Diagonals in elementary cellular automaton 30

by Michael Brunnbauer (/), 2019-12-09

Abstract

The usual pattern starting from a single cell with state 1 / black is examined. It is contructed from a richer structure which yields a progression of polynomials for diagonals from the right. It is shown that each diagonal from the left can be expressed by the progression of polynomials for diagonals from the right and how the connection between left and right diagonals forces the diagonals from the left to eventually become periodic.

The necessary and sufficient condition for period doubling of diagonals from the right is established. It is also shown that periods cannot decrease. The result suggests that predicting period doubling is computationally expensive.

I am a math amateur and not familiar with the status quo of research regarding cellular automaton 30 so there is probably nothing genuinely new here. This is a fun exercise in the first place.

Introduction

An elementary cellular automaton is a one-dimensional cellular automaton with two possible states and a rule to determine the state of a cell in the next generation - which depends on the state of the cell and its two neighbors.

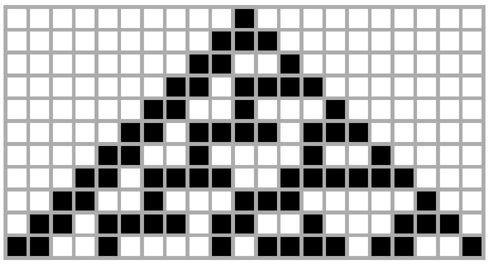
A cell with its two neighbors can have 2^3 possible states and the rule for the automaton has to specify the resulting state of the cell for each of them. This yields $2^{(2^3)}$ possible rules which are usually named by a number called the Wolfram code. This scheme was proposed by Stephen Wolfram. Here is Rule 30:

111	110	101	100	011	010	001	000	Current pattern
0	0	0	1	1	1	1	0	New state for middle cell

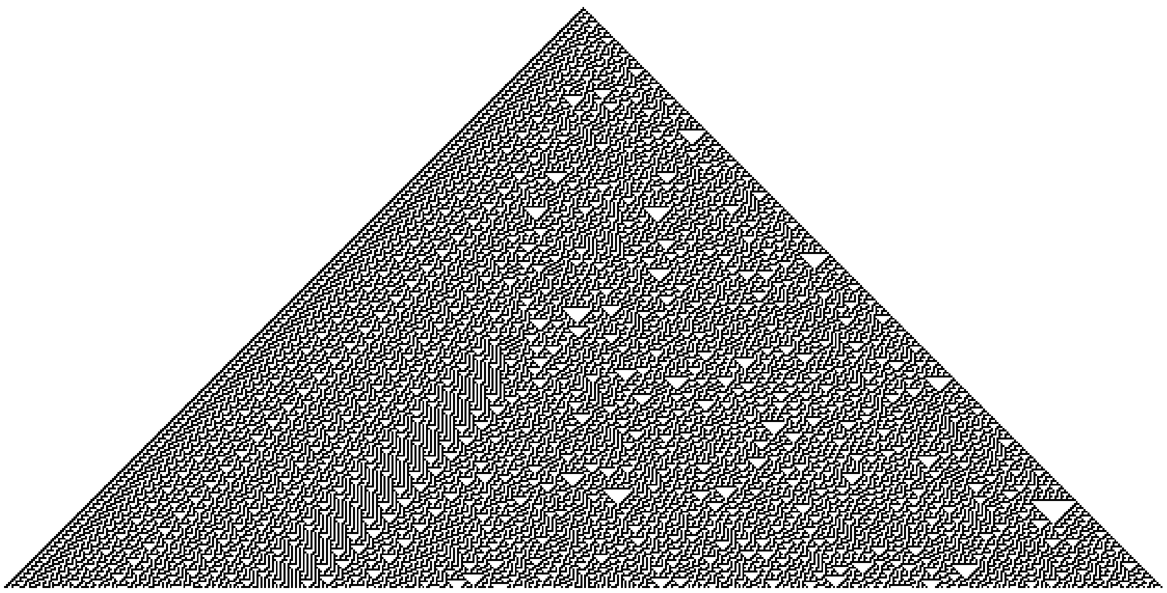
If the possible states of the cell and its two neighbors are ordered this way, the Wolfram code for the automaton is simply the new states read as binary number.

