Cryptography (CS362)

Assignment - 1

**U19CS012**

The below cipher text was generated using **Caesar Cipher**.

G JOYGJBGTZGMK UL ZNK MKTKXGR YAHYZOZAZOUT IOVNKX OY ZNGZ HUZN YKTJKX

GTJ XKIKOBKX SAYZ IUSSOZ ZNK VKXSAZKJ IOVNKX YKWAKTIK ZU SKSUXE. G IUSSUT

ZKINTOWAK LUX GBUOJOTM ZNOY OY ZU AYK G QKECUXJ LXUS CNOIN ZNK IOVNKX YKWAKTIK

IGT HK MKTKXGZKJ. LUX KDGSVRK, AYOTM ZNK QKECUXJ IOVNKX, CXOZK UAZ

ZNK QKECUXJ LURRUCKJ HE ATAYKJ RKZZKXY OT TUXSGR UXJKX GTJ SGZIN ZNOY GMGOTYZ

ZNK VRGOTZKDZ RKZZKXY

1. Write a program to perform **Brute Force attack** on the given cipher.

**Input**

**Any Cipher-Text** generated using Caesar Cipher Encryption algorithm.

**Output**

**Key value** using which cipher text was generated.

Also **Decrypted** **Message** is Generated in separate File.

**Note**

* **Automate** the process of identifying the legitimate plaintext generated from each key e.g. assume that the plaintext was **English text**.
* Your program should include **Logic** that can identify English text in the **Brute Force attack**.
* Submit in form of Folder that contain :
* **Source** code
* **Executable** file
* **Steps** to run your program.

**Code**

*"""Code to Decrypt the Text {Encrypted using Caesar Cipher}*

*Arguments: \* Text File - File Containing the Encrypted Text*

*It will generate "output.txt" File containing the Decrypted Text*

*@Author - [U19CS012] {BHAGYA VINOD RANA}*

*"""*

*# For Checking if the Word is Valid in English Dictionary or Not*

*import* enchant

def decrypt(ciphertext, key):

*"""*

*This function acts like a Casear Cipher.*

*Replaces an input string with another string a fixed number of spaces farther down the alphabet*

*Arguments:*

*\* ciphertext (string) - any upper case or lower case letter string*

*\* key (integer) - any integer value to shift to a new letter*

*"""*

    decrypted = ""

*for* c *in* ciphertext:

*if* c.isupper():

            starting\_ascii = ord('A')

*# Calculate the Alphabet Index*

            alpha\_index = ord(c) - starting\_ascii

*# Increment it be 'key Positions'*

            mod\_26 = (alpha\_index + key) % 26

            decrypted += chr(starting\_ascii + mod\_26)

*elif* c.islower():

            starting\_ascii = ord('a')

*# Calculate the Alphabet Index*

            alpha\_index = ord(c) - starting\_ascii

*# Increment it be 'key Positions'*

            mod\_26 = (alpha\_index + key) % 26

            decrypted += chr(starting\_ascii + mod\_26)

*else*:

            decrypted += c

*# raise ValueError('Input is Not a Letter')*

*return* decrypted

def solve():

*"""*

*This will take input [Cipher Text] from input.txt & Give Decrypted Text in 'output.txt'*

*"""*

*# Context Manager 'with' for File Input*

*with* open('input.txt', 'r') *as* f:

*# for English Dictionary*

        d = enchant.Dict("en\_US")

*# Not used f.read()/f.readlines()/f.readline() - to Avoid Running Out of Memory*

*for* cryptic\_text *in* f:

            max\_valid\_token = 0

            final\_plain\_text = ""

            final\_key = 0

*# Brute Force Attack*

*for* i *in* range(0, 26):

                plain\_text = decrypt(cryptic\_text, i)

*# Get Tokens of the Plain Text*

                plain\_txt\_token = plain\_text.split()

*# Count the Number of Valid Tokens in Plain Text*

                valid\_tokens\_cnt = 0

*for* token *in* plain\_txt\_token:

*if* d.check(token) == True:

                        valid\_tokens\_cnt += 1

*# Update if the valid\_token\_count is maximum*

*if*(valid\_tokens\_cnt > max\_valid\_token):

                    final\_plain\_text = plain\_text

                    final\_key = i

                    max\_valid\_token = valid\_tokens\_cnt

*# print("For key {}, Decrypted Text:\n {} \n".format(final\_key, final\_plain\_text))*

*with* open("output.txt", "a") *as* output\_file:

                output\_file.write(final\_plain\_text)

*# Call to main solve() Function*

*if* \_\_name\_\_ == "\_\_main\_\_":

    solve()

Pre-requisites:

* Python3
* Enchant Module

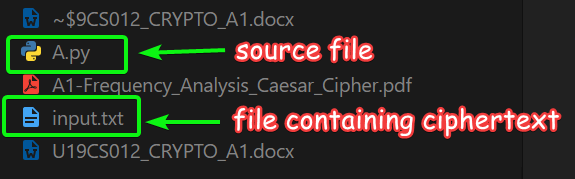
For downloading the **Enchant** Module, use below Command in your Command Line.

pip install pyenchant

**Execution Instruction Steps**

1.) Add the “Encrypted” Cipher Text [Caesar Cipher Algorithm] in input.txt File

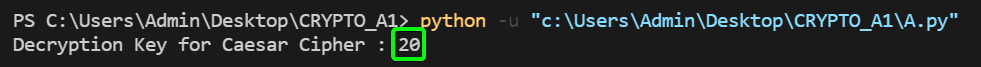
2.) Open Terminal in Folder where Both Source Code {A.py} & input.txt are present.



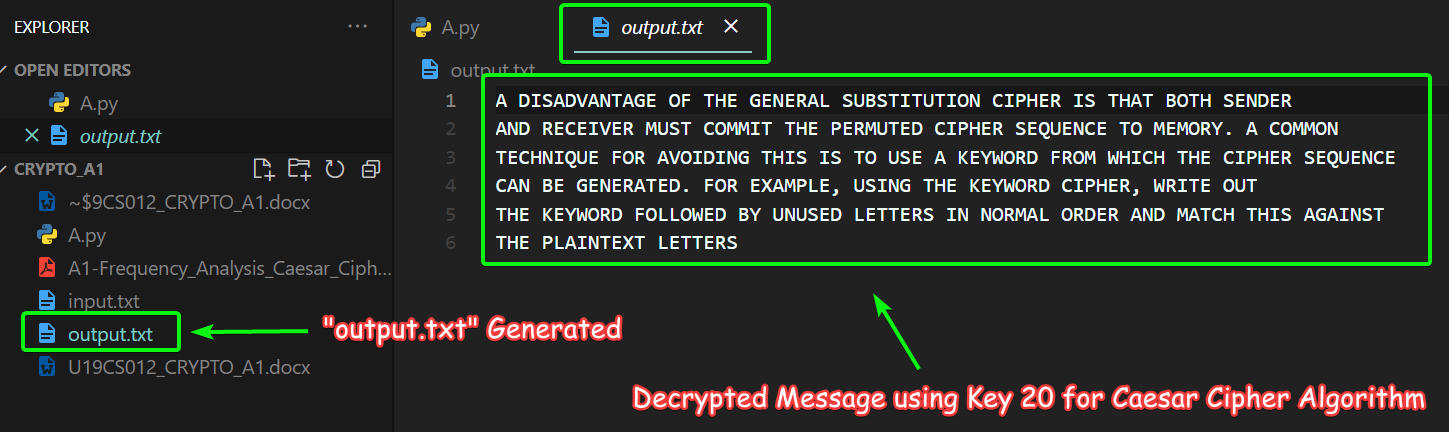
3.) Type the Below Command:

python -u "c:\Users\Admin\Desktop\CRYPTO\_A1\A.py"

