User Guide

To use tihs program:

1. Change ciphertext in ciphertext.txt to be your texts
2. Under Change Directory and Import Modules, change directory path to where you store this program.
3. Frequency Strips and Second Keyword needs human supervision, and this is the only part in this program that requires human supervision. Allign strips generated by this program, and enter your second keyword

Encryption and decryption details can be found here: Link

Change Directory and Import Modules

In [1]:

```
# change directory to current folder, change this as you run it
%cd /content/drive/My\ Drive/jupyter/autopunc
import decrypt
from split_words import wordninja
from decrypt import vigenere_decryption as vdcp
from decrypt import mono_decryption as mdcp
import re
######## For notebook display only #############
# to disable output wrapping in notebook
from IPython.display import HTML, display
def set_css():
  display(HTML('''
  <style>
    pre {
        white-space: pre;
    }
  </style>
  '''))
get_ipython().events.register('pre_run_cell', set_css)
# to add outputwrap for the plaintext
import textwrap
wrapper = textwrap.TextWrapper(width=80,
    initial_indent=" " * 4,
    subsequent_indent=" " * 4,
    break_long_words=False,
    break_on_hyphens=False)
```

```
[WinError 3] The system cannot find the path specified: '/content/drive/My\\ Drive/jupyter/autopunc'
C:\Users\User\Desktop\Polyalphabetic-Decoder
```

Handling ciphertext

In [2]:

```
f = open("ciphertext.txt", "r")          # ciphertext.txt contains the cipher-text, change the text as you use this program
msg = f.read()
msg = ''.join(filter(str.isalpha, msg))
msg = msg.upper()                      # now msg is a string that consists of only upper case letters
print("ciphertext:\n", msg)
```

```
ciphertext:
 URDPKGDIJRGIWPTZPKVKSZTATGDTLMUOTZSUDBVFLKCQNQHABSSXFQGQXGFJOKKIVRDWGVZIZVTORHWZUKRSHSZTSHSFEAHLIHLRZUABPZUABVKCOPKUWMIRZCMIYUWDPLOFFIZZRKUVLLRDSKBTBAJRMXBQRHXTOKGZIGWRSHLSGBPZBMVOUVLDMGYHHVZIDPTOKDPTUVLPMXVGSOCDTZSKFT
ULZFASZMXQDMAPGZHLFFBAQWLXVNXHWNQDWUVSSOUTOKYDAYIGNKGVIDMZJTPRLFFJARYTTVGJSQYZKHHAGBFDMARFGRZVGSKZWRIRRSLBAJCOPKGJLVNNKFBNIXKFVLVGSOCDTZEUCOVMQXZVTUVLPMXCGNTIRKIYZOTHNNUTUT
```

Calculating Index of Coincidence and Finding the Second Keyword Length

In [3]:

```
second_keyword_length = vdcp().second_key_word_length(msg)
```

```
Suppose keyword length is 4. Frequency of letters:
A 3   B 4   C 1   D 3   E 1   F 4   G 7   H 2   I 4   J 2   K 4   L 5   M 4   N 3   O 4   P 5   Q 2   R 5   S 8   T 6   U 5   V 3   W 5   X 1   Y 1   Z 6
IC4 =  0.03787081843046497
Suppose keyword length is 5. Frequency of letters:
```

```
A 2   B 0   C 2   D 0   E 0   F 0   G 9   H 0   I 2   J 2   K 3   L 7   M 1   N 3   O 6   P 0   Q 6   R 3   S 5   T 0   U 10   V 0   W 0   X 2   Y 1   Z 14
IC5 =  0.08225108225108224
IC5 > IC4, second keyword length is 5
```

Frequency Strips and Second Keyword

This part needs human supervision to find the second keyword. Alligning strips and using it to find the second keyword. Second keyword of my example ciphertext is GRAPH.

In [4]:
```python
vdcp().visualize_strips(msg, second_keyword_length)
```

```
|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  2  |  0  |  2  |  0  |  0  |  0  |  9  |  0  |  2  |  2  |  3  |  7  |  1  |  3  |  6  |  0  |  6  |  3  |  5  |  0  |  1  |  0  |  0  |  2  |  1  |  1  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  0  |     |     |     |     |  4  |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  0  |  3  |  5  |  7  |  0  |  6  |  0  |  1  |  5  |  3  |  8  |  1  |  0  |  0  |  2  |  2  |  0  |  7  |  3  |  0  |  3  |  7  |  5  |  6  |  2  |  2  |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  4  |  1  |  1  |  4  |  1  |  7  |  9  |  6  |  2  |  0  |  5  |  6  |  4  |  0  |  4  |  0  |  1  |  4  |  1  |  1  |  0  |  0  |  3  |  0  |  1  |  2  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  2  |     |     |     |     |     |     |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  2  |  8  |  0  |  6  |  0  |  2  |  2  |  3  |  6  |  2  |  0  |  2  |  0  |  2  |  0  |  1  |  3  |  0  |  7  |  1  |  5  |  7  |  2  |  3  |  0  |  5  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  0  |     |     |     |     |     |     |     |     |     |     |

|  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  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  6  |  1  |  0  |  0  |  1  |  1  |  1  |  5  |  1  |  1  |  6  |  2  |  8  |  5  |  3  |  2  |  0  |  6  |  4  |  1  |  0  |  9  |  0  |  0  |  3  |  3  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  0  |     |     |     |     |     |     |
```

