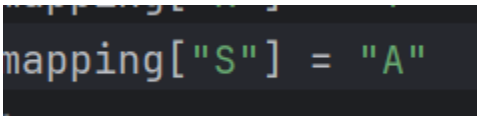
Nên ta có thể đổi lại như sau



Decrypted Text:

The tntle iemeis tt the sttiw's usno sotsctonst, the Dsij Ltid Ssfito, yht no so esilnei sce giested the Toe Inoc, slltynoc hnu tt ifle the tthei Inocs tm Ptyei cnveo tt ueo, dysives, sod elves, no hns gsupsnco tt gtokfei sll tm Unddle-esith. Mitu htuelw becnoonocs no the Shnie, s htbbnt lsod ieunonsgeot tm the Eoclnsh gtfotiwsnde, the sttiw isoces sgitss Unddle-esith, mtlltynoc the kfest tt destitw the Toe Inoc, seeo usnolw thitfch the ewes tm the htbbnts Mitdt, Ssu, Ueiiw, sod Pnppno. Sndnoc the htbbnts sie the ynxsid Csodslm, the ueo Sisctio sod Btituni, the elm Lectlss, sod the dysim Cnuln, yht fonte ss the Gtupsow tm the Inoc no tidei tt isllw the Miee Petples tm Unddle-esith scsnost Ssfito's siunes sod cnve Mitdt s ghsoge tt destitw the Toe Inoc no the mnies tm Utfot Dttu.

Tại đoạn này “s htbbnt lsod ieunonsgeot tm the Eoclnsh gtfotiwsnde,”, Ta có thể thấy chữ “s” nó đứng một mình mà trong tiếng Anh chỉ có mạo từ a mới đứng một mình như vậy nên ta có thể ánh xạ lại như sau:

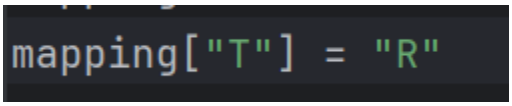


Decrypted Text:

The tntle iemeis tt the sttiw's uano aotactonst, the Daij Ltid Safito, yht no ao eailnei ace gieated the Toe Inoc, alltynoc hnu tt ifle the tthei Inocs tm Ptyei cnveo tt ueo, dyaives, aod elves, no hns gaupanco tt gtokfei all tm Unddle-eaith. Mitu htuelw becnoonocs no the Shnie, a htbbnt laod ieunonsgeot tm the Eoclnsh gtfotiwsnde, the sttiw iaoces agitss Unddle-eaith, mtlltynoc the kfest tt destitw the Toe Inoc, seeo uanolw thitfch the ewes tm the htbbnts Mitdt, Sau, Ueiiw, aod Pnppno. Andnoc the htbbnts aie the ynxaid Caodalm, the ueo Aiactio aod Btituni, the elm Lectlas, aod the dyaim Cnuln, yht fonte as the

Gtupaow tm the Inoc no tidei tt iallw the Miee Petples tm Unddle-eaith acanost Safito's aiunes aod cnve Mitdt a ghaoge tt destitw the Toe Inoc no the mnies tm Utfot Dttu.

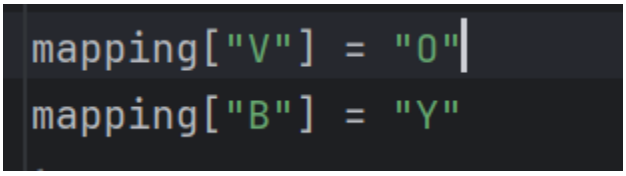
Ở đây ta có thể thấy chữ eaith trong "Unddle-eaith" giống như là earth nên là ta có thể ánh xạ như sau:



Decrypted Text:

The tntle remers tt the sttrw's uano aotactonst, the Darj Ltrd Safrto, yht no ao earlner ace greated the Toe Rnoc, alltynoc hnu tt rfle the tther Rnocs tm Ptyer cnveo tt ueo, dyarves, aod elves, no hns gaupanco tt gtokfer all tm Unddle-earth. Mrtu htuelw becnoonocs no the Shnre, a htbbnt laod reunonsgeot tm the Eoclnsh gtfotrwsnde, the sttrw raoces agrtss Unddle-earth, mtlltynoc the kfest tt destrtw the Toe Rnoc, seeo uanolw thrtfch the ewes tm the htbbnts Mrtdt, Sau, Uerrw, aod Pnppno. Andnoc the htbbnts are the ynxard Caodalm, the ueo Aractro aod Btrtunr, the elm Lectlas, aod the dyarm Cnuln, yht fonte as the Gtupaow tm the Rnoc no trder tt rallw the Mree Petples tm Unddle-earth acanost Safrto's arunes aod cnve Mrtdt a ghaoge tt destrtw the Toe Rnoc no the mnres tm Utfot Dttu.

