密码学作业 1

赵伯俣 2021302181156

2023 年 9 月 25 日

1 vigenere 密码的加密程序

以下将介绍使用 vigenere 密码的加密程序将一段明文加密成相应的密文

1.1 编写思路

首先将明文中的空格去掉, 只保留明文中的字母, 然后利用 vigenere 的方法将明文的下标与密文的下标相加之后模除 26 即可得到密文的下标, 将密文输出即可。

1.2 源代码

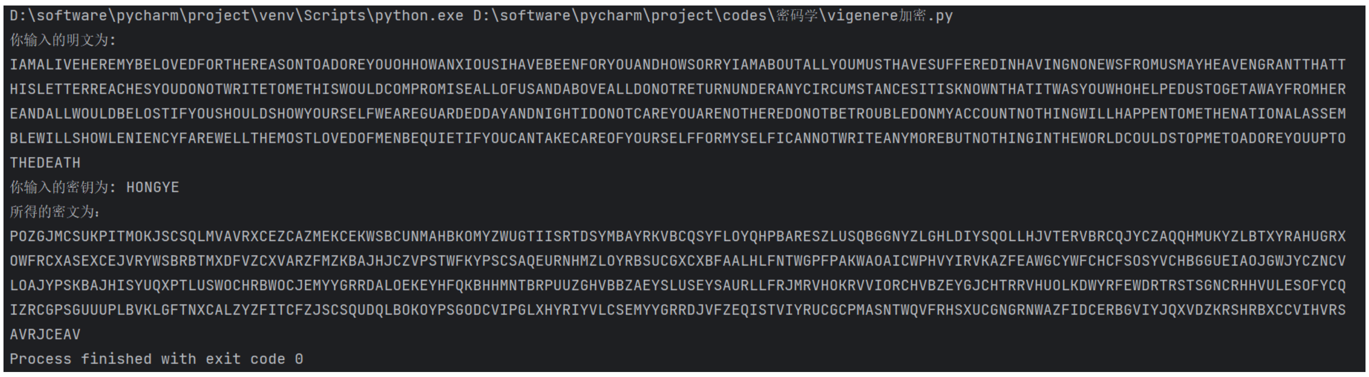
```python
letters = ['A', 'B', 'C', 'D', 'E',
           'F', 'G', 'H', 'I', 'J',
           'K', 'L', 'M', 'N', 'O',
           'P', 'Q', 'R', 'S', 'T',
           'U', 'V', 'W', 'X', 'Y', 'Z']

M = ("I am alive here, my beloved, for the reason to adore you. Oh!How anxious I have been for you and how sorry "
     "I am about all you must have suffered in having no news from us. May heaven grant that this letter reaches "
     "you. Do not write to me, this would compromise all of us and above all,do not return under any "
     "circumstances. It is known that it was you who helped us to get away from here and all would be lost if you "
     "should show yourself.We are guarded day and night. I do not care you are not here. Do not be troubled on my "
     "account. Nothing will happen to me. The national assemble will show leniency. Farewell the most loved of "
     "men. Be quiet if you can take care of yourself.For myself I cannot write any more, but nothing in the world "
     "could stop me to adore you up to the death.")

K = "hongye"


def c_alpha(cipher):
    """去掉密文中的非字母且全部转换为大写"""
    cipher_alpha = ''
    for i in range(len(cipher)):
        if cipher[i].isalpha():
            cipher_alpha += cipher[i]

    return cipher_alpha.upper()


def printf(output):
    """将要输出的字符串每150个一行进行输出"""
    i = 0
    for x in output:
        if i == 150:
            print("")
            i = 0
        print(x, end='')
        i += 1
    print("")


def change(M, K):
    """将明文通过密钥加密为密文"""
    m = -1
    k = -1
    result = []
    # index代表当前正在加密的明文标号
    for index in range(len(M)):
        for i in range(len(letters)):

            if letters[i] == M[index]:
                m = i  # m代表当前明文字母的字母表顺序
            if letters[i] == K[index % len(K)]:
                k = i  # k代表当前密钥字母的字母表顺序
        c = (m + k) % 26
        result.append(letters[c])
    return result


if __name__ == "__main__":
    M = c_alpha(M)  # 明文
    K = c_alpha(K)  # 密钥

    print("你输入的明文为:")
    printf(M)
    print("你输入的密钥为:", K)

    C = change(M, K)

    print("所得的密文为: ")
    printf(C)
```

Listing 1: vigenere 加密程序代码

1.3 结果演示

1.3.1 加密过程 1

明文为:I am alive here, my beloved, for the reason to adore you. Oh!How anxious I have been for you and how sorry I am about all you must have suffered in having no news from us. May heaven grant that this letter reaches you. Do not write to me, this would compromise all of us and above all,do not return under any circumstances. It is known that it was you who helped us to get away from here and all would be lost if you should show yourself.We are guarded day and night. I do not care you are not here. Do not be troubled on my account. Nothing will happen to me. The national assemble will show leniency. Farewell the most loved of men. Be quiet if you can take care of yourself.For myself I cannot write any more, but nothing in the world could stop me to adore you up to the death.
密钥为:hongye
得到的密文如下图所示