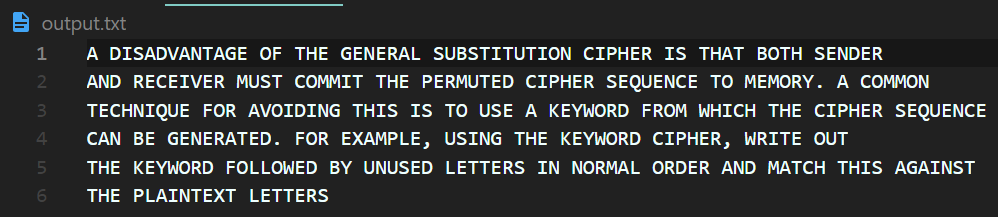
As shown below



“output.txt” Generated



**Output**



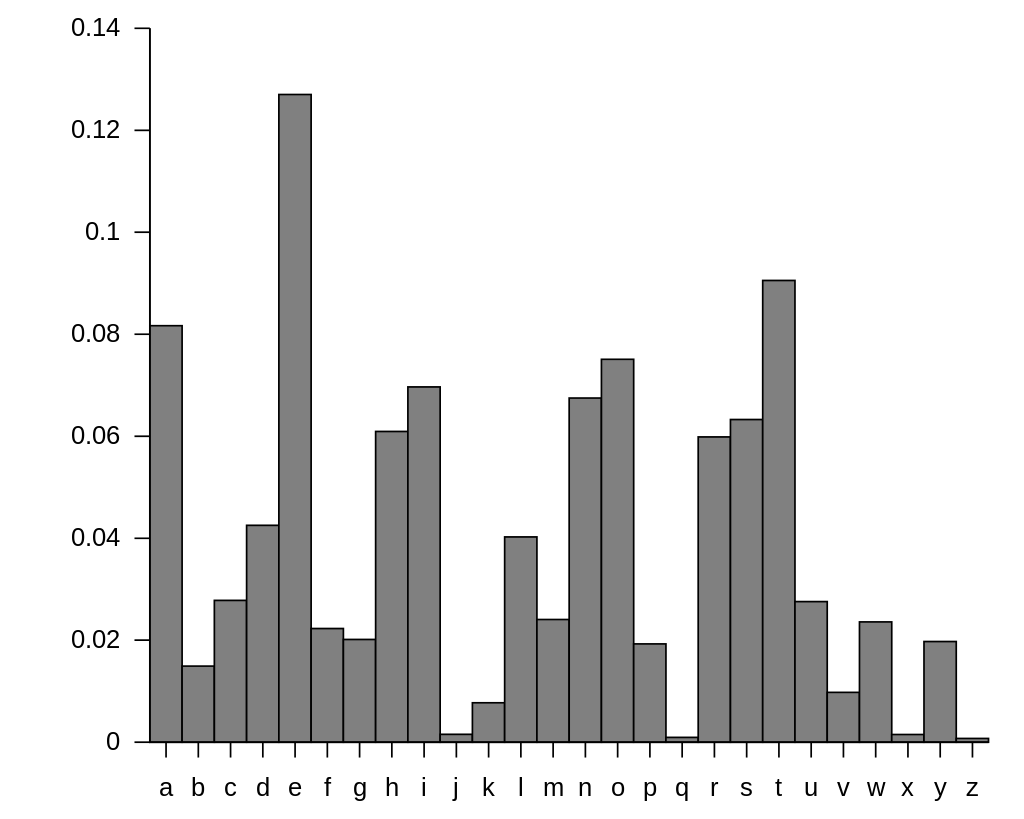
2. To illustrate the use of **Frequency** **analysis** for breaking the cipher.

You can use the program given in below link which will help you to carry out frequency analysis attack for such cipher text produced by a Mono-Alphabetic cipher. [<http://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html>]

Your goal is to find the **Plaintext**, as well as the **key** employed for the above encryption. Clearly explain the **Methodology** **i.e.** how you could break the code **step by step** while performing the frequency analysis.

Frequencies Analysis: using known character frequencies to **decrypt** a cipher

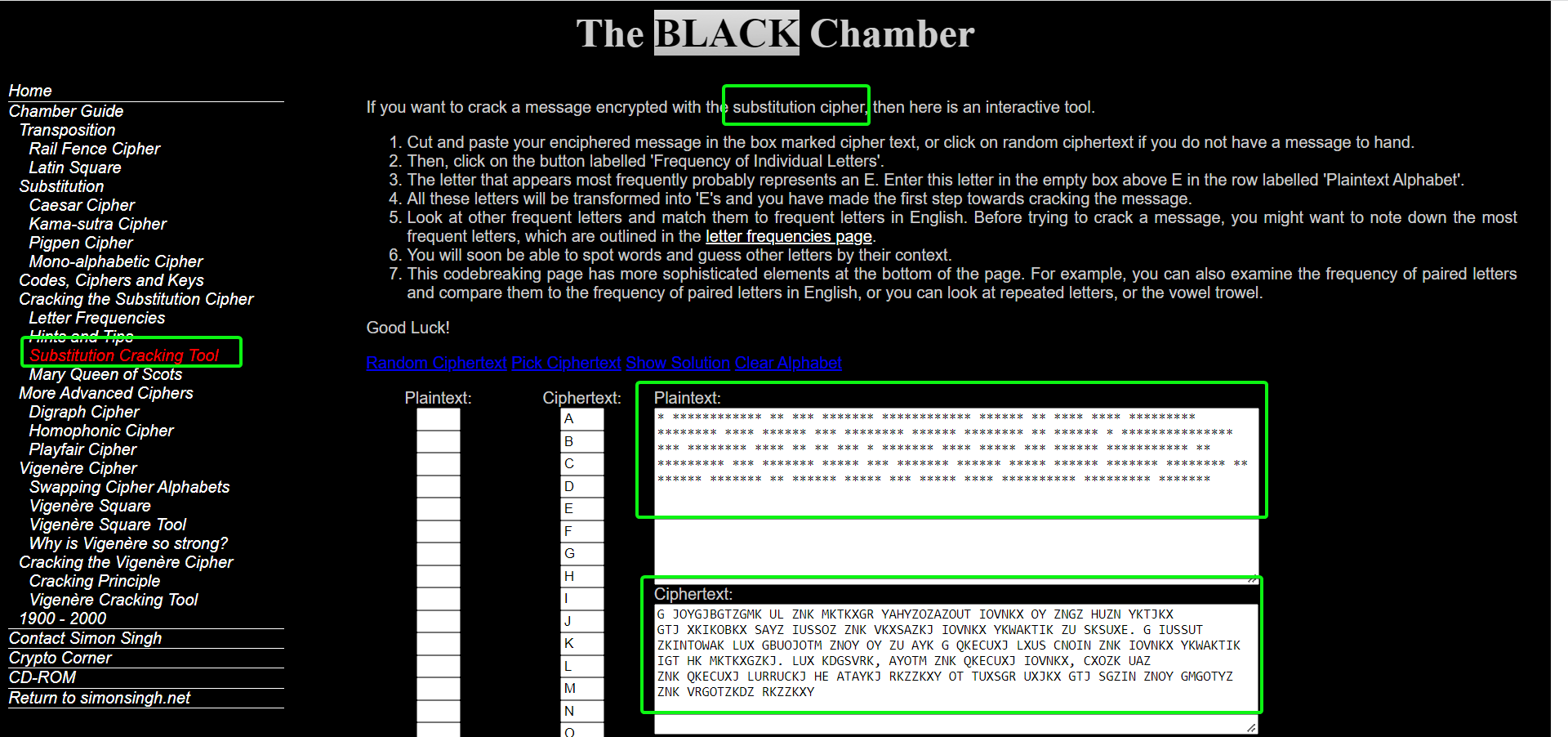
Since, we are working with a Mono-Alphabetic Cipher, we should examine the frequencies of the **letters** of **English** Alphabets.



