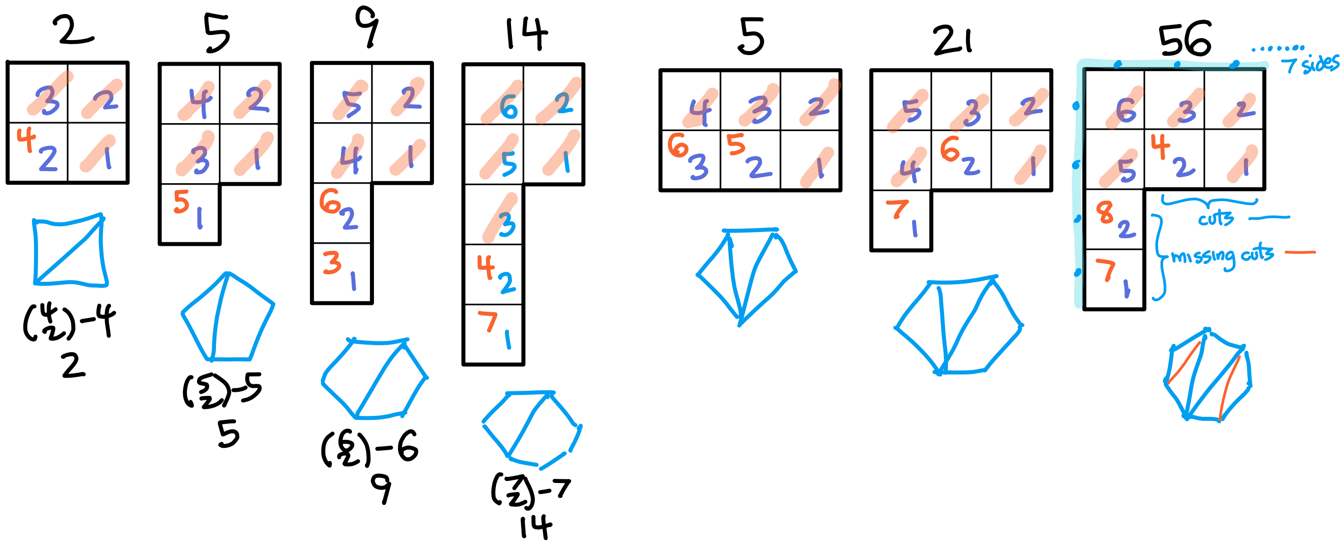
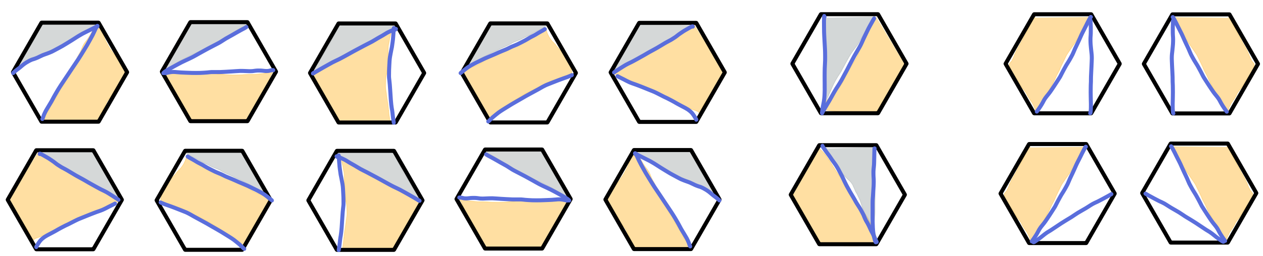
March 3, 2021

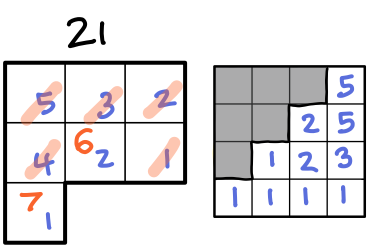
These are my notes as I look for an alternative construction.

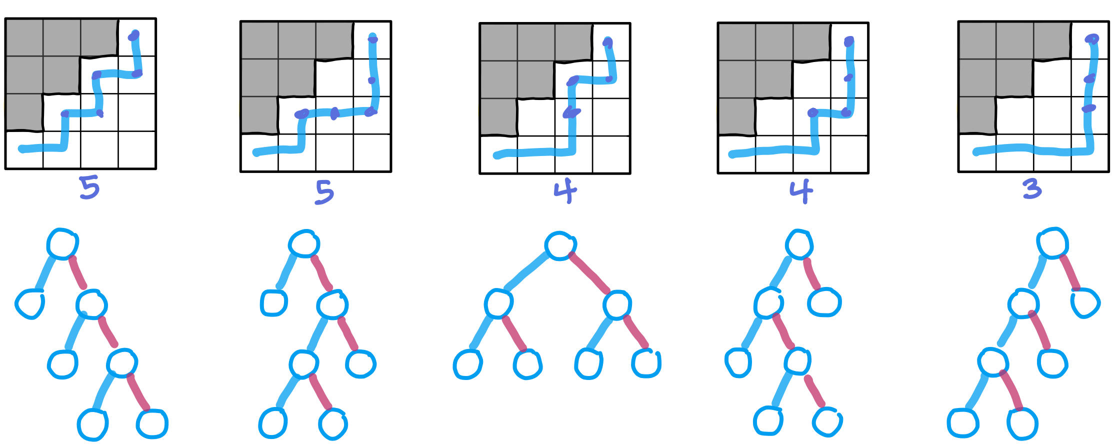
Thinking of Stanley correspondence as lattice walks with pauses