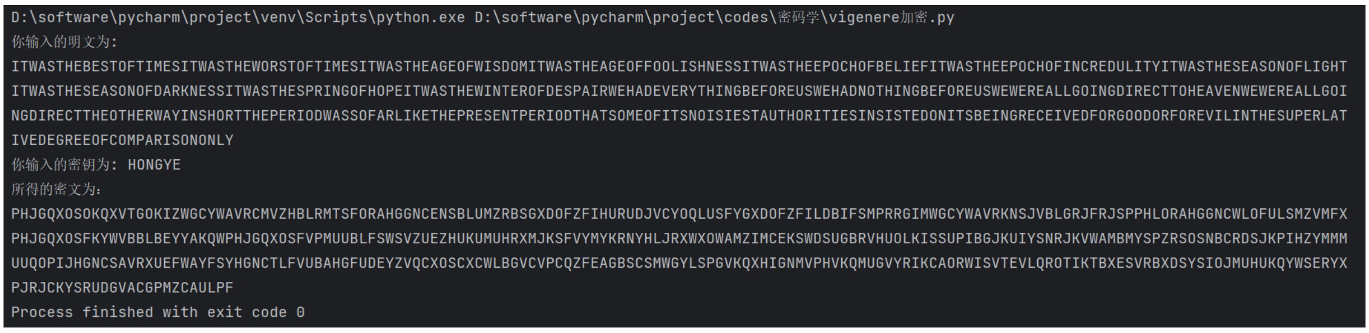


```
D:\software\pycharm\project\venv\Scripts\python.exe D:\software\pycharm\project\codes\密码学\vigenere加密.py
你输入的明文为:
IAMALIVEHEREMYBELOVEDFORTHEREASONTOADOREYOUOHHOWANXIOUSIHAVEBEENFORYOUANDHOWSORRYIAMABOUTALLYOUMUSTHAVESUFFEREDINHAVINGNONEWSFROMUSMAYHEAVENGRANTTHATT
HISLETTERREACHESYOUDONOTWRITETOMETHISWOULDCOMPROMISEALLOFUSANDABOVEALLDONOTRETURNUNDERANYCIRCUMSTANCESITISKNOWNTHATITWASYOUWHOHELPEDUSTOGETAWAYFROMHER
EANDALLWOULDBELOSTIFYOUSHOULDSHOWYOURSELFWEAREGUARDEDDAYANDNIGHTIDONOTCAREYOUARENOTHEREDONOTBETROUBLEDONMYACCOUNTNOTHINGWILLHAPPENTOMETHENATIONALASSEM
BLEWILLSHOWLENIENCYFAREWELLTHEMOSTLOVEDOFMENBEQUIETIFYOUCANTAKECAREOFYOURSELFFORMYSELFICANNOTWRITEANYMOREBUTNOTHINGINTHEWORLDCOULDSTOPMETOADOREYOUUPTO
THEDEATH
你输入的密钥为: HONGYE
所得的密文为:
POZGJMCSUKPITMOKJSCSQLMVAVRXCEZCAZMEKCEKWSBCUNMAHBKOMYZWUGTIISRTDSYMBAYRKVBCQSYFLOYQHPBARESZLUSQBGGNYZLGHLDIYSQOLLHJVTERVBRCQJYCZAQQHMUKYZLBTXYRAHUGRX
OWFRCXASEXCEJVRYWSBRBTMXDFVZCXVARZFMZKBAJHJCZVPSTWFKYPSCSAQEURNHMZLOYRBSUCGXCXBFAALHLFNTWGPFPAKWAOAICWPHVYIRVKAZFEAWGCYWFCHCFSOSYVCHBGGUEIAOJGWJYCZNCV
LOAJYPSKBAJHISYUQXPTLUSWOCHRBWOCJEMYYGRRDALOEKEYHFQKBHHMNTBRPUUZGHVBBZAEYSLUSEYSAURLLFRJMRVHOKRVVIORCHVBZEYGJCHTRRVHUOLKDWYRFEWDRTRSTSGNCRHHVULESOFYCQ
IZRCGPSGUUUPLBVKLGFTNXCALZYZFITCFZJSCSQUDQLBOKOYPSGODCVIPGLXHYRIYVLCSEMYYGRRDJVFZEQISTVIYRUCGCPMASNTWQVFRHSXUCGNGRNWAZFIDCERBGVIYJQXVDZKRSHRBXCCVIHVRS
AVRJCEAV
Process finished with exit code 0
```

图 1: 加密程序运行结果 1

1.3.2 加密过程 2

明文为:It was the best of times.It was the worst of times. It was the age of wisdom. It was the age offoolishness. It was the epoch of belief. It was the epoch of incredulity. It was the season of light. It was the season of darkness. It was the spring of hope. It was the winter of despair. We had everything before us. We had nothing before us. We were all going direct to heaven. We were all going direct the other way. In short, the period was so, far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
密钥为:hongye
得到的密文如下图所示



```
D:\software\pycharm\project\venv\Scripts\python.exe D:\software\pycharm\project\codes\密码学\vigenere加密.py
你输入的明文为:
ITWASTHEBESTOFTIMESITWASTHEWORSTOFTIMESITWASTHEAGEOFWISDOMITWASTHEAGEOFFOOLISHNESSITWASTHEEPOCHOFBELIEFITWASTHEEPOCHOFINCREDULITYITWASTHESEASONOFLIGHT
ITWASTHESEASONOFDARKNESSITWASTHESPRINGOFHOPEITWASTHEWINTEROFDESPAIRWEHADEVERYTHINGBEFOREUSWEHADNOTHINGBEFOREUSWEWEREALLGOINGDIRECTTOHEAVENWEWEREALLGOI
NGDIRECTTHEOTHERWAYINSHORTTHEPERIODWASSOFARLIKETHEPRESENTPERIODTHATSOMEOFITSNOISIESTAUTHORITIESINSISTEDONITSBEINGRECEIVEDFORGOODORFOREVILINTHESUPERLAT
IVEDEGREEOFCOMPARISONONLY
你输入的密钥为: HONGYE
所得的密文为:
PHJGQXOSOKQXVTGOKIZWGCYWAVRCMVZHBLRMTSFORAHGGNCENSBLUMZRBSGXDOFZFIHURUDJVCYOQLUSFYGXDOFZFILDBIFSMPRRGIMWGCYWAVRKNSJVBLGRJFRJSPPHLORAHGGNCWLOFULSMZVMFX
PHJGQXOSFKYWVBBLBEYYAKQWPHJGQXOSFVPMUUBLFSWSVZUEZHUKUMUHRXMJKSFVYMYKRNYHLJRXWXOWAMZIMCEKSWDSUGBRVHUOLKISSUPIBGJKUIYSNRJKVWAMBMYSPZRSOSNBCRDSJKPIHZYMMM
UUQOPIJHGNCSAVRXUEFWAYFSYHGNCTLFVUBAHGFUDEYZVQCXOSCXCWLBGVCVPCQZFEAGBSCSMWGYLSPGVKQXHIGNMVPHVKQMUGVYRIKCAORWISVTEVLQROTIKTBXESVRBXDSYSIOJMUHUKQYWSERYX
PJRJCKYSRUDGVACGPMZCAULPF
Process finished with exit code 0
```

