Plug and Play

Ethan D. Bolker
Debra K. Borkovitz
Katelyn Lee
Adam Salachi

July 27, 2021

1 Introduction

Note: *Describe the source of the problem and the group makeup*

> Given a sequence $0, 1, 1, 2, 2, 3, 3, ..., n, n$ find the number of ways to arrange the elements in the sequence so that there's exactly one digit between the two 1's, two digits between the two 2's, three digits between the two 3's, so on and so forth, and there are exactly n digits between the two n's.
>
> For $n = 2$, a possible solution would be 12102. For $n = 5$, a possible solution would be 53141352402.[1]

Note: *Reformulate and generalize to plugs on a strip. Pictures of our 3d printed incarnations.*

2 Plugs, Strips and Puzzles

Our mathematical model for a plug is a bit string, for example 1011 . Our current convention is that there are no leading or trailing 0 bits. (We may want to relax that convention at some time.)

The *length* of a plug is the length of the string; this one has length 4. The *number of prongs* is the number of 1 bits — 3 in this example.

Each plug has a *plugnumber* when we interpret its bit string as a binary integer. So 1011 is number $1 + 4 + 8 = 13$.[1]

Since plugs begin and end with 1 bits, the plugs of length n have plugnumbers the odd integers between $2^{n-1} + 1$ and $2^n - 1$. [2]

[1] This convention reads increasing bit significance from left to right, so the first bit is the units bit. We can change our minds and set the plugnumber for 1011 to $1 + 2 + 8 = 11$ if we wish, but let's decide soon and stick to our decision.

[2] This equivalence will need revision if we allow leading and trailing 0 s in plug bitstrings.

We will want ways to talk about plugs other than specifying their bit strings or plugnumbers. In any context we can give a plug any name we like, or have variables whose values can be plugs.

The Stackexchange question that triggered our project asked about the 2 prong plugs. We'll call those the *classic plugs* and give them their own names: T_k for the two prong plug of length k, It has $k - 2$ 0 bits between its 2 end prongs. T_k has plugnumber $2^{k-1} + 1$. The stackexchange question uses the single prong plug 1 instead of the mathematically more natural length 2 two prong plug T_2 with no gap between the prongs.

A *power strip* (or just a *strip*) models a place to plug in plugs. Think of it as a finite sequence of slots some of which are occupied by the prongs of plugs. So a power strip is a pair consisting of an array of slots and a set of (plug, offset) pairs such that all the plugs can be inserted simultaneously at the specified offsets. It is a *solved* if every slot in the strip is filled.

A *plug puzzle*, or, for us, just a puzzle, specifies the types amd numbers of plugs you are allowed to use to fill a strip — something like

$$\mathbb{E}(\text{list of allowable plug types}, \text{restrictions on numbers of plugs of each type})$$

A *solution* of length n is a strip with n slots filled with plugs that are consistent with the restrictions in the puzzle specification.

Given a puzzle we try to understand the number and shape of solutions. often as a function of the size of the strip.

Note: *Remember to say somewhere (perhaps not here) that plugs and solutions have a left to right direction. You can't turn them around. Perhaps add an electric system not quite analogy.*

The stackexchange post posed a sequence of puzzles:

$$\mathbb{T}_k = \mathbb{E}(\{\text{the one prong plug}, T_3, T_4, \ldots. T_k\}, \text{exactly 1 of each }).$$

The only possible solutions are strips of length $2k + 1$. The question asks for an efficient algorithm to count them.

3 Anything goes

That puzzle spurred our investigations,but it is much too hard to start out with. In particular, we discovered that restricting the number of plugs of each type is a stumbling block. So for a while we will study just puzzles that specify the allowed plug types, with no limit on the number of each that can appear in a solution.

In the *anything goes* puzzle $\mathbb{A}$ you may use any plugs as oftem as you wish:

$$\mathbb{A} = \mathbb{E}(\text{all plug types})$$

Counting solutions begins to put us in touch with some classical notions in combinatorics.

Theorem 1. *Length n solutions for the plug puzzle that allows arbitrary many instances of any plug correspond to the partitions of the n-element set $[n] = \{1, 2, \ldots, n\}$.*

Note: *Example here — wait until the good LaTeX plug representation is done.*

The Bell numbers $\{1, 2, 3, 5, 15, 52, \ldots$ count these.