Tiếp theo ta có thể thấy từ này "destrtw" khá giống với destroy nên ta có thể ánh xạ lại như sau



Decrypted Text:

The tntle remers to the story's uano aotacoonst, the Darj Lord Safroo, yho no ao earlner ace greated the Ooe Rnoc, alloynoc hnu to rfle the other Rnocs om Poyer cnveo to ueo, dyarves, aod elves, no hns gaupanco to gookfer all om Unddle-earth. Mrou houely becnoonocs no the Shnre, a hobbnt laod reunonsgeot om the Eoclnsh gofotrysnde, the story raoces agross Unddle-earth, molloynoc the kfest to destroy the Ooe Rnoc, seeo uanoly throfch the eyes om the hobbnts Mrodo, Sau, Uerry, aod Pnppno. Andnoc the hobbnts are the ynxard Caodalm, the ueo Aracoro aod Borounr, the elm Lecolas, aod the dyarm Cnuln, yho fonte as the Goupaoy om the Rnoc no order to rally the Mree Peoples om Unddle-earth acanost Safroo's arunes aod cnve Mrodo a ghaoge to destroy the Ooe Rnoc no the mnres om Uofot Doou.

Từ này “earlner” khá giống “earlier” nên ta có thể ánh xạ như sau:

```
mapping["Q"] = "I"
```

Cứ tiếp tục tìm những từ gần giống với các từ gốc tiếng anh chúng ta sẽ từ từ giải mã ra được lại đoạn văn gốc, sau đây là kết quả sau khi điều chỉnh ánh xạ

```
# Optional manual adjustments to improve decryption quality
mapping["H"] = "S"
mapping["R"] = "T"
mapping["S"] = "A"
mapping["T"] = "R"
mapping["V"] = "O"
mapping["B"] = "Y"
mapping["Q"] = "I"
mapping["M"] = "S"
mapping["U"] = "W"
mapping["J"] = "M"
mapping["A"] = "F"
mapping["M"] = "N"
mapping["F"] = "G"
mapping["O"] = "C"
mapping["K"] = "U"
mapping["P"] = "Q"
mapping["Z"] = "K"
mapping["D"] = "Z"
```

Decrypted Text:

The title refers to the story's main antagonist, the Dark Lord Sauron, who in an earlier age created the One Ring, allowing him to rule the other Rings of Power given to men, dwarves, and elves, in his campaign to conquer all of Middle-earth. From homely beginnings in the Shire, a hobbit land reminiscent of the English countryside, the story ranges across Middle-earth, following the quest to destroy the One Ring, seen mainly through the eyes of the hobbits Frodo, Sam, Merry, and Pippin. Aiding the hobbits are the wizard Gandalf, the men Aragorn and Boromir, the elf Legolas, and the dwarf Gimli, who unite as the Company of the Ring in order to rally the Free Peoples of Middle-earth against

Sauron's armies and give Frodo a chance to destroy the One Ring in the fires of Mount Doom.

- Từ cách làm trên ta có thể đưa ra các bước làm chung:

+Bước 1: Tìm những từ liên quan đến mạo từ, động từ phổ biến và sơ hữu cách như { a, an, the, is, ‘s, am are,…} để ánh xạ

+Bước 2: Sau khi hoàn thành bước 1, ta có thể tìm một số từ ngắn có thể đoạn được như “eaith” thành “earth”, “destrtw” thành “destroy”

+Bước 3: Đoán nghĩa những từ còn lại có thể đoán được, một số từ chỉ cần ánh xạ một hai từ thì nên ưu tiên trước những từ có thể đoán được nhưng cần 3 đến 4 từ ánh xạ trở lên

