brief explaination of quiz4 109550027 紀竺均

1.Using the number of vowels to detect ciphertext rectangles

```
def counting(n,m):
    vowel = [0]*n
    j=0
    for i in text :
        if i != " ":
            if(j==n):
                j=0
            if i=='A'or i=='E' or i=='I' or i=='O' or i=='U':
                vowel[j]+=1
            j+=1

    ans1=0
    for i in range(n):
        ans1+=abs(vowel[i]-m*0.4)

    print(ans1)
    return ans1

n1=7
m1=11
n2=11
m2=7
if counting(n1,m1)>counting(n2,m2):
    n=n2
    m=m2
else:
    n=n1
    m=m1
print(n,m)
```

I use the counting function to determine the matrix's rows and cols. (Same as quiz 2) As a result, this is a 11*7 matrix.

2. Using plaintext bigrams and trigrams to calculate conditional probabilities for Markov decision processing.

```
d3={}
d2={}
def markov():
    plain = plaintextref.translate({ord(c): None for c in string.whitespace})
    #print(plain)

    for i in range(len(plain)-2):
        j=i+1
        k=i+2
        temp=""
        temp+=plain[i]
        temp+=plain[j]
        temp+=plain[k]
        if temp in d3:
            d3[temp]+=1
        else:
            d3[temp]=1

    #print(d3)
    for i in range(len(plain)-1):
        j=i+1
        temp=""
        temp+=plain[i]
        temp+=plain[j]
        if temp in d2:
            d2[temp]+=1
        else:
            d2[temp]=1

    #print(d2)
```

I use the markov function to count the probabilities of 2 characters and 3 characters, then, I save them in dictionary d2 and d3.

```
strcol=[""]*m
ind=0
c=0
for t in text:
    if c==n:
        c=0
        ind+=1
    if t != " ":
        strcol[ind]+=t
        c+=1
for i in range(m):
    print(strcol[i])
firstcol = 2
seccol = 5
ndone = [0,1,2,3,4,5,6]
resultcol=[firstcol,seccol]
ndone.remove(firstcol)
ndone.remove(seccol)
```

In this section, I slice the ciphertext into strings of columns. The result is:

'EOEYEGTRNPS'

'ECEHHETYHSN'

'GNDDDDETOCR'

'AERAEMHTECS'

'EUSIARWKDRI'

'RNYARANUEYI'

'NTTCEIETUS'

Followed the hint, the first column should be 'GNDDDDETOCR' and the second column should be 'RNYARANUEYI'

3. Using MDP to recover columnar transposition ciphers

```
for i in range(2,m):
    maxprob=0
    choosecol=-1
    for j in ndone:
        probs=0
        for w in range(m):
            char2=""
            char2 += strcol[firstcol][w]
            char2 += strcol[seccol][w]

            char3 = char2+strcol[j][w]
            if char2 not in d2 or char3 not in d3:
                continue
            else:
                probs+=math.log(26*(d3[char3]/d2[char2]))
        if probs>maxprob:

            maxprob=probs
            choosecol=j
    ndone.remove(choosecol)
    resultcol.append(choosecol)
    firstcol=seccol
    seccol=choosecol

plaintext=""
for i in range(n):
    for col in resultcol:
        plaintext+=strcol[col][i]
print(plaintext)
```

Every time I want to decide the next column, I run through the undone columns and count the total conditional probabilities of the column.

```
probs += math.log(26*(d3[char3]/d2[char2]))
```

Find the most appropriate column with the highest probability, remove it from the undone columns and redo this to find the next column.

Luckily, I find out the origin plaintext:

```
GREECEANNOUNCEDYESTERDAYITHADREACHEDAGREEMENTWITHTURKEYTOENDTHE
CYPRUSCRISISNS
```