It is easy to see that the rule can be expressed as left cell XOR (middle cell OR right cell).

If started from one black cell (state 1), the first few iterations look like this:



And here are the first 300 iterations:



Diagonals from the right

The diagonals on the right side of rule 30 started from a single black cell are periodic and double their period at irregular intervals. The sequence of periods is A094605 (http://oeis.org/A094605) on OEIS and begins with

1, 2, 2, 4, 8, 8, 16, 32, 32, 64, 64, 64, 64, 64, 64, 128, 256, 256, 256, 256, 256, 256, 256, 256, 512, 1024, 1024, 2048, 2048, 4096, ...

The n_{th} cell of the m_{th} diagonal on the right side can be recursively defined with the function $\mathrm{R}(\mathrm{m},\mathrm{n})$ as follows:

$$\mathrm{R}(\mathrm{m},0) = 0$$

$$\mathrm{R}(0,\mathrm{n}) = 0$$

$$\mathrm{R}(1,\mathrm{n}) = 1$$

$$\mathrm{R}(\mathrm{m},\mathrm{n}) = \begin{cases} \mathrm{R}(\mathrm{m},\mathrm{n}-1) + \mathrm{R}(\mathrm{m}-1,\mathrm{n}) + \mathrm{R}(\mathrm{m}-2,\mathrm{n}) + \mathrm{R}(\mathrm{m}-1,\mathrm{n}) \cdot \mathrm{R}(\mathrm{m}-2,\mathrm{n}) & \text{for even } \mathrm{m} > 1 \\ \mathrm{R}(\mathrm{m},\mathrm{n}-1) + \mathrm{R}(\mathrm{m}-1,\mathrm{n}-1) + \mathrm{R}(\mathrm{m}-2,\mathrm{n}) + \mathrm{R}(\mathrm{m}-1,\mathrm{n}-1) \cdot \mathrm{R}(\mathrm{m}-2,\mathrm{n}) & \text{for odd } \mathrm{m} > 1 \end{cases}$$

where $\mathrm{R}(\mathrm{m},\mathrm{n}-1)+\ldots$ accounts for the XOR with the right cell and $\mathrm{R}(\mathrm{m}-1,\mathrm{n}) + \mathrm{R}(\mathrm{m}-2,\mathrm{n}) + \mathrm{R}(\mathrm{m}-1,\mathrm{n}) \cdot \mathrm{R}(\mathrm{m}-2,\mathrm{n})$ accounts for the OR of the middle and right cell. It can be seen that the m_{th} diagonal is computed from diagonal $\mathrm{m}-1$ and $\mathrm{m}-2$.

The n_{th} cell of the m_{th} diagonal on the right side in the cellular automaton is then $\mathrm{R}(\mathrm{m},\mathrm{n}) \bmod 2$.

This construction is similar to Pascal's triangle, where the value of a cell $\bmod 2$ is the corresponding cell in elementary cellular automaton 60.

The recursion over n can be replaced with a sum, so for right side diagonals, I adopt the notation $\mathrm{R}_{\mathrm{m}}(\mathrm{n})$ instead of $\mathrm{R}(\mathrm{m},\mathrm{n})$.

$$R_m(n) = \begin{cases} \sum_{i=1}^{n} R_{m-1}(i) + R_{m-2}(i) + R_{m-1}(i) \cdot R_{m-2}(i) & \text{for even } m > 1 \\ \sum_{i=1}^{n} R_{m-1}(i-1) + R_{m-2}(i) + R_{m-1}(i-1) \cdot R_{m-2}(i) & \text{for odd } m > 1 \end{cases}$$

Let's look at the first few diagonals:

$$R_2(n) = \sum_{i=1}^{n} R_1(i) + R_0(i) + R_1(i) \cdot R_0(i) = \sum_{i=1}^{n} 1 = n$$

$$R_3(n) = \sum_{i=1}^{n} R_2(i-1) + R_1(i) + R_2(i-1) \cdot R_1(i) = \sum_{i=1}^{n} i - 1 + i$$

$$= \frac{n^2}{2} + \frac{n}{2} - n + \frac{n^2}{2} + \frac{n}{2} = n^2$$

$$R_4(n) = \sum_{i=1}^{n} R_3(i) + R_2(i) + R_3(i) \cdot R_2(i) = \sum_{i=1}^{n} i^2 + i + i^3$$

$$= \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} + \frac{n^2}{2} + \frac{n}{2} + \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4} = \frac{3n^4 + 10n^3 + 15n^2 + 8n}{12}$$

Using Faulhaber's formula, every $\sum_{i=1}^{n} i^x$ can be converted into a $(x+1)_{th}$ degree polynomial of n. As the only operations involved are addition, multiplication and summation over polynomials, every $R_m(n)$ is a polynomial of n.