4 Factoring

There is more information in a a solved strip than the shape of the partition that defines it. For example, two copies of 101 solve a strip of length 4. So do two copies of 11 . Both solutions correspond to a partition of the four element set of slots into two sets of two slots each, but their geometry is different.

In a power strip of length n there are $n-1$ *gaps* between slots. The *thickness* or *thickness array* is a the sequence of integers that counts the number of plugs that cover each of the gaps. A solution (that is, a filled strip) is *prime* or *atomic*if its thickness is never 0. Any solution is a concatenation of prime filled strips: the *factors*.

When a puzzle specifies no restrictions on the number of each kind of plug then factors and products (concatenations) of solutions are themselves solutions. When that happens we can relate the number of prime solutions and the number of solutions.

Let $S(n)$ be the number of solved strips of length n and $S'(n)$ the number of prime solutions of length n.

Let $P(n)$ be the number of solved strips that use n plugs and $P'(n)$ the number of prime solutions that use n plugs.

NOT QUITE RIGHT YET. P and P' must be carefully defined to count the number of prime plugs actually used in solutions of some length or bounded length. To see why, note that for the anything goes puzzle there are infinitely many one plug primes: the plugs with all bits 1 .

When necessary we will write $S(\mathbb{E}, n)$ when we need to make clear which puzzle we are counting.

Some of these counts may be 0. Consider the puzzle that uses just the classic plug 11 . Then $S(n) = 1$ if n is even, 0 if n is odd. $S'(2) = 1$ and 0 otherwise.

We did a brute force calculation for the number of prime solutions to the anything goes puzzle, and found $\{1, 1, 2, 6, 22, \ldots\}$. That sequence seems to be A074664 in The On-Line Encyclopedia of Integer Sequences. The comments there about permutations and partitions strongly support

Conjecture 2. *The prime solutions to problem $\mathbb{A}$ are characterized by the equivalent conditions defining the OEIS sequence A074664.*

This easy theorem might help prove the conjecture. It shows that each of S and S' (or P and P') determines the other.

Theorem 3. *Suppose that every factor of a solved strip solves its smaller strip and any concatenation of solved strips is a solution.*

Then the first k for which $S(k) > 0$ is the same as the first k for which $S'(k) > 0$. For that k we have $S(k) = S'(k)$ and for all n

$$S(n) = S'(1)S(n-1) + S'(2)S(n-2) + \cdots + S'(n-1)S(1) + S'(n). \quad (4.1)$$

The same assertions hold for the counts $P(n)$ and $P'(n)$.

Proof. Each solved strip begins with a prime solution of length $k \leq n$ which can be chosen $S'(k)$ ways and ends with one of the $S(n-k)$ solved strips of length $n-k$. Replace "length" by "number of plugs" to prove the second assertion. □

Equation 4.1 id as kind of convolution. It appears in the OEIS discussion of OEIS A074664. In Section 13 we discuss it further.

Note: *So far we've used thickness only to find factors. But there is probably more information to exploit. What's the maximum thickness? What patterns are possible in the thickness vector?*

5 Disallowing the Plug 1

Note: *Also a section by Debbie. Also again, not really a theorem and notation should probably be changed (I just changed S' to B' because we use S' for soemthing else). Table needs to be better labeled. I have similar work for when the plug 1 is allowed that I ran out of time before I put up.*

Note: *Use some letter other than S below since that's now used for solution counts.*

Theorem 4. *Let $B'(n,k)$ represent the number of set partitions with no singletons, i.e. the number of solutions to the plug problem with k plugs, where the plug 1 is not allowed. Then $B'(n,k) = kB'(n-1,k) + (n-1)B'(n-2,k-1)$, with $B'(2,1) = 1$ and $B'(n,k) = 0$ when $k > \frac{n}{2}$*

Proof. Consider adding position $n-1$ in a strip of length n (where position numbering starts from 0) to form a solution with k plugs. We can add a prong at position $n-1$ to one of the existing plugs in a solution for a strip of length $n-1$ with k plugs in $kB'(n-1,k)$, which counts all possibilities where position $n-1$ is part of a plug with 3 or more prongs. If position $n-1$ is part of a plug with 2 prongs, then we have $n-1$ choices for the other end of the plug, and we have a total of $(n-1)B'(n-2,k-1)$ solutions. □

