Plug and Play

Ethan D. Bolker
Deborah Borkovitz
Katelyn Lee
u Adam Salachi

June 19, 2021

1 Introduction

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.

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.

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]

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

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.

$$\{1011, 1001, 100\}$$

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

A *power strip* (or just a *strip* models a place to plug in plugs. Think of it as a finite sequence of slots subsets of which can be occupied by the prongs of a plug.

A plug can be *inserted* in a strip at an integer offset m if the plugs 1 bits shifted by m correspond to empty slots. A pair consisting of a strip and a set of (plug, offset) pairs and a strip is *compatible* if all the plugs can be inserted at the specified offsets. It is a *solution* 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 puzzle consisting of the first n two prong puzzles.

We think that problem is too hard to tackle first. We may return to it. Here's an easy one to start with - hardly worth calling a theorem, but it begins to put us in touch with some classical notions in combinatorics.

Theorem 1. *For each integer n, the plug puzzle solutions for the strip of length n correspond to the partitions of the n-element set $[n] = \{1, 2, \ldots, n\}$.*

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

6 Thickness

There is more to the previous theorem than just counting partitions.

Define thickness, as a sequence indexed by the spaces bewteen slots in a strip. That allows us to consider unbreakable solutions (thickness always positive) and maximum thickness.