Final Project

Polyalphabetic Decoding

AMS 303

Graph Theory

Wenhan Gao

State University of New York at Stony Brook

November 2020

1 Computing I.C.

Counting and computing will be done by python. Codes are attached at the end of this report.
Given text:
305 Gao, Wenhan

```
MSFNI SWDNY SIXCW SYOHE QEDCQ XKLJB SBSZO JSHHM EPXHZ ESSTE
JSXMM CSSRG APHXY FYXLC YSZQC KCLJB DSOLM SSZZF XKPNC RWPXN
AXBNB USOCO XANCT UMGLW AQZZC JPUHP ASXDW EIAJW WLGPF QMRTW
CPNEQ GUAXA UPXDF QXELE JYBBC CSBIO JSZCA KIDXC VWZXB CWLLO
JPNFT CDBIO JSZZR WSXZA XKNQT SWDQC GMZCS WSXBH EKSTO AOOHM
CXRTM ZSXZC EAENC UIXIT CAGPT MWTEY CSRAH IBSCX BSQTA
```

Suppose keyword length 4, frequency of letters:

```
A4 B2 C6 D1 E3 F2 G1 H3 I4 J3 K0 L2 M3
N3 O1 P1 Q4 R1 S7 T4 U2 V0 W4 X5 Y2 Z6
```

By the formula: I.C. $= \frac{1}{N(N-1)} \sum f_i(f_i - 1) = 0.0410958904109589$
Suppose keyword length 5, frequency of letters:

```
A5 B1 C8 D1 E5 F1 G2 H0 I1 J7 K2 L0 M2
N0 O0 P0 Q3 R1 S6 T0 U4 V1 W3 X4 Y1 Z1
```

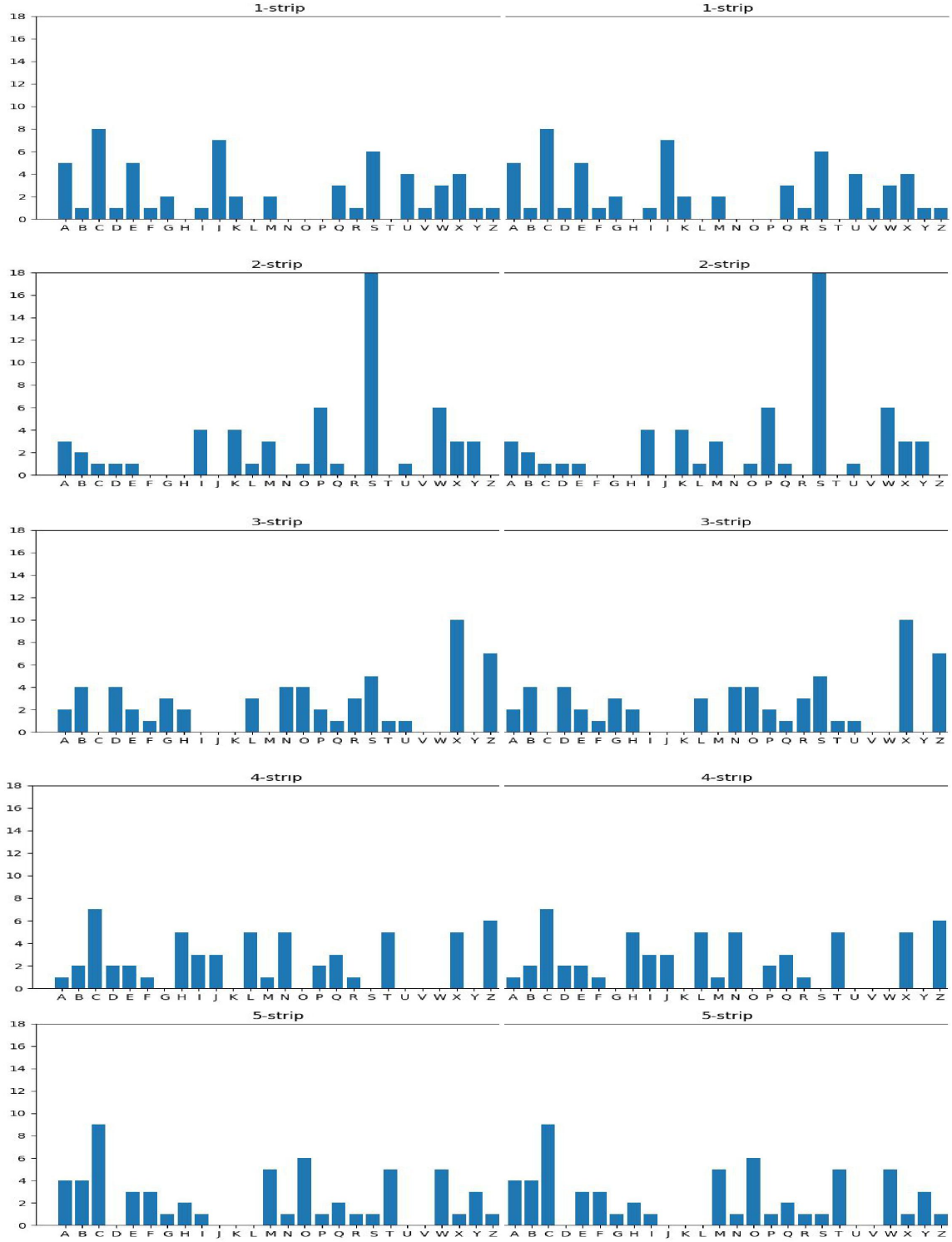
By the formula: I.C. $= \frac{1}{N(N-1)} \sum f_i(f_i - 1) = 0.061367621274108705$

