Q1 Commands

5 Points

List the commands used in the game to reach the first ciphertext.

Commands used

● to follow the Trail : **climb**

● to read the written Message : **read**

● to enter the Caves : **enter**

● to read the message written on glass panel : **read**

After executing the above commands in sequence, we reached our firs

Q2 Cryptosystem

5 Points

What cryptosystem was used in this level?

●*Simple* **Substitution Cipher** *was used to* **encode** *the plain text and the te*

● *We used* **Automated Frequency Analysis using Quadgrams** *to* **decod**

Q3 Analysis

25 Points

What tools and observations were used to figure our the cryptosystem? (Explain in less than 100 words)

■**OBSERVATIONS** : –

i. When we first saw the cypher text, we assumed it to be a "caesar cipher" or
a "substitution cipher".

ii. We first tried to break the cipher with all possible 26 shifts of letters
and conclude that it is not breakable with caesar cipher.
Now, we tried Substitution cipher with frequency analysis.

iii. The keen observation we made while performing frequency analysis is that spaces
are shuffled, and words are not bound together.

iv. Instead of manually performing frequency analysis, we came up with the
thought of automating frequency analysis with a program.

v. Instead of considering individual letter frequencies, we considered
the frequencies of Quadgrams (four-letter pairs, such as --'tion','vity').

vi. We collected a Quadgram source file for ENGLISH, in which "Quadgram and its
score", Score tells its frequency in English literature corpus.

vii. The central theme of the algorithm is --> Calculating sum of all possible Quadgram
Scores by making substitution of letters iteratively and the key
with max Quadgrams Score is considered to be optimal.

■**ALGORITHM**:-

1) **Score()** function calculates the sum of Quadgram scores of ciphertext.

2)Calculate the initial Score of ciphertext with **INITIAL_KEY**.

3)Now, Perform a Random Swap of the letter by key and calculate new Ciphertext score of quadgrams by calling Score () function.

4)If the new Score is optimal, then we accept the above swap in key or else we reject the swap.

5)perform Step 3 and Step 4 iteratively around 10^5 iterations.

6)Now, we have the optimal key., and so perform substitution of each letter in ciphertext based on the KEY.

End.

★ **Note**:- Run the program at least 3,4 times to get the preferred answer.

■**Manual corrections** : –

▶ There is a possibility of atmost 1 or 2 individual letters mismatch., which we correct the mismatch Manually.

▶ This is unavoidable because, for small paragraphs, some letters frequency would be too low, and they will not contribute much to score calculation., and might end up in the wrong places.

$\mathbf{Decrypting DIGITS}$:-

- In ciphertext we have =>
  ("substitution cipher in which digits have been shifted by $\mathbf{2}$ places")

- In plaintext =>
  lets say we have digit $\mathbf{"x"}$ in plaintext
  During encryption it becomes "$x + x\ mod 10 = 2$"

  Solution for $\mathbf{x}$ ={1 or 6}

  - Shifting by x = 1 password has failed.
  - Shifting by x = 6 password has $\mathbf{passed}$.

- Hence DIGITS are shifted by $\mathbf{6}$ places.

- NEW PASSWORD = $\mathbf{iRqy3U5qdgt}$.

■**Pros**:

This algorithm works well for any substitution cipher where even if spaces are jumbled, and words are not bound together.

Q4 Mapping

10 Points

What is the plaintext space and ciphertext space?
What is the mapping between the elements of plaintext space and the elements of ciphertext space? (Explain in less than 100 words)

$\underline{\textbf{ciphertextspace}}$ => "omkf pi hdn cmgef icphsck .H krg vphqkc c, fic mco kqgf ioqag eo qfcmckf oq ficpihdn cm .Kg dcgeficu hfcm pi hdn cmklo uuncdgmc oqfc mc kfoq afihqfiokgq c!Fi cpgy cvkc yeg mfio kdck kha cokh kodjuck vn k fofvfo gqpojicmoqli opiyoa of kihsc nccqki oefc ynr2 juhpck. Fi c jhkklgm yok oMxr9V1x ya flofigvffic xvgfck. Fio kokfice"

$\underline{\textbf{plaintextspace}}$ => "irst ch amb eroft hecaves .A syo ucanse e, the rei snot hingo fi nterest in thechamb er .So meofthel ater ch amb erswi llbemore inte re stin gthanthison e!Th ecod euse dfo rthi smes sag eisa simples ub s tituti oncipherinwh ichdig it shave beensh ifte dby6 places. Th e passwor dis iRqy3U5q dg twithoutthe quotes. Thi sisthef"

$\underline{\textbf{The correct interpretation of the plain text is}}$:
" This is the first chamber of the caves. As you can see, there is nothing of interest in the chamber. Some of the later chambers will be more interesting than this one! The code used for this message is a simple substitution cipher in which digits have been shifted by 6 places. The password is iRqy3U5qdgt without the quotes."

