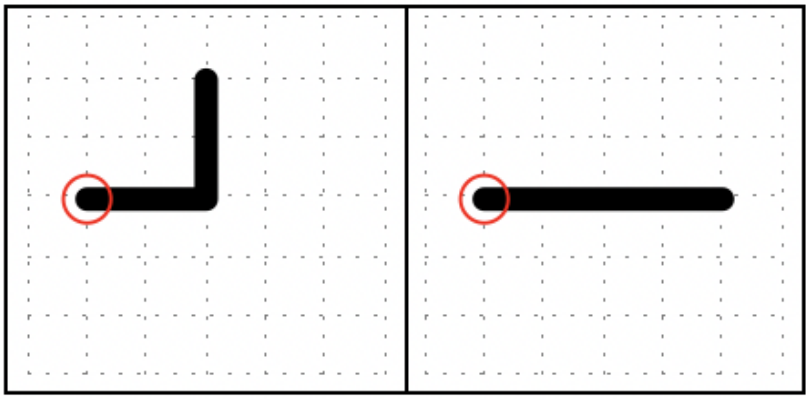
Twitter Thread by platonic solid snake



platonic solid snake
@joshmillard

New thread because new day + progress. My Fogleworm configuration search is now pretty darned working-ish. Here's three stages for N=3: finding unique candidates, generating valid configurations of candidates, and filtering that down to unique configs. N(3) = 2, as hoped!

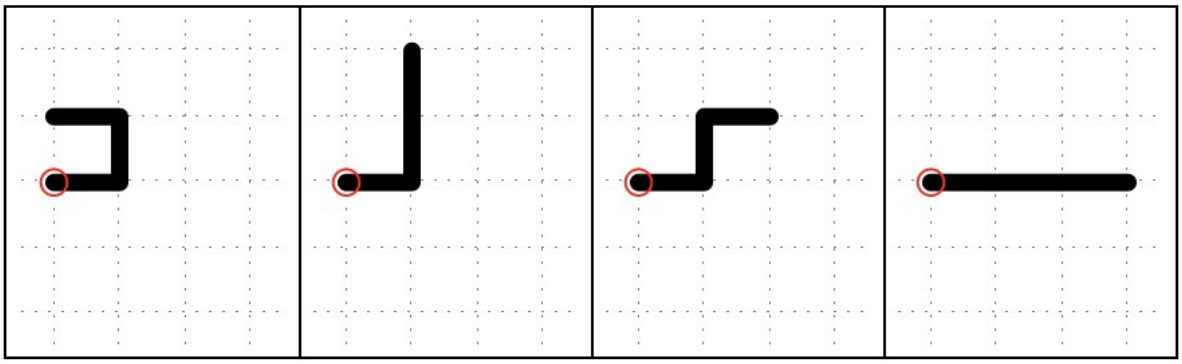
```
(* --- commence to outputtin' --- *)
order: 3
build time: 0.000509 sec
plot time: 0.000063 sec
valid time: 0.000044 sec
unique time: 0.000743 sec
total time: 0.001359 sec
raw: 2
valid: 2
unique: 2
```



```
draw time: 0.001939 sec
```

The same output for N=4 looks like this. I'm expecting N(4) = 96. I got...97. Here's a fun game, find the two grids among the 97 in that last pair of images that are congruent under transformation. [screams in math] But this is CLOSE, which is GREAT.

