"Calculator Game" - Max Figura - 2024/10/17

The Calculator Game is a sort of cellular automoton that operates on a finite, linear sequence of digits. It has two primary rules to generate the next state from the previous:

(1) For each contiguous sequence of three or more 0s, set the middle items to one or two 1s (for sequences of even or odd length, respectively). For example:

```
00      00000      00000000
00      00100      00011000
```

(2) For each nonzero item, read from left to right, move that item a number of spaces equal to its value, alternating leftward or rightward, starting with right (or "forward"):

```
0100      010040
0010      401000
```

Items that would move past the end (or start) of the sequence instead "wrap around" and resume count at the start (or end):

```
02301
10023
```

Any items which land in the same space sum together:

```
0100200      00010105000
0030000      06002000010
```

Traditionally, values larger than 9 will then be removed, effectively shortening the sequence as a whole. This represented with a decimal point that is ignored for all subsequent operations:

```
000100630009003
61.010000130000
10.001171000140
```

01.510010111007

Observations

As a piece of arbitrarly-defined recreational mathematics, the Calculator Game has not been studied with much rigor. However, some notable observations have been made about its behavior:

- Periodic states do exist. Most common are 2-cycles, usually occurring when each item alternates between moving forward and moving backward, without any landing on the same space or any 1s being generated:

  0100302200
  0310020020
  0100302200

  Fixed points can also occur when items "swap" places with another another item of the same value (or have a value evenly divisible by the sequence length):

  01102020 0040
  01102020 0040

  Other periods are also possible, at least in trivial cases:

  100
  010
  001
  100

- For any finite starting sequence, the space of potential reachable states is also finite (as the length never increases and there exist only eleven possible values for each space). Because each iteration is dependent only on the single state before it, this means that any

orbit must eventually reach a periodic state, and the system cannot truly be chaotic.

- When starting with a palindromic state of even length, every subsequent state must likewise be a palindrome of even length.

Implementation and the General Case

Originally the Calculator Game was, true to its name, always performed on a calculator. This was a manually process which did not make use of any functionalities and could be done just as well on paper. But the system can be automated easy enough, and if the sequence is stored as an array of numbers instead of a string of digits, it raises a question: why cap out after 9? A variant of the game was constructed using Python, which does not clip the values at all. In some ways this becomes an entirely different process, whose behavior may eventually diverge from that of the classic Calculator Game. Most notably, we see a greater diversity of pseudo-periodic states, whose orbits do not return to the same state but still exhibit something similar to periodicity:

```
[27, 0, 1, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 28]
[0, 0, 1, 1, 1, 0, 28]
[0, 0, 1, 1, 0, 1, 28]
[0, 0, 1, 1, 0, 0, 29]
[29, 0, 1, 1, 0, 0, 0]
[0, 30, 0, 0, 1, 1, 0]
[0, 0, 0, 31, 0, 0, 1]
[0, 1, 0, 0, 0, 1, 31]
[0, 0, 32, 1, 1, 0, 0]
[0, 0, 1, 0, 0, 1, 32]
[0, 0, 0, 33, 1, 0, 0]
[0, 34, 0, 1, 0, 0, 0]
[34, 0, 1, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 35]
[0, 0, 1, 1, 1, 0, 35]
[0, 0, 1, 1, 0, 1, 35]
[0, 0, 1, 1, 0, 0, 36]
[36, 0, 1, 1, 0, 0, 0]
[0, 37, 0, 0, 1, 1, 0]
[0, 0, 0, 38, 0, 0, 1]
[0, 1, 0, 0, 0, 1, 38]
[0, 0, 39, 1, 1, 0, 0]
```

```
[0, 0, 1, 0, 0, 1, 39]
[0, 0, 0, 40, 1, 0, 0]
[0, 41, 0, 1, 0, 0, 0]
[41, 0, 1, 0, 0, 1, 0]
```

Here one item accumulates to grow larger and larger, such that after 13 iterations it manages to return to the same position with a value exactly 7 more than it had previously. If we flatten each value to be less than or equal to the length a la modulo operator, we see true periodicity:

```
[6, 0, 1, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 7]
[0, 0, 1, 1, 1, 0, 7]
[0, 0, 1, 1, 0, 1, 7]
[0, 0, 1, 1, 0, 0, 1]
[1, 0, 1, 1, 0, 0, 0]
[0, 2, 0, 0, 1, 1, 0]
[0, 0, 0, 3, 0, 0, 1]
[0, 1, 0, 0, 0, 1, 3]
[0, 0, 4, 1, 1, 0, 0]
[0, 0, 1, 0, 0, 1, 4]
[0, 0, 0, 5, 1, 0, 0]
[0, 6, 0, 1, 0, 0, 0]
[6, 0, 1, 0, 0, 1, 0]
```

Although clipping at 9 has not yet been implemented in the program, one could instead consider some mutable `maxValue` attribute of the system to allow for a general case, where the maximum value before a space is removed could be made higher or lower. It is unknown how exactly this could impact the system's behavior, but the case without clipping could be considered as one where `maxValue` is arbitrarily large, lending some insight up until one item grows past that bar and diverges.