$\underline{\mathbf{M}APPING\ BETWEEN\ ELEMENTS}\ (element - wise)$

ciphertext elements = ['c', 'f', 'k', 'o', 'i', 'g', 'h', 'm', 'q', 'p', 'd', 'n', 'v', 'e', 'y', 'a', 'u', 'j', 'l', 'r', 'x', 's', 'b',

't', 'w', 'z', '!' , '.' , ',' , 6, 5, 3]
plaintext elements   = ['e', 't', 's', 'i', 'h', 'o', 'a', 'r', 'n', 'c', 'm', 'b', 'u', 'f', 'd', 'g', 'l', 'p', 'w', 'y', 'q', 'v', 'j', 'y', 'x', 'z', '!' , '.' , ',' , 2, 1, 9]

note:- last 4 alphabets (b,t,w,z) in cipher text elements have "ZERO" occurrence in ciphertext
so their mapping to key does not matter., and
$\underline{Uppercase\ letters\ also\ followed\ same\ mapping\ as\ Lowercase\ letters}.$

•In alphabetical order

CIPHERTEXT to PLAINTEXT Mapping

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
g	j	e	m	f	t	o	a	h	p	s	w	r	b	i	c	n	y	v	x	l	u	k	q	d	z

Q5 Password

5 Points

What is the final command used to clear this level?

iRqy3U5qdgt

Q6 Codes

0 Points

Upload any code that you have used to solve this level english_quadgrams.txt ⬇ Download

```
Large file hidden. You can download it using the button above.
```

ar_assgn1.ipynb ⬇ Download

In [1]:

```
from random import randint
from math import log
```

In [2]:

```
#SCORE FUNCTION

def score():
    sc = 0     #score

    #FOR EACH POSSIBLE QUARDGRAM IN CIPHERTEXT
    for i in range(len(cipher_text)-3) :      #as last three letters
cannot form quadgram
        term = ""

        for j in range(4):    #quadgram_size
            letter = cipher_text[i+j]
            if letter in ind :    #DEALING WITH ONLY ALPHABTES in
ciphertext
                term += key[ind[letter]]             #REPLACING WITH
RESPECTIVE KEY LETTER and CONCATINATING to string "term"

        if term in quad:
            sc += log(quad[term]/n,10)
    return sc
```

In [3]:

```
#CREATING A DICTIONARY FOR QUADGRAMS
quad ={}

#READING FILE
file = open("english_quadgrams.txt")
for i in file:
    a,b = i.split()
    quad[a.lower()] = int(b)

#LENGTH OF THE QUADGRAM DICTIONARY
n = len(quad)
```

```
n = len(quad)
```

In [4]:

```
#CIPHER TEXT
cipher="omkf pi hdn cmgef icphsck .H krg vphqkc c, fic mco kqgf ioqag
eo qfcmckf oq ficpihdn cm .Kg dcgeficu hfcm pi hdn cmklo uuncdgmc
oqfc mc kfoq afihqfiokgq c!Fi cpgy cvkc yeg mfio kdck kha cokh
kodjuck vn k fofvfo gqpojicmoqli opiyoa of kihsc nccqki oefc ynr2
juhpck. Fi c jhkklgm yok oMxr9V1x ya flofigvffic xvgfck. Fio kokfice"
cipher_text=cipher.lower()
r=''
for i in cipher_text:
    if ord(i)>=97 and ord(i)<=122:
        r+=i
cipher_text=r
```

In [5]:

```
alphabets = {}
#ALL ALPHABETS
for i in range(26):
    alphabets[chr(i+97)] = 0

#CALCULATING FREQUENCY OF EACH LETTER IN CIPHERTEXT
for i in cipher_text:
    if ord(i)>=97 and ord(i)<=122:
        alphabets[i]+=1

#converting into a list
alphabets = list(alphabets.items())
#SORTING ON FREQUNCIES
alphabets.sort(reverse=True, key = lambda i:i[1])
```

In [6]:

```
#WE ASSIGN A PRIORITY TO EACH LETTER BASED ON ITS ABOVE FREQUENCIES
(LIKE INDEX)
ind={}
for i in range(len(alphabets)):
    ind[alphabets[i][0]]=i

#ind
```

ALGORITHM STARTS NOW

In [13]:

```
#INITIAL KEY
key='etaoinshrdlucmwfygpbvkxjqz'
key=list(key)

#SCORE
sc = score()

#PERFORMING ITERATIVELY

for _ in range(12000):
    #TWO RANDOM INDICES
    a = randint(0,len(key)-1)
    b = randint(0,len(key)-1)

    #SWAPPING LETTERS IN THE KEY
    key[a],key[b] = key[b],key[a]

    #NEWSCORE
    new_sc=score()

    if new_sc>sc:          #IF OPTIMAL
        sc=new_sc
    else:                  #ELSE REVERSE THE ABOVE SWAP
        key[a],key[b]=key[b],key[a]
```

