Plug and Play

Ethan D. Bolker
Deborah Borkovitz
Katelyn Lee
Adam Salachi

June 22, 2021

1 Introduction

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

2 Plugs

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]

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 will give them their own names: T_n for the two prong plug of length n, It has $n - 2$ 0 bits between its 2 end prongs. [3]

That question used the the single prong plug number 1 instead of the double prong plug T_2 with no gap between its prongs.

[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.

[3] Perhaps we want the nontraditional n_T, emphasizing the n in our name for the two prong plug of that length.

3 Plug Puzzles

Note: *The discussion here pays a lot of attention to reusing plugs, and whether two instances of the same plug are the same or different. Our work so far seems to have focussed on allowing multiple instances of plugs and not distinguishing among them. So we should probably design vocabulary that makes that easiest. If so, this section needs a rewrite.*

A *plug puzzle* is a finite multiset[4] of plugs. A *labeled* plug puzzle is a set of labeled plugs. The fact that it's a set implies that all the labels are distinct. When no two of the plugs have the same shape (bit string) the strings or the plug numbers can serve as labels.

The distinction between ordinary and labeled plug sets is analogous to the distinction common in combinatorial counting questions where the balls being distributed into boxes are identical or distinguishable.

In Python we would model a labeled plug puzzle as a dictionary, with key the label and value the plug.

For example, consider the labeled puzzle

$$\{(a, 1011), (b, 1001), (c, 1001), (d, 101)\}$$

The *length* of a plug puzzle is the sum of the lengths of its plugs. This puzzle has length 9.

4 Power strip

Note: *Not sure whether to think of a power strip as a place to plug in plugs or as a strip with some plugs in place. For now, the latter seems to require fewer words.*

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.

5 An easy interesting question

The stackexchange question asked for a fast algorithm to count the number of solutions for the plug puzzles consisting of one each of the first n two prong puzzles.

We think that problem is too hard to tackle first. We may return to it. Here's the natural first one to start with, Given an empty strip of length n, find all the plug puzzle solutions, allowing multiples of any kinds of plugs. We'll call that problem $\mathbb{E}$, for easy. The answer is hardly worth calling a theorem, but it begins to put us in touch with some classical notions in combinatorics.

[4] A multiset is a "set that allows for repetitions of elements".

Theorem 1. *For each integer n, solved strips for the plug puzzles that allow 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.

6 Thickness, factoring

There is more information in a a solved power 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 . All of those plugs correspond to a partition 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* if its thickness is never 0. Any solution is a concatenation of prime filled strips: the *factors*.

Whether or not the factors are themselves solutions to a particular problem depends on the problem specifications. For the stackexchange problem nontrivial factors are never solutions since solutions must use all the two prong plugs up to some length. For the most general problem discussed in the preceding section every prime factor is a solution for its shorter strip. When that happens we can relate the number of prime solutions and the number of solutions.

Let b_n be the number of solved strips of length n and p_n the number of prime solutions of length n.

Some of the counts b_n and p_n may be 0. Consider the problem that uses just the classic two prong plug 11 . Then $b_n = 1$ if n is even, 0 if n is odd. $p_2 = 1$ and $p_n = 0$ when $n \neq 2$.

The sequence $\{1, 1, 2, 6, 22, \ldots\}$ counts the prime solutions to problem $\mathbb{E}$. That sequence seems to be A074664 in The On-Line Encyclopedia of Integer Sequences. The comments there about permutations and partitions strongly supports

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

This easy theorem might help prove the conjecture:

Theorem 3. *Suppose that every factor of a solved strip solves its smaller strip. Then*

$$b_n = p_1 b_{n-1} + p_2 b_{n-2} + \cdots + p_{n-1} b_1 + p_n$$

Proof. Each solved strip begins with a prime solution of length $k \leq n$ which can be chosen p_k ways and ends with one of the b_{n-k} solved strips of length $n - k$. □

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?*

7 Specifying Variations

There are (at least) two ways to specify the plugs allowed in a puzzle: which plugs and how many of each. For each of those attributes you can write inclusion or exclusion rules.

Something like

$$\mathbb{E}(\text{prongs } = \{2, 4, 6, \ldots\}, \text{number } = 3 \text{ at most})$$

is the plug puzzle where you can use plugs with just an even number of prongs, at most 3 of each. Then

$$\mathbb{E} = \mathbb{E}(\text{all plugs }, \infty)$$

is the anything goes puzzle while the study of two prong plugs is

$$\mathbb{E}(\{T_k\}, \infty)$$

The puzzle

$$\mathbb{E}(\{T_k\} 1 \text{ exactly})$$

is almost the original stackexchange question — it uses our new meaning for the first plug, and requires one of each but not one of each of the first k.

For any puzzle we try to understand the number and shape of solutions and prime solutions as a function of the size of the strip.

8 Infinite strips

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