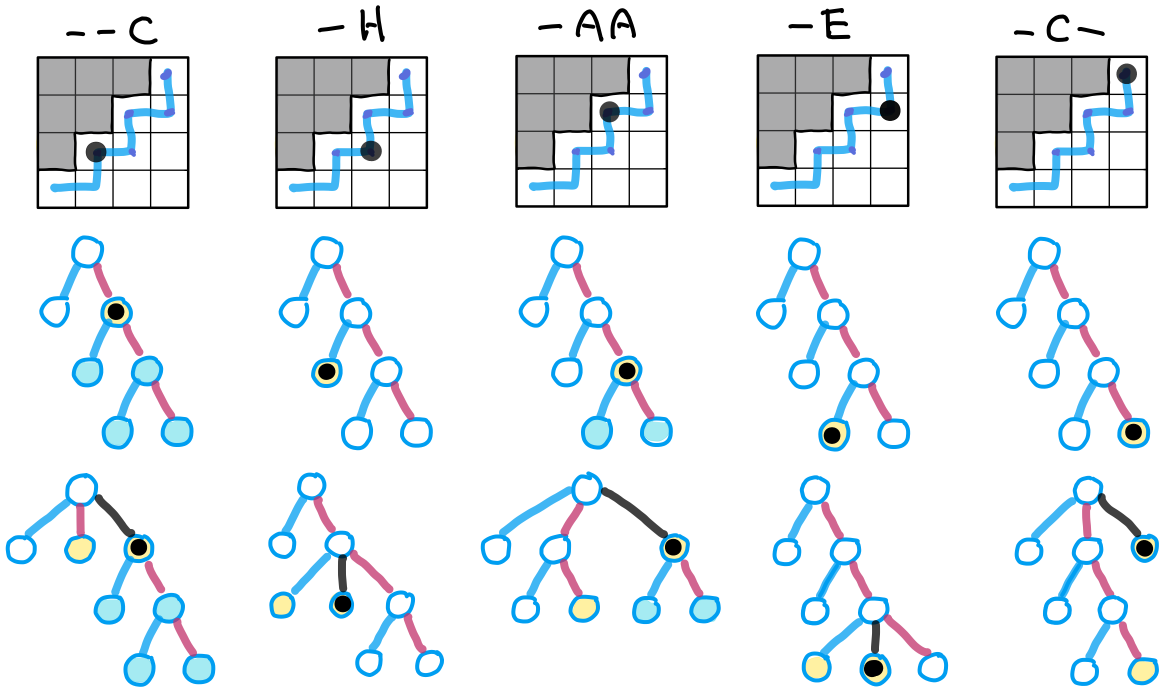


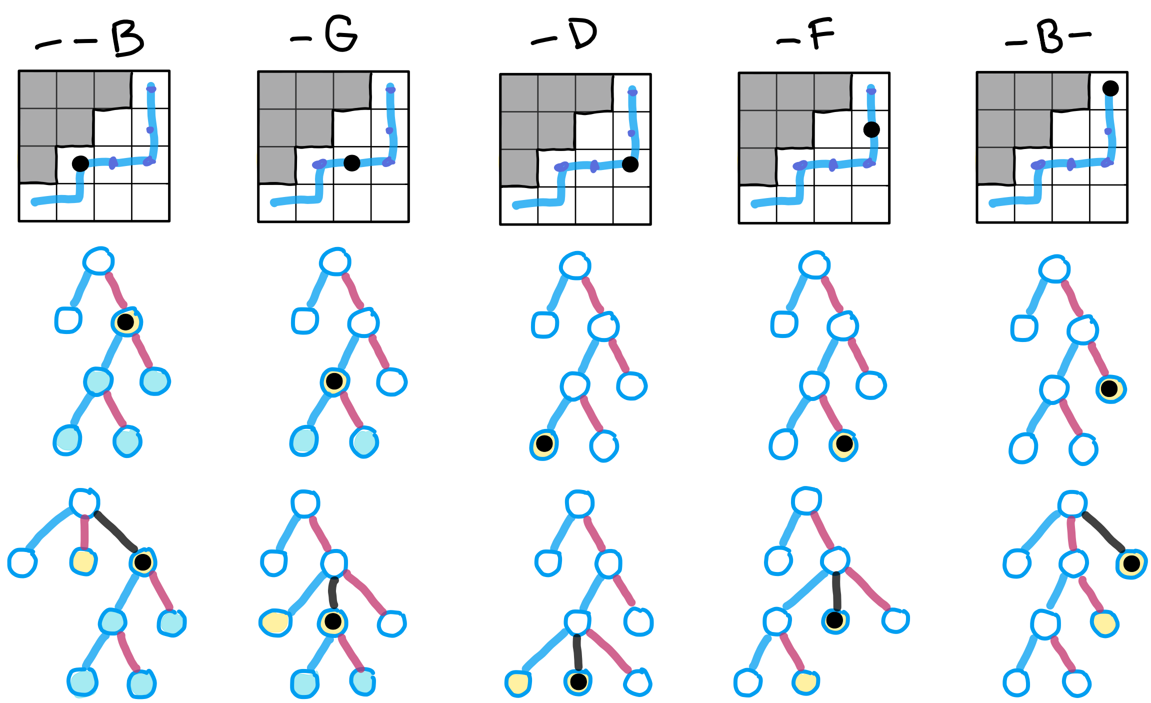


March 5, Friday

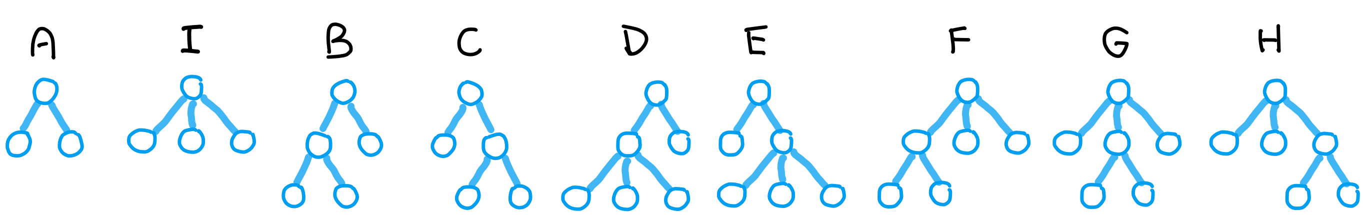




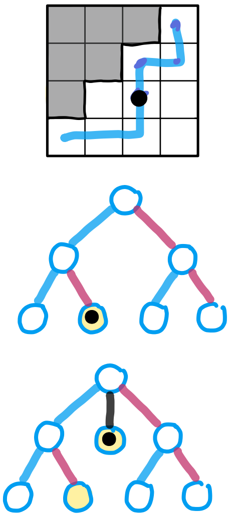




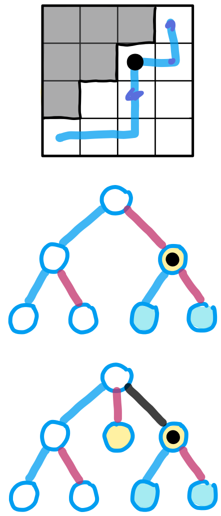
(First try is wrong)