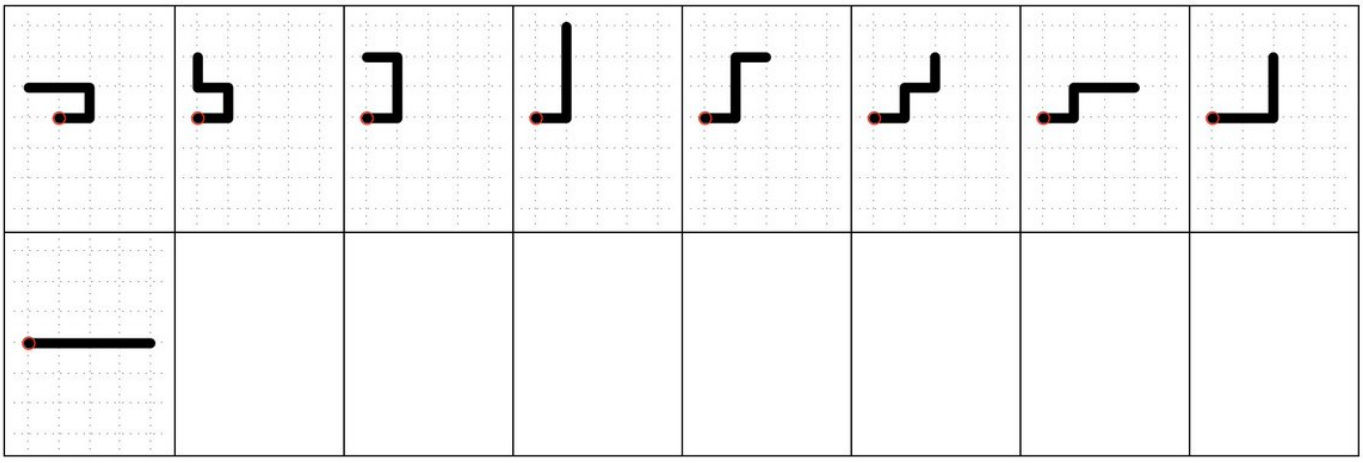
```
raw: 5
valid: 5
unique: 4
```



```
draw time: 0.004201 sec
```

N=5, targeting N(5) = 856. 1087 is...a great deal more than 856.

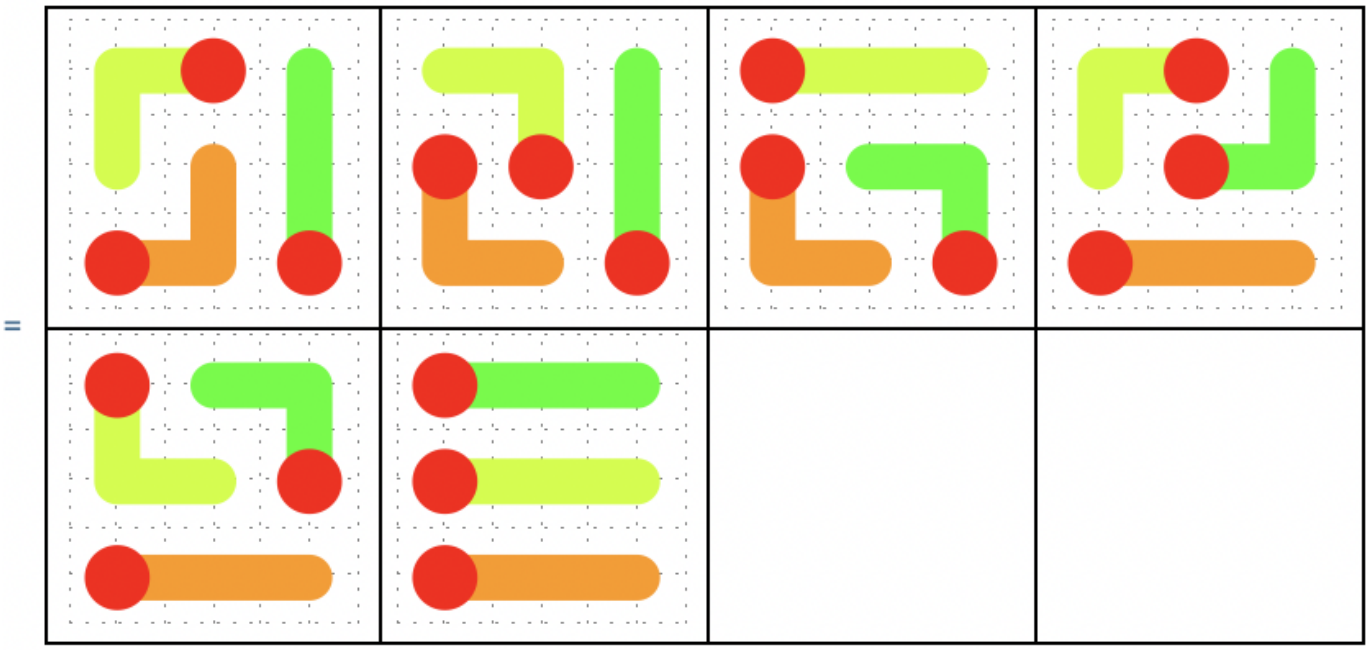
```
order: 5
build time: 0.002443 sec
plot time: 0.000538 sec
valid time: 0.000263 sec
unique time: 0.006835 sec
total time: 0.010079 sec
raw: 13
valid: 13
unique: 9
```