图 2: 加密程序运行结果 2

2 vigenere 密码破译程序

以下将介绍在已知 vigenere 密码的密文的情况下破解其明文以及密钥

2.1 编写思路

2.1.1 通过重合指数法破解密钥的长度

在将密文中的空格去除之后通过 find_key_len 函数遍历从 2 到 50 的所有的密钥长度, 假设当前遍历的密钥长度为 i, 则将密文分成 i 组, 对于每一组调用 count_key_len_CI 函数求其重合指数, 得到密钥长度为 i 时的重合指数, 因为每一个密钥长度对应有数组重合指数, 所以在该次实验中调用 Bias 函数将当前密钥长度的重合指数求其对于 0.065 的偏差, 取偏差最小的一组的密钥长度为该密文的密钥长度。

2.1.2 通过互重合指数确定密钥字符之间的相对位移

在得到了密钥的长度 key_len 之后通过调用 group_k 函数将密钥字符串分成 key_len 组, 通过 count_MIC 函数求得每一组与第一组之间的互重合指数表, 并选取表中互重合指数最接近 0.065 的值的下标作为当前字符相对于第一个字符的偏移量, 从而得到所有字符相对于第一个字符的偏移量。

2.1.3 遍历 26 种密钥字的情况选出合适的密钥字

该过程遍历 26 种密钥的情况, 将密文的前二十位字符拿出用 26 种密钥分别进行解密, 将解密的结果进行分词处理, 由程序员选出最为符合语义的一个密钥并将其输入程序。

2.1.4 将密文字符串通过密钥转换为明文并做英文的分词处理

该步骤是在已知密文已知密钥的情况下将密文字符串进行翻译从而得到明文字符串, 然后将得到的明文字符串进行分词处理, 从而得到一个通顺的句子。

2.2 源代码

```python
import wordninja

letters = ['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']


def c_alpha(cipher):
    """去掉密文中的非字母"""
    cipher_alpha = ''
    for i in range(len(cipher)):
        if (cipher[i].isalpha()):
            cipher_alpha += cipher[i]
    return cipher_alpha


def count_CI(cipher):
    """计算cipher的重合指数"""
    N = [0.0 for i in range(26)]
    cipher = c_alpha(cipher)
    L = len(cipher)
    if cipher == '':
        return 0
    else:
        for i in range(L):  # 计算所有字母的频数，存在数组N当中
            if (cipher[i].islower()):
                N[ord(cipher[i]) - ord('a')] += 1
            else:
                N[ord(cipher[i]) - ord('A')] += 1
    CI_1 = 0
    for i in range(26):
        CI_1 += ((N[i] / L) * ((N[i] - 1) / (L - 1)))
    return CI_1


def Bias(uncip):
    """计算当前key_len的偏差"""
    result = 0.0
    for x in uncip:
        result += pow(abs(0.065 - x), 2)
    result = result / len(uncip)
    return result


def count_key_len_CI(cipher, key_len):
    """计算秘钥长度为 key_len 的重合指数,并返回该组的偏差"""
    un_cip = ['' for i in range(key_len)]  # un_cip 是分组
    aver_CI = 0.0
    count = 0
    for i in range(len(cipher_alpha)):
        z = i % key_len
        un_cip[z] += cipher_alpha[i]

    for i in range(key_len):
        un_cip[i] = count_CI(un_cip[i])
        aver_CI += un_cip[i]

    key_len_bias = Bias(un_cip)
    return key_len_bias