In [5]:
```python
second_keyword = input ("The second keyword found by alligned strips is:")
second_keyword = second_keyword.upper()
```

```
The second keyword found by alligned strips is:graph
```

Vigenere Decryption by the Second Keyword

In [6]:
```python
ciphertext = vdcp().vigenere_decrypt(msg, second_keyword, second_keyword_length)
print("New ciphertext is:\n", ciphertext)
```

```
New ciphertext is:
 OADADAMIUKARWAMTYKGDMITLMAMTWFOXTKLOMBGYFTCBGKQAMLMGFBZKGGQCITKTOLMWROTRZGMIAHHSOTRDAMITDAMOELAFRHWKTDAMITDAMOELOADOFMTKTLMTROFDAEIOFTSTAKFOFUROLEKTMTDAMIUKAHIMITGKBAFRDAFBGMITKMGHOELOFAHHSOTRDAMITDAMOELAFREGDHWMTKLEOT
FETOADSGGQOFUYGKAFOFMTKFLIOHGHHGKMWFOMBOFMITYOTSRGYDAEIOFTSTAKFOFUTLHTEOASSBRTTHSTAKFOFUAFRKTEGDDTFRTKLBLMTDLOADASLGGHTFMGCGKQOFEGDHWMTKXOLOGFKGZGMOELAFRLGYMCAKTRTXTSGHDTFM
```

Trigraph Table

Each trigram in the text is represented as the letter in the middle and its neighbors. For example, GE under A represents the trigram GAE.