The second one is greater. Therefore, the second key word is of length 5.

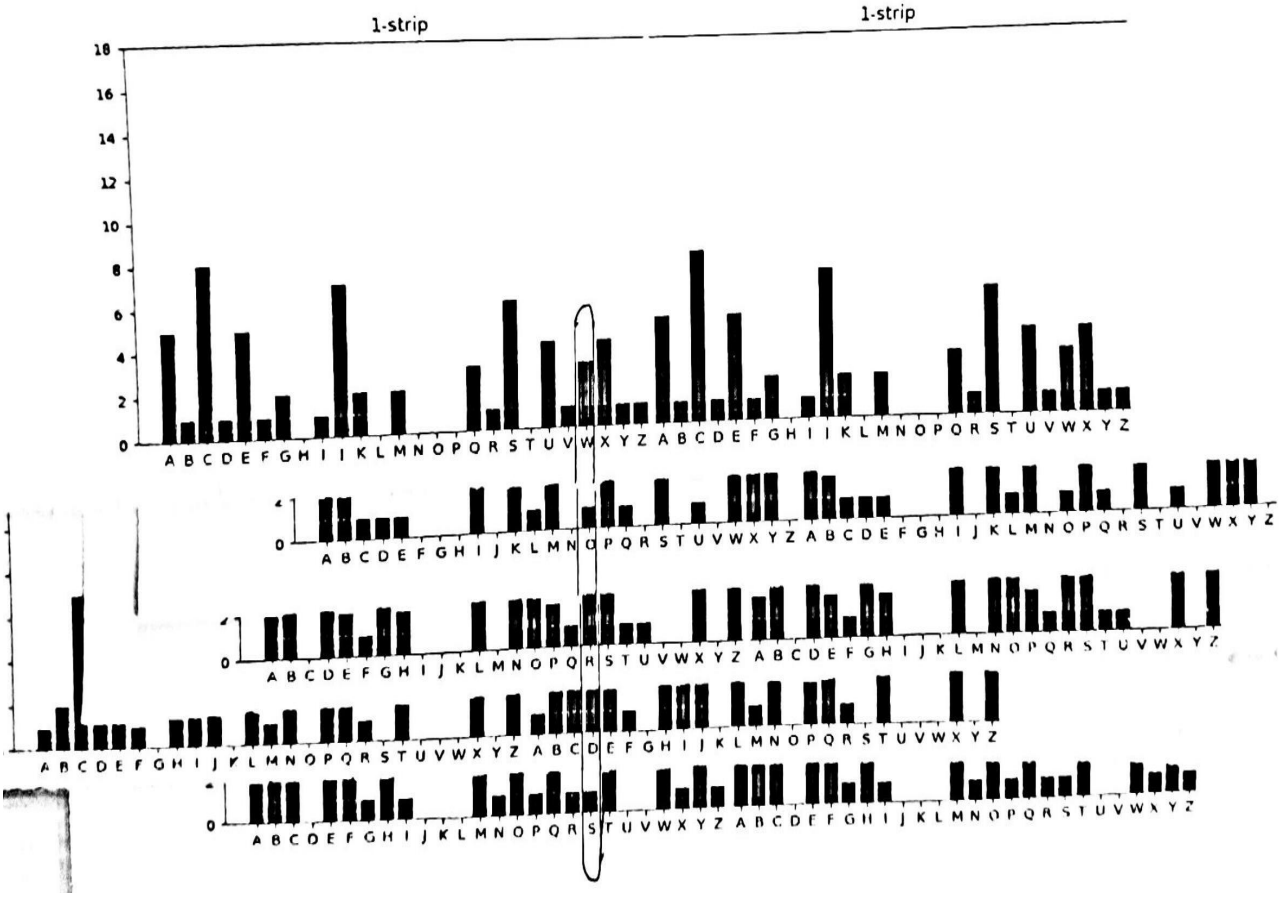
2 Frequency Strips

```
Strip1:|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|
       |5|1|8|1|5|1|2|0|1|7|2|0|2|0|0|0|3|1|6|0|4|1|3|4|1|1|
Strip2:|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|
       |3|2|1|1|1|0|0|0|4|0|4|1|3|0|1|6|1|0|1|0|1|0|6|3|3|0|
Strip3:|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|4|0|4|2|1|3|2|0|0|0|3|0|4|4|2|1|3|5|1|1|0|0|1|0|7|
Strip4:|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|
       |1|2|7|2|2|1|0|5|3|3|0|5|1|5|0|2|3|1|0|5|0|0|0|5|0|6|
Strip5:|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|4|9|0|3|3|1|2|1|0|0|0|5|1|6|1|2|1|1|5|0|0|5|1|3|1|
```