Qua ba bước trên là chúng ta đã hoàn thành việc giải mã

Bài 3: Polyalphabetic

3.1) Hill cipher 2x2

- Phần code:

Mã hóa:

```python
#Tạo một ma trận key 2*2
keyMatrix = [[0] * 2 for i in range(2)]
#Tạo ma trận 2*1 để chứa 2 ký tự cần mã hóa
messageVector = [[0] for i in range(2)]
#Ma trận kết quả sau mã hóa
cipherMatrix = [[0] for i in range(2)]

#Biến key vừa nhập thành một trận 4*4 được đánh số theo thứ tự trong bảng chữ cái
1 usage
def getKeyMatrix(key):
    k = 0
    for i in range(2):
        for j in range(2):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

#Mã hóa bằng cách nhân ma trận keyMatrix * messageVector = cipherMatrix
#Sau khi có cipherMatrix cần mod 26 để nó nằm trong phạm vi của bảng chữ cái
1 usage
def encrypt(messageVector):
    for i in range(2):
        cipherMatrix[i][0] = 0
        for x in range(2):
            cipherMatrix[i][0] += (keyMatrix[i][x] * messageVector[x][0])
        cipherMatrix[i][0] = cipherMatrix[i][0] % 26

#Thuật toán mã hóa HillCipher 2*2
1 usage
def HillCipher(message, key):
    #Tạo khóa
    getKeyMatrix(key)

    #Biến các chữ cái được đánh số giống trong bảng chữ cái
    for i in range(2):
        messageVector[i][0] = ord(message[i]) % 65
    #Mã hóa
    encrypt(messageVector)

    #Biến đoạn code sau thành lại chuỗi
    CipherText = []
    for i in range(2):
        CipherText.append(chr(cipherMatrix[i][0] + 65))
    return "".join(CipherText)
```

```python
#Hàm khởi tạo
def main():
    #Nhập một chuỗi có 2 ký tự nếu sai nhập lại
    while True:
        plaintext = input("Enter a string(2 characters): ")
        if len(plaintext) == 2:
            break
    #Nhập key có 16 ký tự nếu sai nhập lại
    while True:
        key = input("Enter a key(4 characters): ")
        if len(key) == 4:
            break
    #Chuỗi sau mã hóa. In hoa trước khi đi vào mã hóa
    ciphertext = HillCipher(plaintext.upper(), key.upper())
    print(ciphertext)

if __name__ == "__main__":
    main()
```

Giải mã:

```python
# Tạo một ma trận key 2*2
keyMatrix = [[0] * 2 for i in range(2)]
# Tạo ma trận 2*1 để chứa 2 ký tự cần giải mã
messageVector = [[0] for i in range(2)]
# Ma trận kết quả sau giải mã
plainMatrix = [[0] for i in range(2)]

# Biến key vừa nhập thành một ma trận 2*2 được đánh số theo thứ tự trong bảng chữ cái
1 usage
def getKeyMatrix(key):
    k = 0
    for i in range(2):
        for j in range(2):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

# Tìm nghịch đảo modulo 26 của ma trận
1 usage
def modInverseMatrix(matrix):
    det = (matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]) % 26
    det_inv = pow(det, -1, 26)
    inv_matrix = [
        [matrix[1][1] * det_inv % 26, -matrix[0][1] * det_inv % 26],
        [-matrix[1][0] * det_inv % 26, matrix[0][0] * det_inv % 26],
    ]
    for i in range(2):
        for j in range(2):
            inv_matrix[i][j] = inv_matrix[i][j] % 26
    return inv_matrix

# Giải mã bằng cách nhân nghịch đảo ma trận keyMatrix với messageVector để có plainMatrix
1 usage
def decrypt(messageVector, inverseKeyMatrix):
    for i in range(2):
        plainMatrix[i][0] = 0
        for x in range(2):
            plainMatrix[i][0] += (inverseKeyMatrix[i][x] * messageVector[x][0])
        plainMatrix[i][0] = plainMatrix[i][0] % 26
```

```python
# Thuật toán giải mã Hill Cipher 2*2
1 usage
def HillCipherDecrypt(ciphertext, key):
    # Tạo khóa
    getKeyMatrix(key)

    # Tìm nghịch đảo của ma trận khóa
    inverseKeyMatrix = modInverseMatrix(keyMatrix)

    # Biến các chữ cái được đánh số giống trong bảng chữ cái
    for i in range(2):
        messageVector[i][0] = ord(ciphertext[i]) % 65

    # Giải mã
    decrypt(messageVector, inverseKeyMatrix)

    # Biến đoạn code sau thành lại chuỗi
    PlainText = []
    for i in range(2):
        PlainText.append(chr(plainMatrix[i][0] + 65))
    return "".join(PlainText)
```

```python
# Hàm khởi tạo
def main():
    # Nhập một chuỗi đã được mã hóa có 2 ký tự
    while True:
        ciphertext = input("Enter a cipher text (2 characters): ")
        if len(ciphertext) == 2:
            break
    # Nhập key có 4 ký tự
    while True:
        key = input("Enter a key (4 characters): ")
        if len(key) == 4:
            break
    # Chuỗi sau giải mã
    plaintext = HillCipherDecrypt(ciphertext.upper(), key.upper())
    print(plaintext)

if __name__ == "__main__":
    main()
```

