CS181HW1
Xiaoming Huang

1.

I use a C++ program to generate the count of each letter, and order them in descending order, the entire program is called "*1.cpp*". And I also try different substitution in the main() function in the program as well as how I determine some possible matching in the text.

Part A: Counting

count of each letter from A to Z:

6 0 2 2 6 15 10 5 18 0 14 15 1 0 14 0 18 13 11 5 9 4 35 0 6 21

Sort the letters in descending order:

W Z I Q F L K O R S G U E A Y T H V C D M J N X P B

English frequency ranking used:

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

Part B: Decryption

Test 1: Try matching W–Z–I with e–t–a:

tae VQKD LAFLaOFe YOSteKeR taKGAUa tae SeQCeL GY tae tKeeL EQLtOFU RQHHSeR LaQRGVL GF tae UKGAFR TeSGV TOKRL EaOKHeR QFR LQFU OF tae TKQFEaeL QRROFU tG tae HeQEeYAS QDTOQFEe GY tae YGKeLt Q SOUat TKeeMe KALtSeR tae SeQCeL EKeQtOFU Q UeFtSe VaOLHeKOFU LGAFR taQt YOSSeR tae QOK

Note tae is not a word, so test 1 fails

Test 2: tae is not a word, but the is, so try on that

Try matching W–Z–I ---> e–t–h:

the VQKD LAFLhOFe YOSteKeR thKGAUh the SeQCeL GY the tKeeL EQLtOFU RQHHSeR LhQRGVL GF the UKGAFR TeSGV TOKRL EhOKHeR QFR LQFU OF the TKQFEheL QRROFU tG the HeQEeYAS QDTOQFEe GY the YGKeLt Q SOUht TKeeMe KALtSeR the SeQCeL EKeQtOFU Q UeFtSe VhOLHeKOFU LGAFR thQt YOSSeR the QOK

And keep guessing word based on this, process of my guesses(output of each step presented in the cpp program):

- Match Q to the first not used word in EngFreq, which is a;
- some 2-letter word of G*, guess one of it to be "of"
- some 2-letter word of G*, guess one of it to be "of"

- "thKoAUh" --> through
- "oY the treeL" --> "of the trees"
- "SeaCes of the trees" --> "leaves of the trees"
- another common o* is on, so try "oF" with "on"
- "on the grounR" --> "on the ground"
- "sunshOne fOltered through the leaves" --> "sunshine filtered through the leaves"
- "shadoVs on the ground" --> "shadows on the ground"
- "Telow Tirds" --> "below birds"
- "warD sunshine" --> "warm sunshine"
- "sang in the branEhes" --> "sang in the branches"
- "the Heaceful ambiance" --> "the peachful ambiance"
- "breeMe" --> "breeze"

Part C: Results

Plaintext:

the warm sunshine filtered through the leaves of the trees casting dappled shadows on the ground below birds chirped and sang in the branches adding to the peaceful ambiance of the forest a light breeze rustled the leaves creating a gentle whispering sound that filled the air

2.

I wrote a C++ program which has:

- **readFile() function to read text from a txt file**
- **encrypt()**
- **decrypt()**

And tested with the class example

the code is include in “*2.cpp*”

here is a screenshot of the output of my program in terminal:

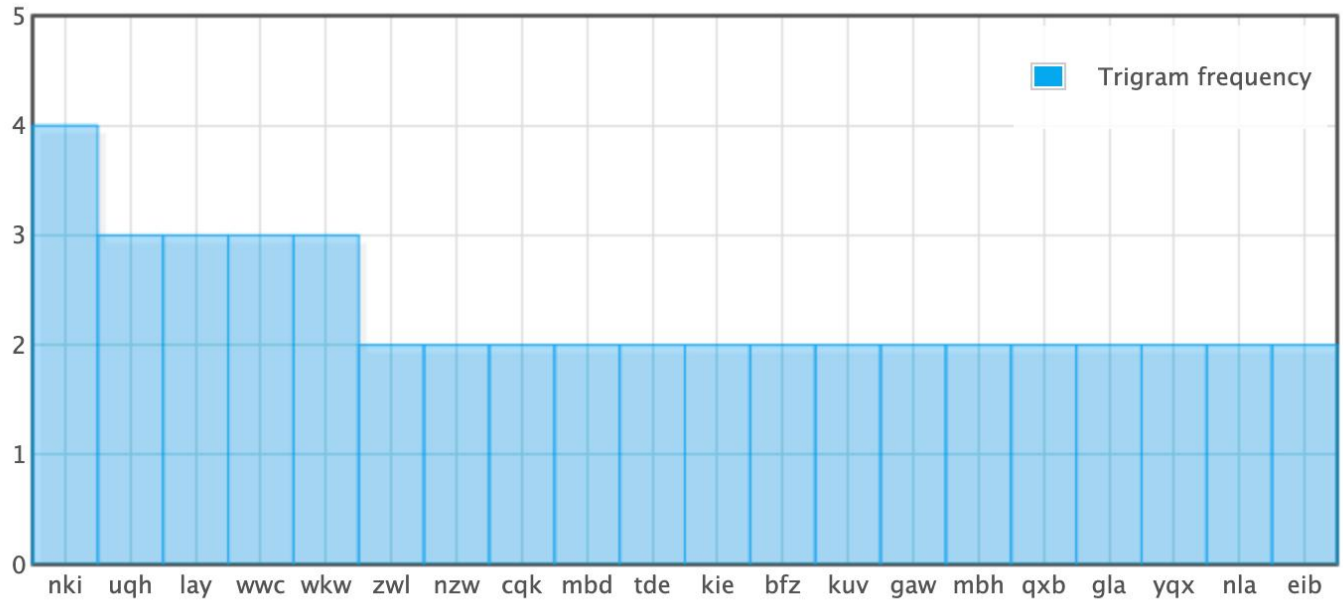
```
Sample ciphertext we use:
wzggqbuawqpvhveirrbvnysttaknkenxosavvwfwfrvxqumhuwwqgwtgziihlocgpnhjmnnmtzqboavv
abcuawohbvrjtampovklgpigfsmfmwvnniyhzyrvqkkiqywwehvjrjwgwewgzhcxucakepwpsnjhvama
hkmehnhuwwvtzguwaclzstsvfxlplzmuywzygagkaofkioblwiargtvrgzitxeofswcrqbtllcmiabfk
ttbwbfenvzsnlytxahuwvgtzstghutvrzwrcglprariltwxwtampotgvwlqhvkhkynwpmpvmwgbjxqnb
tnuxhkwasagvbwbntswmpwfdmhxncezinbdsqarvaihojmneqoalfwmpomqdqgmkuwvgfghusrfaqggg
vavwzyahggwbrgjjbakeaxkgovnkwwkdwiwhdnboaumggbgbmvexaoogypwewgzvgymfrfgglbcuaq


Sample decryption using sample ciphertext:
eveninhisowntimepythagoraswasalegendrumoredtobethesonofthegodapollobyavirginbirt
htohismotherpythaishewassaidtohaveworkedmiraclesconversedwithdaemonsandheardthem
usicofthestarshewasregardedbyhisfollowersassemidivineandtherewasasayingthatamong
rationalcreaturestherearegodsandmenandbeingslikepythagorusitisdifficulttosortout
factfromfictionabouthislifeforhelivedinthatbrilliantbuthazyzonewheremythandhisto
rycollidenoneofhiswritinghassurvivedbutancientsourcesaboundwithreferencestohim


Sample encryption using the plaintext from above:
wzggqbuawqpvhveirrbvnysttaknkenxosavvwfwfrvxqumhuwwqgwtgziihlocgpnhjmnnmtzqboavv
abcuawohbvrjtampovklgpigfsmfmwvnniyhzyrvqkkiqywwehvjrjwgwewgzhcxucakepwpsnjhvama
hkmehnhuwwvtzguwaclzstsvfxlplzmuywzygagkaofkioblwiargtvrgzitxeofswcrqbtllcmiabfk
ttbwbfenvzsnlytxahuwvgtzstghutvrzwrcglprariltwxwtampotgvwlqhvkhkynwpmpvmwgbjxqnb
tnuxhkwasagvbwbntswmpwfdmhxncezinbdsqarvaihojmneqoalfwmpomqdqgmkuwvgfghusrfaqggg
vavwzyahggwbrgjjbakeaxkgovnkwwkdwiwhdnboaumggbgbmvexaoogypwewgzvgymfrfgglbcuaq
```

3.

Part A: Finding keylength

lpiusnw kwcyiez wwcqelx deizabh vthgwbk jrcfshe vgiwxet nxfesga sgxeibg wmldzre wkfrrtb
kmirhng veirorw vhqqsax slzdvnl avixpqm gpbhvrb luyqxvg layxrqx jzlraga layqxbh cmbhsga
wkuvnhl ltmievk sgxkeib fzjhvut hlnkiox lmyugyt afvhgnn kxcwanl ykuvwlt fwqdrgx vpydvga
gnakefy gknkegm zxjdwfb fznkiex ztxzseg laypvrt desdfbn lmbhwnf wthgfbm zmbdxzh jgcqkrj
mtfocyt qbhoino wlhrwgx haugxeh vwyqfyt udikmxx hmnkisb jlniset fhnkiew srshxxg gpcqkuh
opubprt vliqxbp srcgshu lxxljvl zhoohro wkwrqru svelwut devhxre dbhjxub kpcwlnl azbvszx
oayuinz wluqhnz wlbhrpx lpiusnw kwcyiez wwcqejh gwuqhvm ghewlrh fxfhwfm jtphprw truqhga
smbdwzt vxuopga wwcijrk wgwhve

Result:



Space between four “nki” numbered in #1, 2, 3, 4:
1) #1&2: 70 = 2*5*7
2) #2&3: 98 = 2*7*7
3) #3&4: 14 = 2*7
4) #1&3: 168 = 2*2*2*3*7
5) #2&4: 112 = 2*2*2*2*7
6) #1&4: 182 = 2*7*13

Space between three “uqh”:
1) #1&2: 35 = 5*7
2) #2&3: 28 = 2*2*7
3) #1&3: 63 = 3*3*7

Space between three “lay”:

1) #1&2: 14 = 2*7
2) #2&3: 126 = 2*3*3*7
3) #1&3: 140 = 2*2*5*7

Space between three "wwc":
1) #1&2: 483 = 3*7*23
2) #2&3: 56 = 7*8
3) #1&3: 539 = 7*7*11

We can see that 7 is a common factor of all of them,
so mostlikely the keyword has length of 7(k=7)

Part B: Counting number of characters in each column:

I write the program in the C++ file called "3.cpp",

Here is the result column I got:
0th column count:]
[3 0 1 4 0 5 6 3 0 5 5 10 1 1 2 0 1 0 7 1 1 8 12 0 1 4**]**

1th column count:
[5 2 0 1 4 1 5 4 0 0 5 7 8 1 0 7 0 4 0 6 1 2 8 6 0 4**]**

2th column count:
[1 7 11 0 2 4 0 5 9 2 0 2 1 6 1 1 2 0 2 0 8 2 2 4 8 1**]**

3th column count:
[0 1 0 8 2 1 4 11 3 1 8 2 0 0 4 1 12 6 0 0 4 3 4 2 2 2**]**

4th column count:
[3 0 1 0 6 3 2 5 9 2 2 2 1 1 1 4 1 4 10 0 0 6 8 9 0 1**]**

5th column count:
[1 7 0 0 9 3 9 3 1 1 0 2 0 11 1 1 2 12 1 1 4 4 0 2 3 3**]**

6th column count:
[6 6 0 0 3 1 5 7 0 1 3 5 5 2 2 1 0 0 0 11 2 0 5 10 1 4**]**

Organized in a excel file called “*shift_table.xlsx*”:

	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	AA	AB	AC
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25			
2																													
3		3	0	1	4	0	5	6	3	0	5	5	10	1	1	2	0	1	0	7	1	1	8	12	0	1	4	SUM	SHIFT
4	column0							A				E															T	15	6
5													T							A				E				29	18
6					E															T							A	15	25
7																													
8		5	2	0	1	4	1	5	4	0	0	5	7	8	1	0	7	0	4	0	6	1	2	8	6	0	4		
9		A				E															T							15	0
10								A				E															T	14	6
11	column1	T							A				E															16	7
12						T							A				E											18	11
13														T							A				E			20	19
14		E															T							A				20	22
15																													
16	column2	1	7	11	0	2	4	0	5	9	2	0	2	1	6	1	1	2	0	2	0	8	2	2	4	8	1		
17															T							A				E		22	20
18																													
19		0	1	0	8	2	1	4	11	3	1	8	2	0	0	4	1	12	6	0	0	4	3	4	2	2	2		
20	column3				A				E															T				23	3
21					T							A				E												20	10
22																													
23	column4	3	0	1	0	6	3	2	5	9	2	2	2	1	1	1	4	1	4	10	0	0	6	8	9	0	1		
24						A				E															T			24	4
25																													
26	column5	1	7	0	0	9	3	9	3	1	1	0	2	0	11	1	1	2	12	1	1	4	4	0	2	3	3		
27								T							A				E									32	13
28																													
29		6	6	0	0	3	1	5	7	0	1	3	5	5	2	2	1	0	0	0	11	2	0	5	10	1	4		
30	column6	T							A				E															18	7
31														T							A				E			26	19
32	shift of A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		

Each box represent a column in the text, and all the posible shifts sorted below in sum-descending order:

column#	0	1	2	3	4	5	6
shift	18	19	20	3	4	13	19
	25	22		10			7
	6	11					
		7					
		0					
		6					

Keyword try: “18 19 20 3 4 13 19” --> “**student**”

Cite:
http://practicalcryptography.com/cryptanalysis/text-characterisation/monogram-bigram-and-trigram-frequency-counts/#trigram-counts