Looking at the situation modulo 2 only seems to delay the increasing complexity of the polynomials:

$$R_2(n) = n \pmod 2$$

$$R_3(n) = n^2 = n \pmod 2$$

$$R_4(n) = \sum_{i=1}^{n} i + i + i^2 = \sum_{i=1}^{n} i + i + i = \sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2} = \frac{n^2 + n}{2} \pmod 2$$

$$R_5(n) = \sum_{i=1}^{n} \frac{(i-1) \cdot i}{2} + i + \frac{(i-1) \cdot i^2}{2} = \sum_{i=1}^{n} \frac{i^2}{2} - \frac{i}{2} + i + \frac{i^3}{2} - \frac{i^2}{2} = \sum_{i=1}^{n} \frac{i^3}{2} + \frac{i}{2} \pmod 2$$

$$= \frac{1}{2} \cdot (\frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}) + \frac{1}{2} \cdot (\frac{n^2}{2} + \frac{n}{2}) = \frac{n^4 + 2n^3 + 3n^2 + 2n}{8} \pmod 2$$

Diagonals from the left

For diagonals from the left, $R(m, n-1)$ and $R(m-2, n)$ from the definition of $R(m, n)$ switch their roles. The new function will be called L, so all R are also replaced with L.

$$L(m, 0) = 0$$

$$L(0, n) = 0$$

$$L(1,n) = 1$$

$$L(m,n) = \begin{cases} L(m-2,n) + L(m-1,n) + L(m,n-1) + L(m-1,n)\cdot L(m,n-1) & \text{for even } m > 1 \\ L(m-2,n) + L(m-1,n-1) + L(m,n-1) + L(m-1,n-1)\cdot L(m,n-1) & \text{for odd } m > 1 \end{cases}$$

This is the same definition as for diagonals from the right, but with multiplication by $L(m, n-1)$ instead of $L(m-2, n)$ at the end. Because $L(m, n-1)$ is now used twice in the definition, the recursion over n cannot be easily removed (the same applies to the recursion over m). It seems that $L(m, n)$ grows exponential with n.

The development of the periods of $L(m, n) \bmod 2$ with fixed m also is more complex. Periods don't always start at $L(m, 1)$, they double more slowly and can also decrease. Here are the periods and starting points of the first few diagonals:

Periods: 1, 1, 1, 2, 1, 2, 2, 1, 4, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 1, 8, 1, ...

Period start: 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 3, 4, 5, 8, 6, 11, 8, 13, 12, 14, 12, 15, 15, 16, 16, 17, 15, 18, ...

Intersections

By construction, R and L must be equal where the diagonals intersect:

$$L(m,n) = \begin{cases} R(2n, \frac{m}{2}) & \text{for even } m \\ R(2n-1, \frac{m+1}{2}) & \text{for odd } m \end{cases}$$

$$(m > 0, n > 0)$$

This can also be proven by induction (See appendix 1). The intersection in the central column is just a special case of this rule:

$$L(m, \lceil \frac{m}{2} \rceil) = R(m, \lceil \frac{m}{2} \rceil)$$

It is intriguing that one diagonal from the left can be expressed by the growing degree and complexity of the polynomials for the diagonals from the right. One may also ask how those progressions of polynomials eventually become periodic when taken modulo 2. But it can easily be seen that their eventual periodicity is necessary:

To calculate left diagonals modulo 2, right diagonals only have to be computed up to size $\frac{m}{2}$ or $\frac{m+1}{2}$. If only the values modulo 2 are of interest, this is a closed system of fixed size: Two partial diagonals to calculate the next partial diagonal of same size. This system eventually has to become periodic.