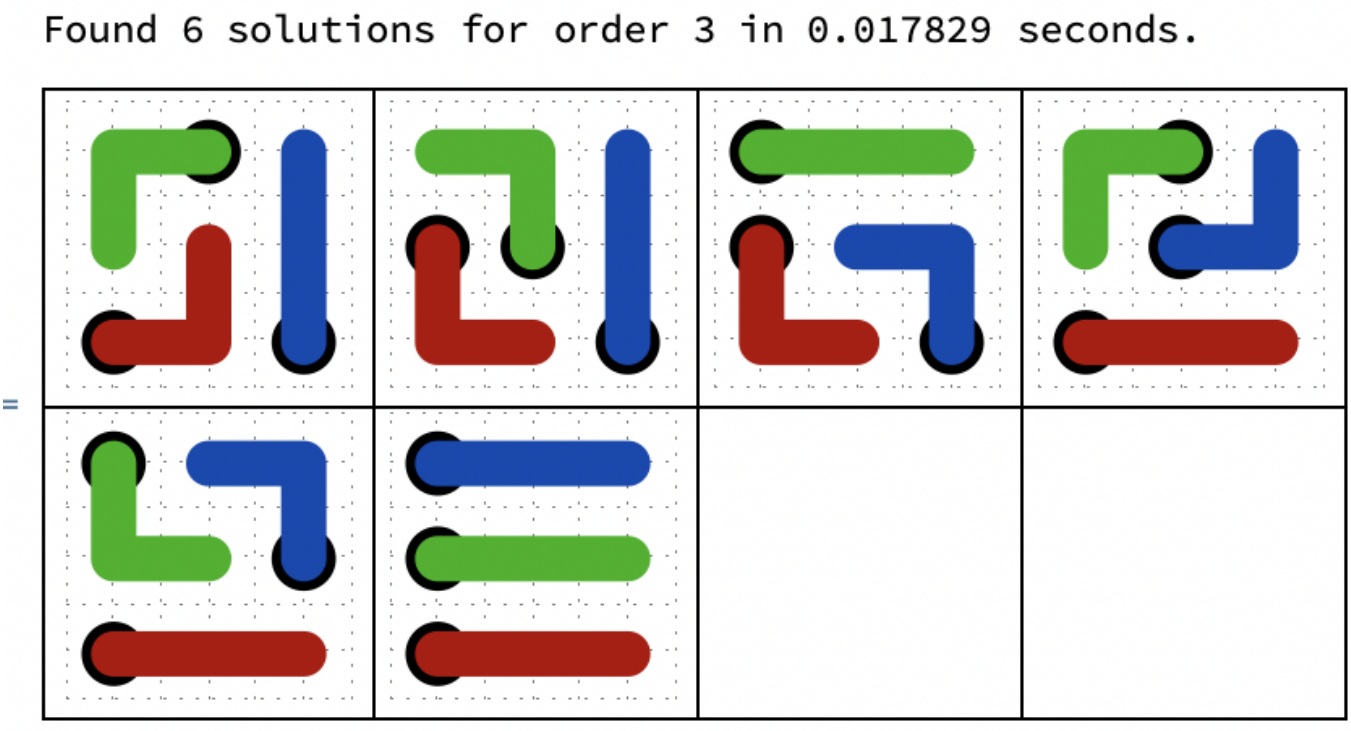
```
draw time: 0.007444 sec
```

I don't dare run N=6 right now, but if I'm lucky and the proportion of search time is going up by "only" two orders of magnitude for each N, it'd probably get done overnight if I start it tonight in this current form.