- Giải thích:

Thuật toán mã hóa hill cipher 2x2 sẽ mã hóa theo kiểu khối. Mỗi lần mã hóa nó sẽ mã hóa 2 ký tự cùng một lúc. Ở đoạn code trên chỉ mã hóa và giải mã một khối, nếu muốn mã hóa một đoạn văn thì cần dùng dùng 1 loop để duyệt qua từng đôi một ký tự.

Thuật toán mã hóa sẽ có các bước sau:

+Nhập 2 chữ cần mã hóa. Chữ này sẽ được lưu dưới dạng ma trận 2x1 được ký hiệu là M

+Nhập một ma trận key 2x2. Ma trận này sẽ được lưu dưới dạng số theo thứ tự của bảng chữ cái. Ký hiệu là K.

+Mã hóa bằng cách E = K.M mod 26

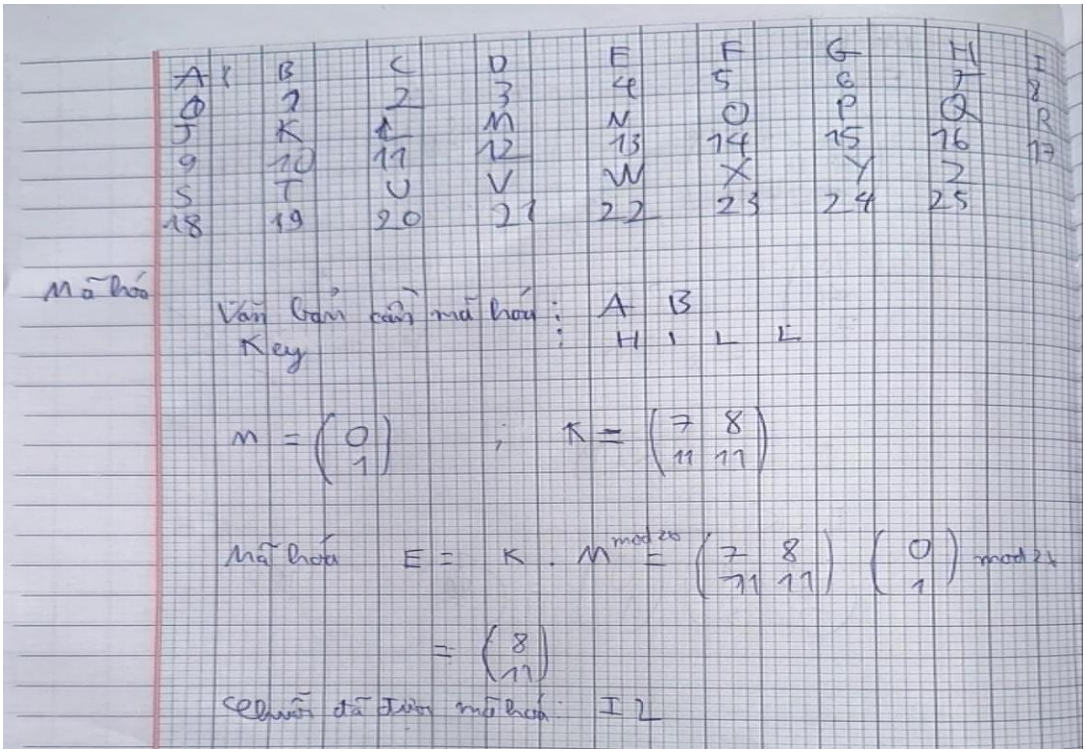
Thuật toán giả mã sẽ có các bước sau:

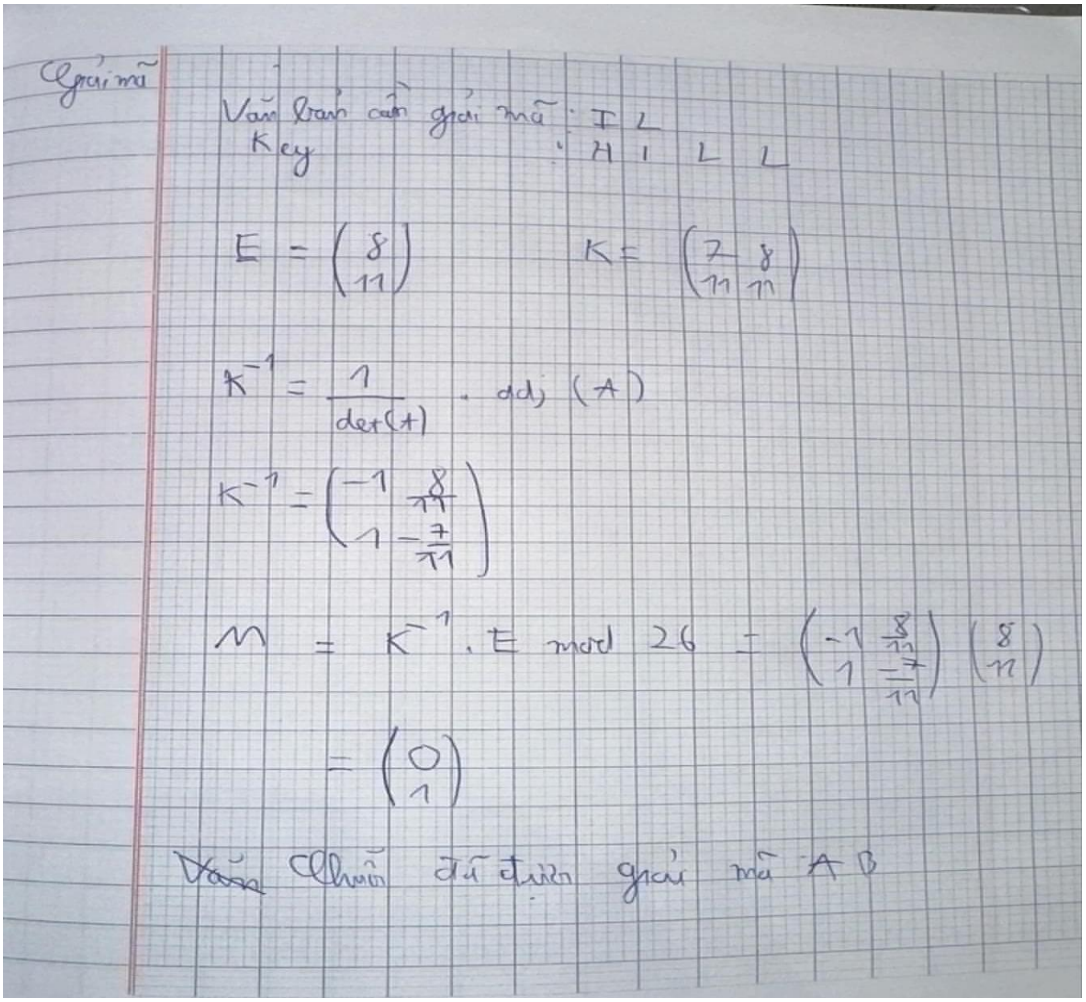
+Nhập 2 chữ cần giải mã. Chữ này sẽ được lưu dưới dạng ma trận 2x1 được hiệu là E

+Nhập một ma trận key 2x2. Ma trận này sẽ được lưu dưới dạng số theo thứ tự của bản chữ cái. Ký hiệu là K.