```

```python

def find_key_len(cipher):
    """找出偏差最小的密钥长度并输出"""
    M = [(1, count_CI(cipher))] + [(0, 0.0) for i in range(49)]
    for i in range(2, 50):
        M[i] = (i, count_key_len_CI(cipher, i))

    M = sorted(M, key=lambda x: x[1])  # 按照数组第二个元素排序

    print("密钥长度为: ", M[1][0])
    return M[1][0]


# 密文
cipher = (
    'krkpekmcwxtvknugcmkxfwmgmjvpttuflihcumgxafsdajfupgzzmjlkyykxdvccyqiwdncebwhyjmgkazybtdf'
    'sitncwdnolqiacmchnhwcgxfzlwtxzlvgqecllhimbnudynagrttgiiycmvyyimjzqaxvkcgkgrawxupmjwqemi'
    'ptzrtmqdciakjudnnuadfrimbbuvyaeqwshtpuyqhxvyaeffldmtvrjkpllsxtrlnvkiajfukycvgjgibubldpp'
    'kfpmkkuplafslaqycaigushmqxcityrwukqdftkgrlstncudnnuzteqjrxyafshaqljsljfunhwiqtehncpkgxs'
    'pkfvbstarlsgkxfibffldmerptrqlygxpfrwxtvbdgqkztmtfsqegumcfararhwerchvygczyzjaacgntgvfktm'
    'jvlpmkflpecjqtfdcclbncqwhycccbgeanyciclxncrwxofqieqmcshhdccughsxxvzdnhwtycmcbcrttvmurql'
    'phxnwddkopqtehzapgpfrlkkkcpgadmgxdlrchvygczkerwxyfpawefsawukmefgkmpwqicnhwlnihvycsxckf')


def count_n(c1, c2):  # 确定两个子串最优的相对偏移量n=k1-k2
    n = 0
    mins = 100
    k = [0.0 for i in range(26)]
    for i in range(26):
        k[i] = count_MIC(c1, c2, i)
        # print(i,k[i])
        if (abs(k[i] - 0.065) < mins):
            mins = abs(k[i] - 0.065)
            n = i
    return n


def count_MIC(c1, c2, n):
    """计算c1字符串和c2字符串的互重合指数, n为k1-k2的偏移量"""
    count_1 = [0 for i in range(26)]
    count_2 = [0 for i in range(26)]
    L_1 = len(c1)
    L_2 = len(c2)
    MIC = 0
    for i in range(L_1):
        if (c1[i].isupper()):
            count_1[ord(c1[i]) - ord('A')] += 1
        elif (c1[i].islower()):
            count_1[ord(c1[i]) - ord('a')] += 1
    for i in range(L_2):
        if (c2[i].isupper()):
            count_2[(ord(c2[i]) - ord('A') + n + 26) % 26] += 1
        elif (c2[i].islower()):
            count_2[(ord(c2[i]) - ord('a') + n + 26) % 26] += 1
    for i in range(26):
        MIC += count_1[i] * count_2[i] / (L_1 * L_2)
    return MIC


def group_k(cipher, key_len):
    """完成分组操作并计算每一组与第一组的最优相对偏移量并返回"""
    N = ['' for i in range(key_len)]
    MIC = [0 for i in range(key_len)]
    s = [0 for i in range(key_len)]
    for i in range(len(cipher)):  # 对密文进行分组
        m = i % key_len
        N[m] += cipher[i]
    for i in range(1, key_len):  # 计算与第一组之间的相对偏移量
        s[i] = count_n(N[0], N[i])  # s[i] = k1-k(i+1)
        MIC[i] = count_MIC(N[0], N[i], s[i])  # MIC[i] = MIC(1,i+1)
    return s


def miyao(key_len, s, k):
    """当密钥的第一个字母的下标为k时，输出其对应的密钥，返回密钥"""
    mi = ['' for i in range(key_len)]
    for i in range(key_len):
        mi[i] = letters[(k - s[i] + 26) % 26]
    print("第一个偏移量为%d,密钥为%s时" % (k, mi))
    return mi


def change(cipher, key_len, key):
    """输入密文字符串，密钥长度以及密钥返回明文的翻译结果"""
    plain = ''
    i = 0
    key_number = []  # 代表key的下标
    for x in key:
        key_number.append(ord(x) - ord('A'))
    while (i < len(cipher)):
        for j in range(key_len):
            if (cipher[i].isupper()):
                plain += chr((ord(cipher[i]) - ord('A') - key_number[j] + 26) % 26 + ord('A'))
            else:
                plain += chr((ord(cipher[i]) - ord('a') - key_number[j] + 26) % 26 + ord('a'))
            i += 1
            if (i == len(cipher)):
                break
    return plain


def printf(output):
    """将要输出的字符串每150个一行进行输出"""
    i = 0
    for x in output:
        if i == 150:
            print("")
            i = 0
        print(x, end='')
        i += 1


if __name__ == "__main__":
    # 得到密文的字符串
    cipher_alpha = c_alpha(cipher)

    # 求得密钥的长度
    key_len = find_key_len(cipher_alpha)

    # 求得密钥的每个字符之间的偏移量
    s = group_k(cipher_alpha, key_len)
    print("密钥每个字符之间的偏移量是", s)

    # 将26个不同的密钥的翻译结果分词之后输出
    for k in range(26):
        key = miyao(key_len, s, k)
        plain = change(cipher_alpha, key_len, key)
        word = wordninja.split(plain[0:20])
        word_result = ''
        for i in range(len(word)):
            word_result += word[i]
            word_result += ' '
        print(word_result)

    # 获取最合适的密钥
    k = int(input("请参考上面的输出, 输入符合语义的结果的偏移量"))
    key = miyao(key_len, s, k)
    print("得到的密钥为:", key)

    # 已知密钥, 将密文转换为明文
    plain = change(cipher_alpha, key_len, key)
    word = wordninja.split(plain)
    word_result = ''
    for i in range(len(word)):
        word_result += word[i]
        word_result += ' '

    # 将明文结果输出
    print("翻译之后的明文为:")
    printf(word_result)
```