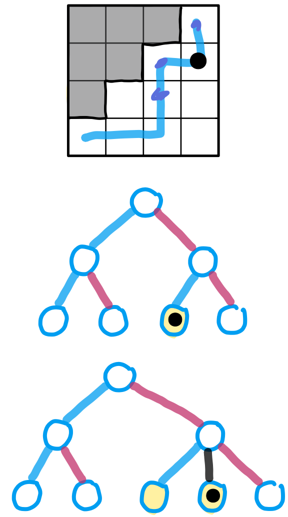
A-A



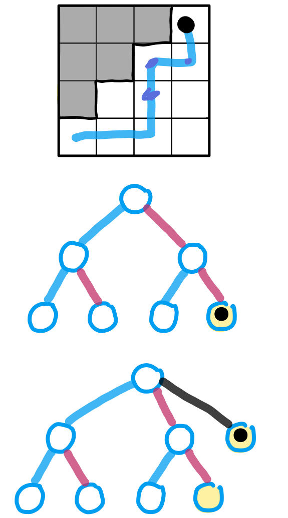
A-A



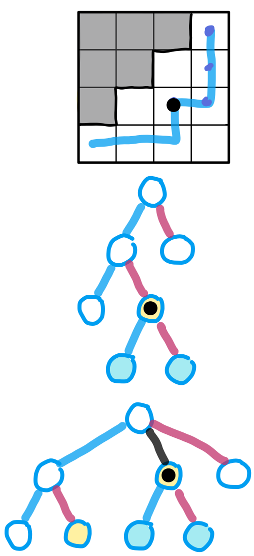
AI



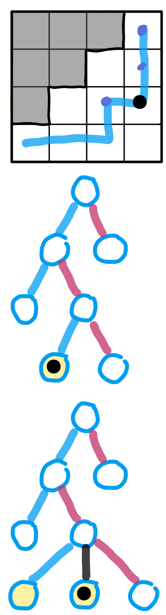
AA-



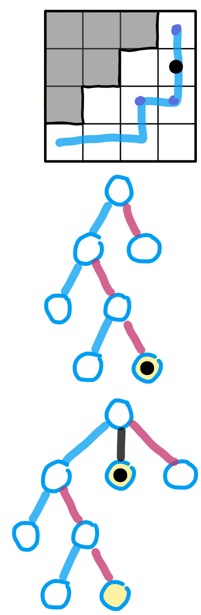
AA-



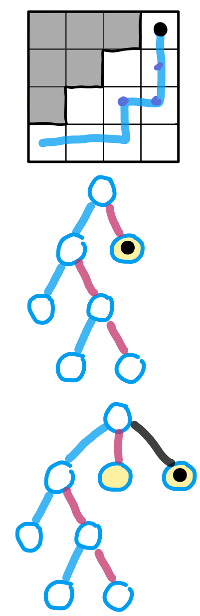
E-



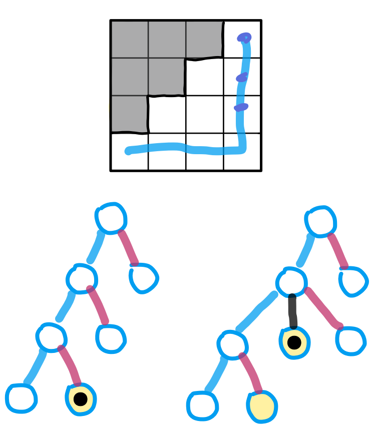
C--



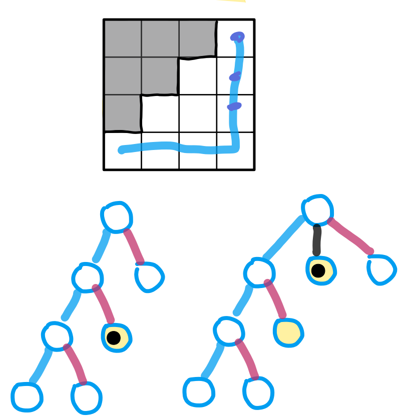
C--



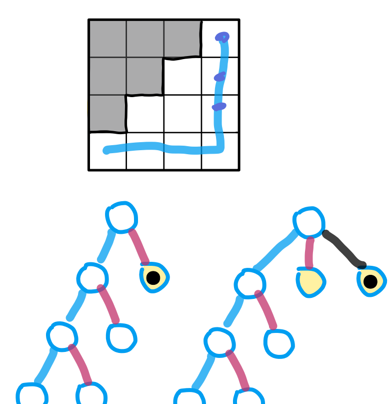
F-

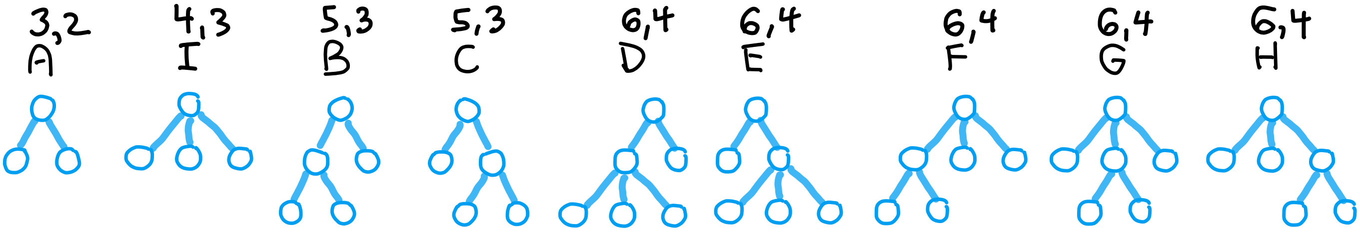


B--



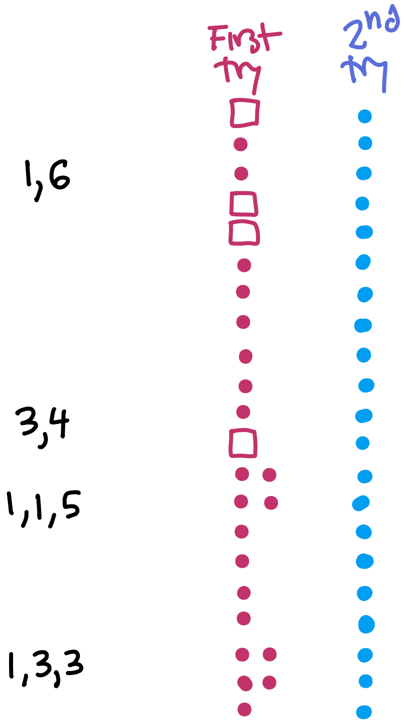
B--





8-root = 7 nodes, 5 leaves

10+2+6+3 = 21 ✓



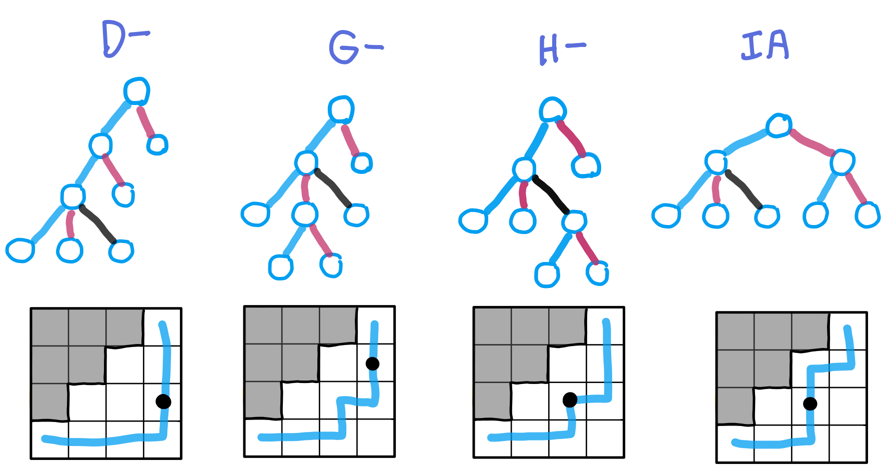
First try, wrong

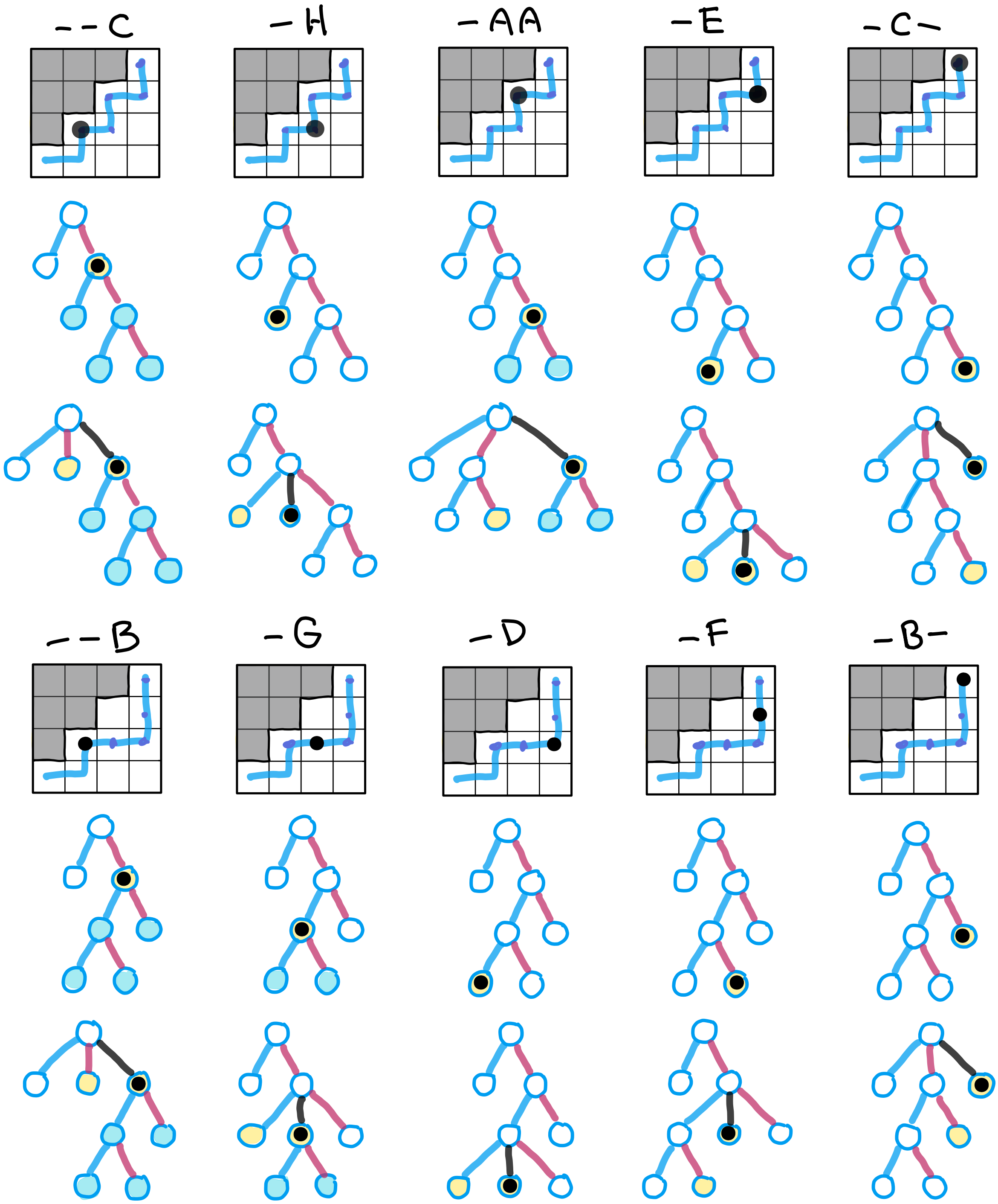
--C	-H	-AA	-E	-C-
--B	-G	-D	-F	-B-

A-A	A-A	AI	AA-
AA-	E-	C--	C--
F-	B--	B--	

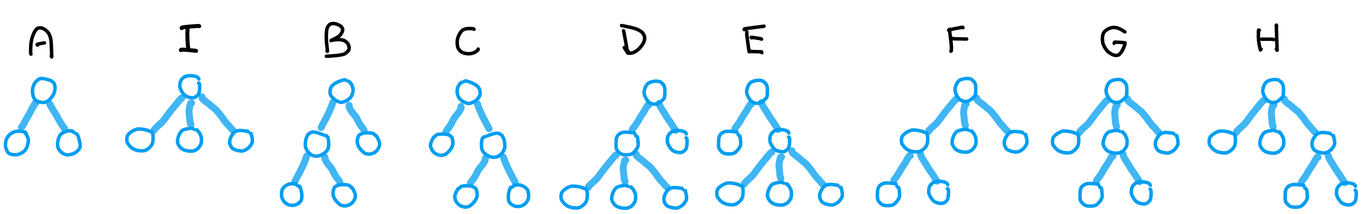
So black floats up red, but not blue

We need a harder example to see black on black rules

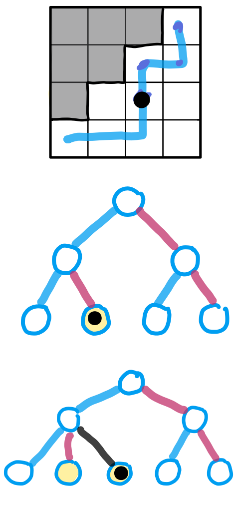




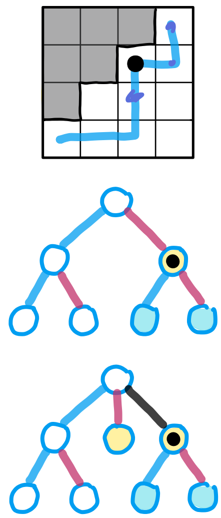
(second try, correct)