+Giải mã bằng cách là $M = K^{-1}.E \bmod 26$

- Ví dụ:






```
Enter a string(2 characters): AB
Enter a key(4 characters): HILL
IL
```

```
Enter a cipher text (2 characters): IL
Enter a key (4 characters): HILL
AB
```

- Lưu ý thuật toán:

Thuật toán này phải chọn key sao cho định thức det(K) khác 0 vì nếu det(K) = 0 thì sẽ không có ma trận nghịch đảo của K và không giải mã được.

3.2) Hill cipher 3x3

- Phần code:

Mã hóa:

```python
import numpy as np

# Tạo ma trận key 3x3
keyMatrix = [[0] * 3 for i in range(3)]
# Ma trận 1x3 để chứa 3 ký tự mã hóa
cipherVector = [[0] for i in range(3)]
# Ma trận chứa kết quả giải mã
plainMatrix = [[0] for i in range(3)]

# Biến key thành ma trận 3x3
1 usage
def getKeyMatrix(key):
    k = 0
    for i in range(3):
        for j in range(3):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

# Tìm nghịch đảo modulo 26 của một số
1 usage
def modInverse(a, m=26):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return None
```

```python
# Tính ma trận nghịch đảo modulo 26
1 usage
def getInverseKeyMatrix():
    matrix = np.array(keyMatrix)

    # Tính định thức và nghịch đảo định thức modulo 26
    det = int(round(np.linalg.det(matrix))) % 26
    det_inv = modInverse(det, m: 26)

    if det_inv is None:
        raise ValueError("Ma trận không khả nghịch modulo 26!")

    # Tính ma trận adjugate (phụ hợp)
    adjugate = np.round(np.linalg.inv(matrix) * np.linalg.det(matrix)).astype(int) % 26

    # Nhân với nghịch đảo của định thức mod 26
    inverseMatrix = (adjugate * det_inv) % 26
    return inverseMatrix.astype(int)

# Nhân ma trận nghịch đảo với ciphertext để giải mã
1 usage
def decrypt(cipherVector, inverseKeyMatrix):
    for i in range(3):
        plainMatrix[i][0] = 0
        for x in range(3):
            plainMatrix[i][0] += (inverseKeyMatrix[i][x] * cipherVector[x][0])
        plainMatrix[i][0] = plainMatrix[i][0] % 26

# Thuật toán giải mã Hill Cipher 3x3
```

```python
# Thuật toán giải mã Hill Cipher 3×3
1 usage
def HillCipherDecrypt(ciphertext, key):
    # Tạo khóa
    getKeyMatrix(key)

    # Tính ma trận nghịch đảo của key
    inverseKeyMatrix = getInverseKeyMatrix()

    # Chuyển ciphertext thành số
    for i in range(3):
        cipherVector[i][0] = ord(ciphertext[i]) % 65

    # Giải mã
    decrypt(cipherVector, inverseKeyMatrix)

    # Chuyển về dạng chữ cái
    plaintext = "".join([chr(plainMatrix[i][0] + 65) for i in range(3)])
    return plaintext
```

```python
# Hàm khởi chạy chương trình
def main():
    # Nhập ciphertext (3 ký tự)
    while True:
        ciphertext = input("Enter a ciphertext(3 characters): ").upper()
        if len(ciphertext) == 3:
            break
    # Nhập key (9 ký tự)
    while True:
        key = input("Enter a key(9 characters): ").upper()
        if len(key) == 9:
            break
    try:
        plaintext = HillCipherDecrypt(ciphertext, key)
        print("Decrypted text:", plaintext)
    except ValueError as e:
        print("Error:", e)

if __name__ == "__main__":
    main()
```

