Advanced Algorithms Assignment 3

Anthony Medico

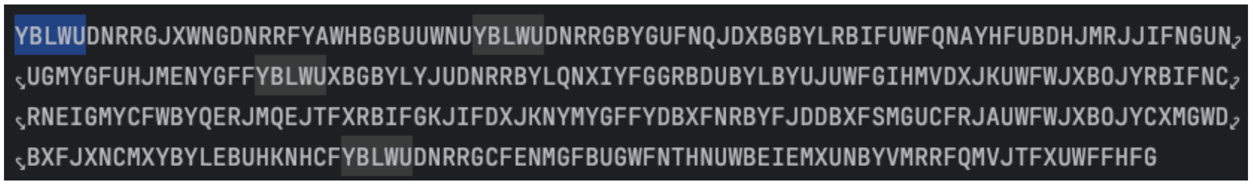
Question 1

To solve the Substitution Cipher, methods similar to those shown in class were used – using statistical information of the relative frequencies of the 26 letters in the English language.

I wrote a program to assist with the process, with different tools to aid with the following:

- Count the frequencies of each letter in the ciphertext and display each as a percentage (to compare with the statistical frequencies).
- Add a substitution (guess) to the permutation, e.g., `F = e` where `F` is the ciphertext character and `e` is the guessed corresponding plaintext character.
- Clear the guesses, if needing to start over.
- Printing the ciphertext with the guessed letters shown above, or a dash if no guess made yet for that character
- Once the solution was found, added an option to solve and print the result instantly.

In addition to the above tools, I was able to use my IDE (PyCharm) console to help me locate repeated portions of text. The screenshot below shows (in blue) that I manually highlighted the first five letters of the ciphertext, `YBLWU`, and the lighter gray highlights show that three more repetitions of this string occur. The IDE did this automatically, and I used this functionality constantly during this task to look for repetitions like this.



```
YBLWUDNRRGJXWNGDNRRFYAWHBGBUUWNUYBLWUDNRRGBYGUFNQJDXBGBYLRBIFUWFQNAYHFUBDHJMRJJIFNGUN
UGMYGFUHJMENYGFFYBLWUXBGBYLYJUDNRRBYLQNXIYFGGRBDUBYLBYUJUWFGIHMVDXJKUWFWJXBOJYRBIFNC
RNEIGMYCFWBYQERJMQEJTFXRBIFGKJIFDXJKNYMYGFFYDBXFNRBYFJDDBXFSMGUCFRJAUWFWJXBOJYCXMGWD
BXFJXNCMXYBYLEBUHKNHCFYBLWUDNRRGCFENMGFBUGWFNTHNUWBEIEMXUNBYVMRRFQMVJTFXUWFFHFG
```

The table below shows the frequency of occurrence of the 26 ciphertext letters which I obtained from my helper program.

A	0.9%	H	2.7%	O	0.6%	V	0.9%
B	9.3%	I	2.7%	P	0%	W	5.4%
C	2.1%	J	6.3%	Q	1.8%	X	5.1%
D	4.2%	K	1.2%	R	6.3%	Y	8.1%
E	2.4%	L	2.7%	S	0.3%	Z	0%
F	10.2%	M	4.2%	T	0.9%		
G	6.9%	N	6.9%	U	7.5%		

Next, I constructed this chart to organize the information above.

```
Highest frequency is E (12%)
- Possible characters: F (10.2%), B (9.3%)

Next tier: T, A, O, I, N, S, H, R (6-9% each)
-Possible characters:
       B (9.3%)
       Y (8.1%)
       U (7.5%)
       N (6.9%)
       G (6.9%)
       J (6.3%)
       R (6.3%)
       W (5.4%)

Next tier: D, L (4%)
-Possible characters:
       X (5.1%)
       M (4.2%)
       D (4.2%)

Next tier: U, C, M, W, F, G, Y, P, B (1.5-2.8%)
-Possible characters:
       M,D (4.2%)
       H,I,L (2.7%)
       E (2.4%)
       C (2.1%)
       Q (1.8%)
       K (1.2%)
       A,T,V (0.9%)

Last tier: V, K, J, X, Q, Z (< 1%)
       A,T,V (0.9%)
       O (0.6%)
       S (0.3%)
       P,Z (0%)
```

With this information I began attempting to figure out the substitutions.

- The first few hours were a bunch of guesses that didn't seem to lead anywhere, but a summary of that was guessing F = e or B = e, followed by random guesses of a bunch of two letter repetitions: HE, ER, RE, ES, followed by some three letter like THE, AND, ING.
- Below I show the detailed process once I started making actual progress.

- Guess: F = e; F is the highest frequency
- Guess: UWF = the; UWF appears five times
- Figure out ES or ER
  - Guess G = r; this looked a bit odd
  - Guess G = s; seemed to make sense, went with this
- In the result so far, I spotted `th_t`; and guessed the word is `that`
  - Guess N = a
- Noticed the first 10 letters `YBLWUDNRRG` appeared 3 times, and the first 5 letters of that `YBLWU` was repeated by itself elsewhere.
  - Determined this was probably two words:
    - `---ht -a--s`
    - `YBLWU DNRRG`
  - Thought that `ht` was a big odd, and that the only way that would work is if it was `ght` with the silent `gh`
  - Guess L = g
  - Thought of the word 'ought', which means Y = o, B = u
    - But B is a high frequency letter, and u is not
    - Needed to find two higher frequency letters for Y and B
    - Guessed the word is 'night'
    - Guess Y = n
    - Guess B = i
  - For the second word `DNRRG = -a--s`, tried to figure out the double RR
    - R is a higher frequency letter, and all that's left of those were o,r
    - Checked both of those but neither made sense
    - In the next tier at 4% are the letters d,l
      - Tried both of these, dd didn't quite make sense
      - 'll' would form a lot of words though, e.g.
        - balls, calls, falls, halls, malls, walls
    - Guess R = l
  - For these words I now had:
    - `night -alls`
    - `YBLWU DNRRG`
  - Decided to come back to this later for the missing letter since there wasn't one clear answer
- At this point I looked over the whole ciphertext to try to pick out other words, and saw the following, guessing the word is 'instead':
    - `instea-`
    - `BYGUFNQ`
  - Guess Q = d
- Spotted the following and guessed 'rising':
    - `-ising`
    - `XBGBYL`

    - With the word ‘rising’, maybe `-alls` is ‘falls’, also spotted `-allen` which could be ‘fallen’; opposite words show a relation
    - Guess X = r
    - Guess D = f
- Guessed `-r` between two deciphered words was ‘or’
    - Guess J = o
- Guessed `li-e` between two deciphered words was ‘like’
    - Guess I = k
- Guessed ‘rising like the da-‘ suggested it was ‘day’
    - Guess A = y; didn’t make sense
    - There was an ‘n’ after so guessed da-n was ‘dawn’
    - Guess A = w
- Guessed `wh-` is ‘why’
    - Guess H = y
- Guessed `fro-thehori-on` is ‘from the horizon’
    - Guess K = m
    - Guess O = z
- Guessed `ifyo-look` is ‘if you look’
    - Guess M = u
- Guessed `you-ansee` is ‘you can see’
    - Guess E = c
- Guessed `skyu-from` is ‘sky up from
    - Guess V = p
- The words `-lack` and `-ehind` both had the same cipher character C
    - Guessed ‘black’ and ‘behind’
    - Guess C = b
- Guessed cloudco-er is ‘cloud cover’
    - Guess T = v
- Lastly, `fire-ustbelow`; could be ‘must’, ‘gust’, ‘just’
    - Only letters left are j, q, x
    - Guess S = j

Final result:

Cipher:	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
Plaintext:	`wibfcesykomguaz-dljvtphrn-`

Decrypted text on next page.

```
nightfallsorhasfallenwhyisitthatnightfal
YBLWUDNRRGJXWNGDNRRFYAWHBGBUUWNUYBLWUDNR

lsinsteadofrisinglikethedawnyetifyoulook
RGBYGUFNQJDXBGBYLRBIFUWFQNAYHFUBDHJMRJJI

eastatsunsetyoucanseenightrisingnotfalli
FNGUNUGMYGFUHJMENYGFFYBLWUXBGBYLYJUDNRRB

ngdarknessliftingintotheskyupfromthehori
YLQNXIYFGGRBDUBYLBYUJUWFGIHMVDXJKUWFWJXB

zonlikeablacksunbehindcloudcoverlikesmok
OJYRBIFNCRNEIGMYCFWBYQERJMQEJTFXRBIFGKJI

efromanunseenfirealineoffirejustbelowthe
FDXJKNYMYGFFYDBXFNRBYFJDDBXFSMGUCFRJAUWF

horizonbrushfireoraburningcitymaybenight
WJXBOJYCXMGWDBXFJXNCMXYBYLEBUHKNHCFYBLWU

fallsbecauseitsheavyathickcurtainpulledu
DNRRGCFENMGFBUGWFNTHNUWBEIEMXUNBYVMRRFQM

povertheeyes
VJTFXUWFFHFG
```

Plaintext: From *The Handmaid's Tale* by Margaret Atwood (I googled it after deciphering)

"Night falls. Or has fallen. Why is it that night falls, instead of rising, like the dawn? Yet if you look east, at sunset, you can see night rising, not falling; darkness lifting into the sky, up from the horizon, like a black sun behind cloudcover. Like smoke from an unseen fire, a line of fire just below the horizon, brushfire or a burning city. Maybe night falls because it's heavy, a thick curtain pulled up over the eyes."

Question 2

Summary: To solve the Vigenere Cipher, I calculated the index of coincidence for values of m between 1 and 9, determined from those results that m = 6, then calculated the mutual index of coincidence for m = 6, and used that result to determine the key which was (9, 0, 1, 1, 4, 17) representing the word JABBER. I then used the key to decipher the text character by character.

The way I wrote the program was in phases. Each phase calculated what was needed for the next. Once I obtained the results, I added more code to the program to use those results for the next phase. The final output of the program displays all of this data leading up to and including the final deciphered text.

Phase 1: Determine m using index of coincidence calculation.

For m = 1 to 9, I divided the ciphertext into slices of every m letters (rows of length m), but the rows needed to be columns instead, so I transposed the matrix to make it in the proper format for the index of coincidence calculation.

For each m above, I did the index of coincidence calculation and displayed the results. It worked as described in class, and m = 6 stood out with values much close to 0.065. Below is a summary of the results for this phase, produced by my program, only up to m = 6:

```
m = 1
CWBTFIRL…
0.040

m = 2
CBFRMKWU…
WTILJRDI…
0.044, 0.041

m = 3
CTRJWIUI…
WFLKDIIC…
BIMRUJUK…
0.045, 0.047, 0.050

m = 4
CFMWIUXH…
WIJDJIVZ…
BRKUUCFM…
TLRIIKTG…
0.046, 0.044, 0.041, 0.041
```

```
m = 5
CIKIUVMR…
WRRIIFGW…
BLWJCTZD…
TMDUKHSH…
FJUIXZIJ…
0.037, 0.036, 0.039, 0.041, 0.042
```

```
m = 6
CRWUXMWU…
WLDIVGDE…
BMUUFZHJ…
TJIITSJO…
FKICHIQX…
IRJKZRSY…
0.062, 0.055, 0.069, 0.054, 0.055, 0.065
```

<u>Phase 2</u>: Calculate the mutual index of coincidence using m = 6.

I created a global variable to save the matrix for m = 6 from phase 1. With this matrix now stored, I calculated the mutual index of coincidence using it and the probabilities (of frequencies in English) from the textbook. Since m = 6, this was done for i = 1 to 6: Below are the results with the values closest to 0.065 highlighted.

i = 1	0.033 0.036 0.041 0.039 0.028 0.044 0.033 0.031 0.043 **0.062** 0.034 0.028 0.050 0.045 0.033 0.035 0.043 0.027 0.036 0.037 0.040 0.031 0.048 0.045 0.041 0.042
i = 2	**0.060** 0.038 0.033 0.036 0.047 0.042 0.033 0.037 0.040 0.033 0.031 0.046 0.038 0.035 0.038 0.051 0.041 0.033 0.041 0.043 0.035 0.031 0.042 0.029 0.031 0.038
i = 3	0.037 **0.069** 0.040 0.030 0.031 0.042 0.034 0.035 0.038 0.040 0.034 0.035 0.043 0.042 0.039 0.038 0.048 0.040 0.029 0.034 0.045 0.034 0.035 0.049 0.035 0.025
i = 4	0.038 **0.059** 0.045 0.031 0.038 0.037 0.032 0.037 0.043 0.036 0.037 0.040 0.035 0.036 0.042 0.045 0.044 0.040 0.028 0.032 0.039 0.041 0.037 0.042 0.039 0.029
i = 5	0.038 0.037 0.033 0.044 **0.062** 0.046 0.032 0.035 0.038 0.032 0.034 0.035 0.033 0.033 0.041 0.042 0.043 0.042 0.041 0.040 0.045 0.035 0.030 0.040 0.036 0.033
i = 6	0.029 0.032 0.042 0.044 0.043 0.041 0.048 0.038 0.030 0.036 0.051 0.031 0.028 0.040 0.038 0.031 0.039 **0.065** 0.041 0.034 0.035 0.041 0.034 0.037 0.038 0.034

Counting the index for each value highlighted above we get (9, 0, 1, 1, 4, 17) which corresponds to the keyword JABBER

Phase 3: Use the keyword (shift amounts) to decrypt to plaintext.

To decrypt the text, I used an array with the shift amounts [9, 0, 1, 1, 4, 17]. I then iterated through each letter of the ciphertext and did the following:

- Get the character from the ciphertext string
- Get the correct shift amount from the key for this part of the string by using `K[i % 6]`
- Converted the ciphertext character to an alphabetic number (A = 0, B = 1, C = 2, …)
- Subtracted the shift amount from the character number, mod 26
- Converted the new number back to its ascii value but lowercase this time
- Retrieved the actual character from the ascii value
- Printed the plaintext character

The result:

```
twasbrilligandtheslithytovesdidgyreandgimbleinthewabeallmimsywer
etheborogovesandthemomerathsoutgrabebewarethejabberwockmysonthej
awsthatbitetheclawsthatcatchbewarethejubjubbirdandshunthefrumiou
sbandersnatchhetookhisvorpalswordinhandlongtimethemanxomefoeheso
ughtsorestedhebythetumtumtreeandstoodawhileinthoughtandasinuffis
hthoughthestoodthejabberwockwitheyesofflamecamewhifflingthrought
hetulgeywoodandburbledasitcame
```

Phase 4: Extreme confusion.

I wasn't sure if this was correct but there were a lot of correct English words. Since it started with the word 'Twas, I thought maybe it's a poem. I ended up googling it and learning it was in fact a 'nonsense' poem called *Jabberwocky by Lewis Carroll*.

Question 3

Summary: To solve this cipher I brute force factored n into the two primes p and q. With such a small value of n it took no time at all. I implemented and made use of the Square and Multiply, Chinese Remainder Theorem, and Multiplicative Inverse algorithms learned in class, to help decrypt each given y value. Once I had the x value, I then converted it to its corresponding string. I printed out all four possibilities for each x, and then manually chose the 4-letter slice that made the most sense given the overall context.

Factoring n: To do this efficiently, I only had to check primes less than the square root of n. My algorithm did the following:

- Iterate through the numbers `3` to `floor(sqrt(n))`
- For each number `i`, first check if it's a prime
- If so, check if `n mod i = 0` (evenly divisible by n)
- If so, check if `n/i` is prime
- If so, two prime factors found

Running this algorithm produced only 1 result: p = 691, q = 719

Decrypting the ciphertext: To do this I did the following for each given integer `y`:

- Calculate congruences for the current value of y
    - Implemented square and multiply algorithm to do this efficiently
    - Used the formula $\pm y^{\frac{p+1}{4}} (mod\ p)$ and $\pm y^{\frac{q+1}{4}} (mod\ q)$
- Used the Chinese Remainder Theorem on all four cases above to get each deciphered value x
    - Which also made use of the Multiplicative Inverse algorithm
- Converted each x into its corresponding text
    - Divided x by 26^3 to get c_0 from the quotient
    - Divided the remainder by 26^2 to get c_1 from that quotient
    - Repeated similarly two more times to get all c values
    - Converted each c to its corresponding ascii value
    - Stored the corresponding letters
- Stored all four possible answers into a global array that will hold every possible plaintext, and printed this once the program was finished for the entire ciphertext
- From the results, I manually chose the x that made the most sense
    - These last two steps are shown below:

Results: Below is the program output of all plaintexts. I went through manually and highlighted the correct ones for each column.

```
kylp izuu wswz izyc ayhc cpfx kylp forg etyo urpe rfec toff gvog vnrr dtmg vhpz
mktd louz ells xljx jnqk reyi mktd opst skdh yxdn gfbf pekg xpqi ehxk reis }ehv
pwfs qsdw xvnd evoy stil lcan pwfs nrgc jwvo djvi wbxq ncop erin xzbl lcqd acra
ring theb fobw that |irt zrsy ring wshp xnah hpjr lbut istq vlkp gthe ynmp gziw
```

```
tdlw eewi ttua eryt hing izyc hscn hoon uarc gets lnld atsh nkfo elig htge ioqi
zwyt lgwl {pkf ebzy oqxh xljx show izuk feot xven ttfo ukgd iawx oxpq wdqi zdkd
ckac rack broq yeyx nqbo evoy jzjz thel xckc elui inth hwss ugby njjf gdin ddos
jdmz yccn inev xpac uylp that uowi uski ight wcfd qtns |ngo owth xvqp unsr tsin
```

Plaintext: Excerpt from *Anthem* by Leonard Cohen

“Ring the bells that still can ring. Forget your perfect offering. There is a crack, a crack in everything that’s how the light gets in, that’s how the light gets in.”