Periodicity of diagonals from the right

For the analysis of periodicity modulo 2, I am changing back to the simpler picture and redefine $R_m(n)$ with addition modulo 2 (XOR) and $A + B + A \cdot B$ replaced by the logical OR $A \vee B$. I call the resulting function $R'_m(n)$.

$$R_m(n) \bmod 2 = R'_m(n) = \begin{cases} \sum_{i=1}^{n} R'_{m-1}(i) \vee R'_{m-2}(i) & \text{for even } m > 1 \\ \sum_{i=1}^{n} R'_{m-1}(i-1) \vee R'_{m-2}(i) & \text{for odd } m > 1 \end{cases}$$

In words: To get $R'_m(n)$ for even $m > 1$, do a summation (modulo 2) over the logical OR of the two preceding diagonals up to cell n. For odd $m > 1$, shift the last diagonal by one before doing the OR.

If $R'_{m-1} \vee R'_{m-2}$ is periodic with basic period p, the summation can cause a period doubling:

Behaviour of periods under summation modulo 2

Let $f(x)$ with range $\{0, 1\}$ be periodic with basic period p:

$$\forall x > 0 : f(x + p) = f(x)$$

$$\forall p_1 < p : \exists x f(x + p_1) \neq f(x)$$

Then if $\frac{n}{p} = a$ with remainder b:

$$\sum_{i=1}^{n} f(i) = f(1) + \ldots + f(p) + f(1 + p) + \ldots + f(n)$$

$$= a \bmod 2 \cdot \sum_{i=1}^{p} f(i) + \sum_{i=1}^{b} f(i)$$

It can be seen that the second term (the sum up to b) is periodic with period p. If the sum of $f(x)$ up to p is 1, the first term amounts to addition of 1 while a is odd and 0 while a is even. If the sum of $f(x)$ up to p is 0, the second term stays unchanged.

So the necessary and sufficient condition for period doubling of the whole sum is that the sum up to p is 1 and this causes the new basic period to be the sequence generated by the second term concatenated with the inverse of itself.

$$\exists n : p = 2^n$$

$$g(x) = \sum_{i=1}^{x} f(i)$$

Case 1

$$\sum_{i=1}^{p} f(i) = 1$$

It follows:

$$\forall x > 0 : g(x + 2p) = g(x)$$

$$\forall x > 0 : g(x + p) \neq g(x)$$

$2p$ is also the basic period of $g(x)$:

$$\forall p_1 < 2p : \exists x g(x + p_1) \neq g(x)$$

Proof by contradiction:

$$\exists p_1 < 2p : \forall x\, g(x + p_1) = g(x)$$

Let p_1 divide 2p. Then p_1 is a power of 2. As p is not a period of $g(x)$, p_1 cannot be p - so it must also divide p. But then p would again be a period of $g(x)$ - which is impossible. But if p_1 does not divide 2p, $g(x)$ would have two periods that are not multiples of each other - which is also impossible. So $2p$ must be the basic period of $g(x)$.

Case 2

$$\sum_{i=1}^{p} f(i) = 0$$

It follows:

$$\forall x > 0 : g(x + p) = g(x)$$

$$g(p) = 0$$

The last statement means that each period p ends with 0. If where was another period p_1 smaller than p, it would imply that p is a multiple of p_1 and that period p_1 also has to end with 0 - so the summation at $p_1 + 1$ starts fresh and yields p_1 again. This forces $f(x)$ to be periodic with period $p_1 < p$, which is a contradiction - so the basic period of $g(x)$ must be p again.

Summary

It is established that $g(x)$ will have twice or the same basic period of $f(x)$ depending on whether the sum of $f(x)$ up to p modulo 2 is 1.

Overall behaviour of periods

Here are the first few periods of $R'_m(n)$:

```
1
10
10
1100
10110100
10101000
1010011101011000
11001010101011110011010101010000
10111010011010101101010101010000
1010110010110101010110011001111101010011010010101010011001100000
```

The typical concatenation of a sequence with its inverse at every period doubling can be seen. So in accordance with the last result, we suspect that $R'_{m-1} \vee R'_{m-2}$ (with R'_{m-1} shifted or not) always has a basic period p that is the maximum of the basic periods of R'_{m-1} and R'_{m-2}. This would mean that the summation over $R'_{m-1} \vee R'_{m-2}$ (with R'_{m-1} shifted or not) up to p would determine whether there is a period doubling or not.