Giải mã:

```python
import numpy as np

# Tạo ma trận key 3x3
keyMatrix = [[0] * 3 for i in range(3)]
# Ma trận 1x3 để chứa 3 ký tự mã hóa
cipherVector = [[0] for i in range(3)]
# Ma trận chứa kết quả giải mã
plainMatrix = [[0] for i in range(3)]

# Biến key thành ma trận 3x3
1 usage
def getKeyMatrix(key):
    k = 0
    for i in range(3):
        for j in range(3):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

# Tìm nghịch đảo modulo 26 của một số
1 usage
def modInverse(a, m=26):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return None
```

```python
# Tính ma trận nghịch đảo modulo 26
1 usage
def getInverseKeyMatrix():
    matrix = np.array(keyMatrix)

    # Tính định thức và nghịch đảo định thức modulo 26
    det = int(round(np.linalg.det(matrix))) % 26
    det_inv = modInverse(det, m: 26)

    if det_inv is None:
        raise ValueError("Ma trận không khả nghịch modulo 26!")

    # Tính ma trận adjugate (phụ hợp)
    adjugate = np.round(np.linalg.inv(matrix) * np.linalg.det(matrix)).astype(int) % 26

    # Nhân với nghịch đảo của định thức mod 26
    inverseMatrix = (adjugate * det_inv) % 26
    return inverseMatrix.astype(int)

# Nhân ma trận nghịch đảo với ciphertext để giải mã
1 usage
def decrypt(cipherVector, inverseKeyMatrix):
    for i in range(3):
        plainMatrix[i][0] = 0
        for x in range(3):
            plainMatrix[i][0] += (inverseKeyMatrix[i][x] * cipherVector[x][0])
        plainMatrix[i][0] = plainMatrix[i][0] % 26
```

```python
# Thuật toán giải mã Hill Cipher 3×3
1 usage
def HillCipherDecrypt(ciphertext, key):
    # Tạo khóa
    getKeyMatrix(key)

    # Tính ma trận nghịch đảo của key
    inverseKeyMatrix = getInverseKeyMatrix()

    # Chuyển ciphertext thành số
    for i in range(3):
        cipherVector[i][0] = ord(ciphertext[i]) % 65

    # Giải mã
    decrypt(cipherVector, inverseKeyMatrix)

    # Chuyển về dạng chữ cái
    plaintext = "".join([chr(plainMatrix[i][0] + 65) for i in range(3)])
    return plaintext

# Hàm khởi chạy chương trình
def main():
    # Nhập ciphertext (3 ký tự)
    while True:
        ciphertext = input("Enter a ciphertext(3 characters): ").upper()
        if len(ciphertext) == 3:
            break
    # Nhập key (9 ký tự)
    while True:
        key = input("Enter a key(9 characters): ").upper()
        if len(key) == 9:
            break
    try:
        plaintext = HillCipherDecrypt(ciphertext, key)
        print("Decrypted text:", plaintext)
    except ValueError as e:
        print("Error:", e)
```

- Giải thích code:

Thuật toán tương tự như hill cipher 2x2 chỉ khác chuỗi mã hóa là 3 ký tự và ma trận key là 3x3

3.3) Hill cipher 4x4

- Phần code:

Phần mã hóa:

```python
#Tạo một ma trận key 4*4
keyMatrix = [[0] * 4 for i in range(4)]
#Tạo ma trận 1*2 để chứa 2 ký tự cần mã hóa
messageVector = [[0] for i in range(4)]
#Ma trận kết quả sau mã hóa
cipherMatrix = [[0] for i in range(4)]

#Biến key vừa nhập thành một trận 4*4 được đánh số theo thứ tự trong bảng chữ cái
1 usage
def getKeyMatrix(key):
    k = 0
    for i in range(4):
        for j in range(4):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

#Mã hóa bằng cách nhân ma trận keyMatrix * messageVector = cipherMatrix
#Sau khi có cipherMatrix cần mod 26 để nó nằm trong phạm vi của bảng chữ cái
1 usage
def encrypt(messageVector):
    for i in range(4):
        cipherMatrix[i][0] = 0
        for x in range(4):
            cipherMatrix[i][0] += (keyMatrix[i][x] * messageVector[x][0])
        cipherMatrix[i][0] = cipherMatrix[i][0] % 26

```

```python
#Thuật toán mã hóa HillCipher 4*4
1 usage
def HillCipher(message, key):
    #Tạo khóa
    getKeyMatrix(key)

    #Biến các chữ cái được đánh số giống trong bảng chữ cái
    for i in range(4):
        messageVector[i][0] = ord(message[i]) % 65
    #Mã hóa
    encrypt(messageVector)

    #Biến đoạn code sau thành lại chuỗi
    CipherText = []
    for i in range(4):
        CipherText.append(chr(cipherMatrix[i][0] + 65))
    return "".join(CipherText)

#Hàm khởi tạo
def main():
    #Nhập một chuỗi có 4 ký tự nếu sai nhập lại
    while True:
        plaintext = input("Enter a string(4 characters): ")
        if len(plaintext) == 4:
            break
    #Nhập key có 16 ký tự nếu sai nhập lại
    while True:
        key = input("Enter a key(16 characters): ")
        if len(key) == 16:
            break
    #Chuỗi sau mã hóa. In hoa trước khi đi vào mã hóa
    ciphertext = HillCipher(plaintext.upper(), key.upper())
    print(ciphertext)

if __name__ == "__main__":
    main()
```