gonna get colorful, testing out graphics stuff

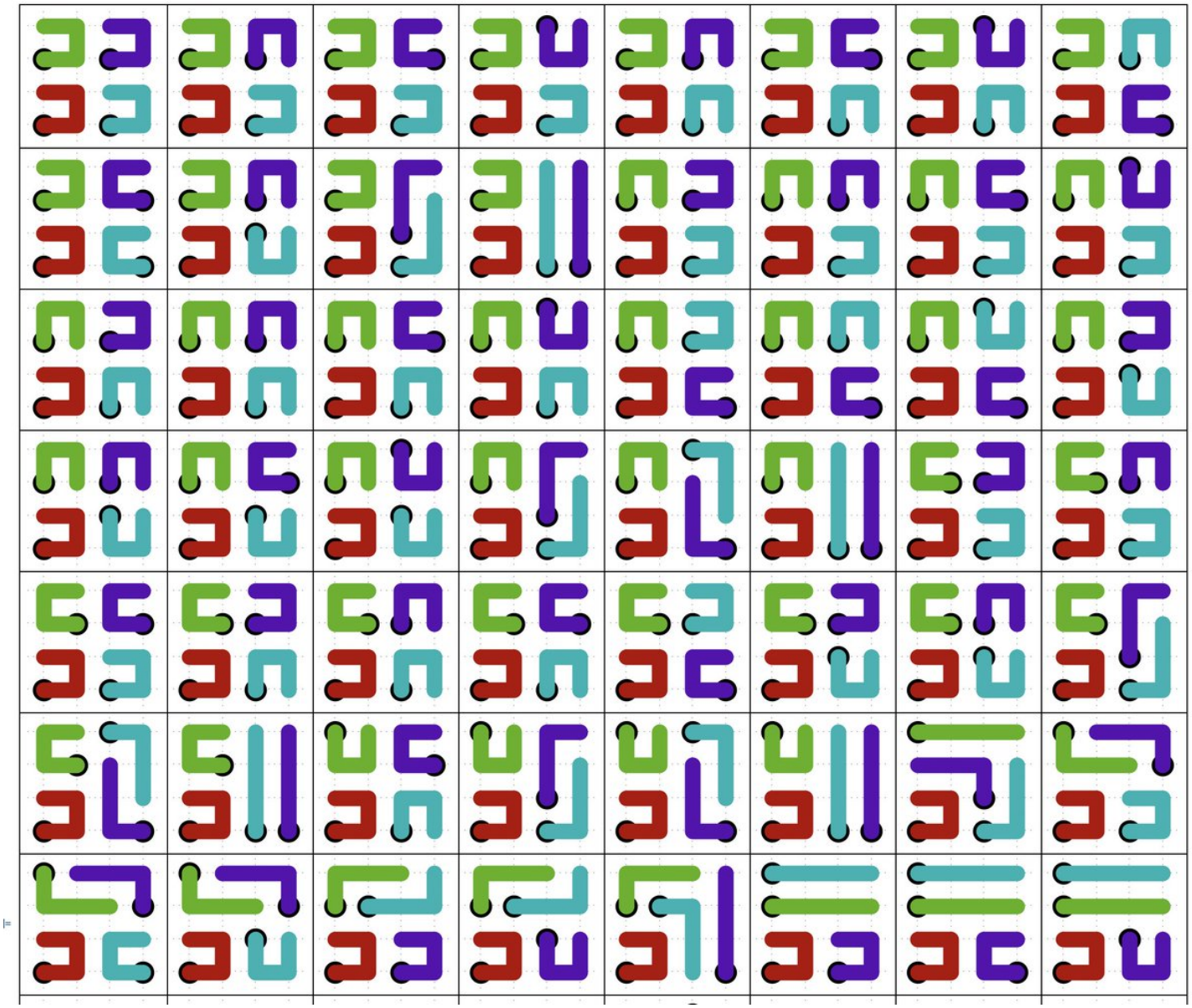


Maybe a little less eye-searing? The thought here is (a) color is nice, and (b) if I really am gonna try and scan over 97 solutions looking for a duplicate, explicitly ordering the worms with color may help look for oddities etc.