In [7]:
```python
print("Trigraph Table:")
mdcp().trigraph_table(ciphertext)
```

```
Trigraph Table:


| 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   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 35  | 8   | 4   | 22  | 14  | 32  | 28  | 16  | 15  | 0   | 25  | 20  | 34  | 0   | 35  | 0   | 4   | 16  | 11  | 46  | 6   | 0   | 7   | 3   | 6   | 3   |
| OD  | MG  | TB  | AA  | OL  | WO  | KD  | AH  | MU  | -   | UA  | TM  | AI  | -   | A   | -   | KA  | AW  | HO  | MY  | IK  | -   | RA  | OT  | TK  | BK  |
| DD  | CG  | QI  | AA  | OL  | YT  | BY  | HS  | MT  | -   | YG  | KO  | AT  | -   | FX  | -   | GC  | WO  | TT  | IL  | FR  | -   | TF  | KO  | GF  | RG  |
| DM  | FZ  | GG  | GM  | AI  | GB  | BK  | RW  | CT  | -   | TL  | MM  | DI  | -   | LM  | -   | GO  | TZ  | HO  | MW  | IK  | -   | MR  | TT  | UG  | GG  |
| KR  | KA  | MA  | RA  | LK  | AR  | MF  | AI  | MA  | -   | GQ  | OM  | LA  | -   | TL  | -   | KO  | TD  | DG  | XK  | FY  | -   | HK  | -   | TO  | -   |
| WM  | FG  | -   | TA  | OL  | OM  | KG  | GO  | MT  | -   | ZG  | EA  | AT  | -   | RT  | -   | -   | FH  | TR  | FC  | FT  | -   | HM  | -   | GD  | -   |
| MM  | MO  | -   | TA  | OL  | OD  | GQ  | AH  | MT  | -   | TT  | EO  | OB  | -   | ST  | -   | -   | TO  | TT  | IK  | FA  | -   | MF  | -   | GM  | -   |
| QM  | SR  | -   | TA  | RG  | OT  | ZM  | HS  | EO  | -   | WT  | TM  | AL  | -   | ME  | -   | -   | UO  | AS  | KO  | -   | -   | HM  | -   | -   | -   |
| IH  | LL  | -   | AO  | LO  | KO  | TK  | DW  | MU  | -   | TT  | OE  | LG  | -   | ME  | -   | -   | FD  | SB  | OR  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | FA  | FT  | OU  | BM  | OG  | HM  | -   | AF  | EO  | LW  | -   | LA  | -   | -   | TD  | HT  | OR  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | TA  | AI  | AR  | MH  | GH  | MT  | -   | ET  | EA  | GI  | -   | DF  | -   | -   | FE  | AL  | ID  | -   | -   | -   | -   | -   | -   |
| LF  | -   | -   | RA  | TO  | AB  | ED  | HG  | MT  | -   | UA  | KE  | AI  | -   | RF  | -   | -   | SG  | TG  | KD  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | RA  | TG  | OA  | SG  | LT  | MT  | -   | GB  | FI  | AO  | -   | IF  | -   | -   | BT  | -   | ID  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | TA  | FG  | AR  | GQ  | TS  | LO  | -   | TM  | TH  | AI  | -   | FF  | -   | -   | FK  | -   | MK  | -   | -   | -   | -   | -   | -   |
| OD  | -   | -   | GH  | OL  | TE  | YK  | GT  | MT  | -   | TL  | KB  | AO  | -   | RL  | -   | -   | FT  | -   | KL  | -   | -   | -   | -   | -   | -   |
| DE  | -   | -   | AS  | -   | OU  | HH  | DW  | EO  | -   | GA  | BM  | FT  | -   | HE  | -   | -   | FL  | -   | MR  | -   | -   | -   | -   | -   | -   |
| TK  | -   | -   | YA  | -   | AO  | HK  | GD  | -   | -   | TF  | DO  | LT  | -   | LF  | -   | -   | TT  | -   | FS  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | GD  | -   | OM  | RY  | -   | -   | -   | GM  | SG  | TT  | -   | ST  | -   | -   | -   | -   | SA  | -   | -   | -   | -   | -   | -   |
| KH  | -   | -   | DT  | -   | KL  | ED  | -   | -   | -   | AF  | OO  | AI  | -   | ME  | -   | -   | -   | -   | KM  | -   | -   | -   | -   | -   | -   |
| BF  | -   | -   | TL  | -   | WO  | LG  | -   | -   | -   | AF  | EA  | II  | -   | ET  | -   | -   | -   | -   | MD  | -   | -   | -   | -   | -   | -   |
| DF  | -   | -   | AA  | -   | OM  | GH  | -   | -   | -   | RT  | RG  | GI  | -   | TA  | -   | -   | -   | -   | IG  | -   | -   | -   | -   | -   | -   |
| FH  | -   | -   | GH  | -   | OT  | MC  | -   | -   | -   | TL  | -   | KG  | -   | QF  | -   | -   | -   | -   | IK  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | HT  | -   | KO  | CK  | -   | -   | -   | GQ  | -   | AI  | -   | FF  | -   | -   | -   | -   | OR  | -   | -   | -   | -   | -   | -   |
| DM  | -   | -   | -   | -   | OU  | ED  | -   | -   | -   | TX  | -   | AO  | -   | IH  | -   | -   | -   | -   | ID  | -   | -   | -   | -   | -   | -   |
| LF  | -   | -   | -   | -   | KO  | OF  | -   | -   | -   | FG  | -   | WT  | -   | FM  | -   | -   | -   | -   | MK  | -   | -   | -   | -   | -   | -   |
| OD  | -   | -   | -   | -   | OU  | KZ  | -   | -   | -   | AT  | -   | FT  | -   | BF  | -   | -   | -   | -   | OF  | -   | -   | -   | -   | -   | -   |
| KF  | -   | -   | -   | -   | AR  | ZM  | -   | -   | -   | -   | -   | KW  | -   | YT  | -   | -   | -   | -   | EO  | -   | -   | -   | -   | -   | -   |
| DE  | -   | -   | -   | -   | TR  | LY  | -   | -   | -   | -   | -   | OB  | -   | IF  | -   | -   | -   | -   | MK  | -   | -   | -   | -   | -   | -   |
| TK  | -   | -   | -   | -   | TM  | SH  | -   | -   | -   | -   | -   | FI  | -   | FF  | -   | -   | -   | -   | IY  | -   | -   | -   | -   | -   | -   |
| OS  | -   | -   | -   | -   | OE  | -   | -   | -   | -   | -   | -   | LT  | -   | EA  | -   | -   | -   | -   | OS  | -   | -   | -   | -   | -   | -   |
| TK  | -   | -   | -   | -   | GK  | -   | -   | -   | -   | -   | -   | FG  | -   | FF  | -   | -   | -   | -   | FS  | -   | -   | -   | -   | -   | -   |
| UF  | -   | -   | -   | -   | AR  | -   | -   | -   | -   | -   | -   | WT  | -   | LA  | -   | -   | -   | -   | SA  | -   | -   | -   | -   | -   | -   |
| OD  | -   | -   | -   | -   | TM  | -   | -   | -   | -   | -   | -   | GO  | -   | QF  | -   | -   | -   | -   | UL  | -   | -   | -   | -   | -   | -   |
| DS  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | YC  | -   | XL  | -   | -   | -   | -   | HE  | -   | -   | -   | -   | -   | -   |
| LF  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | F   | -   | LG  | -   | -   | -   | -   | RT  | -   | -   | -   | -   | -   | -   |
| CK  | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | ME  | -   | -   | -   | -   | TH  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | SA  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | KE  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | DF  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | RK  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | MD  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | HF  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | MK  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | KR  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | RX  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | XS  | -   | -   | -   | -   | -   | -   |
| -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | -   | DF  | -   | -   | -   | -   | -   | -   |
```

Monoalphabetic Decryption by English Trigram Frequencies

The program attempts to decrypt the ciphertext by greedy. This may take a few minutes. Sometimes it is off by just one or two letters, a real human can fix it quickly.