The solutions form a triangle, with n on the vertical axis and k on the horizontal:

1	2	3	4	5
1	0	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	3	0	0
5	1	10	0	0
6	1	25	15	0
7	1	56	105	0

Theorem 5. *Let $B'(n)$ be the sum of the numbers in row n in the table above, i.e. the total number of strips of length n with singletons disallowed. The first few values of $B'(n)$ are $0, 1, 1, 4, 11, 21, 162$. Let $B(n)$ be the nth Bell Number, which counts the total number of strips of length n. Then*

$$B'(n+1) = B(n) - B'(n),$$

that is, the number of strips of length $n+1$ that <u>do not</u> include a singleton plug is equal to the number strips of length n that <u>do</u> include a singleton plug.

Proof. Let $\mathscr{B}'_n$ and $\mathscr{B}''_n$ be respectively, the sets of all strips of length n without singletons and strips of length n with singletons. We define a map from $\mathscr{B}''_n$ to $\mathscr{B}'_{n+1}$ by considering a strip of length n with at least one singleton, and then creating a new plug by joining all the singletons with prong $n+1$. This plug is not a singleton and together with the non-singleton plugs from the original, creates a new strip in $\mathscr{B}'_{n+1}$. The map is easily reversible, by taking a strip of length $n+1$ with no singletons, dropping prong $n+1$ and making everything it's connected to into singletons. Thus we've found a bijection. □

Note: *Language above needs cleaning up, and not sure about notation either.*

6 Puzzles with just one plug type

Note: *When we do enough of these we may see metapatterns.*

- 11...1 This puzzle has just one prime, the plug itself, of length k.
- 100...1 These are the classic plugs.

  Conjecture 6. *Using just the classic plug of length k, the only prime is the one of length $2k-2$ constructed from $k-1$ copies in the obvious way.*

- 1011 This puzzle has no solutions.
- (a, a, b, a, a) tiles an infinite strip, but has no primes. Note that this *block* notation (which maybe we should change) represents a block of ones of length a followed by a block of zeroes of length a, etc. An example is 1100111110011.

- $(k, mk, k, mk, ..., mk, k)$ The plug formed with n blocks of 1's of length k each and $n-1$ blocks of zeroes of length mk each, where $k, m \in \mathbb{N}$, then $m+1$ of these plugs form a prime of length $(m+1)nk$.

  Note: *(DB)This is a generalization of Adam's result for the classic plug. The notation probably needs some clean up. I have been using the subset of positions notation to represent plugs. Then if P is the set of prong positions with first prong at 0, we can use P_k to represent the plug that adds k to every number in P (k can be negative, and more generally we don't have to use P_0 in our puzzle solution – I think this will be important if we want to use this notation for puzzles using more than one type of plug) and we are looking for a pairwise disjoint set of these translations that covers an interval. In the case of the above example, we have $P_0 \cup P_k \cup P_{2k} \ldots \cup P_{mk}$ covers the interval $[0, (m+1)nk]$ – someone please check as there are a lot of variables here. I am also thinking about defining a scaling operation on plugs that multiplies the lengths of the blocks of 0's and 1's by the same thing. If we looked at the strips as continuous instead of discrete, then this change really does just change the unit, it doesn't change how the plugs fit together. Maybe there's another operation that would just duplicate aspects of the plug.... thinking about simpler ways to think about what I wrote here.... these are related to the classic plugs if you can just multiply the block sizes by the same thing and then also sort of repeat the same thing....*

  Conjecture 7. *The plugs of the form $(k, mk, k, mk, ..., mk, k)$ are the only plugs that form primes with only one type of plug.*

Question 1. *Are there any single plug puzzles with more than one prime?*

Question 2. *Can we characterize the single plug puzzles with no solutions?*

7 Puzzles with two plug types

8 Puzzles using 2 Prong Plugs

Note: *There seem to be two attempts at this count. Perhaps one from Adam (moved here) or perhaps both from Deb. I'm not sure what is being counted. We should look at this with our codified definitions.*

First attempt:

Using only classic plugs of length 2^k where $k > 0$:

length	S'	S
2	1	1
4	1	2
6	1	4
8	3	11
10	12	33
12	14	86
14	–	–

Note: *Deb's work:*

In this section we study the two prong plug puzzle

$$\mathbb{T} = (\{T_k | k = 1, 2, \ldots\})$$

This is a variant of the stackexchange puzzle, using T_0 instead of the single prong plug (1), with no restrictions on the number of each kind of plug.