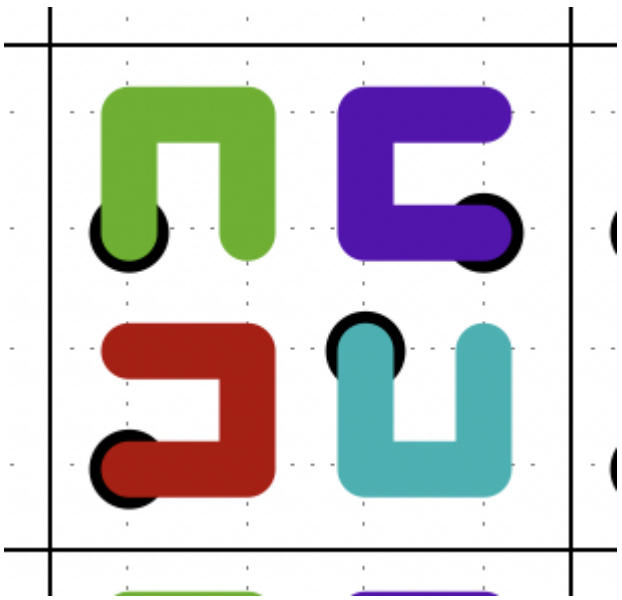


So here's a chunk of the N=4 output with colors assigned in order of placement 1st-4th: red, green, turquoise, purple. (Temporary palette just to get going, it's not great.) Makes it a little easier to parse for me visually, plus now we can see explicit order.

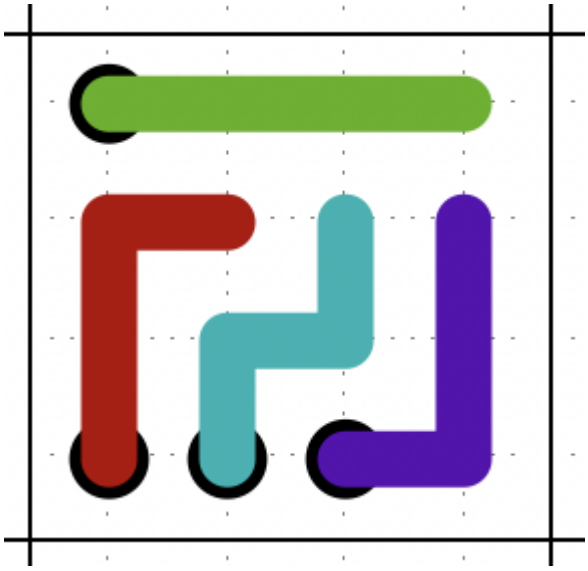
filtered solutions: 97



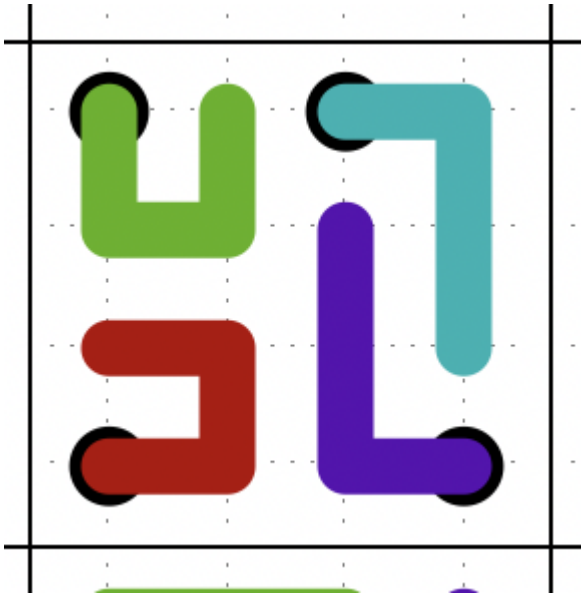
One assumption I have about my code is that in these, the canonical solutions I'm producing, the head of each worm is at the bottom-most row it can be, and then in the left most column. This solution demonstrates that nicely for all four rotations of this 2x2 yoga worm.



