《密码学》第一章作业

1.18

1.29

附上第二题的代码：

recover.py：用于暴力推测密钥长度的值，并且将密文进行第一步恢复

```
import collections
import string
from collections import Counter

s =
'IYMYSILONRFNCQXQJEDSHBUIBCJUZBOLFQYSCHATPEQGQJEJNGNXZWHHGWFSUKULJQACZKKJOAAHGKEM
TAFGMKVRDOPXNEHEKZNKFSKIFRQVHHOVXINPHMRTJPYWQGJWPUUVKFPOAWPMRKKQZWLQDYAZDRMLPBJKJ
OBWIWPSEPVVQMBCRYVCRUZAAOUMBCHDAGDIEMSZFZHALIGKEMJJFPCIWKRMLMPINAYOFIREAOLDTHITDV
RMSE'

def create_ascii_to_num():
    return {c: i for i, c in enumerate(sorted(set(string.ascii_uppercase)))}

def create_num_to_ascii():
    return {i: c for i, c in enumerate(sorted(set(string.ascii_uppercase)))}

def create_ascii_back(num_to_ascii):
    return {b: num_to_ascii[(a + 25) % 26] for b, a in
create_ascii_to_num().items()}

ascii_to_num = create_ascii_to_num()
num_to_ascii = create_num_to_ascii()
ascii_back = create_ascii_back(num_to_ascii)

def calculate_ic(text):
    n = len(text)
    frequency = collections.Counter(text)
    ic = sum(f * (f - 1) for f in frequency.values()) / (n * (n - 1))
    return ic

def back_cipher(ciphertext, l):
    after = ciphertext
    for i in range(l, len(ciphertext), l):
        after = after[:i] + ''.join([ascii_back[asc] for asc in after[i:]])
    return after

def find_key_length(ciphertext, max_len=15):
    average_ics = []
    for m in range(1, max_len + 1):
        a = back_cipher(ciphertext, m)
        ics = []
        for i in range(m):
            subsequence = a[i::m]
            ic = calculate_ic(subsequence)
            ics.append(ic)
        average_ics.append((m, sum(ics) / len(ics)))
    print(average_ics)
    # 找出最可能的密钥长度
    likely_length = max(average_ics, key=lambda x: (x[1] - 0.065))[0]
    return likely_length
```

```
# 调用 find_key_length 函数并传入密文
key_length = find_key_length(s)
print("最可能的密钥长度是:", key_length)


def recover(ciphertext):
    answer=''
    for i in range(49):
        for j in range(5):
            answer=answer+chr(((ord(ciphertext[i*5+j])-ord('A'))+26-
i%26)%26+ord('A'))
    #补上剩下的最后一位，直接手动添加就行
    answer=answer+chr(((ord(ciphertext[49*5+0])-ord('A'))+3)%26+ord('A'))

    return answer

ciphertext='IYMYSILONRFNCQXQJEDSHBUIBCJUZBOLFQYSCHATPEQGQJEJNGNXZWHHGWFSUKULJQACZ
KKJOAAHGKEMTAFGMKVRDOPXNEHEKZNKFSKIFRQVHHOVXINPHMRTJPYWQGJWPUUVKFPOAWPMRKKQZWLQDY
AZDRMLPBJKJOBWIWPSEPVVQMBCRYVCRUZAAOUMBCHDAGDIEMSZFZHALIGKEMJJFPCIWKRMLMPINAYOFIR
EAOLDTHITDVRMSE'
print(recover(ciphertext))

#IYMYSHKNMQDLAOVNGBAPDXQEXXEPUWIFZKSLVATMHWIYIAVAEXDNPMXWVLUHIYIZXDNPMXWVAMMSRVPX
DKPQWTEAMXXFVMPLRGURLYQOLWVAMMSZBMRSKPUWLRAYSHKXQVUVKFPNZVOLPIIOXTINAVWVZNIGKWEFD
IVQCPILXINNIETTIPMTHKPQQDJBQRVROURVRZFMRLTMXTRVPXTTPZMRFTAVTUXQVHFVMPXKGURIYMNYHZ
VQWH
```