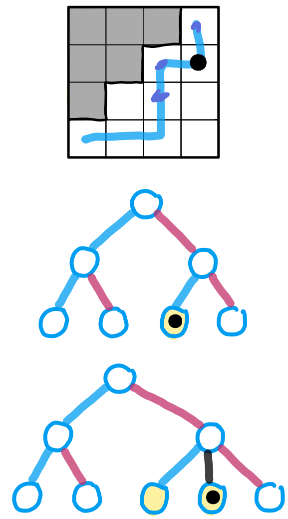
IA



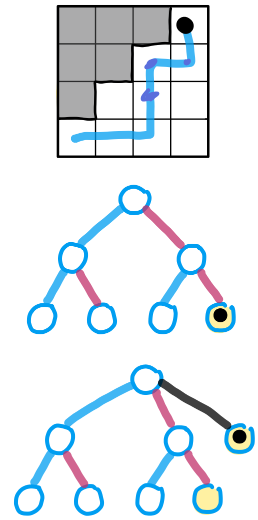
A-A



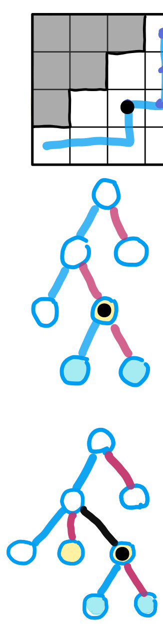
AI



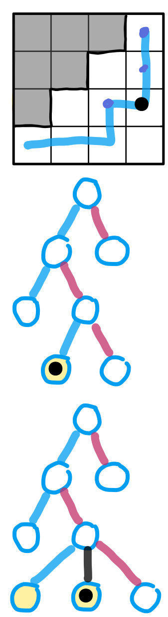
AA-



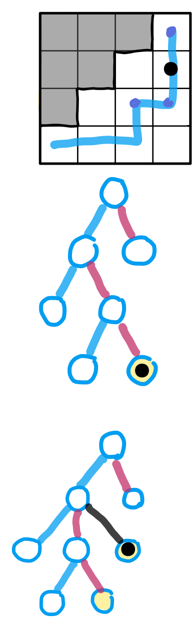
H-



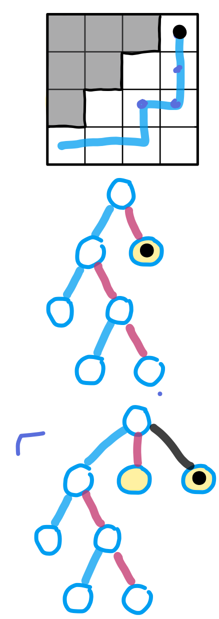
E-



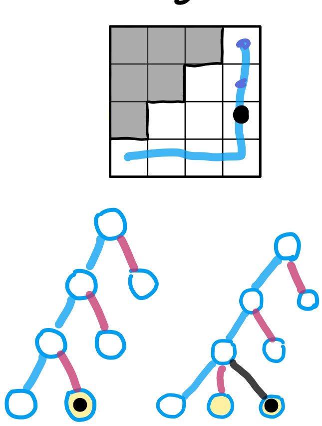
G-



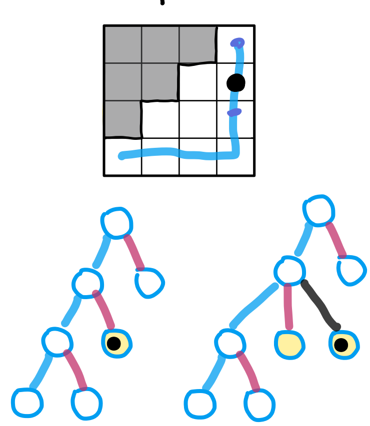
C--



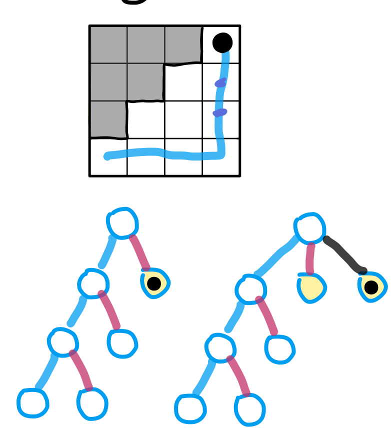
D-

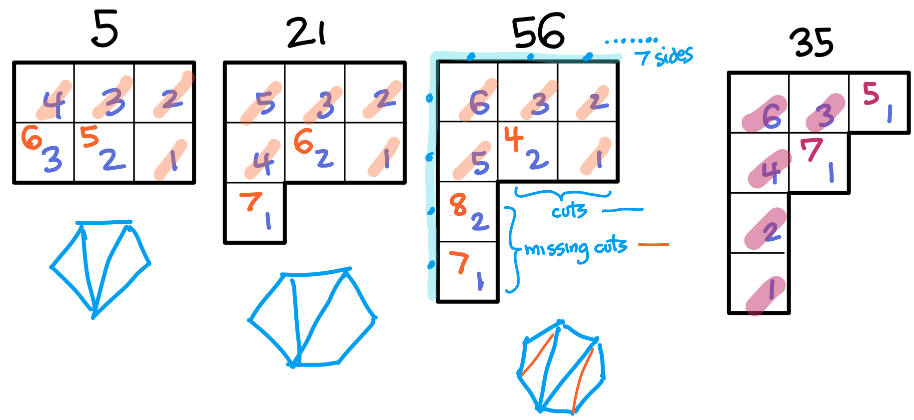


F-

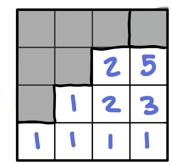


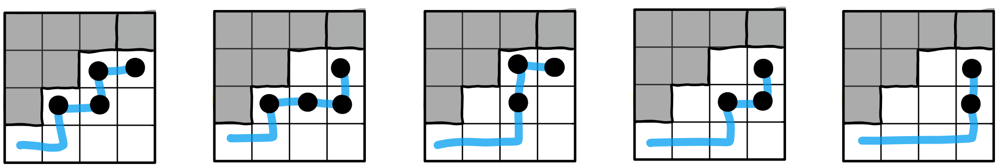
B--





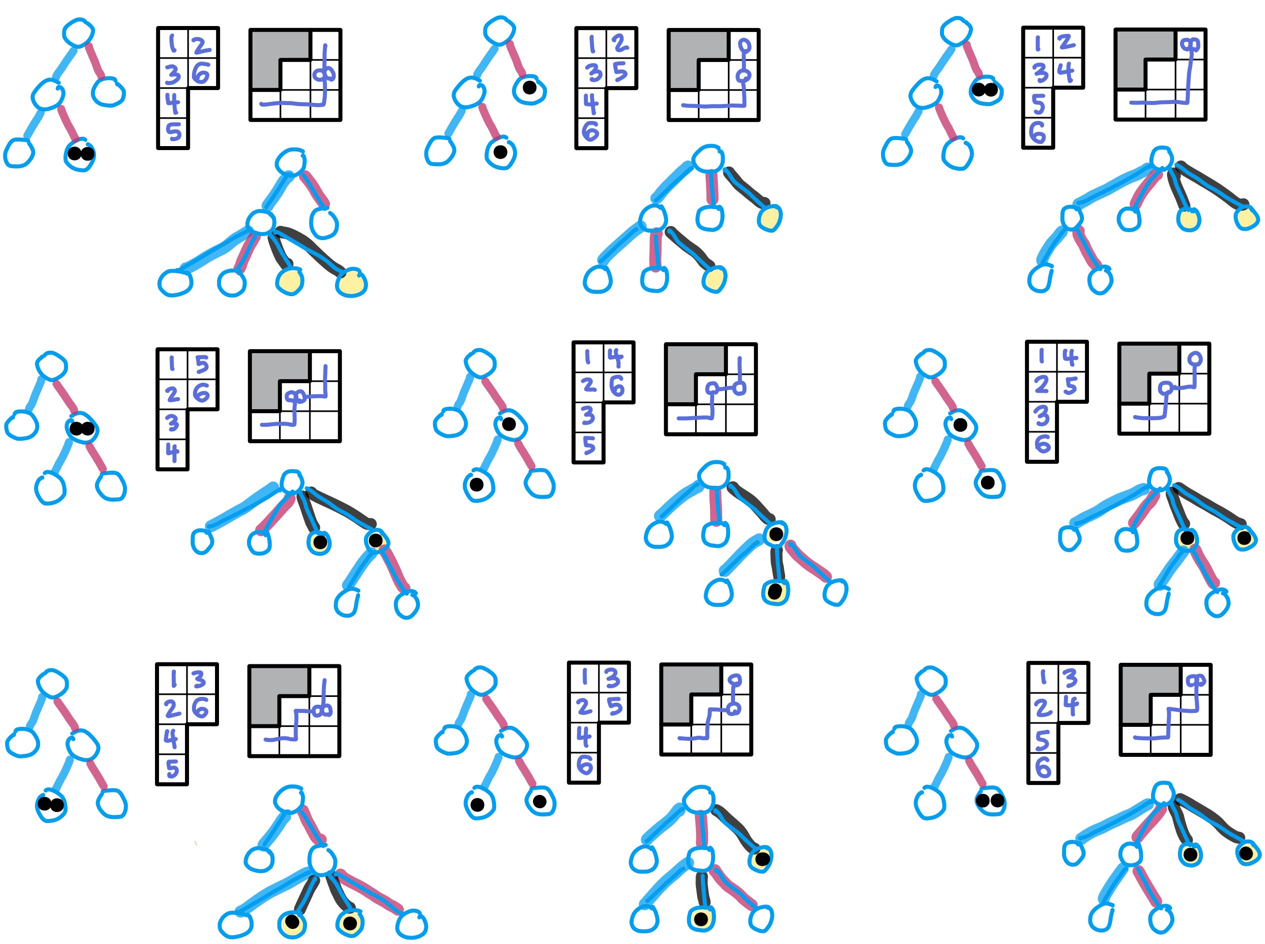
We now understand broken tableaus as incomplete trees.

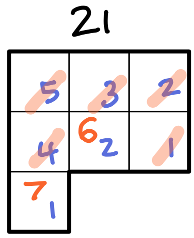




$4+\binom{4}{2}=10$ $4+\binom{4}{2}=10$ $3+\binom{3}{2}=6$ $3+\binom{3}{2}=6$ $2+\binom{2}{2}=3$

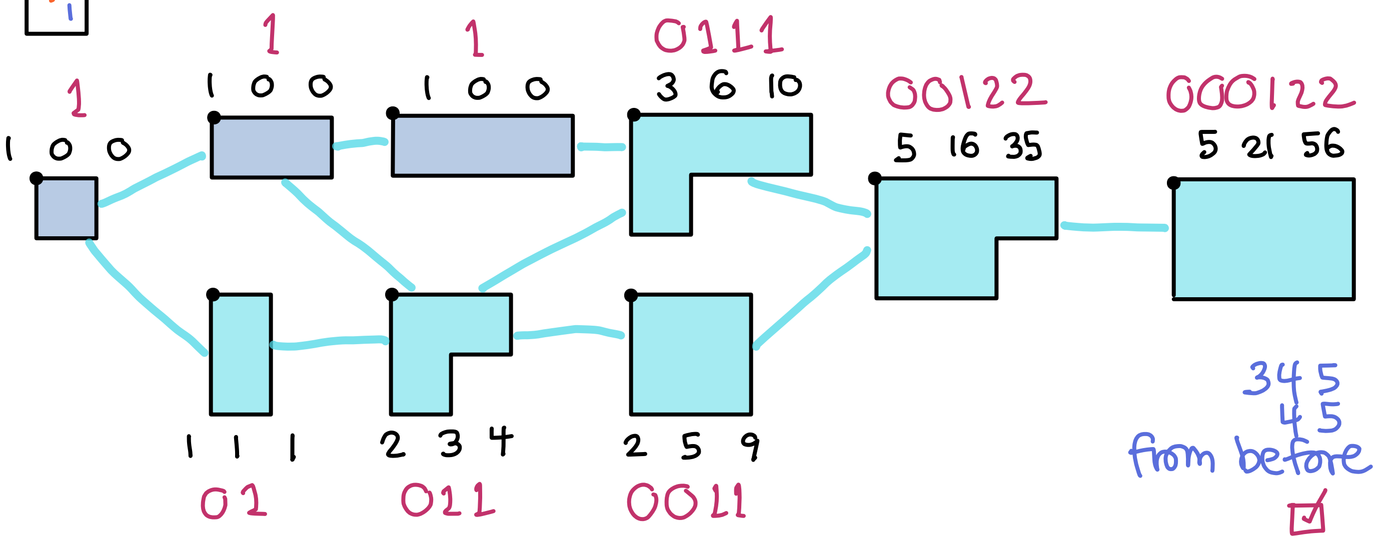
Process multiple dots in sequence as encountered, depth first search





General principle: Allow pauses in all walks.

Here, only once two rows



$g(t)=\sum a_n t^n$, n pauses

$$g(t) = a(t)+b(t)+tg(t)$$
$$(1-t)g(t) = a(t)+b(t)$$

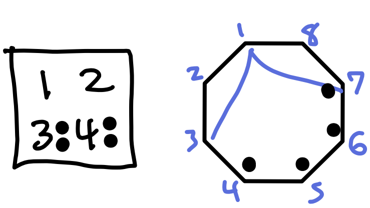
write $a + \frac{b}{1-t} + \frac{c}{(1-t)^2} + \ldots$ as a b c

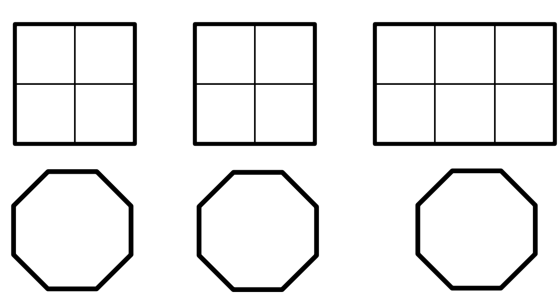
There should be a generalized hook length formula here.