In [8]:
```
# get_key takes 3 arguments: ciphertext, initial_trials, swaptrials; increase trial numbers may increase accurary
key = mdcp().get_key(ciphertext, 50, 2500)
```

In [9]:
```
print("The best key that the program generated is: \n")
print("Plain Alphabet:      ",  "".join(mdcp().alpha))
print("Encryption Alphabet: ",  "".join(key))
```

```
The best key that the program generated is: 

Plain Alphabet:       ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encryption Alphabet:  AZERTYUIOPQSDFGHJKLMWXCVBN
```

In [10]:
```
plaintext =  mdcp().cipher_to_plain_with_key(ciphertext, key)
print("The plaintext decrypted with above key is:\n")
print(plaintext)
```

```
The plaintext decrypted with above key is:

IAMAMATHGRADUATEFROMTHESTATEUNIVERSITYOFNEWYORKATSTONYBROOKWHEREISTUDIEDBOTHAPPLIEDMATHEMATICSANDPUREMATHEMATICSIAMINTERESTEDINMACHINELEARNINGDISCRETEMATHGRAPHTHEORYANDMANYOTHERTOPICSINAPPLIEDMATHEMATICSANDCOMPUTERSCIEN
CEIAMLOOKINGFORANINTERNSHIPOPPORTUNITYINTHEFIELDOFMACHINELEARNINGESPECIALLYDEEPLEARNINGANDRECOMMENDERSYSTEMSIAMALSOOPENTOWORKINCOMPUTERVISIONROBOTICSANDSOFTWAREDEVELOPMENT
```

Word Segmentation

This part uses an open source code wordninja. The segmentation is not perfect, but it is good enough for people to comprehend.

In [11]:
```python
segmented_plaintext = wordninja.split(plaintext)
print("The segmented plaintext is:\n")
print(wrapper.fill(" ".join(segmented_plaintext).lower()))
```

```
The segmented plaintext is:

    iam a math graduate from the state university of new york at stony brook
    where i studied both applied mathematics and pure mathematics iam interested
    in machine learning discrete math graph theory and many other topics in
    applied mathematics and computer science iam looking for an internship
    opportunity in the field of machine learning especially deep learning and
    recommend er system siam also open to work in computer vision robotics and
    software development
```

Sample Decoding With Known Second Keyword

In [16]:
```python
# filename = "ciphertext2.txt"; second keyword = hail
# increase trial numbers in main.py to increase accuracy
# this is the "main.py" script
filename = input("Enter ciphertext txt file name including extension .txt:")
f = open(filename, "r")          # ciphertext.text contains the cipher-text, change the text as you use this program
msg = f.read()
msg = ''.join(filter(str.isalpha, msg))
msg = msg.upper()                 # now msg is a string that consists of only upper case letters
second_keyword = input("Enter second keyword:")
ciphertext = vdcp().vigenere_decrypt(msg, second_keyword, len(second_keyword))
key = mdcp().get_key(ciphertext, 50, 5000)
plaintext =  mdcp().cipher_to_plain_with_key(ciphertext, key)
segmented_plaintext = wordninja.split(plaintext)
print("The best key that the program generated is: \n")
print("Plain Alphabet:      ",  "".join(mdcp().alpha))
print("Encryption Alphabet: ",  "".join(key))
print("The segmented plaintext is:")
print(" ".join(segmented_plaintext).lower())
```

```
Enter ciphertext txt file name including extension .txt:ciphertext2.txt
Enter second keyword:hail
The best key that the program generated is:

Plain Alphabet:       ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encryption Alphabet:  YVZDGJMUFPSINROWBEHLKQCAXT
The segmented plaintext is:
the election of iman katr u wp to succeed her father as the president of the united states has been challenged by w any of the country s leading legal groups they are suspicious of the national mote count vh ich said th
e revere one will ion motes for tru wp and only tv o thousand motes for her opponent oprah v in frey
```

The correct plaintext is:

the election of ivanka trump to succeed her father as the president of the united states has been challenged by many of the country's leading legal groups. they are suspicious of the national vote count which said there were one million votes for trump and only two thousand votes for her opponent oprah winfrey.