vigenerecipher.py：用于维吉尼亚密钥的破解

```
import vigenerecipher
def keyword(Ciphertext, keylength):
    ListCiphertext = list(Ciphertext)
    Standard = {'A': 0.082, 'B': 0.015, 'C': 0.028, 'D': 0.043, 'E': 0.127, 'F':
0.022, 'G': 0.020, 'H': 0.061,
                'I': 0.070, 'J': 0.002, 'K': 0.008, 'L': 0.040, 'M': 0.024, 'N':
0.067, 'O': 0.075, 'P': 0.019,
                'Q': 0.001, 'R': 0.060, 'S': 0.063, 'T': 0.091, 'U': 0.028, 'V':
0.010, 'W': 0.023, 'X': 0.001,
                'Y': 0.020, 'Z': 0.001}

    while True:
        KeyResult = []

        for i in range(keylength):

            PresentCipherList = ListCiphertext[i::keylength]
            QuCoincidenceMax = 0
            KeyLetter = "*"

            for m in range(26):
                QuCoincidencePresent = 0

                for Letter in set(PresentCipherList):
                    LetterFrequency = PresentCipherList.count(Letter) /
len(PresentCipherList)
```

```
                    k = chr((ord(Letter) - 65 - m) % 26 + 65)
                    StandardFrequency = Standard[k]
                    QuCoincidencePresent = QuCoincidencePresent + LetterFrequency
* StandardFrequency

                if QuCoincidencePresent > QuCoincidenceMax:
                    QuCoincidenceMax = QuCoincidencePresent
                    KeyLetter = chr(m + 65)

            print("第", i + 1, "个密钥字母为:", KeyLetter, "对应的重合互指数为:",
QuCoincidenceMax)

            KeyResult.append(KeyLetter)

        Key = "".join(KeyResult)
        break

    return Key


if __name__ == '__main__':

    Ciphertext = input("输入密文：").upper()
    KeyResult = keyword(Ciphertext, 5)

    print("密钥最可能为：", KeyResult, "\n")

#密钥最可能为：PRIME
```

final_decode.py：用于最后的维吉尼亚密码的明文求解

```
import string

def vigenere_decrypt(text, key):

    lower_tab = string.ascii_lowercase
    upper_tab = string.ascii_uppercase
    digit_tab = string.digits

    plain_text = ''
    key_index = 0
    for char in text:
        if char.isupper():
            offset = ord(key[key_index % len(key)].upper()) - ord('A')
            plain_text += upper_tab[(upper_tab.index(char) - offset) % 26]
            key_index += 1
        elif char.islower():
            offset = ord(key[key_index % len(key)].lower()) - ord('a')
            plain_text += lower_tab[(lower_tab.index(char) - offset) % 26]
            key_index += 1
        elif char.isdigit():
            offset = ord(key[key_index % len(key)].upper()) - ord('A')
            plain_text += digit_tab[(digit_tab.index(char) - offset) % 10]
            key_index += 1
        else:
```

```
            plain_text += char

    return plain_text

if __name__ == '__main__':
    secret_key = 'PRIME'
    cipher_text =
'IYMYSHKNMQDLAOVNGBAPDXQEXXEPUWIFZKSLVATMHWIYIAVAEXDNPMXWVLUHIYIZXDNPMXWVAMMSRVPX
DKPQWTEAMXXFVMPLRGURLYQOLWVAMMSZBMRSKPUWLRAYSHKXQVUVKFPNZVOLPIIOXTINAVWVZNIGKWEFD
IVQCPILXINNIETTIPMTHKPQQDJBQRVROURVRZFMRLTMXTRVPXTTPZMRFTAVTUXQVHFVMPXKGURIYMNYHZ
VQWH'
    plain_text = vigenere_decrypt(cipher_text, secret_key)
    print(f'解密后得到的明文是{plain_text}')

#解密后得到的明文是
THEMOSTFAMOUSCRYPTOLOGISTINHISTORYOWESHISFAMELESSTOWHATHEDIDTHANTOWHATHESAIDANDTO
THESENSATIONALWAYINWHICHHESAIDITANDTHISWASMOSTPERFECTLYINCHARACTERFORHERBERTOSBOR
NEYARDLEYWASPERHAPSTHEMOSTENGAGINGARTICULATEANDTECHNICOLOREDPERSONALITYINTHEBUSIN
ESS
```