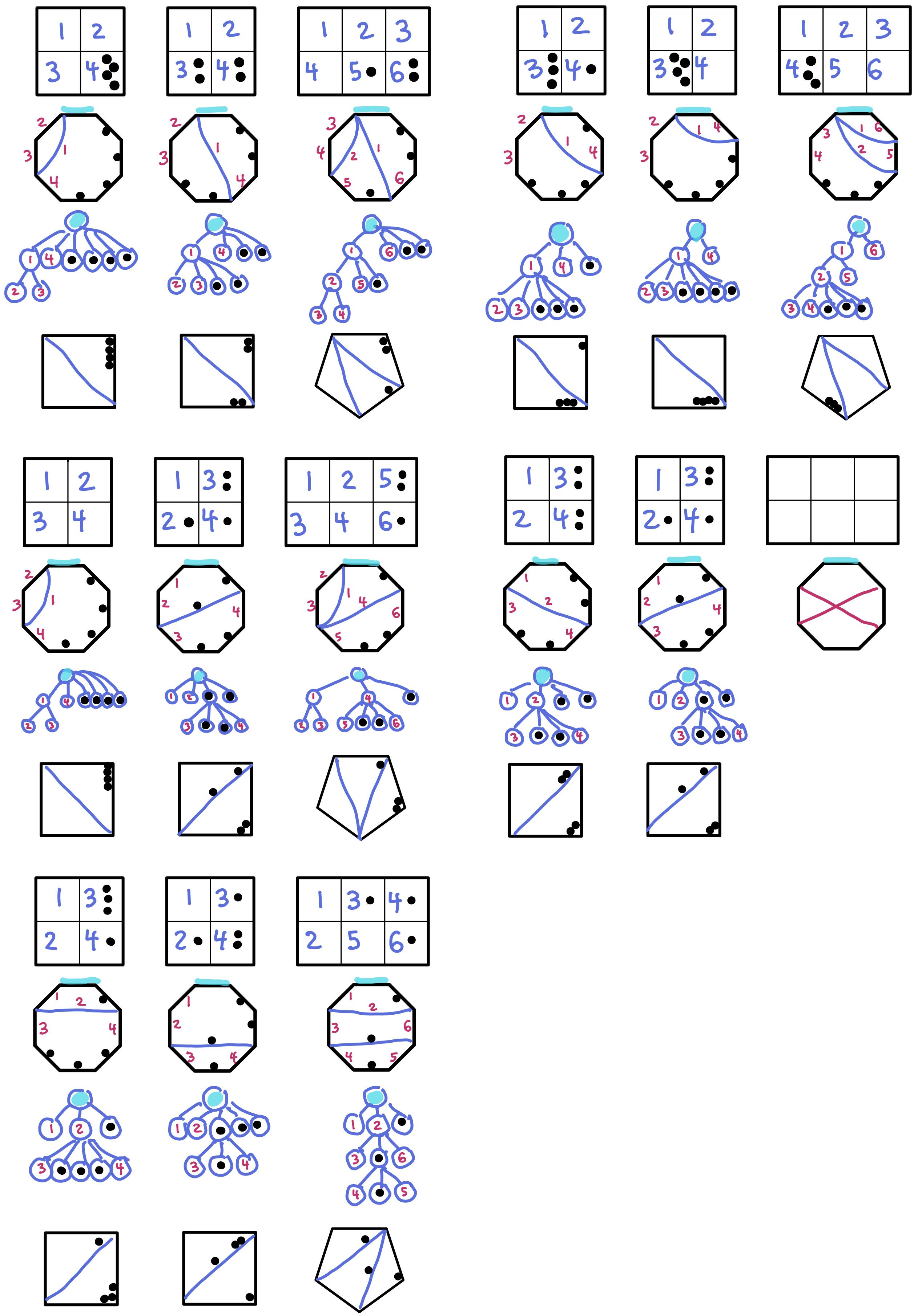
Sunday March 7

Want entire associahedron polytope to map through correspondence

When are two arcs noncrossing?