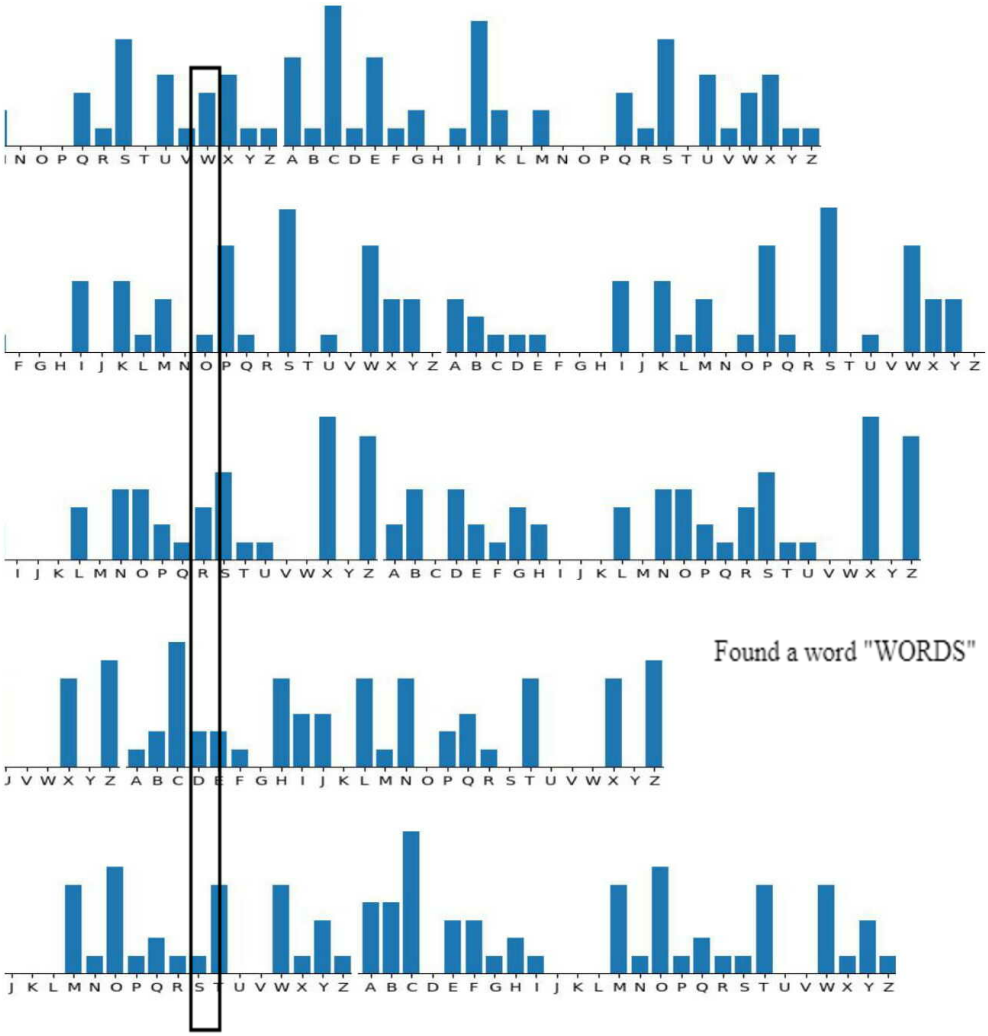
Making histograms:



Aligning strips:




```
Strip1:|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|A|B|C|D|E|F|G|H|I|J|K|L|M|
       |0|0|0|3|1|6|0|4|1|3|4|1|1|5|1|8|1|5|1|2|0|1|7|2|0|2|
Strip2:|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|A|B|C|D|E|
       |0|0|0|4|0|4|1|3|0|1|6|1|0|1|0|1|0|6|3|3|0|3|2|1|1|1|
Strip3:|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|A|B|C|D|E|F|G|H|
       |0|0|0|3|0|4|4|2|1|3|5|1|1|0|0|1|0|7|2|4|0|4|2|1|3|2|
Strip4:|U|V|W|X|Y|Z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|
       |0|0|0|5|0|6|1|2|7|2|2|1|0|5|3|3|0|5|1|5|0|2|3|1|0|5|
Strip5:|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|A|B|C|D|E|F|G|H|I|
       |0|0|0|5|1|6|1|2|1|1|5|0|0|5|1|3|1|4|4|9|0|3|3|1|2|1|
```



Thus, the second keyword is WORDS.

3 Vigenere Decrypt By Second Keyword

Resulting cryptogram by second keyword:

```
QEOKQ WIMKG WUGZE WKXEM UQMZY BWUGJ WNBWW NEQEU IBGEH IEBQM
NEGJU GEBOO EBQUG JKGIK CEINK OOUGJ HEXIU WEIWN BWYKK VIYUV
EJKKJ YEXZW BMWZB YYPIE ECIWK NBDEX EEGAE IUJGE AXPMN UYAQE
GBWBY KGJUI YBGAN UJNIM NKKYK GEKFW NEIZI OUMUK ZIIUJ GIUIW
NBWCB GPKFW NEIWZ AEGWI BWWNB WIMNK KYIZA AEGYP IWBQW EAXEU
GJAQU DEGWK IMNKK YUGFB GMPMB QICBG GEAXP MNBZF FEZQI
```

4 Trigraph Table for Cryptogram

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
10	22	4	2	33	5	25	2	27	11	23	0	13	17	6	6	11	0	0	0	20	2	26	7	13	10
GE	YW	KE	BE	QO	KW	KW	EI	WM	GW	OQ	-	IK	WB	EK	YI	E	-	-	-	WG	KI	QI	KE	ZB	GE
EX	NW	EI	UE	ZW	KW	UZ	JE	UB	GU	MG	-	EU	WE	BO	XM	KW	-	-	-	MQ	UE	GU	EI	WK	MY
YQ	IG	WB	-	XM	GB	UJ	-	HE	GK	WX	-	QZ	ME	OE	GK	UM	-	-	-	WG	-	EK	EZ	IU	XW
GN	EQ	IB	-	NQ	ZF	BE	-	GK	GH	JG	-	QN	IK	KO	YI	EE	-	-	-	EI	-	BU	EE	JE	WB
ZE	EO	-	-	QU	FE	EJ	-	EN	EK	IC	-	BW	WB	OU	MM	BM	-	-	-	JG	-	JN	AP	BY	II
ZA	EQ	-	-	GH	-	UE	-	XU	KY	NO	-	PN	KB	IU	XM	BU	-	-	-	QG	-	BW	AE	YP	KI
AE	NW	-	-	IB	-	UJ	-	EW	UG	YK	-	IN	MU	-	-	AE	-	-	-	OG	-	WN	AP	UA	WA
EX	WM	-	-	NG	-	KI	-	VY	GU	KV	-	UU	AU	-	-	BW	-	-	-	IW	-	UE	-	BK	IA
JQ	ZY	-	-	GB	-	UJ	-	PE	UN	JK	-	IN	JI	-	-	AU	-	-	-	YV	-	IN	-	IB	BF
EX	ND	-	-	OB	-	EA	-	CW	UG	KJ	-	IN	MK	-	-	BI	-	-	-	IJ	-	BY	-	KK	EQ
-	GW	-	-	CI	-	JE	-	EU	GA	WN	-	GP	WE	-	-	ZI	-	-	-	NY	-	ZB	-	KI	-
-	WY	-	-	HX	-	EB	-	UY	-	YG	-	PB	WB	-	-	-	-	-	-	JI	-	MZ	-	GP	-
-	YG	-	-	WI	-	KJ	-	NM	-	NK	-	PN	WE	-	-	-	-	-	-	NJ	-	IK	-	KU	-
-	NW	-	-	VJ	-	BA	-	EZ	-	KY	-	-	WB	-	-	-	-	-	-	OM	-	BB	-	-	-
-	CG	-	-	YX	-	KE	-	ZO	-	YG	-	-	MK	-	-	-	-	-	-	MK	-	FN	-	-	-
-	IW	-	-	IE	-	JI	-	ZI	-	EF	-	-	MK	-	-	-	-	-	-	IJ	-	IN	-	-	-
-	NW	-	-	EC	-	BP	-	IU	-	UZ	-	-	MB	-	-	-	-	-	-	II	-	BC	-	-	-
-	WQ	-	-	DX	-	EW	-	GU	-	PF	-	-	-	-	-	-	-	-	-	EG	-	FN	-	-	-
-	FG	-	-	XE	-	EY	-	UW	-	NK	-	-	-	-	-	-	-	-	-	QD	-	IZ	-	-	-
-	MQ	-	-	EG	-	UJ	-	EW	-	KY	-	-	-	-	-	-	-	-	-	YG	-	GI	-	-	-
-	CG	-	-	AI	-	EW	-	WB	-	WI	-	-	-	-	-	-	-	-	-	-	-	BW	-	-	-
-	NZ	-	-	GA	-	UF	-	WM	-	NK	-	-	-	-	-	-	-	-	-	-	-	WN	-	-	-
-	-	-	-	QG	-	BM	-	YZ	-	KY	-	-	-	-	-	-	-	-	-	-	-	BI	-	-	-
-	-	-	-	GK	-	BG	-	PW	-	-	-	-	-	-	-	-	-	-	-	-	-	IB	-	-	-
-	-	-	-	NI	-	GE	-	KM	-	-	-	-	-	-	-	-	-	-	-	-	-	QE	-	-	-
-	-	-	-	NI	-	-	-	QC	-	-	-	-	-	-	-	-	-	-	-	-	-	GK	-	-	-
-	-	-	-	AG	-	-	-	Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	AG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	WA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	XU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	DG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	GA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	FZ	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1: Trigraph Table

Notice that WNE repeated 3 times and WNBW repeated 4 times. Therefore, we can deduce that WNE is “the” and WNBW is “that”, and from here, we can try to find letter to letter correspondence:

A	E	I	M	Q	U	X
d	e	s	c	r	i	b
B	F	J	N	R	V	Y
a	f	g	h	j	k	l
C	G	K	O	S	W	Z
m	n	o	p	q	t	u
D	H	L	P	T		
v	w	x	y	z		

where the key word is describ(describe).

5 Plain English Text

By above letter to letter correspondence, we can decrypt the cryptogram to plain text:

```
repor tscon tinue tobec ircul ating thatt herei sanew searc
hengi neapp earin gonso mesho pping websi testh atloo kslik
egoog lebut actua llyse emsto haveb eende signe dbych ildre
natal ongis landh ighsc hoolo neoft hesus picio ussig nsist
hatma nyoft hestu dents attha tscho olsud denly start edbei
ngdri vento schoo linfa ncyca rsman nedby chauf feurs
```

And we can slice this text properly to produce English text:

Reports continue to be circulating that there is a new search engine appearing on some shopping websites that looks like google but actually seems to have been designed by children at a Long Island high school. One of the suspicious signs is that many of the students at that school suddenly started being driven to school in fancy cars manned by chauffeurs.

We can also find how the text is encoded:

```
English:  D A M V E F N W S G O X C H P Y R J Q Z I K T B L U
C o d e:  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
Coding1:  W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
Coding2:  O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
Coding3:  R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
Coding4:  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
Coding5:  S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
```

6 Python Code

Following pages are codes of the python program that is been used.

```
 1 from matplotlib import pyplot as plt
 2 import numpy as np
 3 from collections import Counter
 4 from tabulate import tabulate
 5 import itertools
 6 import six
 7
 8 f = open("303.txt", "r")
 9 msg = f.read()
10 msg = msg.replace(" ", "")
11 msg = msg.replace("\n", "")
12
13
14 def find_all(msg, chra):
15     num = msg.find(chra)
16     num_list = [num]
17     while num != -1:
18         num = msg.find(chra, num + 1)
19         num_list.append(num)
20     return num_list[0:len(num_list)-1]
21
22
23 # output a number, the second key word length
24 def second_key_word_length():
25     N = 0; sum = 0; str = ""
26     print("Suppose keyword length 4:")
27     for i in range(0, len(msg), 4):
28         str = str + msg[i]
29     for char in [chr(x) for x in range(ord('A'), ord(
   'Z') + 1)]:
30         lst = find_all(str, char)
31         sum = sum+len(lst)*(len(lst)-1)
32         N = N + len(lst)
33         print(char, len(lst))
34     IC4 = sum / N / (N - 1)
35     print(N)
36     print("IC = ", IC4)
37     N = 0; sum = 0; str = ""
38     print("\nSuppose keyword length 5:")
39     for i in range(0, len(msg), 5):
40         str = str + msg[i]
41     for char in [chr(x) for x in range(ord('A'), ord(
   'Z') + 1)]:
42         lst = find_all(str, char)
```

```
43          sum = sum+len(lst)*(len(lst)-1)
44          N = N + len(lst)
45          print(char, len(lst))
46      IC5 = sum / N / (N - 1)
47      print("IC = ", IC5)
48      if IC5 > IC4:
49          print("Second key word length is 5")
50          return 5
51      else:
52          print("Second key word length is 4")
53          return 4
54
55
56  sec_len = second_key_word_length()
57
58
59  def frequency_strip(string, sec_len):
60      output = []
61      alphab = []
62      for a in [chr(x) for x in range(ord('A'), ord('Z'
    ) + 1)]:
63          alphab.append(a)
64      for j in range(0, 5):
65          sum = 0
66          freq = []
67          print(j+1, "th strip")
68          str1 = ""
69          for i in range(j, len(msg), 5):
70              str1 = str1 + msg[i]
71          for char in [chr(x) for x in range(ord('A'),
    ord('Z') + 1)]:
72              lst = find_all(str1, char)
73              freq.append(len(lst))
74              print(char, len(lst))
75              sum = sum + len(lst)*(len(lst)-1)
76          output.append(freq)
77          print("\n")
78      return alphab, output
79
80
81  alpha_freq = frequency_strip(msg, sec_len)
82  alphab = alpha_freq[0]
83  freq1 = alpha_freq[1][0]
84  freq2 = alpha_freq[1][1]
```

```
 85 freq3 = alpha_freq[1][2]
 86 freq4 = alpha_freq[1][3]
 87 freq5 = alpha_freq[1][4]
 88 nums1 = []
 89 for i in range(0,26):
 90     nums1.append(str(freq1[i]))
 91 nums1 = list(map(list, six.moves.zip_longest(*nums1
    )))
 92 print( "\n"+tabulate(nums1, headers=alphab, tablefmt
    ="github"))
 93 nums2 = []
 94 for i in range(0,26):
 95     nums2.append(str(freq2[i]))
 96 nums2 = list(map(list, six.moves.zip_longest(*nums2
    )))
 97 print( "\n"+tabulate(nums2, headers=alphab, tablefmt
    ="github"))
 98 nums3 = []
 99 for i in range(0,26):
100     nums3.append(str(freq3[i]))
101 nums3 = list(map(list, six.moves.zip_longest(*nums3
    )))
102 print("\n"+ tabulate(nums3, headers=alphab, tablefmt
    ="github"))
103 nums4 = []
104 for i in range(0,26):
105     nums4.append(str(freq4[i]))
106 nums4 = list(map(list, six.moves.zip_longest(*nums4
    )))
107 print("\n"+ tabulate(nums4, headers=alphab, tablefmt
    ="github"))
108 nums5 = []
109 for i in range(0,26):
110     nums5.append(str(freq5[i]))
111 nums5 = list(map(list, six.moves.zip_longest(*nums5
    )))
112 print("\n"+ tabulate(nums5, headers=alphab, tablefmt
    ="github"))
113
114
115 def visualize():
116     plt.bar(alphab, height=freq1)
117     plt.xticks(alphab, alphab)
118     axes = plt.gca()
```

```
119     axes.set_ylim([0, 18])
120     plt.title('1-strip')
121     plt.show()
122     plt.bar(alphab, height=freq2)
123     plt.xticks(alphab, alphab)
124     axes = plt.gca()
125     axes.set_ylim([0, 18])
126     plt.title('2-strip')
127     plt.show()
128     plt.bar(alphab, height=freq3)
129     plt.xticks(alphab, alphab)
130     axes = plt.gca()
131     axes.set_ylim([0, 18])
132     plt.title('3-strip')
133     plt.show()
134     plt.bar(alphab, height=freq4)
135     plt.xticks(alphab, alphab)
136     axes = plt.gca()
137     axes.set_ylim([0, 18])
138     plt.title('4-strip')
139     plt.show()
140     plt.bar(alphab, height=freq5)
141     plt.xticks(alphab, alphab)
142     axes = plt.gca()
143     axes.set_ylim([0, 18])
144     plt.title('5-strip')
145     plt.show()
146     #print(tabulate(freq2, headers=alphab, tablefmt
    ="github"))
147
148 # visualize()
149
150
151
152 def trigraph(msg):
153     output = []
154     for chra in [chr(x) for x in range(ord('A'), ord
    ('Z')+1)]:
155         lst = find_all(msg, chra)
156         neighbors = []
157         end = len(msg)-1
158         for i in lst:
159             if i == 0:
160                 neighbors.append(msg[1])
```

```
161             elif i == end:
162                 neighbors.append(msg[end-1])
163             else:
164                 neighbors.append(msg[i-1]+msg[i+1])
165         output.append((chra, neighbors))
166         print(chra, len(neighbors))
167         a = dict(Counter(neighbors))
168         print(a)
169     return output
170 
171 
172 # input cipher text and key(all cap letter)
173 def vigenere_decrypt(t, key):
174     num0 = ord(key[0]) - 65
175     num1 = ord(key[1]) - 65
176     num2 = ord(key[2]) - 65
177     num3 = ord(key[3]) - 65
178     num4 = ord(key[4]) - 65
179     de_str = ""
180     temp = 0
181     for char in t:
182         number = ord(char) - 65
183         if temp % 5 == 0:
184             number = (number - num0) % 26
185         elif temp % 5 == 1:
186             number = (number - num1) % 26
187         elif temp % 5 == 2:
188             number = (number - num2) % 26
189         elif temp % 5 == 3:
190             number = (number - num3) % 26
191         else:
192             number = (number - num4) % 26
193         letter = chr(number + 65)
194         de_str = de_str + str(letter)
195         temp = temp + 1
196     print(de_str)
197     return de_str
198 
199 
200 #  WORDS is the second keyword
201 chra_nei = trigraph(vigenere_decrypt(msg, "WORDS"))
202 letters = []
203 freq = []
204 counts = []
```

```
205 for i in range(0,26):
206     counts.append(len(chra_nei[i][1]))
207     freq.append(chra_nei[i][1])
208     letters.append(chra_nei[i][0])
209 
210 for j in range(0,26):
211     freq[j].insert(0, len(freq[j]))
212 
213 freq = list(map(list, six.moves.zip_longest(*freq,
    fillvalue='-')))
214 print("\n")
215 print(tabulate(freq, headers=letters, tablefmt="
    github"))
216 
217 
218 
219 
220 
```