And scanning over the output solutions, this seems to hold true for most shapes in most configurations, as in every example below. But! But.



But I see certain configurations of Elle that violate that expectation! Like the turquoise one in these first two examples, or the red Elle in the latter two. So that's weird! Is it meaningfully messing with my code? Not sure!

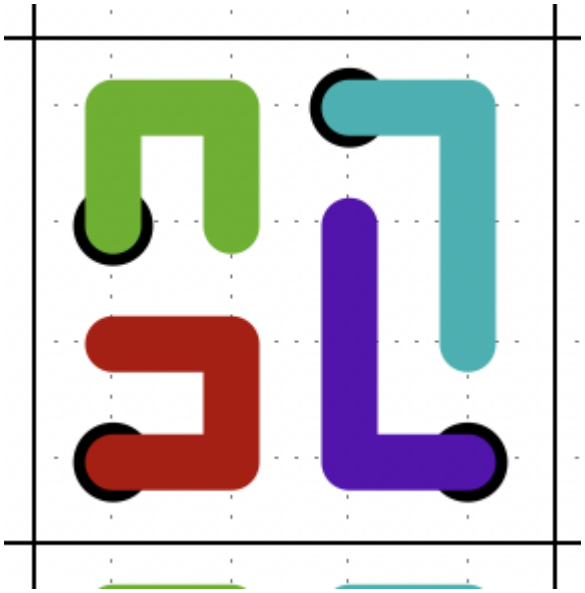


The headedness of the worms doesn't matter for the the underlying problem; I'm just tracking them because my worms are stored as ordered lists of coordinates and comparing configurations of worms means controlling for that. My output should always be head-normalized, and yet!

Here's my canonizeWorm[] code. The logic is, take a worm as a list of coords (e.g. {{2,1},{2,2},{1,2},{1,1}} for a left-facing Yoga worm), and compare the first and last element. If the first one is "smaller", return original worm. If not, then return the list reversed.

```
(* given a worm w, check to see if the head is smaller than the tail in some coordinate space rubric; if so,
return w; if not, reverse the sequence of coordinate pairs and return that *)
canonizeWorm[w_] := Module[{tempw, first, last}, (
    tempw = w;
    first = First[w];
    last = Last[w];
    If[(first[[1]] < last[[1]]) || (first[[2]] < last[[2]]),
     Return[w],
     Return[Reverse[w]]
    ]
   )];
```

And it seems generally to be working, but why not for this blue Elle or this red Elle, respectively? For the red one, that worm as shown is {{1,2},{1,1},{2,1},{3,1}}, which...hmm, is what my canonize function would call smaller than the reverse if it prioritizes the first coord.



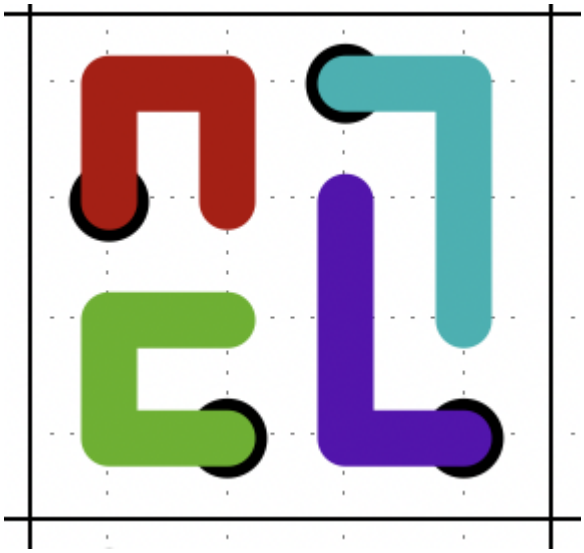
But if I prioritize the second coord, then for that same red Elle = {{1,2},...,{3,1}} the logic is
if 2 < 1 (it's not, obviously)
OR if 1 < 3 (it is)
THEN reverse the worm"