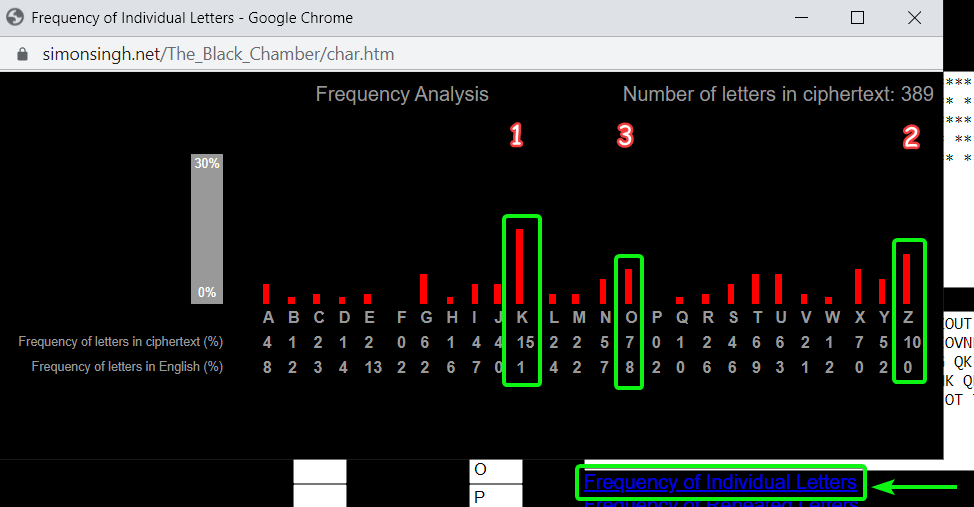
Above is list of average frequencies for letters in the English language. So, for example, the letter **E** accounts for **12.7%** of all letters in English, whereas **Z** accounts for 0.1 %. {**The Average Distribution**}

1.) We will be using the Below Frequency Analysis Tool {since it does not have Ads}