The proof idea is that every diagonal from the right must terminate with an increasing number of zeros and that the number of zeros at the end of half its period can never catch up to match it. Thus $R'_{m-1} \vee R'_{m-2}$ (with R'_{m-1} shifted or not) always must have a basic period p that is the maximum of the basic periods of R'_{m-1} and R'_{m-2}.

Growth of zeros at the end

For even m:

$$R'_m(n) = \sum_{i=1}^{n} R'_{m-1}(i) \vee R'_{m-2}(i)$$

Let z be the number of zeros at the end of a period (with a one before them). Then this transition table applies regardless whether period doubling occurs or not:

$R'_{m-2}(i)$	$R'_{m-1}(i)$	$R'_m(n)$
$z = 0$	$z = 0$	$z = 1$
z	z	$z + 1$
z	$z + 1$	$z + 1$

For odd m:

$$R'_m(n) = \sum_{i=1}^{n} R'_{m-1}(i-1) \vee R'_{m-2}(i)$$

the transition table looks like this (again regardless whether period doubling occurs or not):

$R'_{m-2}(i)$	$R'_{m-1}(i)$	$R'_m(n)$
$z = 0$	$z = 0$	$z = 1$
z	z	z
z	$z + 1$	$z + 1$

For even and odd m:

Every start in the whole transition system leads to cyclic behaviour that adds a zero every second step - possibly with an initiation period of 1-2 steps.

Growth of zeros in the middle

After each period doubling, the number of zeros at the end are matched by the same number of ones at the end of half the period. But the situation regarding trailing zeros for the partial sum is like for the full sum - except that in the situation where the full sum would double its period, the partial sum gets all trailing zeros replaced by ones. So the number of zeros at the end of the partial sum can never catch up before period doubling occurs again - which has to happen eventually.

Outlook

It was established that the period of diagonals from the right doubles if the number of ones (or black cells) in a combination of the last two diagonals is odd and otherwise stays the same.

This combination is the logical OR of the basic period of the last diagonal (shifted by one for odd diagonals) with the basic period of the second to last diagonal (doubled if the period has half the size of the last diagonal).

At first glance, the number of ones (or black cells) in that combination does not seem to be reduceable to a generalized property of the previous two diagonals so the period doubling cannot be predicted without calculating the values in the periods - but that may well be a premature conclusion. To my current knowledge, this is still an unsolved problem.

Appendix 2 contains a Python program to compute and output the period lengths of diagonals from the right until it runs out of memory. With a 32bit Python interpreter, it came this far:

1, 2, 2, 4, 8, 8, 16, 32, 32, 64, 64, 64, 64, 64, 64, 128, 256, 256, 256, 256, 256, 256, 256, 256, 512, 1024, 1024, 2048, 2048, 4096, 4096, 4096, 4096, 4096, 8192, 8192, 16384, 32768, 32768, 65536, 65536, 131072, 131072, 262144, 262144, 262144, 262144, 262144, 524288, 1048576, 1048576, 2097152, 2097152, 2097152, 4194304, 8388608, 8388608, 8388608, 16777216, 16777216, 33554432, 33554432, 33554432, 67108864, 134217728

Appendix 3 contains a Python program to compute and output the offsets and period lengths of diagonals from the left from partial diagonals from the right. The memory requirement of this calculation only grows linear with m so it can run much longer. Here are the first 410 (offset,period) pairs for diagonals from the left:

(0, 1), (0, 1), (1, 1), (0, 2), (0, 1), (0, 2), (0, 2), (0, 1), (0, 4), (0, 1), (1, 4), (2, 4), (0, 4), (3, 4), (4, 4), (5, 4), (8, 4), (6, 4), (11, 4), (8, 4), (13, 4), (12, 4), (14, 4), (12, 2), (15, 4), (15, 4), (16, 4), (16, 4), (17, 1), (15, 8), (18, 1), (18, 8), (20, 8), (19, 8), (23, 8), (19, 8), (28, 8), (22, 8), (29, 8), (30, 8), (35, 8), (31, 8), (36, 8), (34, 8), (39, 8), (36, 4), (41, 8), (38, 8), (47, 8), (40, 8), (48, 8), (48, 8), (50, 8), (48, 8), (51, 8), (50, 8), (54, 8), (50, 8), (55, 8), (55, 8), (56, 8), (56, 8), (56, 8), (59, 8), (56, 4), (60, 8), (61, 4), (63, 8), (62, 8), (64, 8), (64, 8), (63, 8), (64, 8), (62, 8), (65, 8), (63, 8), (66, 8), (69, 8), (68, 8), (71, 8), (70, 4), (74, 8), (72, 8), (75, 8), (74, 8), (82, 8), (76, 8), (83, 8), (83, 8), (79, 8), (82, 8), (81, 8), (81, 8), (81, 8), (83, 4), (82, 8), (82, 8), (82, 8), (83, 8), (83, 8), (84, 8), (86, 8), (84, 8), (89, 8), (86, 8), (94, 8), (92, 8), (97, 8), (95, 8), (98, 8), (99, 8), (99, 8), (100, 8), (101, 8), (101, 8), (104, 8), (100, 8), (105, 8), (104, 8), (106, 8), (106, 8), (104, 2), (105, 8), (104, 8), (105, 8), (104, 8), (106, 8), (104, 8), (107, 8), (106, 8), (105, 8), (106, 8), (105, 8), (108, 8), (106, 8), (110, 8), (108, 8), (111, 2), (111, 8), (108, 8), (111, 8), (109, 8), (112, 8), (110, 8), (113, 8), (112, 2), (111, 8), (111, 2), (113, 8), (111, 8), (114, 8), (113, 8), (115, 8), (113, 8), (116, 8), (118, 8), (119, 8), (119, 8), (120, 8), (120, 8), (121, 8), (121, 8), (120, 8), (122, 8), (122, 8), (119, 8), (121, 8), (122, 8), (122, 8), (124, 8), (122, 8), (125, 8), (126, 8), (126, 8), (128, 8), (128, 8), (130, 8), (129, 8), (132, 8), (129, 8), (138, 8), (132, 8), (139, 4), (140, 8), (140, 8), (142, 8), (140, 8), (148, 8), (143, 8), (149, 8), (151, 8), (150, 8), (152, 8), (150, 8), (153, 4), (152, 8), (155, 8), (152, 8), (156, 8), (159, 8), (157, 8), (160, 8), (159, 8), (163, 8), (161, 8), (164, 4), (167, 8), (165, 8), (169, 8), (167, 8), (172, 8), (169, 4), (173, 8), (175, 8), (174, 8), (176, 8), (176, 4), (174, 8), (175, 8), (174, 8), (175, 8), (174, 8), (170, 8), (174, 8), (175, 8), (175, 8), (176, 8), (176, 8), (177, 8), (176, 8), (174, 8), (176, 8), (177, 8), (177, 8), (176, 8), (179, 8), (179, 8), (176, 8), (178, 8), (179, 8), (179, 8), (177, 8), (179, 8), (178, 8), (181, 8), (178, 8), (183, 8), (180, 8), (185, 8), (182, 8), (186, 8), (184, 8), (188, 8), (186, 8), (190, 8), (188, 8), (191, 8), (193, 8), (194, 8), (194, 8), (193, 8), (195, 8), (196, 8), (200, 8), (196, 8), (203, 8), (204, 8), (205, 8), (207, 8), (207, 8), (208, 8), (208, 8), (210, 8), (211, 8), (215, 8), (212, 8), (218, 8), (220, 8), (226, 8), (221, 8), (230, 8), (225, 8), (231, 8), (231, 8), (232, 8), (233, 8), (234, 4), (236, 8), (238, 4), (238, 8), (240, 4), (239, 8), (238, 8), (238, 8), (238, 8), (238, 8), (238, 8), (237, 8), (238, 8), (238, 8), (239, 8), (239, 8), (240, 8), (238, 8), (242, 8), (240, 8), (243, 8), (242, 8), (241, 8), (241, 8), (240, 8), (241, 8), (242, 8), (241, 8), (241, 8), (241, 8), (242, 8), (242, 8), (243, 8), (244, 8), (243, 8), (245, 8), (245, 8), (244, 8), (242, 8), (243, 8), (241, 8), (243, 8), (243, 8), (246, 8), (240, 8), (250, 4), (248, 8), (252, 8), (251, 8), (253, 8), (253, 8), (254, 8), (253, 8), (258, 8), (257, 8), (259, 8), (260, 8), (260, 8), (260, 8), (259, 8), (259, 8), (256, 8), (260, 8), (262, 8), (261, 8), (264, 8), (267, 8), (265, 8), (269, 8), (267, 8), (273, 8), (269, 8), (278, 8), (272, 8), (280, 8), (278, 4), (283, 8), (280, 4), (285, 8), (282, 8), (286, 8), (288, 8), (287, 8), (289, 8), (289, 8), (288, 8), (289, 8), (289, 8), (292, 8), (290, 8), (293, 8), (292, 8), (292, 4), (291, 8), (291, 8), (292, 8), (294, 8), (295, 8), (295, 8), (297, 8), (295, 8), (298, 8), (297, 8), (299, 4), (299, 8), (298, 8), (296, 8), (297, 8), (296, 8), (297, 8), (298, 8), (296, 8), (302, 8), (301, 1), (298, 16), (300, 1), (299, 16), (300, 16), (300, 16), (302, 16), (300, 16), (303, 16), (303, 16), (304, 16)