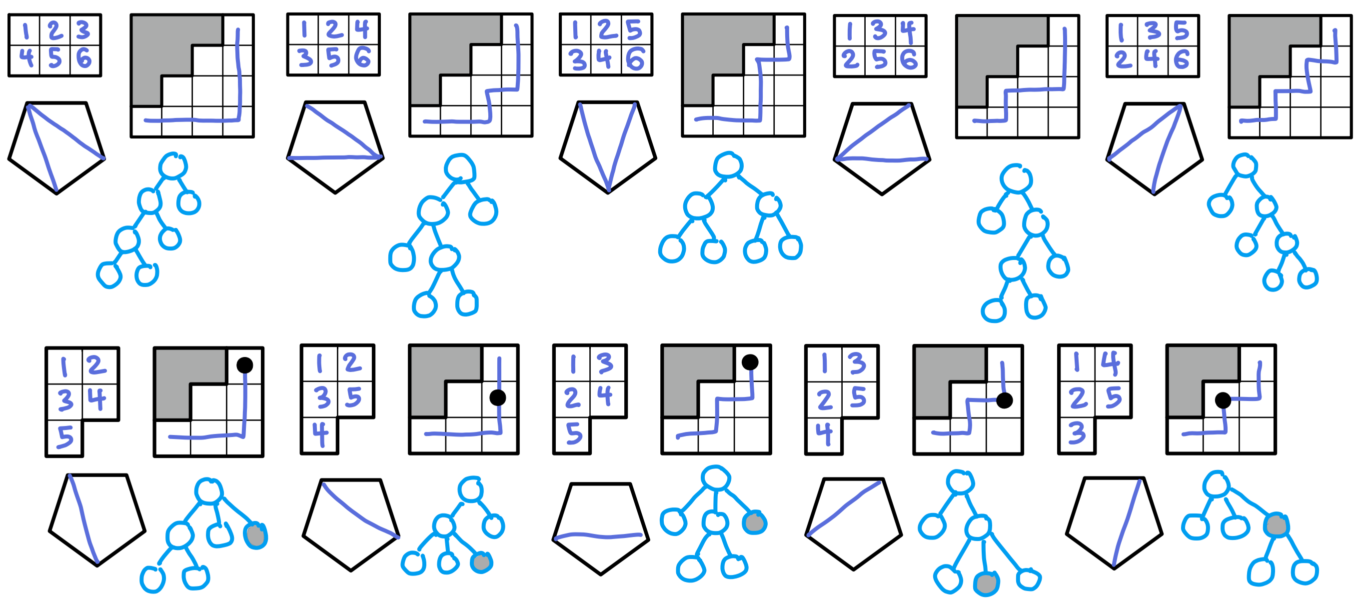


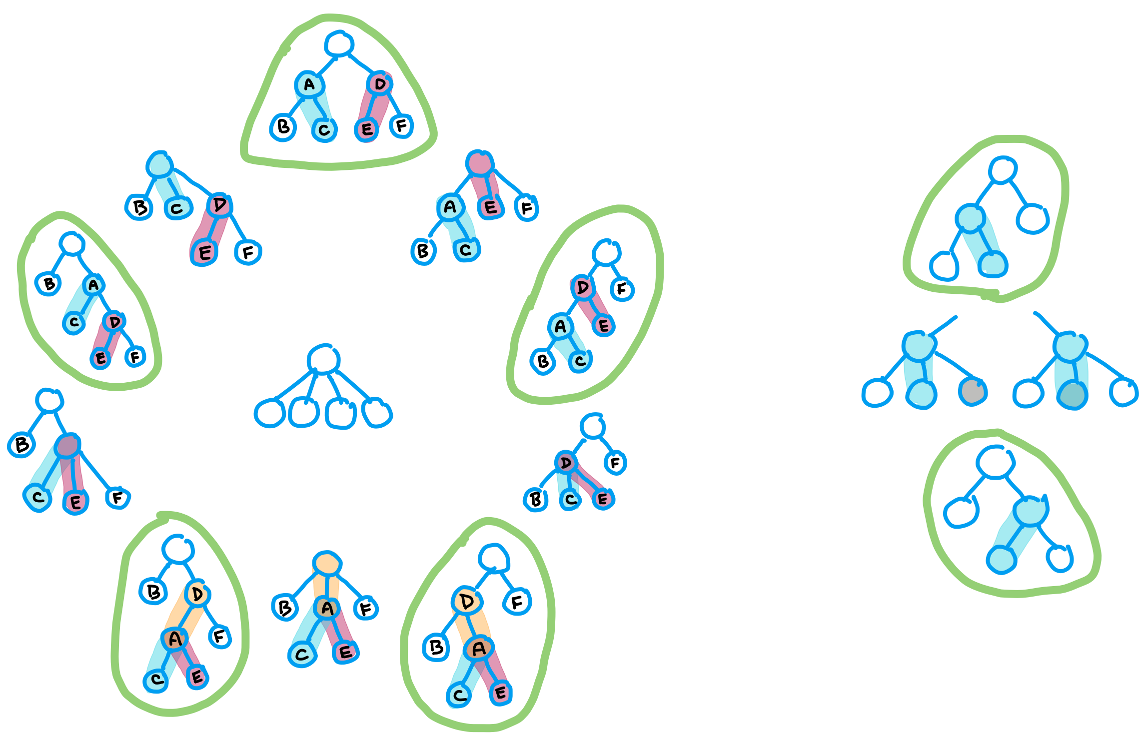
Monday, March 8

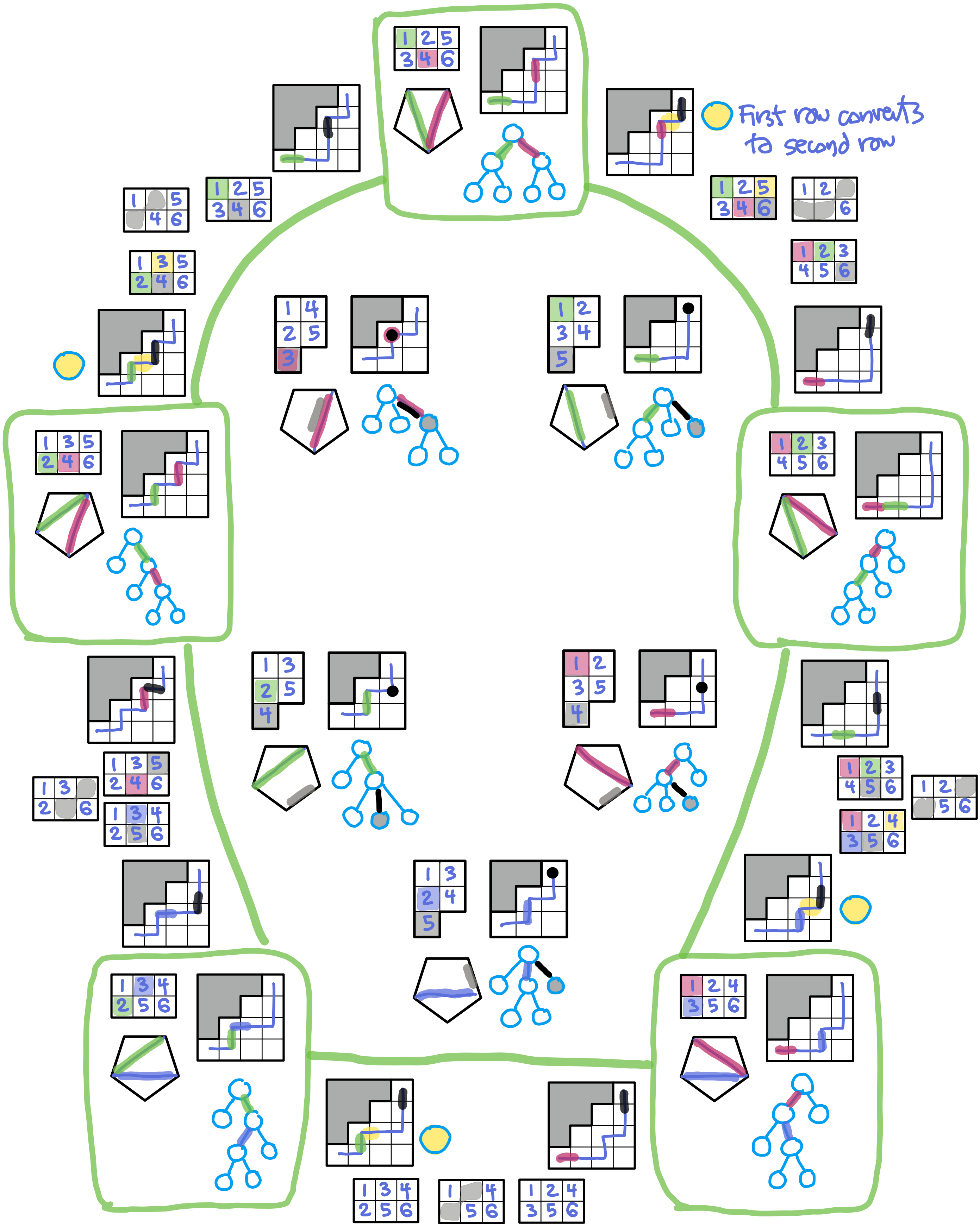
Need to study squarefree ideal on variables = interior edges,
monomials = crossing pairs.

This should have the dual associahedron as minimal resolution.

Meanwhile, start instead trying to understand edges on polytope.

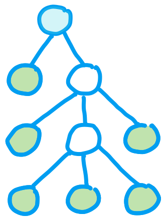


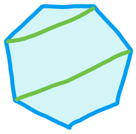



First row converts
to second row

Tuesday, March 9, 2021

Want to confirm rule in general, for nested extra edges.





7-gon

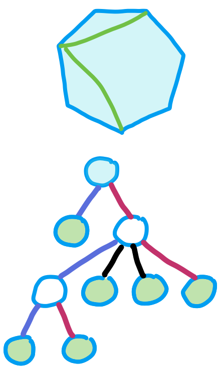
$\binom{7}{2}-7=21-7=14$ interior edges.

$\binom{14}{2}=\frac{14\cdot 13}{2\cdot 1}=7\cdot 13=91$ pairs

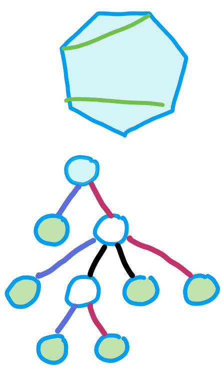
$\binom{7}{4}=\frac{7\cdot\cancel{6}\cdot 5\cdot\cancel{4}}{\cancel{4}\cdot\cancel{3}\cdot\cancel{2}\cdot\cancel{1}}=35$ crossing pairs

$91-35=56$ too many to write all out?

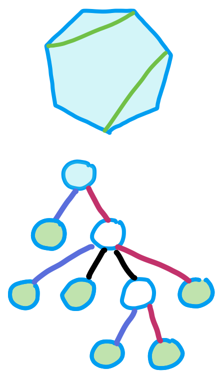
Compare with neighboring trees. $\binom{6}{2}-6=9$ cases with this start