Note: *What is below doesn't really warrant a theorem... maybe a proposition or a lemma or just within the text, not sure. Also the $t(n)$ notation is temporary pending my comment in the Specifying Variations section. Also should we use length n for the strip for consistency, which would make all odd lengths 0, or use $2n$ as I did below?*

Theorem 8. *For the puzzle $\mathbb{T}$ there are no solutions of odd length. For each n,*

$$S(2n) = (2n-1)!! = (2n-1) \cdot (2n-3) \cdots 3 \cdot 1.$$

Proof. Solutions of length $2n$ correspond to partitions of the set of $2n$ slots into subsets of size two. There are $2n-1$ ways to choose a partner for the plug whose first prong occupies the first slot , $2n-3$ ways to choose a partner for the plug whose first prong occupies the next empty slot, and so on. □

Theorem 3 applied to the sequence

$$0, 1, 0, 3, 0, 15, 0, 105, 0, 945, 0, 10395, \ldots$$

shows that the number of length n prime plugs for this puzzle is the sequence

$$0, 1, 0, 2, 0, 10, 0, 74, 0, 706, 0, 8162, \ldots$$

Conjecture 9. *The nonzero entries in that sequence match A000698 in the OEIS. The comment there explictly mentions the convolution construction from the Bell numbers*

We obtain the following table with the first few nonzero values of $S(n)$, along with the ratio $\dfrac{S'(n)}{S(n)}$.

n	$S(n)$	$\frac{S'(n)}{S(n)}$
1	1	1
2	2	0.67
3	10	0.67
4	74	0.70
5	706	0.75
6	8162	0.79

9 $11\ldots1$

Write

$$R_k = 11\ldots1 \quad (k \text{ 1's})$$

for the plug of length n with n prongs.

Let

$$d = (d_1, d_2, d_3, \ldots)$$

be an infinite bit string (a sequence of 0's and 1's) and $\mathbb{R}(d)$ the puzzle using just the plugs R_k specified by d:

$$\mathbb{R}(d) = \mathbb{E}(\{R_k \mid d_k = 1\}).$$

Then d is the sequence $S'(\mathbb{R}(d), n)$ that counts prime solutions of length n; its convolution counts all the solutions.

Every solution of length n is a concatenation of plugs R_k that corresponds to a *composition* of n: a way to write n as an ordered sum of nonnegative integers k for which $d_k = 1$.

Several interesting combinatorial sequences come up in this context.

- When all plugs R_k are allowed, $d = S' = (1, 1, \ldots)$ and $S(n) = 2^{n-1}$
- When only the single prong plug is allowed, $d = S' = (1, 0, 0, \ldots)$ and $S(n) = 1$ for all n.
- When $d = S' = (1, 1, 0, 0, \ldots)$ the total count $S(n) = F_n$, the nth Fibonacci number.
- When $d = S' = (1, 1, 1, 0, 0, \ldots)$ the total count $S(n)$ is the nth Tribonacci number.
- If $d_k = 1$ for just two values s and t then $S(n)$ is the number of nonnegative integral solutions to the diophantine equation

  $$sx + ty = n.$$

  when the order of the $s + t$ summands is taken into account.

We have been calculating S from S' using convolution, but there is a recursion for S directly:

$$S(n,d) = \sum\{S_{n-k} \mid k < n \text{ and } d_k = 1\} \tag{9.1}$$

We want to study how $S(n,d)$ grows (for fixed d). If d is the sequence of all 1's then $S(n,d) = 2^{n-1}$ grows exponentially with growth factor 2. The Fibonacci sequence grows exponentially with asymptotic growth factor $\varphi = (1+\sqrt{5})/2 \approx 1.618$. If $d = (1,0,0,\ldots)$ then $S(n)$ is identically 1 and "grows" exponentially with growth factor 1.