Appendix

Appendix 1

Proof of:

$$L(m,n) = \begin{cases} R(2n, \frac{m}{2}) & \text{for even } m \\ R(2n-1, \frac{m+1}{2}) & \text{for odd } m \end{cases}$$

$$(m > 0, n > 0)$$

We start by proving:

$$L(m,1) = \begin{cases} R(2, \frac{m}{2}) & \text{for even m} \\ R(1, \frac{m+1}{2}) & \text{for odd m} \end{cases}$$

Proof:

$$R(2, \frac{m}{2}) = \frac{m}{2} \text{ (see section "Diagonals from the right")}$$

$$R(1, \frac{m+1}{2}) = 1 \text{ (by definition)}$$

$$L(m,1) = \begin{cases} L(m-2,1) + L(m-1,0) = 1 = R(1, \frac{m+1}{2}) & \text{for odd } m > 1 \\ L(m-2,1) + L(m-1,1) = L(m-2,1) + 1 = \frac{m}{2} = R(2, \frac{m}{2}) & \text{for even } m > 1 \end{cases}$$

$$L(1,1) = 1 = R(1,1) \text{ (by definition)}$$

Induction step. From:

$$L(m,n) = \begin{cases} R(2n, \frac{m}{2}) & \text{if m even} \\ R(2n-1, \frac{m+1}{2}) & \text{if m odd} \end{cases}$$

and:

$$L(m_1, n+1) = \begin{cases} R(2n+2, \frac{m_1}{2}) & \text{for even } m_1 < m \\ R(2n+1, \frac{m_1+1}{2}) & \text{for odd } m_1 < m \end{cases}$$

we prove:

$$L(m, n+1) = \begin{cases} R(2n+2, \frac{m}{2}) & \text{if m even} \\ R(2n+1, \frac{m+1}{2}) & \text{if m odd} \end{cases}$$

Proof for even m:

$$L(m, n+1)$$

$$= L(m-2, n+1) + L(m-1, n+1) + L(m,n) + L(m-1, n+1) \cdot L(m,n)$$

$$= R(2n+2, \frac{m-2}{2}) + R(2n+1, \frac{m}{2}) + R(2n, \frac{m}{2}) + R(2n+1, \frac{m}{2}) \cdot R(2n, \frac{m}{2})$$

$$= R(2n+2, \frac{m}{2})$$

Proof for odd m:

$$L(m, n+1)$$

$$= L(m-2, n+1) + L(m-1, n) + L(m, n) + L(m-1, n) \cdot L(m, n)$$

$$= R(2n+1, \frac{m-1}{2}) + R(2n, \frac{m-1}{2}) + R(2n-1, \frac{m+1}{2}) + R(2n, \frac{m-1}{2}) \cdot R(2n-1, \frac{m+1}{2})$$