<https://www.simonsingh.net/The_Black_Chamber/substitutioncrackingtool.html>



2.) Watch the Frequency of Individual Letters

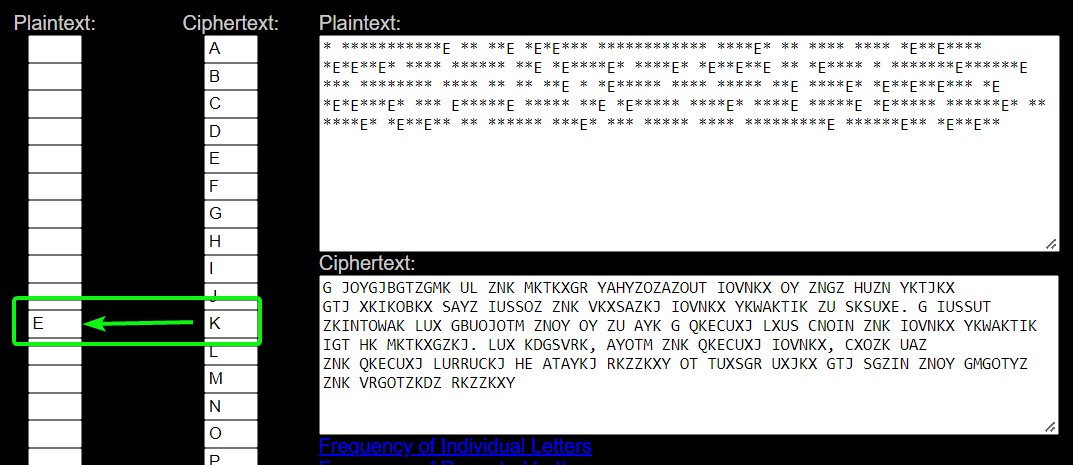


We can observe that Letter “K” has maximum frequency [15], followed by Letter “Z” [10], and then Letter “O”.

Observations:

(a) Since Letter ‘’K” is most Frequent in Given Cipher Text & If we consider Average Distribution of English Alphabet Frequencies, Letter “E” is most Frequent. {It may/may not be Right Substitution}