If d uses only finitely many plugs define the order $o(d)$ of d as the largest k for which $d_k = 1$. Then information about $g(d)$ is encoded in the the polynomial whose coefficients are the negatives of the bits in d together with a leading 1.

$$p_d(x) = x^{o(d)} - \sum_{d_k=1} x^{o(d)-k}$$

For example

$$p_{11}(x) = x^2 - x - 1$$

and

$$p_{1011}(x) = x^4 - x^3 - x - 1.$$

It's known (cite reference) that $S(n,d)$ grows with an asymptotic growth factor $g(d)$ that is the largest real root of $p_d(x)$.

A monic polynomial is a *recursion polynomial* if the coefficient of each power less than the highest is 0 or -1 and the constant term is -1. Recursion polynomials correspond bijectively to recursions defined by Equation 9.

Theorem 10. *For each bit sequence d the sequence $S(n,d)$ grows exponentially with an asymptotic growth factor $g(d)$ that satisfies $1 \le g(d) \le 2$.*

Proof. If d uses only finitely many plugs then recursion 9 implies that $S(n,d)$ does have a finite exponential growth rate $g(d)$, which must be at least 1 since the single prong puzzle solutions grow at that rate. For any d, $S(n,d)$ grows no faster than $S(n, \text{ all 1's})$ so has asymptotic growth rate at most 2.

There's a nice alternative proof for finite bit sequences that uses the recursion polynomial $p_d(x)$. Its value at 1 can't be positive , since the first term and the last term cancel and the other terms are negative. (It's strictly less than 0 except for the recursion polynomial $p_d(x) = x + 1$ for the single prong plug puzzle.). The value $p_d(2) < 2$ since it's the difference between $2^{o(d)}$ and an integer less than that (since we have its binary representation). The polynomial is clearly increasing for $x > 2$ so the largest real root is between 1 and 2. □

Different puzzle problems can have the same growth rate. $g(11) = \varphi$ since that d generates the Fibonacci numbers. Calculation shows $g(1011)$ and $g(101011)$ have the same rate (to many decimal places), though they (necessarily) generate different sequences. We can understand this kind of coincidence several ways.

If you expand the Fibonacci sequence recursion one level deeper you see that

$$F(n) = F(n-1) + F(n-2) = F(n-1) + F(n-3) + F(n-4)$$

so the sequences $d = 11$ and $d = 1011$ define the same recursion with different initial conditions. Asymptotically they grow at the same rate.

The recursion polynomials are related:

$$\begin{aligned} p_{1011}(x) &= x^4 - x^3 - x - 1 \\ &= (x^2+1)(x^2-x-1) \\ &= (x^2+1)p_{11}(x). \end{aligned}$$

Therefore p_{1011} and p_{11} have the same roots, so the same largest real root.

Theorem 11. *If d and h are finite bit strings and*

$$p_h(x) = v(x)p_d(x)$$

for some polynomial $v(x)$ that is positive when $x > 1$ then $g(d) = g(h)$.

Proof. The recursion polynomials have the same largest real root. □

Corollary 12. *For bit string $d = z1$*

$$g(z1) = g(z0z1).$$

Proof. The concatenation of the strings $z0$ and $z1$ corresponds to adding a shift of the recursion polynomial $p_{z1}(x)$ by a factor of $x^{o(d)}$ to itself so that the leading and constant terms align and hence to the factorization

$$p_{z0z1}(x) = (x^{o(d)} + 1)p_{z1}(x),$$

□

These shifts are not the only only way to construct bit strings with the same growth rate. You can't even require that the polynomial $v(x)$ have coefficients 0 and 1. For example

$$\begin{aligned} p_{10001}(x) &= x^5 - x^4 - 1 \\ &= (x^2-x+1)(x^3-x-1) \\ &= (x^2-x+1)p_{011}(x). \end{aligned}$$

Question 3. *Theorem 11 shows that every recursion polynomial is a factor of other recursion polynomials of larger degree. Which recursion polynomials are "prime", so the roots of the tree of their multiples? Can one recursion polynomial ever divide another? (Probably not.)*

We suspect that the converse of Theorem 11 is true.

Conjecture 13. *If d and h are finite bit strings with $o(d) < o(h)$ then $g(d) = g(h)$ only if*

$$p_h(x) = v(x)p_d(x)$$

for some polynomial $v(x)$ that is positive when (insert appropriate positivity hypothesis here).