Listing 2: vigenere 破解程序代码

2.3 代码运行结果演示

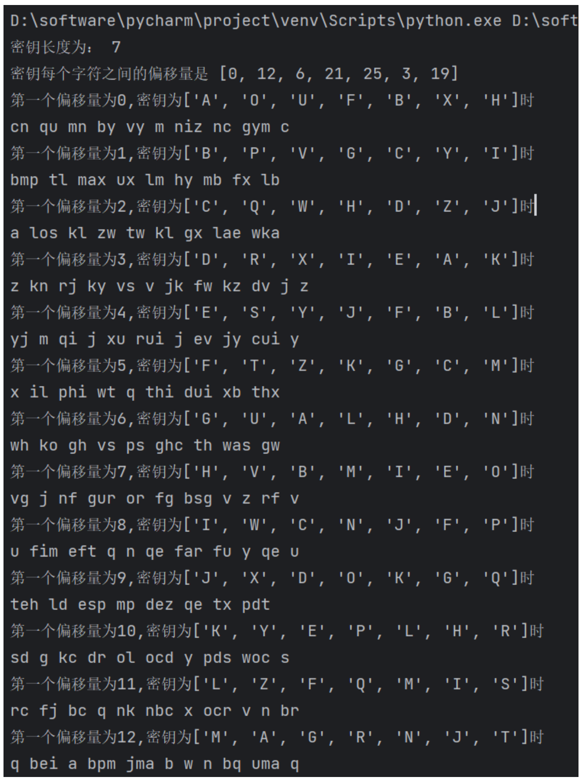
2.3.1 破译过程 1

已知密文为:

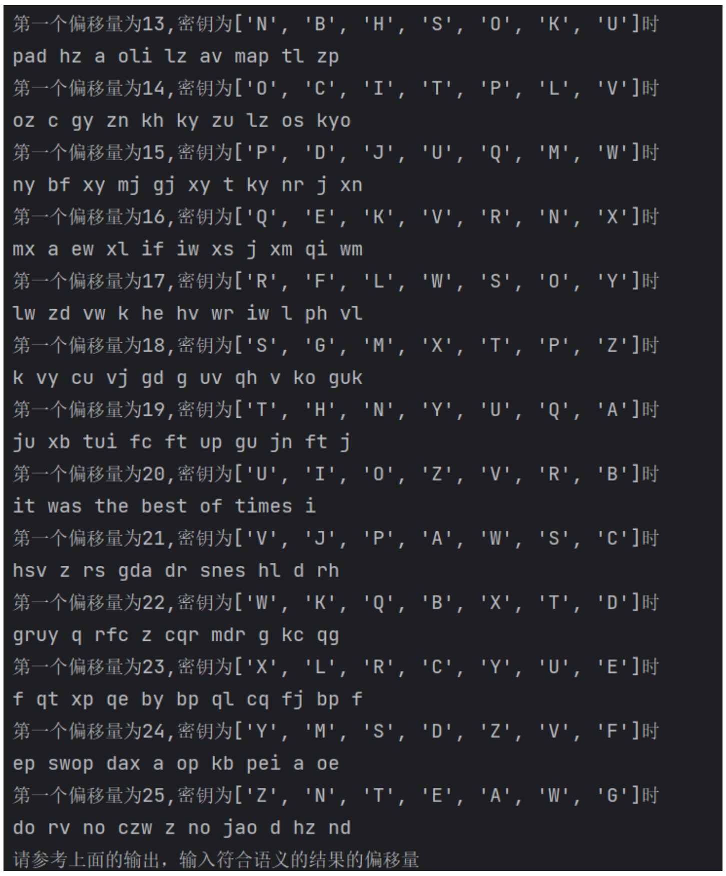
```
        cbkznkiyjsrofgnqadnzuqigscvxizgsjwucusrdkxuahgzrhywtvdjeiuwsrrtnpszbvpzncngztbvsrnzuqigscvf
        jwqgjwcytwdazuqigscvfjwqgjwjhkfdylmcbmhonbmbvdnvbmwbnacjaphhonbmbvdnvbmwbnaublsbdnjjneoroyf
        mxfhixpzpcozzuqigscvxcvhdmfgxmgovzsqmvzyvwyzmsczoajsejifoakdcrehwhgdehvmtnmvvmesvzifutzfjzo
        alwqztunwvdvmfhesvzifutzfjzoalwqztunpsnoyfleoxdetbwfsoyfjmfhjuxuagnarsfqydoyfjzsrzeujmfhjuu
        bihrjdfinwsnepcawdnkbobvnmzucmghijjmbscjejnapddehlmqddmfxncqbfpxwfejifpqzhikiyaiozimubwuzuf
        azsdjwdiudzmztivcmgp