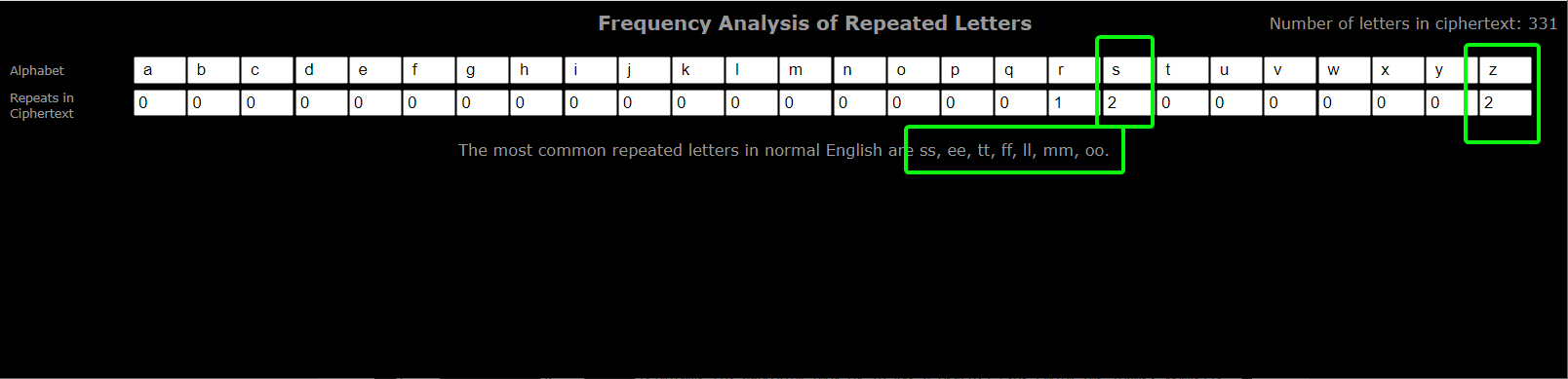
∴ K -> E

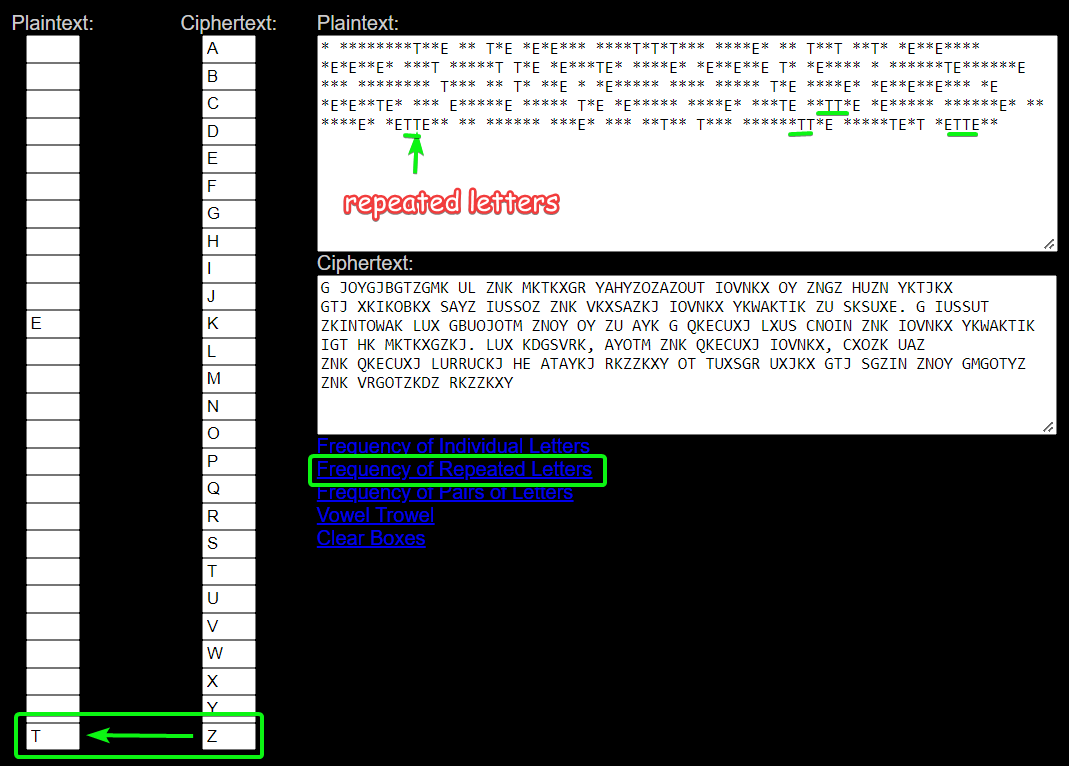


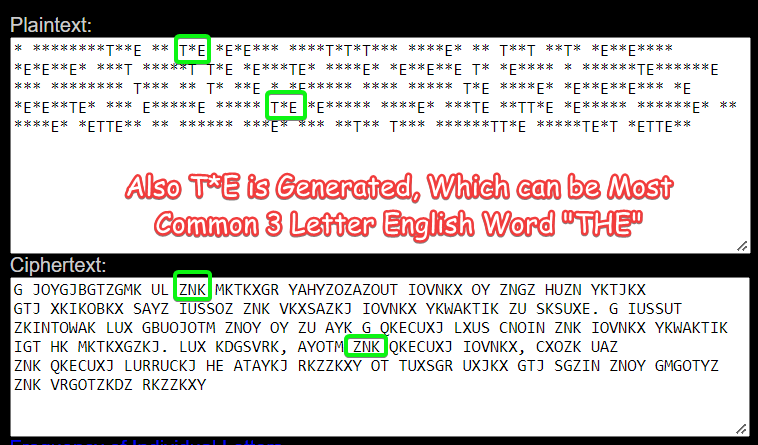
We don’t see any Abnormal English Words being generated by this Substitution.

(b) Since Letter ‘’Z” is **Next** most Frequent in Given Cipher Text & If we consider Average Distribution of English Alphabet Frequencies, Letter “T” is Next most Frequent. {It may/may not be Right Substitution}

∴ Z -> T



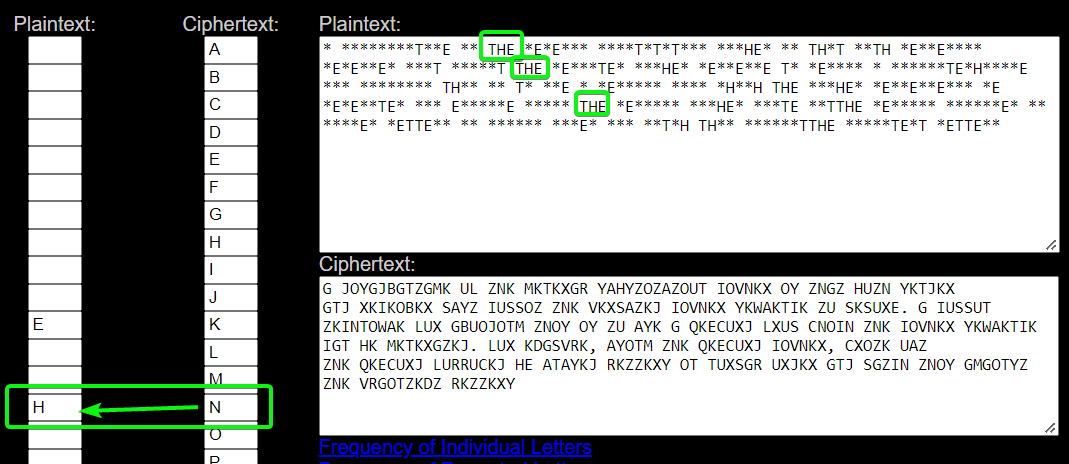




Since, Z -> T, also leads to Most Common Repeated Letters in Normal English & 3 Letter Frequent Word Occurrence, This Substitution is also Fine.