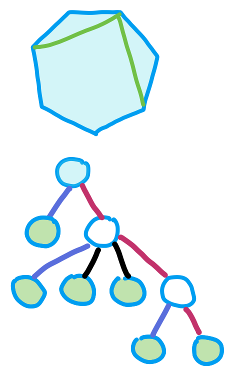
1	3	4
2	5	8
6		
7		



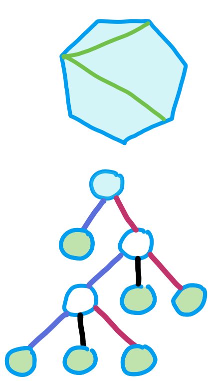
1	3	5
2	6	8
4		
7		



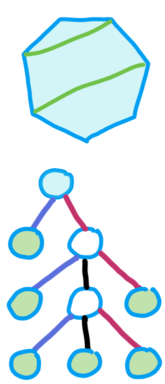
1	3	6
2	7	8
4		
5		



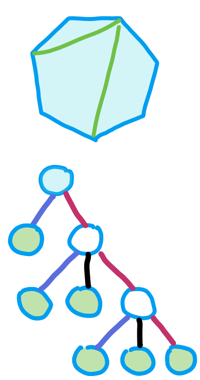
1	3	7
2	6	8
4		
5		



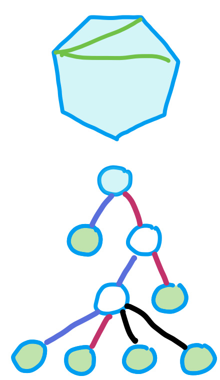
1	3	4
2	6	8
5		
7		



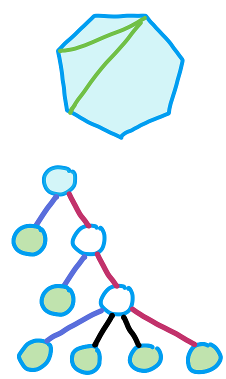
1	3	5
2	7	8
4		
6		



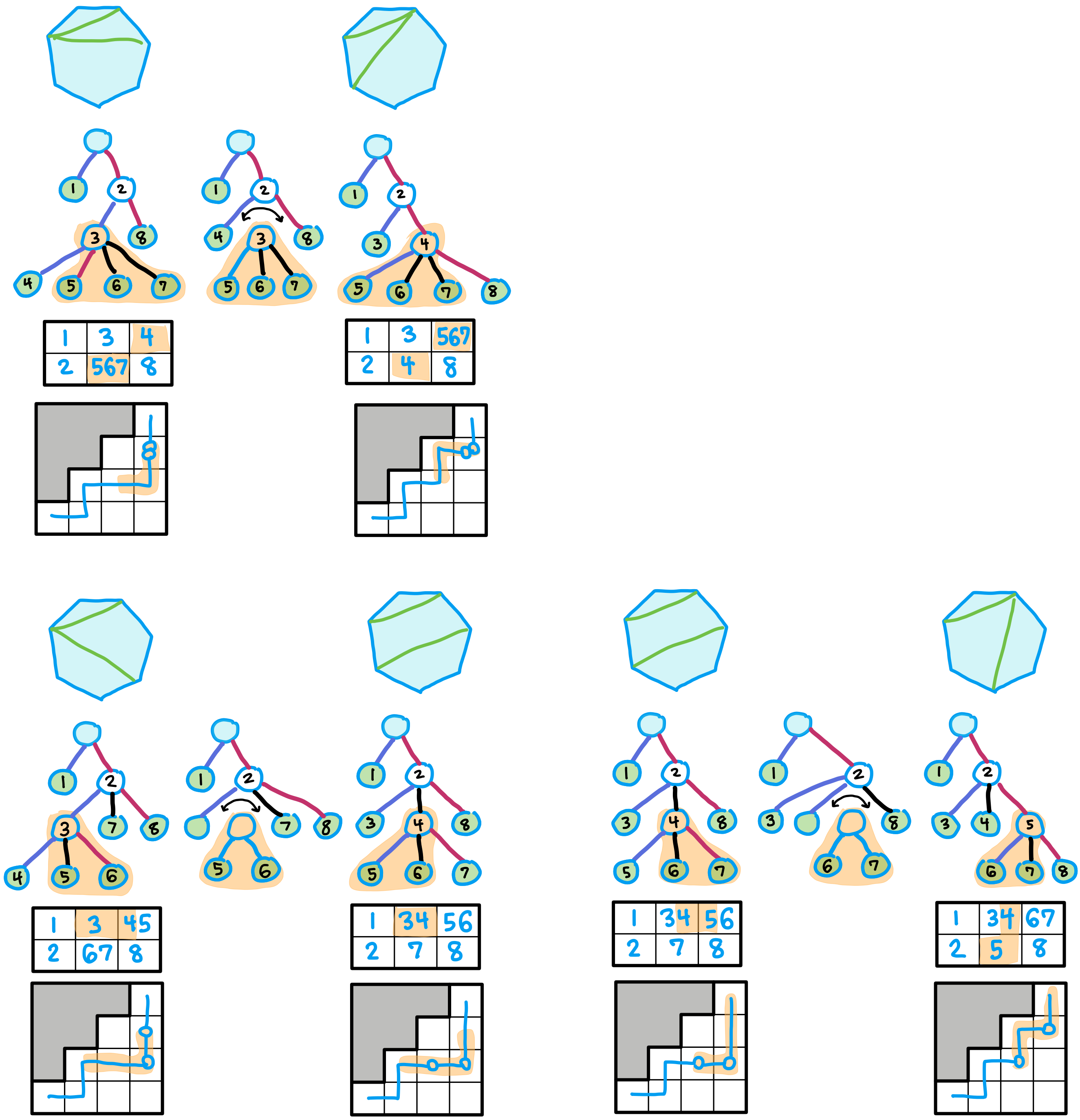
1	3	6
2	5	8
4		
7		



1	3	4
2	5	8
6		
7		

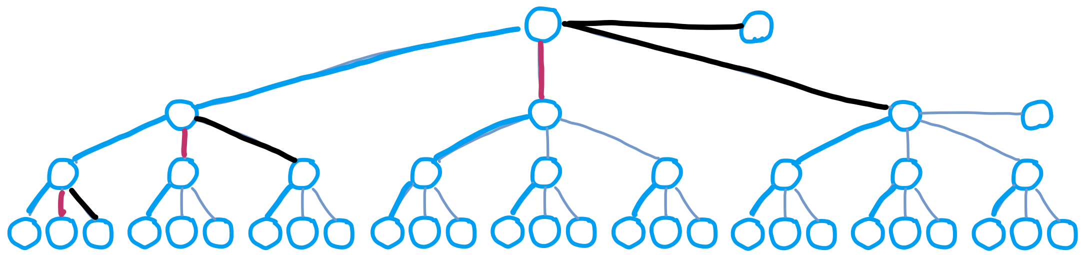


1	3	5
2	4	8
6		
7		



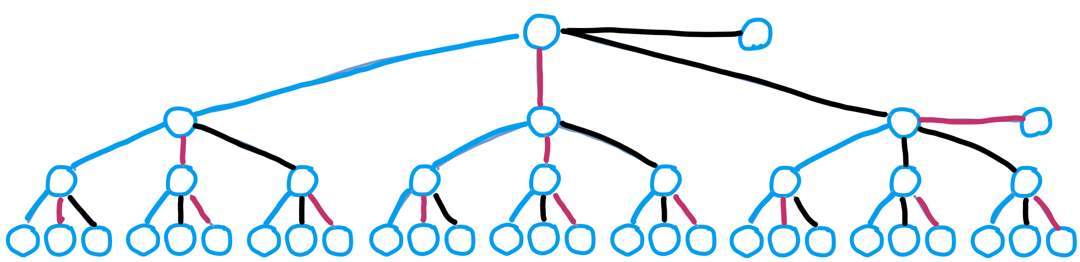
1 3 4
2 567 8
1 3 567
2 4 8
1 3 45
2 67 8
1 34 56
2 7 8
1 34 56
2 7 8
1 34 67
2 5 8

Thursday March 11 What are rules in general?

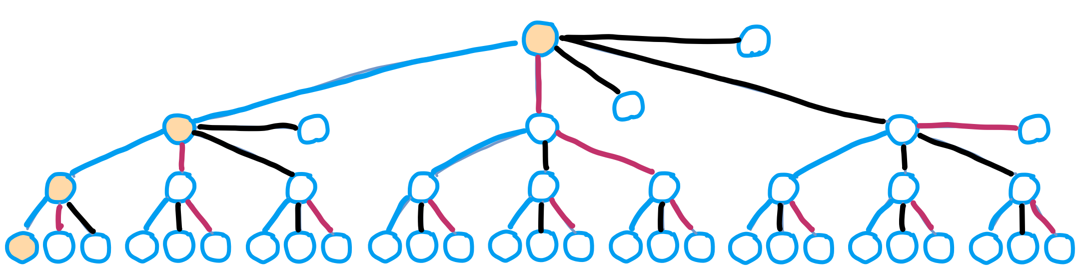


Left (blue) always descends to create new node
Use is forced, no question when to use

Right (red), Extra (black) have variant rules based on context
Right always rises to attach to nearest available host
Rules for black need to be pinned down.
For standard Young tableau correspondence, no black till first red
Given this, need red to rise, so goes after blue again
This sets rules for leftmost blue spine

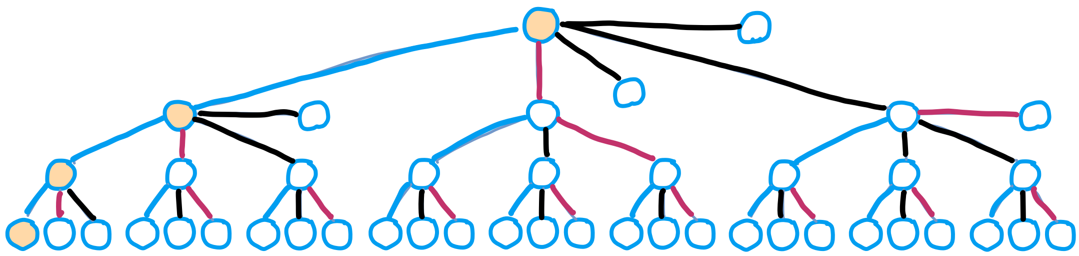


Perhaps rules are local, in which case locally leftmost is same
Starting in middle, always ambiguity whether black rises
Need red as divider between cases
Same issues in middle, so leftmost spine can't be local?



Off the leftmost spine, red closes the level.
Black climbs to find an open level.

To find a potential flaw, what changes if leftmost spine isn't special?

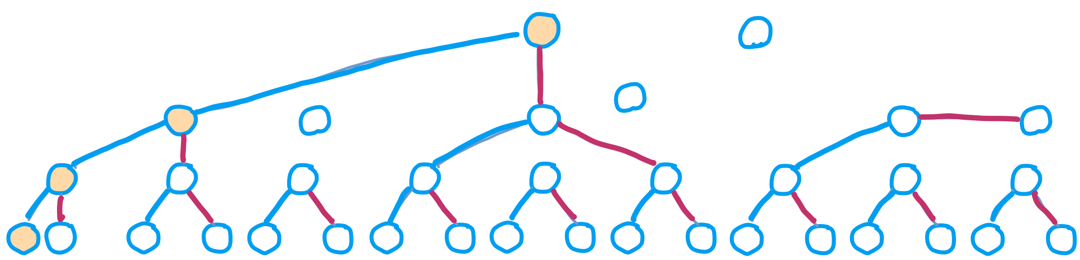


Reason the other way, down from root
How will we ever get black edges last in traversal?
Now, how do we avoid two reds in a row, down spine?
Leftmost spine has to be special.

The SYT (standard Young Tableau) is just a fancy way of writing words in {L, R, E} so initial segments never have more R than L, and no E can appear before first R.

So we want bijection between words and our trees.
They are essentially the same thing.
Tree traversal is extra structure on words, strip tree to get words.

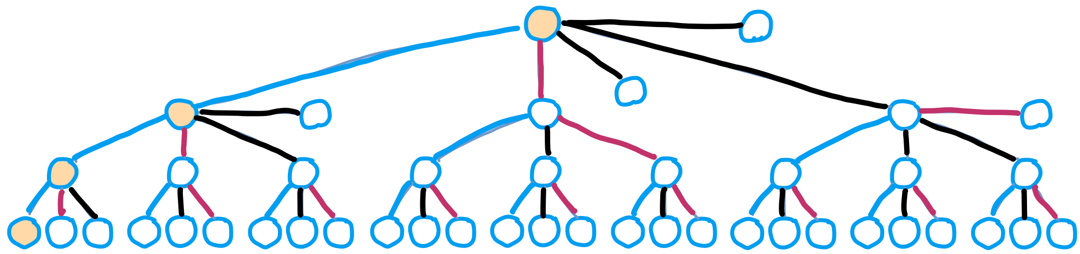
Need to see there is a unique tree for each word.



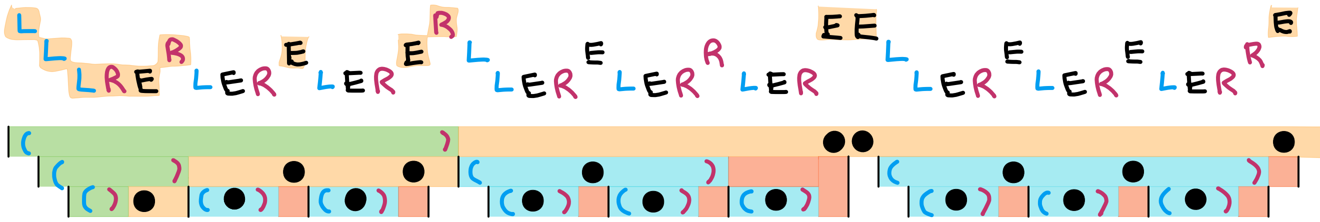
Remove black E edges, left with forest.
Enough to see that words lift to tree in order, and every E has unambiguous position in tree.

Seems to work, still feels unexpected.
Need code to check bijection for many cases.

Because trees are valence ≥ 2, leftmost doesn't need to climb multiple steps ever. Anywhere else, need possibility.



For each black edge or cluster, explain exactly its position.



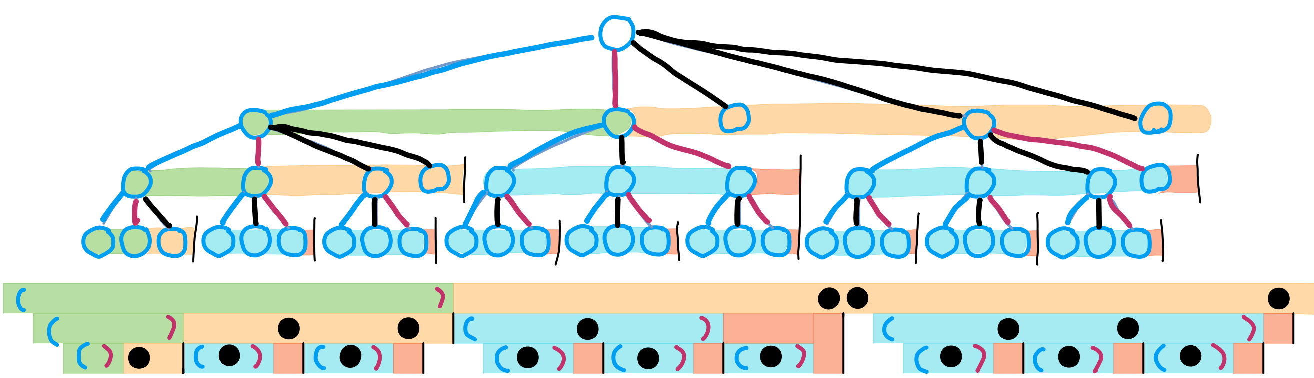
Each layer has a state, given by color bar
Each new symbol amends these states

- (always creates a new layer, green from spine else blue
-) turns green to yellow, rises through yellow turning it off, and turns off blue
- ● parks in first available layer, without changing state
  not allowed in green

This is a push-down automata, one could imagine general rules for similar trees.

We could generalize this to include all standard Young tableaus? Or more?
Gist of theorem now is that color scheme is unique per tree.

Can a standard PDA automata pop several symbols? Why not?
Is the general enumeration solvable? Hook length formula?