```

破译结果如下图所示

求得密钥长度和偏移量后遍历 26 种可能的密钥结果



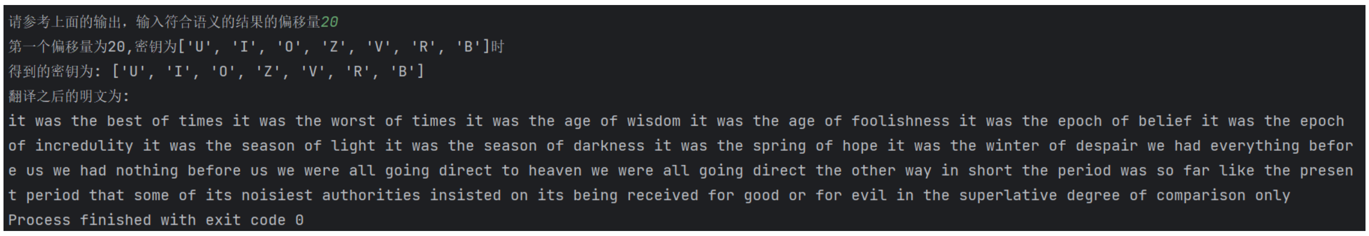
```
D:\software\pycharm\project\venv\Scripts\python.exe D:\soft
密钥长度为： 7
密钥每个字符之间的偏移量是 [0, 12, 6, 21, 25, 3, 19]
第一个偏移量为0,密钥为['A', 'O', 'U', 'F', 'B', 'X', 'H']时
cn qu mn by vy m niz nc gym c
第一个偏移量为1,密钥为['B', 'P', 'V', 'G', 'C', 'Y', 'I']时
bmp tl max ux lm hy mb fx lb
第一个偏移量为2,密钥为['C', 'Q', 'W', 'H', 'D', 'Z', 'J']时
a los kl zw tw kl gx lae wka
第一个偏移量为3,密钥为['D', 'R', 'X', 'I', 'E', 'A', 'K']时
z kn rj ky vs v jk fw kz dv j z
第一个偏移量为4,密钥为['E', 'S', 'Y', 'J', 'F', 'B', 'L']时
yj m qi j xu rui j ev jy cui y
第一个偏移量为5,密钥为['F', 'T', 'Z', 'K', 'G', 'C', 'M']时
x il phi wt q thi dui xb thx
第一个偏移量为6,密钥为['G', 'U', 'A', 'L', 'H', 'D', 'N']时
wh ko gh vs ps ghc th was gw
第一个偏移量为7,密钥为['H', 'V', 'B', 'M', 'I', 'E', 'O']时
vg j nf gur or fg bsg v z rf v
第一个偏移量为8,密钥为['I', 'W', 'C', 'N', 'J', 'F', 'P']时
u fim eft q n qe far fu y qe u
第一个偏移量为9,密钥为['J', 'X', 'D', 'O', 'K', 'G', 'Q']时
teh ld esp mp dez qe tx pdt
第一个偏移量为10,密钥为['K', 'Y', 'E', 'P', 'L', 'H', 'R']时
sd g kc dr ol ocd y pds woc s
第一个偏移量为11,密钥为['L', 'Z', 'F', 'Q', 'M', 'I', 'S']时
rc fj bc q nk nbc x ocr v n br
第一个偏移量为12,密钥为['M', 'A', 'G', 'R', 'N', 'J', 'T']时
q bei a bpm jma b w n bq uma q
```



```
第一个偏移量为13,密钥为['N', 'B', 'H', 'S', 'O', 'K', 'U']时
pad hz a oli lz av map tl zp
第一个偏移量为14,密钥为['O', 'C', 'I', 'T', 'P', 'L', 'V']时
oz c gy zn kh ky zu lz os kyo
第一个偏移量为15,密钥为['P', 'D', 'J', 'U', 'Q', 'M', 'W']时
ny bf xy mj gj xy t ky nr j xn
第一个偏移量为16,密钥为['Q', 'E', 'K', 'V', 'R', 'N', 'X']时
mx a ew xl if iw xs j xm qi wm
第一个偏移量为17,密钥为['R', 'F', 'L', 'W', 'S', 'O', 'Y']时
lw zd vw k he hv wr iw l ph vl
第一个偏移量为18,密钥为['S', 'G', 'M', 'X', 'T', 'P', 'Z']时
k vy cu vj gd g uv qh v ko guk
第一个偏移量为19,密钥为['T', 'H', 'N', 'Y', 'U', 'Q', 'A']时
ju xb tui fc ft up gu jn ft j
第一个偏移量为20,密钥为['U', 'I', 'O', 'Z', 'V', 'R', 'B']时
it was the best of times i
第一个偏移量为21,密钥为['V', 'J', 'P', 'A', 'W', 'S', 'C']时
hsv z rs gda dr snes hl d rh
第一个偏移量为22,密钥为['W', 'K', 'Q', 'B', 'X', 'T', 'D']时
gruy q rfc z cqr mdr g kc qg
第一个偏移量为23,密钥为['X', 'L', 'R', 'C', 'Y', 'U', 'E']时
f qt xp qe by bp ql cq fj bp f
第一个偏移量为24,密钥为['Y', 'M', 'S', 'D', 'Z', 'V', 'F']时
ep swop dax a op kb pei a oe
第一个偏移量为25,密钥为['Z', 'N', 'T', 'E', 'A', 'W', 'G']时
do rv no czw z no jao d hz nd
请参考上面的输出，输入符合语义的结果的偏移量
```

图 3: 密钥长度, 偏移量和 26 种密钥运行结果

选择最为符合语义的密钥结果后输出相应的明文如下图所示



```
请参考上面的输出，输入符合语义的结果的偏移量20
第一个偏移量为20,密钥为['U', 'I', 'O', 'Z', 'V', 'R', 'B']时
得到的密钥为: ['U', 'I', 'O', 'Z', 'V', 'R', 'B']
翻译之后的明文为:
it was the best of times it was the worst of times it was the age of wisdom it was the age of foolishness it was the epoch of belief it was the epoch
of incredulity it was the season of light it was the season of darkness it was the spring of hope it was the winter of despair we had everything befor
e us we had nothing before us we were all going direct to heaven we were all going direct the other way in short the period was so far like the presen
t period that some of its noisiest authorities insisted on its being received for good or for evil in the superlative degree of comparison only
Process finished with exit code 0
```