(c) From Observation (b), T\*E [Plaintext] -> ZNK [Cipher],

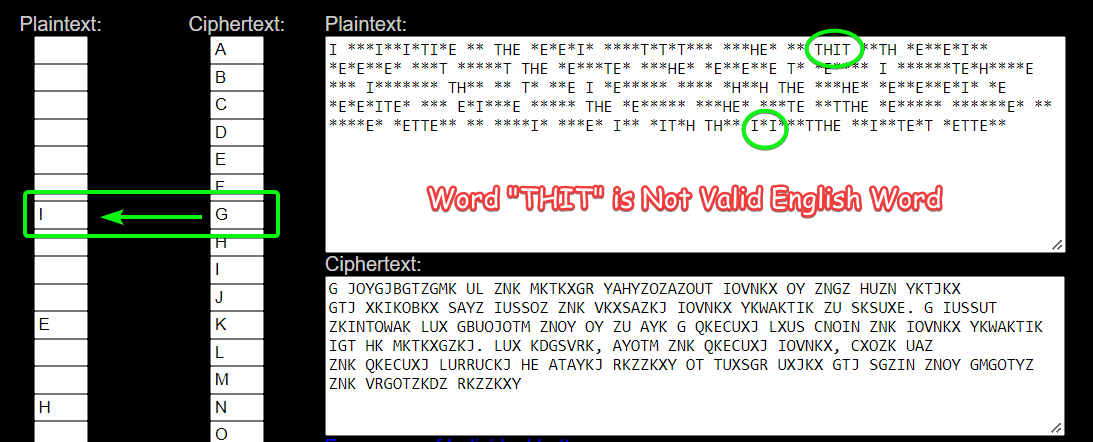
∴ N -> H



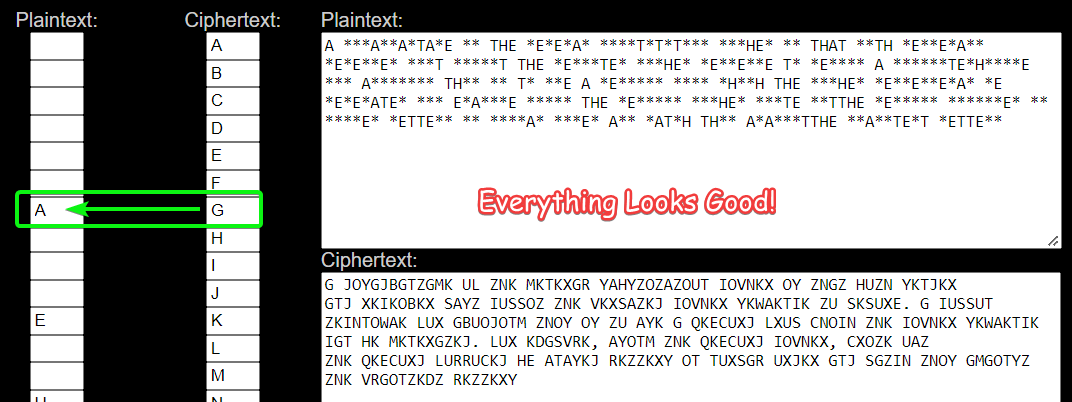
(d) Since One-letter words in English are “A” and “I”,

∴ We can Safely Predict that Letter “G” -> Either “A” or “I”.

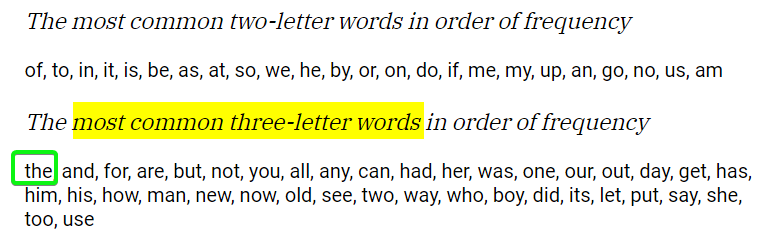
Lets Try G->I



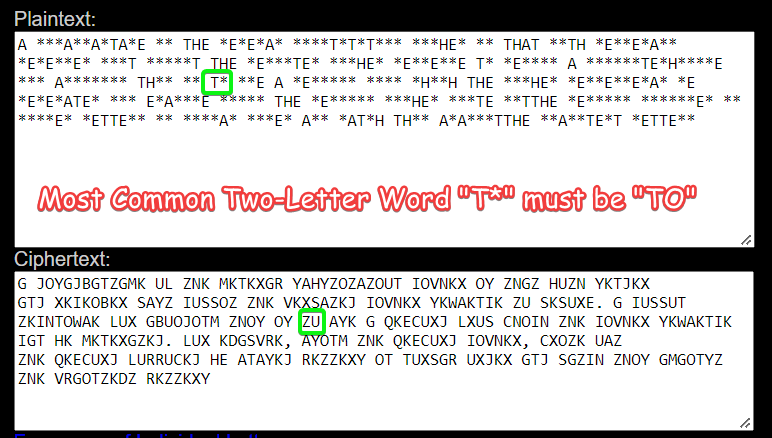
Since, Word “THIT” is Not Valid English Word, ∴ Substituting G->A would Lead to “THAT” which is Valid English Word. ∴ G -> A



