Some little questions

Tag systems

There are several variations on this;

> `Input:` *A string of* `0`*'s and* `1`*'s.*

We will repeatedly cross off three digits from the front of our string, and tack on digits at the end, by the following rule: If the first crossed-off digit is `0`, we tack on `00`; if the first crossed-off digit is `1`, we tack on `1101`.

There are three possibilities: either our string at some point will have too few letters to cross off, and we *halt*; or we might enter into a *loop*, in which the same strings recur again and again *ad infinitum*; or we might run forever, without looping, the strings eventually growing without bound.

For example, if we begin with the string `10101`, we cross off `101` from the left and tack on `1101` on the right, obtaining ~~`101`~~`01 1101 = 011101`. Repeating this process we obtain ~~`011`~~`101 00 = 10100`, then `001101` and then once again `10100`. So on our third string we enter into a loop of length 2.

> `Decision Problem:` *For a given input, does this process loop, halt or run forever without repeating?*

If the process does loop or halt, we'll find this out, with enough patience. But we would need some idea of what's happening to figure out if it runs forever without repeating.

Try these out — or better yet write a program to do so.

`1··1··1··0··` (the ·'s aren't going to matter)

`1··1··1··0··1··0··`

`1··1··0··1··0··1··`

More to the point, can you give a theorem that sorts the inputs out, deciding which ones are which?

More generally, such a system is specified by an alphabet, a number k of letters to strike off, and rules for strings to add on the end, depending on the first letter struck off. The input is a string in the alphabet and the decision problem is whether or not the procedure will run forever without looping.

Here's another interesting system, with $k = 2$ on alphabet `a`, `b`, `c`, with rules

`a→bc` `b→a` `c→aaa`

Beginning with a string of m `a`'s does this always halt? Can you prove it?

Fractran

A Fractran program is specified by a list of rational numbers, and its input is an integer. At each step, the current integer is multiplied by the first number on the list that gives an integer for the next step. If there's no such fraction, we halt. Again, we might ask, for a given fractran program, does the process halt, repeat, or run forever without repeating?

So for example:

$$\frac{3}{11} \quad \frac{847}{45} \quad \frac{143}{6} \quad \frac{7}{3} \quad \frac{10}{91} \quad \frac{3}{7} \quad \frac{36}{325} \quad \frac{1}{2} \quad \frac{36}{5}$$

Taking as our input 10, we would first multiply by 1/2, obtaining 5; we then multiply by 36/5, obtaining 36; in this manner, we see 858, then 234; then 5577; 1521; 3549; 8281; 910 and then 100 — 10^2; after another thirty-six steps, we have 1000 — 10^3.

In fact, this program calculates the primes; every power of 10 will be a prime power, in order. Write an emulator to test this. How does this work?

Generally speaking, given a fractran program, and input, can we figure out if the program runs forever without repeating?

the Post Correspondence Problem

No big deal... Given a list of allowed pairs of strings, can you select a sequence of pairs so that the concatonation of the first strings in each pair equals that of the second strings in each pair.

For example, for pairs $\begin{bmatrix} \texttt{0} \\ \texttt{100} \end{bmatrix}$, $\begin{bmatrix} \texttt{01} \\ \texttt{00} \end{bmatrix}$, $\begin{bmatrix} \texttt{110} \\ \texttt{11} \end{bmatrix}$, one solution is

$\begin{bmatrix} \texttt{110} \\ \texttt{11} \end{bmatrix} \begin{bmatrix} \texttt{01} \\ \texttt{00} \end{bmatrix} \begin{bmatrix} \texttt{110} \\ \texttt{11} \end{bmatrix} \begin{bmatrix} \texttt{0} \\ \texttt{100} \end{bmatrix}$ — both top and bottom are the sequence `110011100`.

Given a list of pairs, is there a solution?

In this one, we specify that the sequence has to begin with some number n of $\begin{bmatrix} — \\ \texttt{a} \end{bmatrix}$'s where the top is the empty string, and the sequence has to end with some number m of $\begin{bmatrix} \texttt{a} \\ — \end{bmatrix}$'s The middle is pairs of

$\begin{bmatrix} \texttt{aa} \\ \texttt{bc} \end{bmatrix}$, $\begin{bmatrix} \texttt{ab} \\ \texttt{bc} \end{bmatrix}$, $\begin{bmatrix} \texttt{bc} \\ \texttt{a} \end{bmatrix}$, $\begin{bmatrix} \texttt{ca} \\ \texttt{aaa} \end{bmatrix}$, $\begin{bmatrix} \texttt{cb} \\ \texttt{aaa} \end{bmatrix}$

Describe m as a function of n. Can you prove this is correct?

Here are some trickier examples. Try these by hand. Try these by computer. How bad can this be?

$$\left[\begin{array}{c}0\\1\end{array}\right]\left[\begin{array}{c}01\\0\end{array}\right]\left[\begin{array}{c}1\\101\end{array}\right]$$

$$\left[\begin{array}{c}0\\011\end{array}\right]\left[\begin{array}{c}001\\1\end{array}\right]\left[\begin{array}{c}1\\00\end{array}\right]\left[\begin{array}{c}11\\110\end{array}\right]$$

$$\left[\begin{array}{c}0\\00\end{array}\right]\left[\begin{array}{c}0000\\0101\end{array}\right]\left[\begin{array}{c}0001\\10\end{array}\right]\left[\begin{array}{c}101\\1\end{array}\right]$$

$$\left[\begin{array}{c}0\\01\end{array}\right]\left[\begin{array}{c}1\\10\end{array}\right]\left[\begin{array}{c}0010\\0\end{array}\right]$$

String substitutions

In this example, we have alphabet $\mathtt{1}, A, B, C, D$ and the following rules, where the ω's can be anything.

$$\begin{array}{rcl}
A\omega & \to & A\omega\mathtt{1} \\
A\omega\mathtt{1} & \to & B\omega CD\omega\mathtt{1} \\
\omega_1\mathtt{1}C\omega_2\mathtt{1} & \to & \mathtt{1}\omega_1C\mathtt{1}\omega_2 \\
\omega_1BC\omega_2\mathtt{1} & \to & B\omega_1C\omega_2\mathtt{1} \\
\omega_1B\omega_2\mathtt{1}C\omega_3D & \to & B\omega_1\omega_2CD\omega_3 \\
\mathtt{11}BC\omega D\mathtt{1} & \to & \omega\mathtt{1}
\end{array}$$

So for example, starting with $A\mathtt{111}$, we can apply the first and second rules and make the new words $A\mathtt{1111}$ and $B\mathtt{111}CD\mathtt{1111}$. The third rule can be applied to that, giving $\mathtt{1}B\mathtt{11}C\mathtt{1}D\mathtt{111}$. Keep on going!

Show that the strings of just 1's are exactly the primes!

This one seems innocent enough:

`Input:` *Two words α and ω written in the letters* `a`, `b`, `c`, `d`, `e`

We are allowed to make the following seven substitutions:

$$\mathtt{ac}\leftrightarrow\mathtt{ca} \qquad \mathtt{ad}\leftrightarrow\mathtt{da} \qquad \mathtt{bc}\leftrightarrow\mathtt{cb} \qquad \mathtt{bd}\leftrightarrow\mathtt{db}$$
$$\mathtt{ce}\leftrightarrow\mathtt{eca} \qquad \mathtt{de}\leftrightarrow\mathtt{edb} \qquad \mathtt{cdca}\leftrightarrow\mathtt{cdcae}$$

`Decision Problem:` *Is there a sequence of substitutions taking α to ω?*

In fact, each of these systems can encode arbitrary computation, and so has undecidable questions within it.