Note: *Part of that conjecture is pretty strong. It says two recursion polynomials can't share a largest real root unless one is a multiple of the other.*

Question 4. *Infinite bit strings d define puzzles for which there is no recursion of fixed length. But they do exhibit an asympotic growth rate $g(d) \leq 2$. Might $g(d)$ be the largest real root of some constructable power series?*

Note: *Ethan: I think the following questions are less interesting than when I first wrote them down.*

Question 5. *The bitstring sequences d correspond to subsets of the natural numbers. They form a partially ordered set under inclusion, and g is clearly a nondecreasing function with respect to that partial ordering. Can we compare values of g on incomparable subsets?*

If we think of the bit sequences d as binary decimal expansions of real numbers then they parameterize the unit interval (almost). That makes the growth factor function g a function from that interval to $[1, 2]$. What does it look like?

Finite bit strings d index the nodes of an infinite binary tree: take the leff(right) path depending on whether the next entry in d is 0(1). Imagine completing d with all 0's and put the value of $g(d)$ at the node. Then $g(d)$ will be unchanged when you append a 0 to d and move left in the tree. It may increase when you append a 1. Does this structure help us understand g?

10 Non-crossing solutions

Theorem 14. *Let $C(n)$ and $C_{classic}(n)$ represent the number of non-crossing solutions for a strip of length n involving any type of plug and only classic plugs*[3] *respectively. Then $C(n) = C_{classic}(2n)$ where n is any positive integer.*

Proof. There is a bijective function between non-crossing solutions involving only classic plugs and involving any plug. The algorithm to convert between solutions of each type and its inverse follow.

To convert a solution involving any type of plug to a solution of double the length involving only classic plugs, firstly, singletons ('1' plugs) are converted to T_2 plugs. Then, all other plugs are converted into a minimum of two classic plugs: one plug that is double the total length of the initial plug, and other plugs that are each double the length of the number of *gaps* between inner prongs. Note that a classic plug (one without inner prongs) will only result in

[3] Note: I don't recall if classic (or for that matter, non-crossing) has been defined earlier; will match formatting, definitions etc. when I have time online. -Katie

two plugs—one of double the total length and one of double the number of inner gaps, that is $2(n-1)$, where n is the length of the strip. This can be visualized below. **Remember to add: Images of examples for a few individual plugs and a solution**.

For the inverse algorithm, which converts a solution composed of only classic plugs to one that is half the length and involves any kind of plug, first (1), any "loose" T_2 plugs, those that aren't within any other plug, should be converted to singletons. Then (2), any plugs "containing" others should be paired with each plug *directly* below it. By halving the lengths of each of the inner plugs, the lengths of gaps between inner prongs can be found and added to the overall plug of half the length of the originally overarching classic plug. By repeating (1) and (2) until all plugs have been converted, non-crossing classic solutions can be converted to non-crossing free solutions, as below. **Remember to add: Images of examples for a few individual plugs and a solution for inverse**.

It should be noted that these algorithms can be applied to any solution of the originating type. This is clear for the doubling algorithm, and also must be true for the inverse, because all spaces within larger classic plugs must be filled by an appropriate number of other classic plugs—since they are non-crossing, overlapping plugs are not a problem.

Then, because each function can be applied to any non-crossing solution of each type, and (as below) the algorithm and its inverse, or vice-versa, applied in succession to a solution results in the original solution. **Remember to add: Examples going in each direction**. □

11 Asymptotics

Some of the structure in the preceding section generalizes. A puzzle with no restrictions on the number of allowed plugs of each allowable plug type is determined by the set of allowable plug types. That is just a subset of the countable set P of all possible plug types: the partitions of arbitrary integers. So we can think of growth rate as a function g on the power set $\mathcal{P}(P)$. It's weakly increasing on the partial order determined by set inclusion. Its maximum is the growth rate for the Bell numbers, which come from the anythng goes puzzle.

Note: *Perhaps it will be useful to think about aysmptotics in this general context. We might be able to say something about classes of puzzles when exact answers or estimates for particular puzzles are beyond us.*

12 Infinite strips

Note: *Placeholder for work on infinite prime strips.*

With the plugs A =10101 andB = 1011101 you can construct many doubly infinite filled strips since each interlocks with itself and with the other at either end.

The sequence of A's and B's can be made aperiodic in many ways. For example, start with the singly infinite

$$ABAABAAABAAAAB\ldots \tag{12.1}$$

then link it to its reverse.