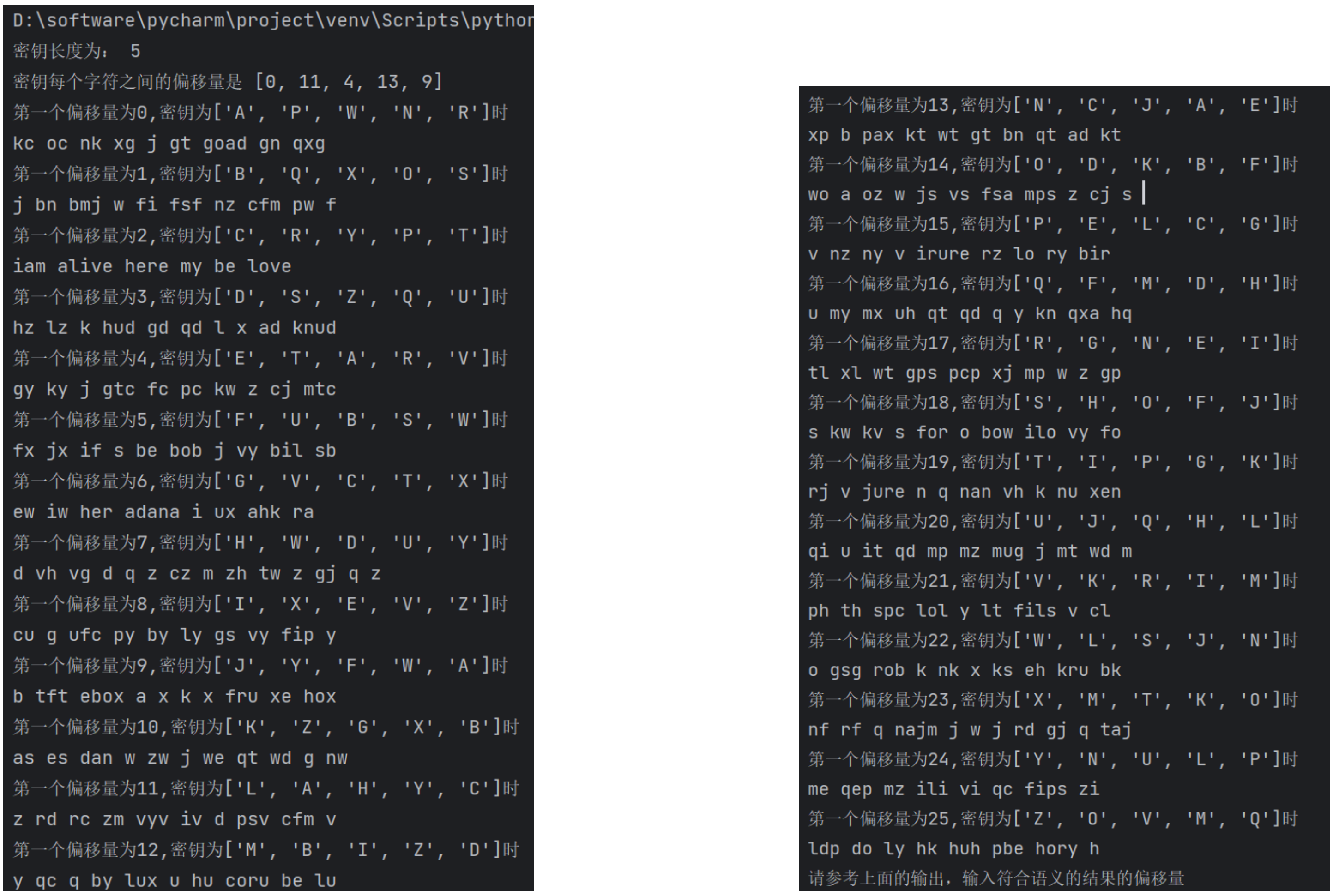
图 4: 得到明文结果

2.3.2 破译过程 2

已知密文为:

```
	krkpekmcwxtvknugcmkxfwmgmjvpttuflihcumgxafsdajfupgzzmjlkyykxdvccyqiwdncebwhyjmgkazybtdf
	sitncwdnolqiacmchnhwcgxfzlwtxzlvgqecllhimbnudynagrttgiiycmvyyimjzqaxvkcgkgrawxupmjwqemi
	ptzrtmqdciakjudnnuadfrimbbuvyaeqwshtpuyqhxvyaeffldmtvrjkpllsxtrlnvkiajfukycvgjgibubldpp
	kfpmkkuplafslaqycaigushmqxcityrwukqdftkgrlstncudnnuzteqjrxyafshaqljsljfunhwiqtehncpkgxs
	pkfvbstarlsgkxfibffldmerptrqlygxpfrwxtvbdgqkztmtfsqegumcfararhwerchvygczyzjaacgntgvfktm
	jvlpmkflpecjqtfdcclbncqwhycccbgeanyciclxncrwxofqieqmcshhdccughsxxvzdnhwtycmcbcrttvmurql
	phxnwddkopqtehzapgpfrlkkkcpgadmgxdlrchvygczkerwxyfpawefsawukmefgkmpwqicnhwlnihvycsxckf