$$= R(2n+1, \frac{m+1}{2})$$

Appendix 2

A Python program to compute and output the period lengths of diagonals from the right until it runs out of memory. If you do not have the bitarray module, install it with "pip install bitarray".

```python
from bitarray import bitarray

# this function does the summation modulo 2 over the already combined previous
# two periods. the result is the basic period of the next diagonal.
def newdiagonal(combination):
    l=len(combination)
    s=bitarray('')
    value=0
    for i in range(2*l):
        if i==l and not value:
            # no period doubling, stop
            break
        value=value^combination[i%l]
        s.append(value)
    return s

# this function combines the previous two basic periods with OR and then
# calculates the basic period of the next diagonal using newdiagonal()
def newperiod(nr,lastperiod,lastlastperiod):
    l1=len(lastperiod)
    l2=len(lastlastperiod)
    assert l2 <= l1
    if l1 > l2:
        assert l1 == l2*2
        # if the second to last period only has half size, repeat it
        lastlastperiod=lastlastperiod*2

    if not nr%2:
        # normal OR for even diagonals
        combination = lastperiod | lastlastperiod
    else:
        # shift previous period before OR for odd diagonals
        combination = (bitarray('0')+lastperiod[:-1]) | lastlastperiod
    return newdiagonal(combination)

# diagonal 1
lastperiod=bitarray('1')
# diagonal 0
lastlastperiod=bitarray('0')
# period of diagonal 1
periods=[1]
# next diagonal is 2
nr=2

while True:
    s=newperiod(nr,lastperiod,lastlastperiod)
    p=len(s)
    periods.append(p)
    print(periods)
    lastlastperiod=lastperiod
    lastperiod=s
    nr+=1
```

Appendix 3

A Python program to compute and output the offsets and period lengths of diagonals from the left. If you do not have the bitarray module, install it with "pip install bitarray".

```python
from bitarray import bitarray

# check whether s is periodic with period p
def isperiodic(s,p):
    l=len(s)
    for pos in range(l):
        pos1=pos+p
        if pos1 >= l:
            break
        if s[pos]!=s[pos1]:
            return False
    return True

# find the basic period of s starting with offset t
def findperiod(s,t=0):
    if t > len(s):
        return None,None
    if not s:
        return t,1
    p=1
    while p <= len(s)//2:
        if isperiodic(s,p):
            return t,p
        # we do not assume periods of diagonals on the left are powers of two
        # although they seem to be
        p=p+1
    return findperiod(s[1:],t+1)

# compute new diagonal from the right up to fixed length
def newdiagonal(nr,last,lastlast):
    l1=len(last)
    l2=len(lastlast)
    assert l2 == l1

    if not nr%2:
        combination = last | lastlast
    else:
        combination = (bitarray('0')+last[:-1]) | lastlast

    s=bitarray('')
    value=0
    for i in range(l1):
        value=value^combination[i]
        s.append(value)
    return s

# offset and period of first diagonal
periods=[(0,1)]
# we start with the second diagonal
nr=2
while True:
    # l is the length of the part of the diagonal from the right we need
    if not nr%2:
        l=nr//2
    else:
        l=(nr+1)//2
    # this is the diagonal number on the right
    n=1
    # the first diagonal is ones only
    last=bitarray('1'*l)
    # the one before that is zeros only
    lastlast=bitarray('0'*l)
    # here we store the computed values for the left diagonal
    diagonal=bitarray('')
    # we also track the state of the right diagonal system to see when it
    # has become periodic
    states={}
    # we stop when we have that many values for the left diagonal (TBD)
    stopatlen=None

    while True:
        if not nr%2 and not n%2:
            diagonal.append(last[-1])
        if nr%2 and n%2:
            diagonal.append(last[-1])

        # if necessary length has been determined we determine offset+period
        # and are done with this diagonal from the left
        if stopatlen is not None and len(diagonal)==stopatlen:
            t,p=findperiod(diagonal)
            assert t is not None
            periods.append((t,p))
            print(periods)
            print()
            break

        # when states start repeating we know how many values we need at most
        if stopatlen is None:
            k=(tuple(last+lastlast))
            if k in states:
                stopatlen=(len(diagonal)+1)*2
            else:
                states[k]=n

        # calculate next partial diagonal from the right
        n+=1
        s=newdiagonal(n,last,lastlast)
        lastlast=last
        last=s

    nr+=1
```