CS 355, Fall, 2019, Project 3

Write a program in Java to perform a known-plaintext attack on a Linear Feedback Shift Register.

See `www.cs.purdue.edu/homes/ssw/cs355/x07.pdf` for definitions.

For the known-plaintext attack, assume that the LFSR has n bits and that you know know a plaintext and corresponding ciphertext of length $2n$ bits. You XOR them to get $2n$ consecutive bits of key. Of course, the initial content of the LFSR is the first n key bits in reverse order. To find the tap vector, you must invert an $n \times n$ matrix modulo 2. Do not use adjoints or determinants to find the inverse matrix. The most reasonable way to invert a matrix is via Gaussian elimination and row reduction.

Suppose X is the $n \times n$ matrix to invert. Form an $n \times 2n$ matrix M with X as the left side and the identity matrix I as the right side. Row reduce the matrix to get I as the left side. Then the right side is the inverse X^{-1}, and $H = YX^{-1} \bmod 2$ in the notation on the class web page. The tap vector is the first row of H.

Follow these steps to row reduce the matrix $M = [m_{ij}]$ modulo 2. Recall that XOR is addition (or subtraction) modulo 2.

For $i = 1$ to n perform steps 1 to 3.
1. Find a pivot: Let j in $i \le j \le n$ be the first nonzero element of column i on or below the main diagonal, that is, $m_{ij} = 1$ but $m_{ik} = 0$ for $i \le k < j$.
2. Swap rows if needed: If $j > i$, swap the entire rows i and j. (If $j = i$, do nothing.)
3. Use the 1 in m_{ii} to make the rest of column i all zero: For $j = 1$ to n, skipping $j = i$, if $m_{ij} = 1$, then replace the entire row j with its former value XOR row i. (This is like subtracting row i from row j.) (If $m_{ij} = 0$, leave row j unchanged.)

The input to your program will be (1) an integer n with $0 < n < 100$ and (2) a string of $2n$ bits representing the known part of the key, The two items are each followed by a newline character, that is, each input item is on its own line.

Your program should should write (1) the initial content R of the LFSR and (2) the tap vector T, with a single newline after each.

Example:

If the input to your program is:

4
00110101

then your program should write:

1100
0011

exactly as shown because Vocareum grades by comparing character strings. (Each line ends with a newline character.)

Example:

If the input to your program is:

14
01011100101101011111000001101

then your program should write:

10110100111010
10000101000001

exactly as shown because Vocareum grades by comparing character strings.

Name your program `solvelfsr.java`. Submit your program to Vocareum by 11:59 PM on the due date. It will be compiled and run ten times with ten different secret input sets, each worth 10 points.

Note that the input and output are the ascii characters for '0' and '1', but the XOR operates on *bits*. An XOR of ascii characters probably won't give the desired result. I suggest that you either (1) convert ascii to bits and use the logical operations in java, or (2) write your own XOR (and AND for matrix multiplcation) for ascii '0' and '1'.