In fact, starting with the Thue Morse sequence [4] you can build such an aperiodic strip without ever using three of a kind in a row:

$$ABBABAABBAABABBA\ldots$$

Conjecture 15. *This sequence is clearly aperiodic in A and B. Check that it is also aperiodic in bits.*

13 Convolution

The calculation in Theorem 3 is (potentially) interesting for its own sake, not just for our applications to plug problems. The functions t2p() and p2t() that convert between total counts and prime counts each invert the other, so each is bijective when applied to arbitrary sequences of integers (or sequences of real numbers, or sequences of elements from any ring).

p2t always increases entries. It's a kind of integration, counting products given prime factors. Its inverse is a kind of differentiation.

If you play with interesting sequences you can find interesting results/coincidences. For example

```
> python3.6 convolve.py   1 0 1 0 1 0 1 0 1
input [1, 0, 1, 0, 1, 0, 1, 0, 1]
p2t: [1, 1, 2, 3, 5, 8, 13, 21, 34]
t2p [1, -1, 2, -3, 5, -8, 13, -21, 34]
```

When you convolve some counting sequence the result may count something related. There are probably theorems lurking here.

The Catalan numbers (1,1,2,5,14,42,132...) convolve almost to themselves under p2t — only the leading 1 is dropped. An interesting application of the Catalan numbers is that the nth Catalan number C_n for $n \geq 0$ counts the number of valid ways to arrange n open parentheses and n closed parentheses.

Suppose we introduce an additional count C'_n for the number of prime arrangements of n open parentheses and n closed parentheses (i.e. arrangements in which the initial open parenthesis and final closed parenthesis are linked). It follows that C'_n is just the number of ways to arrange the parentheses on the interior, of which there are n-1 of each. So, $C'_{n+1} = C_n$.

It's easy to see that $C_1 = C'_1 = 1$ and thus $C'_2 = C_1 = 1$. Passing the list $[C'_1, C'_2] = [1, 1]$ to p2t() in convolve.py outputs $[C_1, C_2] = [1, 2]$. This output

[4] Recursively, start with A. For each sequence of length a power of 2, make a new version swapping A and B and append.

also tells us $C'_3 = C_2 = 2$ — continuing this process we construct the Catalan numbers (omitting the leading 1) through convolution, as shown in full below:

```
python3.6 convolve.py  1 1 2 5 14 42 132 429 1430
input [1, 1, 2, 5, 14, 42, 132, 429, 1430]
p2t: [1, 2, 5, 14, 42, 132, 429, 1430, 4862]
t2p [1, 0, 1, 2, 6, 18, 57, 186, 622]
```

Under t2p they convolve to OEIS A000957:

```
Fine's sequence (or Fine numbers):
number of relations of valence >= 1 on an n-set;
also number of ordered rooted trees with n edges
having root of even degree.
```

p2t and t2p each essentially permutes the set of sequences. Is there any global structure that's revealed?

Question 6. *p2t and t2p each essentially permutes the set of sequences. Is there any global structure that's revealed?*

Question 7.

Is there a continuous form of this kind of self convolution? Something like

$$g(x) = \int_0^x f(t)f(x-t)dt$$

Theorem 16. *Let $S(n)$ and $S'(n)$ be defined as in Theorem 3, with $S(0) = 1$ and $S'(0) = 0$. Let*

$$S(x) = S(0) + S(1)x + S(2)x^2 + \ldots S(n)x^n + \ldots$$

and

$$S'(x) = S'(0) + S'(1)x + S'(2)x^2 + \ldots S'(n)x^n + \ldots$$

be the ordinary generating functions for $S(n)$ and $S'(n)$ respectively. Then

$$S(x) = \frac{1}{1 - S'(x)}$$

Note: it would be better to use lower case letters for the sequences and upper case for the generating function – less confusing and more standard.

Proof. We rewrite the equation in Theorem 3 as

$$S(x) - 1 = S'(x)S(x)$$

and solve for $S(x)$. See [2], p. 177. □

References

[1] Number of ways to arrange pairs of integers with distance constraint math.stackexchange.com/questions/4124452/number-of-ways-to-arrange-pairs-of-integers-with-distance-constraint

[2] Miklós Bóna, *A Walk Through Combinatorics: An Introduction to Enumeration and Graph Theory, 4th edition*, World Scientific, 2017.