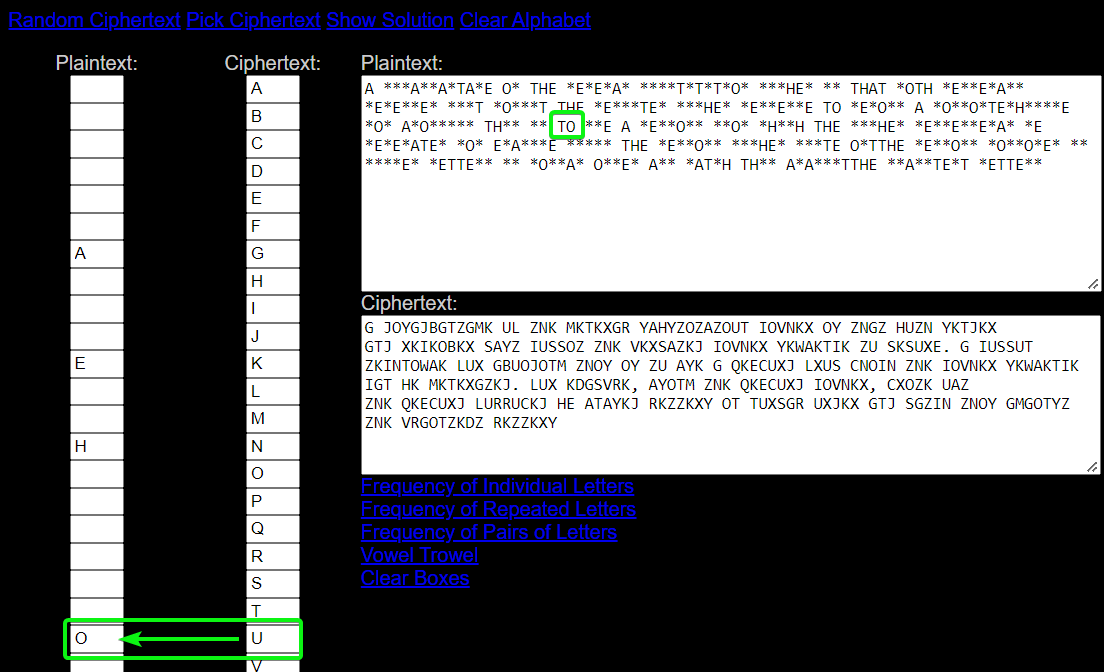
[Most Frequent Substitution - <https://scottbryce.com/cryptograms/stats.html>]



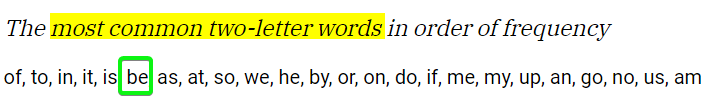
(e) Most Common Two Letter Word Starting with “T” is “TO”



∴ “T**O**” -> “ZU”, U -> O is Valid Substitution.



Similarity, “\*E” -> HK, & Most Common Two Letter Word Ending with E is “BE”.



∴ “**B**E” -> “HK”, H -> B is Valid Substitution.

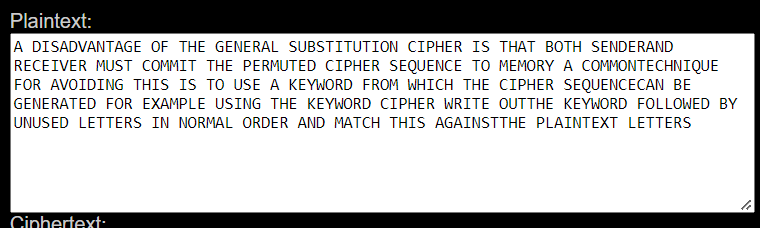
|  |  |  |
| --- | --- | --- |
| Cipher-Letter | Plain-Text | Key {Shift} ( c + key) % 26 |
| K | E | 20 |
| Z | T | 20 |
| N | H | 20 |
| G | A | 20 |
| U | O | 20 |
| H | B | 20 |

Therefore, the Pattern is **Clearly Visible** since its Mono-Alphabetic Caeser.

{We got the Key in First Observation itself, but other observations made first Claim Strong.}



**Message**



After all Substitutions are made, the Cipher Text is **Successfully Decrypted**!

**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA