Nathan Walzer - nwalzer

For problem 1, I made a python program to count the letter frequency for the ciphertext. I then expanded the program to allow me to manually define a key (which I started off as correlating directly with wikipedia-defined letter frequencies) to use a trial and error approach in deciphering. characterCount.py and input.txt are included in this submission.

1a) Ciphertext frequencies

C - 13.93%
B - 9.29%
D - 7.99%
G - 7.71%
F - 7.06%
A - 6.96%
I - 6.5%
E - 5.39%
L - 4.64%
K - 4.36%
H - 4.18%
J - 3.71%
M - 3.44%
N - 2.23%
S - 2.23%
Q - 2.14%
O - 1.76%
P - 1.76%
R - 1.39%
U - 1.39%
T - 0.84%
V - 0.84%
Y - 0.28%
W - 0.0%
X - 0.0%
Z - 0.0%

1b) Decrypted ciphertext

ELECTRICAL AND COMPUTER ENGINEERS DEVELOP AND CREATE PRODUCTS THAT CHANGE THE WORLD AND MAKE OUR LIVES EASIER THE CELL PHONES WE DEPEND ON THE COMPUTERS USED IN NATIONAL SECURITY AND THE ELECTRICAL SYSTEMS THAT MAKE OUR CARS OPERATE WERE ALL CREATED BY ELECTRICAL AND COMPUTER ENGINEERS AT WPI WE KEEP THAT PROGRESS MOVING FORWARD WITH OUR INNOVATIVE RESEARCH AND OUT-OF-THE BOX APPROACHES THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING AT WPI CHALLENGES STUDENTS TO PUSH THEMSELVES TO UNDERSTAND SOCIETYS AND TECHNOLOGYS COMPLEX

ISSUES IN A BROADER CONTEXT THAN WHATS IN FRONT OF THEM WE WANT OUR STUDENTS WHETHER THEY ARE EARNING AN UNDERGRADUATE MINOR OR A DOCTORATE TO TACKLE SOCIETYS MOST PRESSING PROBLEMS AND UNCOVER NEW WAYS OF SOLVING THEM WHETHER ITS DEVELOPING SYSTEMS THAT CAN LOCATE FIREFIGHTERS IN THE MIDDLE OF A BURNING BUILDING OR CREATING NEUROPROSTHETICS THAT LOOK AND FUNCTION LIKE NATURAL LIMBS OUR FACULTY AND STUDENTS ARE AT THE FRONT EDGE OF REMARKABLE INNOVATION WHILE ADVANCING TECHNOLOGIES IS AT OUR CORE WE ALSO TAKE HUMAN CONNECTIONS VERY SERIOUSLY IN ECE WE PRIDE OURSELVES ON THE FAMILY-LIKE ATMOSPHERE WE CULTIVATE; FACULTY STUDENTS AND STAFF ENCOURAGE EACH OTHERS EVERY SUCCESS AND ARE THERE FOR THE CHALLENGES BOTH IN THE CLASSROOM AND IN LIFE

English frequencies according to wikipedia:
E - 11%
S - 8.7%
I - 8.6%
A - 7.8%
R - 7.3%
N - 7.2%
T - 6.7%
O - 6.1%
L - 5.3%
C - 4%
D - 3.8%
U - 3.3%
G - 3%
P - 2.8%
M - 2.7%
H - 2.3%
B - 2%
Y - 1.6%
F - 1.4%
V - 1%
K - 0.97%
W - 0.91%
Z - 0.44%
X - 0.27%
J - 0.21%
Q - 0.19%

1c) Substitution key
The keys for J, Q, and Z can be interchanged since none of those letters appear in the plaintext

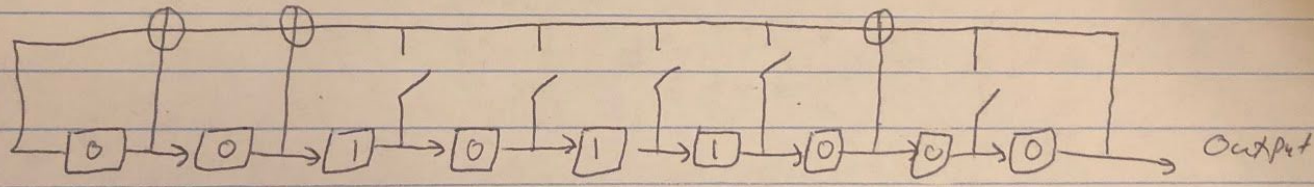
Plaintext -> Ciphertext - plaintext frequency
A -> D - 7.99%
B -> T - 0.84%
C -> L - 4.64%
D -> J - 3.71%
E -> C - 13.93%
F -> P - 1.76%
G -> Q - 2.14%
H -> H - 4.18%
I -> E - 5.39%
J -> X - 0.0%
K -> V - 0.84%
L -> K - 4.36%
M -> N - 2.23%
N -> G - 7.71%
O -> F - 7.06%
P -> S - 2.23%
Q -> Z - 0.0%
R -> A - 6.96%
S -> I - 6.5%
T -> B - 9.29%
U -> M - 3.44%
V -> U - 1.39%
W -> O - 1.76%
X -> Y - 0.28%
Y -> R - 1.39%
Z -> W - 0.0%

For part 2, I built an LFSR python script that told me the period and output for given state and gate arrays. LFSR.py is included in this submission. I did all of part 2 out on paper, but put everything other than the diagrams into text here.

2ia)

2i.

a)



b) This LFSR use polynomial $x^9+x^8+x^7+x^2+1$, which is a 9^{th} degree primitive polynomial, so period = 2^{n+1} = 511

C. First 30 bits,

First bit → Last bit

000110100010000111000010100001

2ib) This LFSR uses polynomial $x^9 + x^8 + x^7 + x^2 + 1$ which is a 9th degree irreducible primitive polynomial, so the period is $2^m-1 = 511$. My python program confirmed this answer.

2ic) The first 30 bits, where first bit is on the left and 30th bit is on the right:
000110100010000111000010100001

2id) I did not know the endian-ness for this problem, so I have the first output bit on the left and the last output bit on the right, just as I had for 2ic

P = 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 0
^ 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 0 0 1
1 1 1 1 0 1 1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1

2iia) *Note: because of a mistake I made, 2iic and 2iid in the picture are incorrect*

C. First 30 bits,

First bit → Last bit

0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 0 0 1

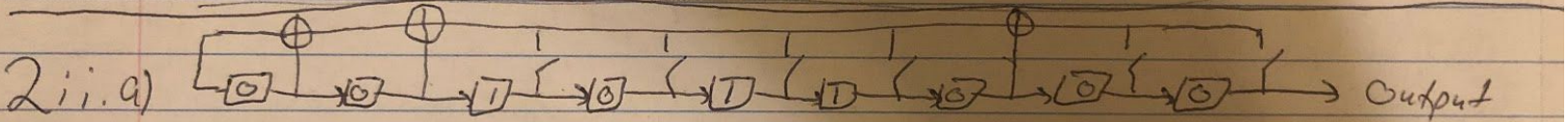
D. I Don't know the endian-ness of this problem, so I solved this problem where the first output bit was the leftmost bit

P = 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0

∧ 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 0 0 1

1 1 1 1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1

2ii.a)



b) This LFSR uses an effective polynomial of $x^7+x^6+x^5+1$ because C_0 and C_1 are both open. The MAX period for a 7^{th} degree polynomial is 127. This polynomial is reducible, however, meaning it has a subset of input states that yeild different periods. According to my Python Program, the maximum period for any input state is 60.

C.) First 30 bits First → last bit

0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0

D. P = 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 0

2iib) This LFSR is essentially using polynomial $x^7 + x^6 + x^5 + 1$ because both C_0 and C_1 are open. The max period for a 7th degree polynomial is usually 127, however this polynomial is reducible, meaning it will contain subsets of initial states that can reach each other. For the given initial state, my python program was able to find a maximum period of 60.

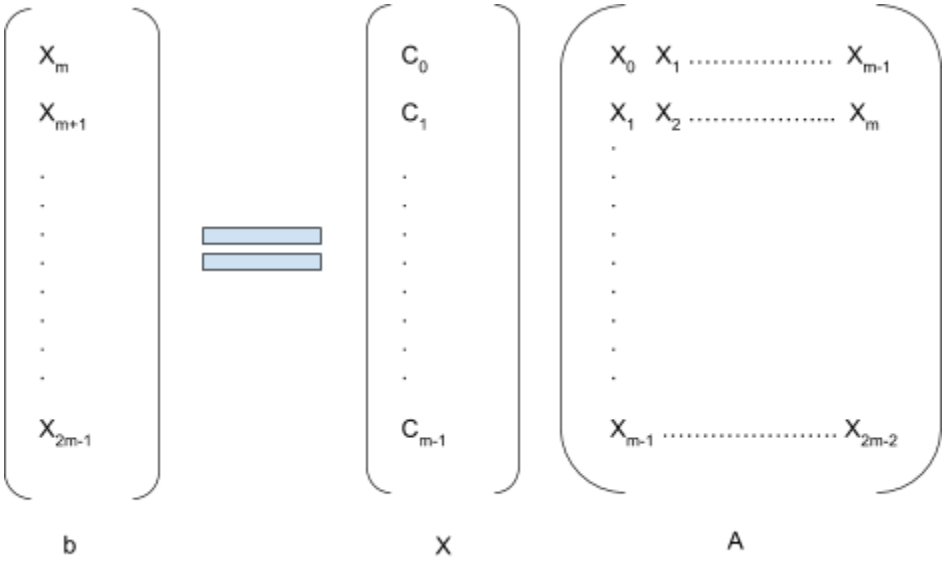
2iic) The first 30 bits, where first bit is on the left and 30th bit is on the right:
000110100010101100011110111111

2id) I did not know the endian-ness for this problem, so I have the first output bit on the left and the last output bit on the right, just as I had for 2ic

P = 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 0
^ 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 1
1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1

Bonus Question:
Consider an LFSR with m registers. Show that you can find gate positions if you have 2m consecutive output bits.

Consider we have LFSR with initial states X_0, X_1...X_{m-1} and gate positions C_0...C_{m-1}. We know that $X_m = C_0X_0 \text{ ^ } C_1X_1 \text{ ^ } ..... \text{ ^ } C_{m-1}X_{m-1}$, $X_{m+1} = C_0X_1 \text{ ^ } C_1X_2 \text{ ^ } ..... \text{ ^ } C_{m-1}X_m$, ….. $X_{2m-1} = C_0X_{m-1} \text{ ^ } C_1X_m \text{ ^ } ..... \text{ ^ } C_{m-1}X_{2m-2}$. We have M equations with M unknowns (C_0-C_{m-1}), since we know that in one period, a state only appears exactly one time, we know that all of these equations are linearly independent. Thus we can set up a matrix equation Ax=b as:



Since we know that A is linearly independent, we know that there exists some A^{-1}, thus we can find the gate values (x) by multiplying both sides of the equation by A^{-1}. $x = A^{-1}b$