Which should produce {{3,1},...,{1,2}} with the head at bottom right. And yet, no change.

```
(* given a worm w, check to see if the head is smaller than the tail in some coordinate space rubric; if so,
return w; if not, reverse the sequence of coordinate pairs and return that *)
canonizeWorm[w_] := Module[{tempw, first, last}, (
    tempw = w;
    first = First[w];
    last = Last[w];
    If[(first[[2]] < last[[2]]) || (first[[1]] < last[[1]]),
     Return[w],
     Return[Reverse[w]]
    ]
   )];
```

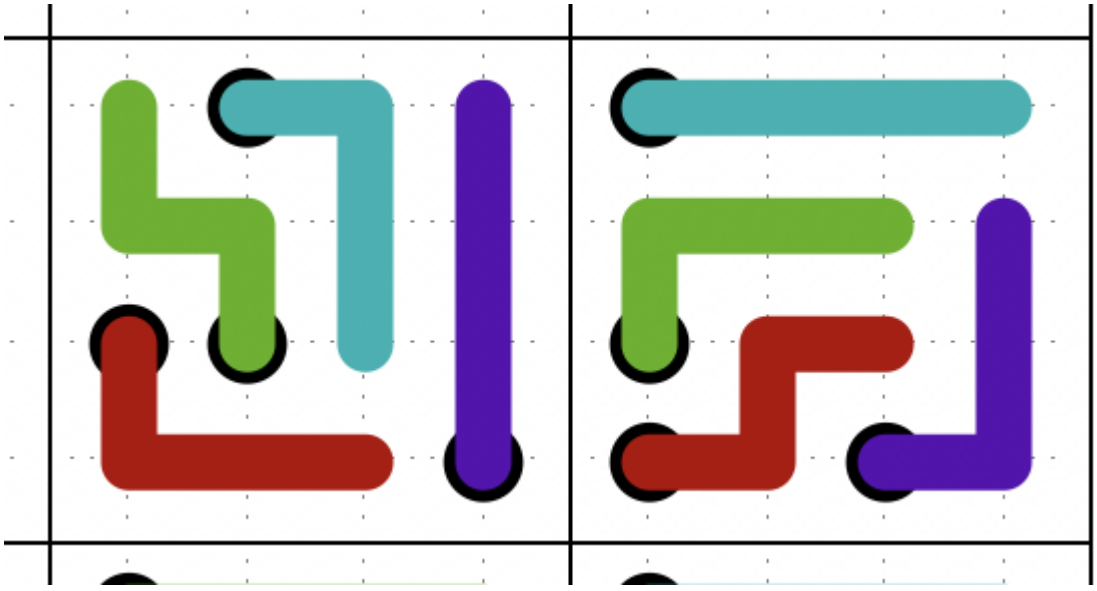
So is there a problem with my logic? Or a problem with my implementation of it? Or a bracket it the wrong place? Or am I being punished for my sins? Or

And why these two specific forms, turqoise and red here respectively? Aside from both being Elles, they also share the property of having a negative slope from tail-to-head or vice versa. But so does the purple Elle in the first example, and yet its head is oriented as expected!



And another mystery in all this: I seem to be producing one extra dupe in here, expecting 96 sets but getting 97. But far more than one set here has this non-canonical Elle problem, so it can't just be "those boards are bad".

Ah ha! Some grimacing and peering turned up a duplicated in my 97 configurations: these two boards are identical under before and after a 90 degree rotation left (ignoring heads, which in this context we should). And the first one has a Weird Elle, head-wise. Hrm.



The odd news: I found this by comparing row-by-row the output of my original N(4)=97 set we've been picking at to a different N(4)=98 set the code produces if I change one constant that shouldn't, in theory, cause a change. Probably related to my bug? Hopefully? Diving back in.