```

破译结果如下图所示求得密钥长度和偏移量后遍历 26 种可能的密钥结果

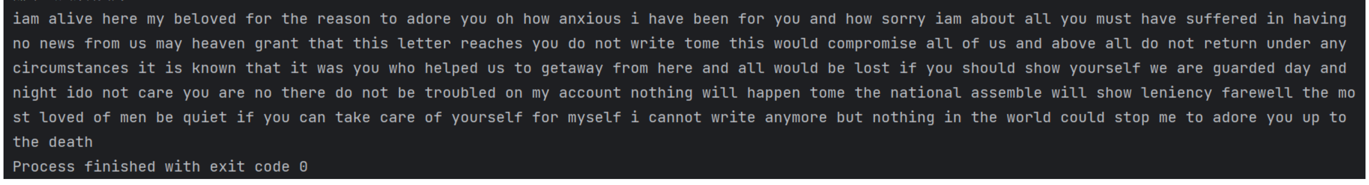


```
D:\software\pycharm\project\venv\Scripts\pythor
密钥长度为： 5
密钥每个字符之间的偏移量是 [0, 11, 4, 13, 9]
第一个偏移量为0,密钥为['A', 'P', 'W', 'N', 'R']时
kc oc nk xg j gt goad gn qxg
第一个偏移量为1,密钥为['B', 'Q', 'X', 'O', 'S']时
j bn bmj w fi fsf nz cfm pw f
第一个偏移量为2,密钥为['C', 'R', 'Y', 'P', 'T']时
iam alive here my be love
第一个偏移量为3,密钥为['D', 'S', 'Z', 'Q', 'U']时
hz lz k hud gd qd l x ad knud
第一个偏移量为4,密钥为['E', 'T', 'A', 'R', 'V']时
gy ky j gtc fc pc kw z cj mtc
第一个偏移量为5,密钥为['F', 'U', 'B', 'S', 'W']时
fx jx if s be bob j vy bil sb
第一个偏移量为6,密钥为['G', 'V', 'C', 'T', 'X']时
ew iw her adana i ux ahk ra
第一个偏移量为7,密钥为['H', 'W', 'D', 'U', 'Y']时
d vh vg d q z cz m zh tw z gj q z
第一个偏移量为8,密钥为['I', 'X', 'E', 'V', 'Z']时
cu g ufc py by ly gs vy fip y
第一个偏移量为9,密钥为['J', 'Y', 'F', 'W', 'A']时
b tft ebox a x k x fru xe hox
第一个偏移量为10,密钥为['K', 'Z', 'G', 'X', 'B']时
as es dan w zw j we qt wd g nw
第一个偏移量为11,密钥为['L', 'A', 'H', 'Y', 'C']时
z rd rc zm vyv iv d psv cfm v
第一个偏移量为12,密钥为['M', 'B', 'I', 'Z', 'D']时
y qc q by lux u hu coru be lu
```

```
第一个偏移量为13,密钥为['N', 'C', 'J', 'A', 'E']时
xp b pax kt wt gt bn qt ad kt
第一个偏移量为14,密钥为['O', 'D', 'K', 'B', 'F']时
wo a oz w js vs fsa mps z cj s
第一个偏移量为15,密钥为['P', 'E', 'L', 'C', 'G']时
v nz ny v irure rz lo ry bir
第一个偏移量为16,密钥为['Q', 'F', 'M', 'D', 'H']时
u my mx uh qt qd q y kn qxa hq
第一个偏移量为17,密钥为['R', 'G', 'N', 'E', 'I']时
tl xl wt gps pcp xj mp w z gp
第一个偏移量为18,密钥为['S', 'H', 'O', 'F', 'J']时
s kw kv s for o bow ilo vy fo
第一个偏移量为19,密钥为['T', 'I', 'P', 'G', 'K']时
rj v jure n q nan vh k nu xen
第一个偏移量为20,密钥为['U', 'J', 'Q', 'H', 'L']时
qi u it qd mp mz mug j mt wd m
第一个偏移量为21,密钥为['V', 'K', 'R', 'I', 'M']时
ph th spc lol y lt fils v cl
第一个偏移量为22,密钥为['W', 'L', 'S', 'J', 'N']时
o gsg rob k nk x ks eh kru bk
第一个偏移量为23,密钥为['X', 'M', 'T', 'K', 'O']时
nf rf q najm j w j rd gj q taj
第一个偏移量为24,密钥为['Y', 'N', 'U', 'L', 'P']时
me qep mz ili vi qc fips zi
第一个偏移量为25,密钥为['Z', 'O', 'V', 'M', 'Q']时
ldp do ly hk huh pbe hory h
请参考上面的输出，输入符合语义的结果的偏移量
```

图 5: 密钥长度, 偏移量和 26 种密钥运行结果

选择最为符合语义的密钥结果后输出相应的明文如下图所示



```
请参考上面的输出，输入符合语义的结果的偏移量2
第一个偏移量为2,密钥为['C', 'R', 'Y', 'P', 'T']时
得到的密钥为: ['C', 'R', 'Y', 'P', 'T']
翻译之后的明文为:
iam alive here my beloved for the reason to adore you oh how anxious i have been for you and how sorry iam about all you must have suffered in having
no news from us may heaven grant that this letter reaches you do not write tome this would compromise all of us and above all do not return under any
circumstances it is known that it was you who helped us to getaway from here and all would be lost if you should show yourself we are guarded day and
night ido not care you are no there do not be troubled on my account nothing will happen tome the national assemble will show leniency farewell the mo
st loved of men be quiet if you can take care of yourself for myself i cannot write anymore but nothing in the world could stop me to adore you up to
the death
Process finished with exit code 0
```

图 6: 得到明文结果

3 作业三

在一次一密中如果去掉密钥全是 0 的情况该算法是否为完全保密, 为什么
答: 如果去掉了密钥是全 0 的情况, 那么在攻击者取得密文的时候会得到信息即明文是不等于自己手中的密文的, 这与完全保密的条件攻击者不能通过密文得到任何有价值的信息相矛盾, 所以说去掉全 0 密钥之后该算法并不会完全保密