Giải mã:

```python
#Tính định thức
2 usages
def calculate_det(matrix):
    # Tách ma trận thành các phần tử riêng biệt
    a, b, c, d = matrix[0]
    e, f, g, h = matrix[1]
    i, j, k, l = matrix[2]
    m, n, o, p = matrix[3]

    # Tính định thức theo công thức khai triển
    det = (
            a * (f * (k * p - l * o) - g * (j * p - l * n) + h * (j * o - k * n))
            - b * (e * (k * p - l * o) - g * (i * p - l * m) + h * (i * o - k * m))
            + c * (e * (j * p - l * n) - f * (i * p - l * m) + h * (i * n - j * m))
            - d * (e * (j * o - k * n) - f * (i * o - k * m) + g * (i * n - j * m))
    )
    return det

# Hàm tạo ma trận khóa 4x4 từ key
1 usage
def getKeyMatrix(key):
    keyMatrix = [[0] * 4 for _ in range(4)]
    k = 0
    for i in range(4):
        for j in range(4):
            keyMatrix[i][j] = (ord(key[k]) - 65) % 26  # A=0, B=1, ..., Z=25
            k += 1
    return keyMatrix
```

```python
# Hàm tính nghịch đảo modulo 26 của định thức
1 usage
def mod_inverse(det, mod=26):
    for i in range(1, mod):
        if (det * i) % mod == 1:
            return i
    raise ValueError("Không tìm thấy nghịch đảo modulo 26.")

# Hàm tính ma trận nghịch đảo (tự code)
1 usage
def modInverseMatrix(matrix):
    det = calculate_det(matrix) % 26

    det_inv = mod_inverse(det)

    # Tính ma trận phụ hợp (adjugate)
    adj = [[0]*4 for _ in range(4)]
    for i in range(4):
        for j in range(4):
            minor = [row[:j] + row[j+1:] for row in (matrix[:i] + matrix[i+1:])]
            minor_det = calculate_det(minor)
            adj[j][i] = ((-1) ** (i + j)) * minor_det  # Transpose và đổi dấu
            adj[j][i] = adj[j][i] % 26

    # Nhân với det_inv và mod 26
    inverse = [[(adj[i][j] * det_inv) % 26 for j in range(4)] for i in range(4)]
    return inverse

```

```python
# Hàm mã hóa
1 usage
def hillCipherEncrypt(plaintext, keyMatrix):
    message = [(ord(c) - 65) % 26 for c in plaintext]
    cipher = [0] * 4
    for i in range(4):
        cipher[i] = sum(keyMatrix[i][j] * message[j] for j in range(4)) % 26
    return ''.join([chr(c + 65) for c in cipher])

# Hàm giải mã
1 usage
def hillCipherDecrypt(ciphertext, keyMatrix):
    inverseKey = modInverseMatrix(keyMatrix)
    message = [(ord(c) - 65) % 26 for c in ciphertext]
    plain = [0] * 4
    for i in range(4):
        plain[i] = sum(inverseKey[i][j] * message[j] for j in range(4)) % 26
    return ''.join([chr(p + 65) for p in plain])

# Hàm chính
def main():
    plaintext = input("Nhập plaintext (4 ký tự): ").upper()
    key = input("Nhập key (16 ký tự): ").upper()

    keyMatrix = getKeyMatrix(key)
    print("Ma trận khóa:")
    for row in keyMatrix:
        print(row)

    try:
        ciphertext = hillCipherEncrypt(plaintext, keyMatrix)
        print("Ciphertext:", ciphertext)
        decrypted = hillCipherDecrypt(ciphertext, keyMatrix)
        print("Giải mã:", decrypted)
    except ValueError as e:
        print(e)
```

- Giải thích code:

Thuật toán tương tự hill cipher 2*2. Chỉ khác là phải có chuỗi mã hóa là 4 ký tự và ma trận key là 4*4