PERFORMING SUBSTITUTION ON CIPHERTEXT WITH THE KEY

In [14]:

```
plain_text = ""

for i in range(len(cipher)):
    if cipher[i].lower() in ind :        #DEALING WITH ONLY ALPHABTES
in ciphertext
        z=key[ind[cipher[i].lower()]]

        if ord(cipher[i])>=97 and ord(cipher[i])<=122:
            pass
        else:
            z=z.upper()

        plain_text+=z
    else:                           #other characters
        plain_text+=cipher[i]


# key[19] = 'y'
print(key)
print(plain_text)
```

```
['e', 't', 's', 'i', 'h', 'o', 'a', 'r', 'n', 'c', 'm', 'b', 'u', 'f', 'd',
irst ch amb eroft hecaves .A syo ucanse e, the rei snot hingo fi nterest ir
```

In [15]:

```
#  last 4 letters differ in each run  (j,x,y,z)
```

MANUAL CORRECTIONS

In [16]:

```
#REPLACING 19th index("z") to "y"
key[19] = 'y'
```

In [17]:

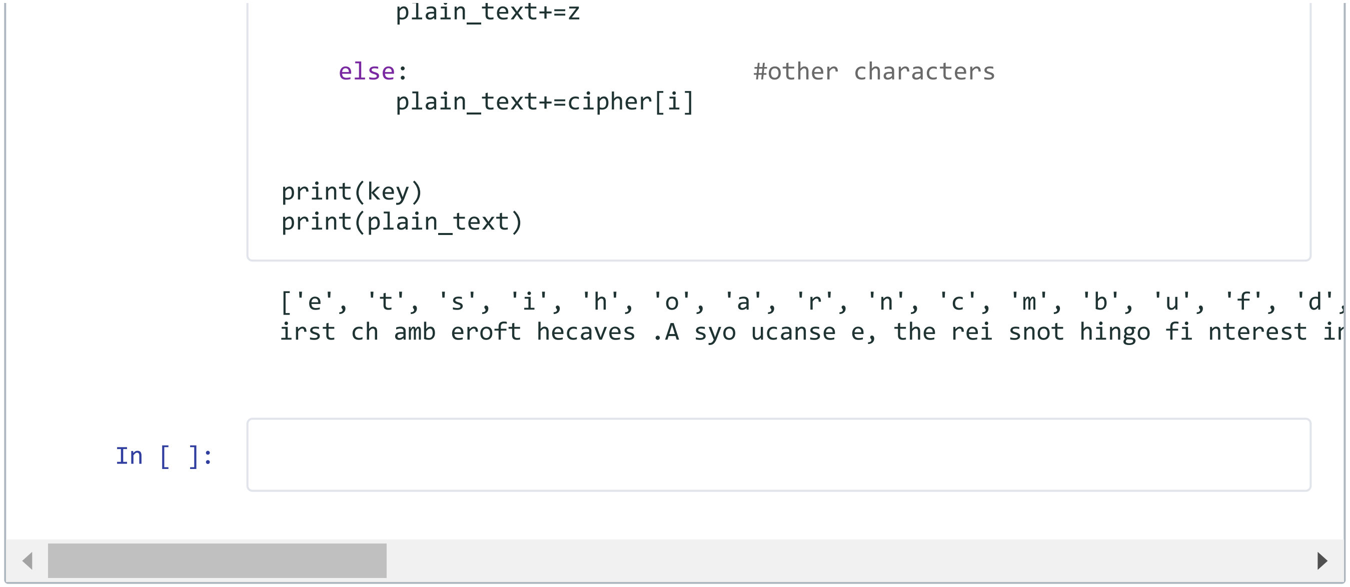
```
#DIGITS ARE MOVED BY 6 places
cipher = cipher.replace('2','6')         #(2-6 = 6)
cipher = cipher.replace('9','3')         #(9-6 = 3)
cipher = cipher.replace('1','5')         #(1-6 = 5)
```

In [18]:

```
plain_text = ""

for i in range(len(cipher)):
    if cipher[i].lower() in ind :       #DEALING WITH ONLY ALPHABTES
in ciphertext
        z=key[ind[cipher[i].lower()]]

        if ord(cipher[i])>=97 and ord(cipher[i])<=122:
            pass
        else:
            z=z.upper()
```



```
        plain_text+=z

    else:                        #other characters
        plain_text+=cipher[i]


print(key)
print(plain_text)
```

```
['e', 't', 's', 'i', 'h', 'o', 'a', 'r', 'n', 'c', 'm', 'b', 'u', 'f', 'd',
irst ch amb eroft hecaves .A syo ucanse e, the rei snot hingo fi nterest ir
```

In [ ]:

Q7 Team Name

0 Points

ANV

Assignment 1

● GRADED

GROUP

Dibbu Amar Raja
Vikas
Idamakanti Venkata Nagarjun Reddy

View or edit group

TOTAL POINTS

50 / 50 pts

QUESTION 1	Commands	**5** / 5 pts
QUESTION 2	Cryptosystem	**5** / 5 pts
QUESTION 3	Analysis	**25** / 25 pts
QUESTION 4	Mapping	**10** / 10 pts
QUESTION 5	Password	**5** / 5 pts
QUESTION 6	Codes	**0** / 0 pts
QUESTION 7	Team Name	**0** / 0 pts