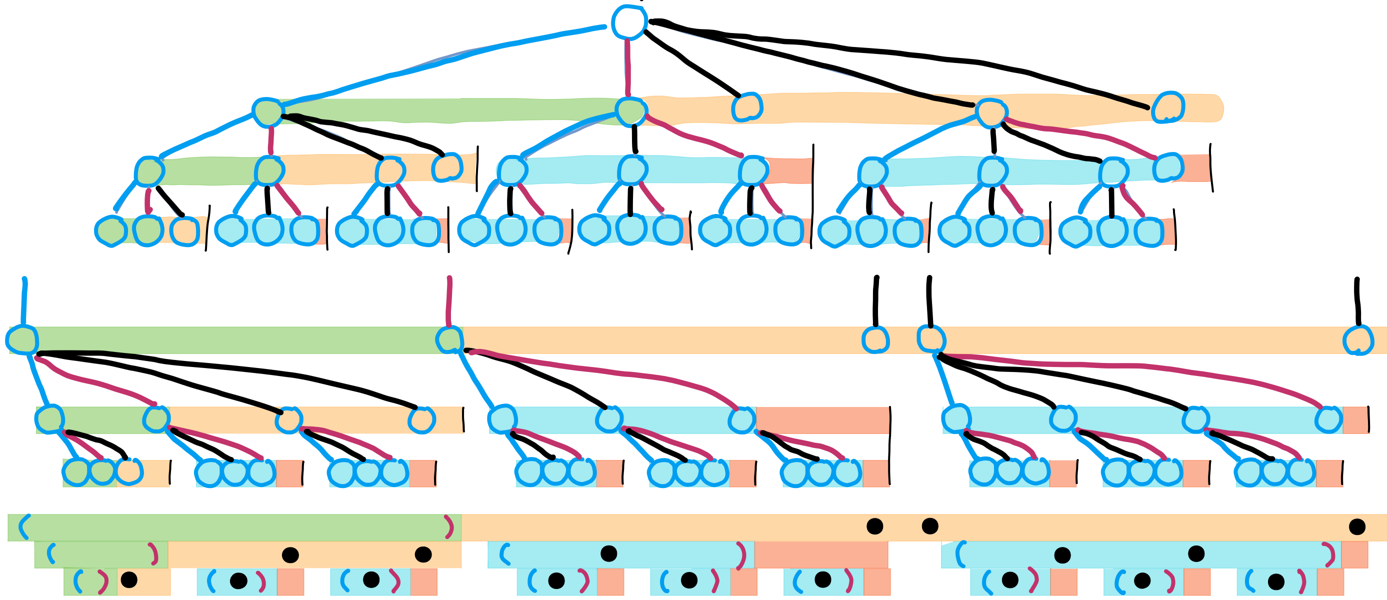


With colors as states, drawing editor could directly write machines such as PDAs.
Could machine learning guess "programs" to solve counting problems?
In this view, every PDA is a tree generator.

Would any PDA that recognizes language work here?

What does it mean that words can be recovered from their state sequences?

Redo to make traversal order and gaps clear.



Second form makes the tree clear, don't need tree edges.

	∅				
(	↓	↓	↓	↓	↓
)		—	↑	—	↑
●			—	—	↑

↓ pushes new state onto stack

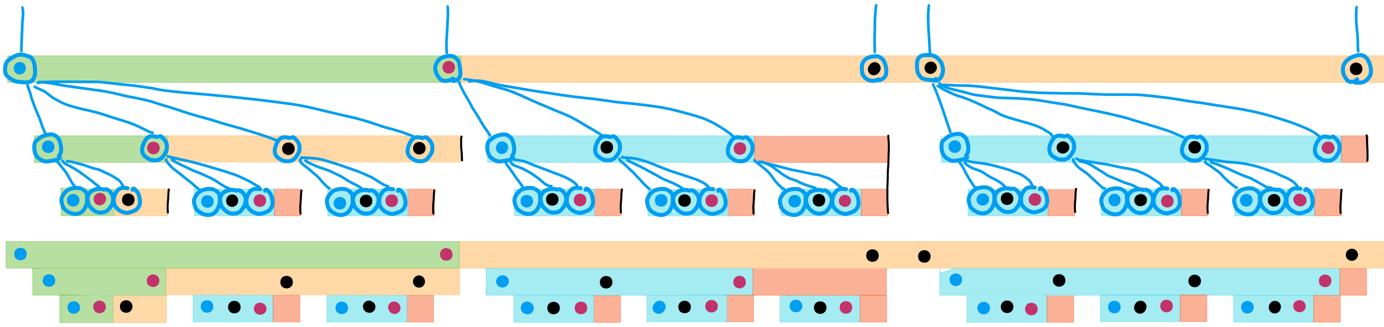
– sets current state

↑ pops state, tries again

\# end of string always pops one state, accept if empty

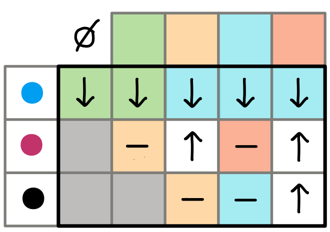
We could/should program this in this generality.

Under what circumstances does the language represent a SYT?



In a way, this is an attempt to pull a "Hilbert basis theorem" on various counting correspondences. Make finding and verifying the correspondence a routine computation.

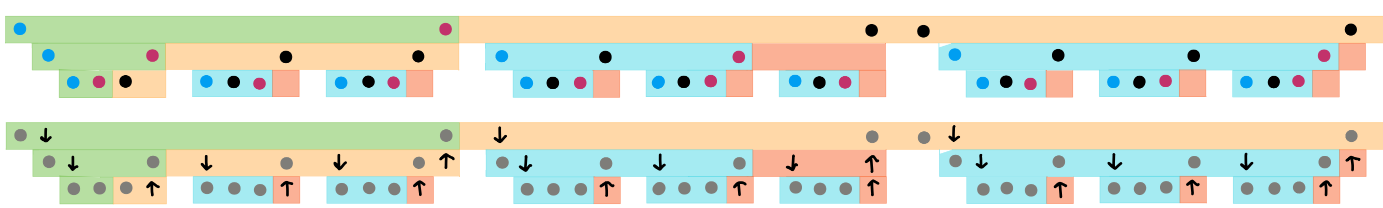
Given a word or a tree there's a word tree connecting them. What's the theory?



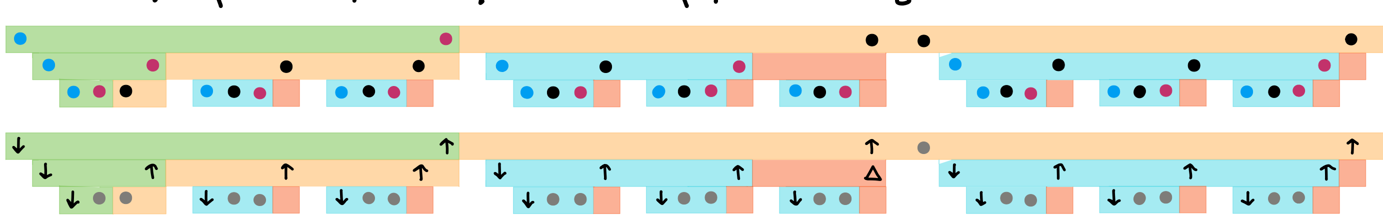
Any PDA table of this form lifts words that it recognizes to uniquely determined trees, nodes labeled by alphabet and levels retaining states of the PDA. (or not

$w \rightarrow wT \rightarrow T$

Lift word to word tree, drop labels to project to tree. What machine does the inverse?



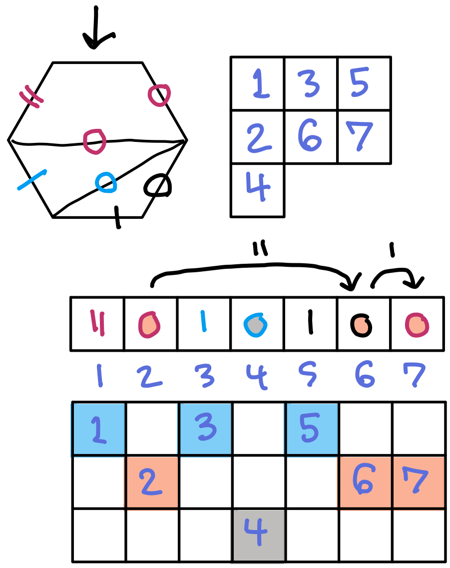
So with one kind of node stamper, need separate symbols for navigation?
Harder to define lift without backtracking.
Easy to have a symbol also go down one level. Multistep ↑↑ is the problem.
Like my Lisp notation, need symbol for empty rise. Augment words. △

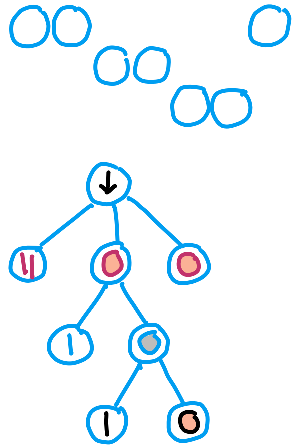


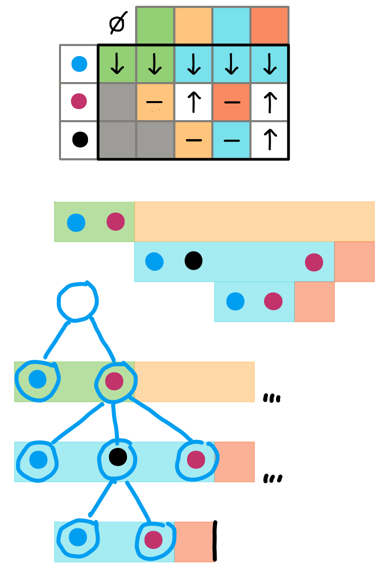
This reveals language issues. Marking a node as last seems to use lookahead.
Remove/fix this. Redefine the machine. Ideally we want transducers, not hardwired so one side is a tree. Sequence is traversal but machine has stack access to last...

Sunday March 14

Review Stanley's construction.







So yes, different construction.
Unlikely to be a PDA.

The machine I want doesn't label nodes in each level till next node.
Reasonable question: Is this transducer algebraic?
We need an end of word symbol to clear stack, and fix last symbols. #
Need more states for a lag transducer like this.
Or, output options, symbol if level extends, another for pop?