Website with slides
(and more...)

Why (almost) all puzzles, and many games, are really just Mazes,

And how to solve them.

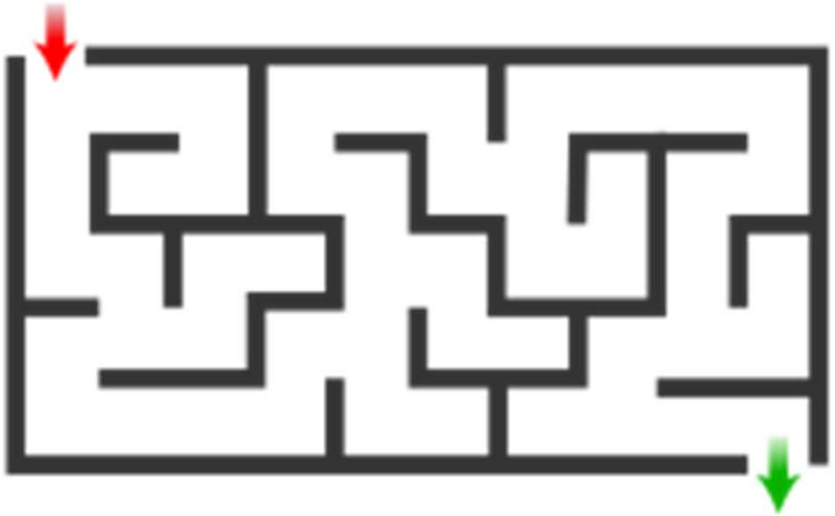
Brian Mintz
Dartmouth Grad Student Seminar
Spring 2024

Mazes

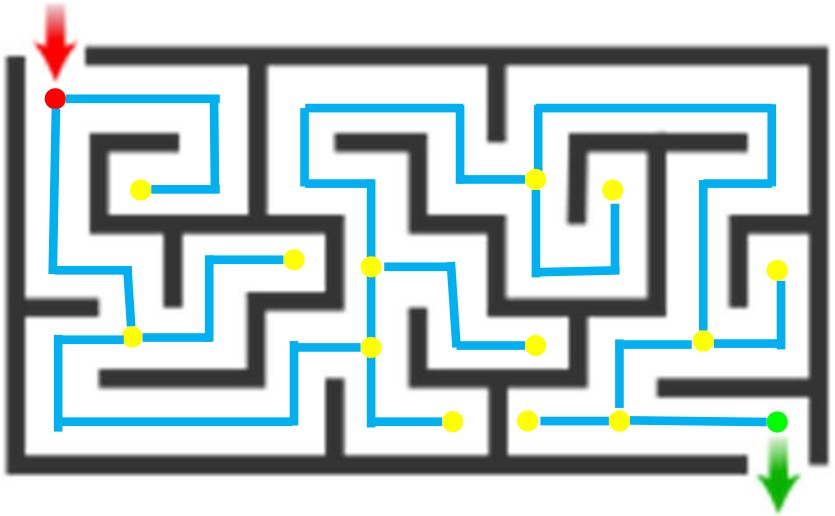
What are They?

And how do you solve them?

Classic Mazes + The dual

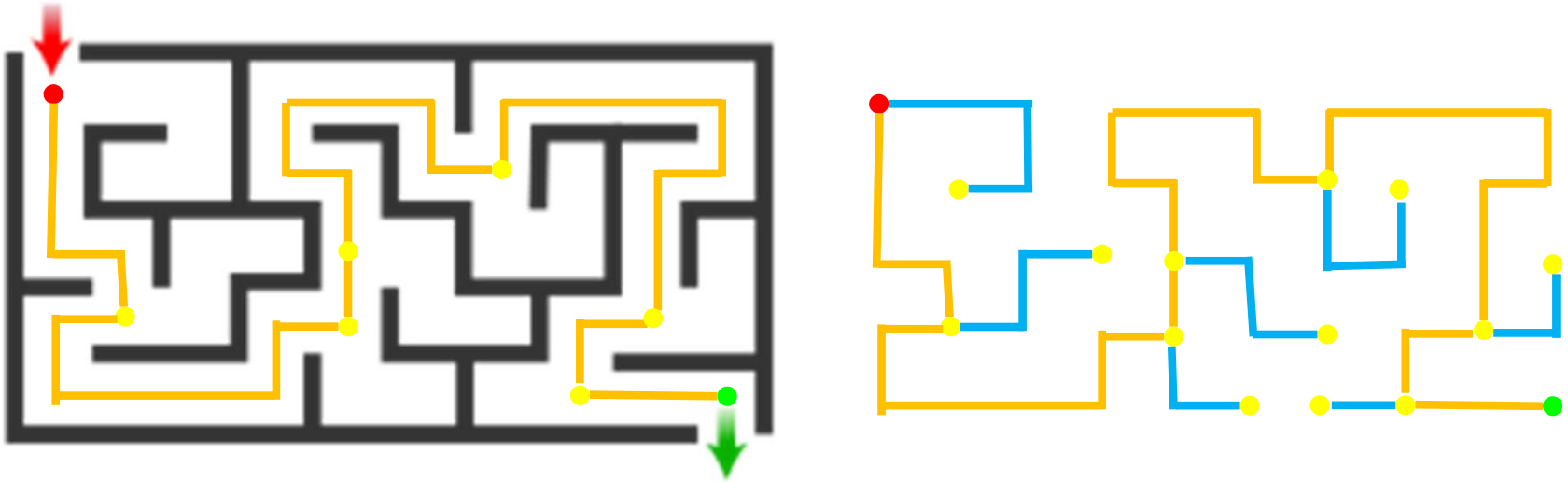


Classic Mazes + The dual



If we highlight the branch points and dead ends, we can summarize all possible positions in this maze with a graph!

Classic Mazes + The dual



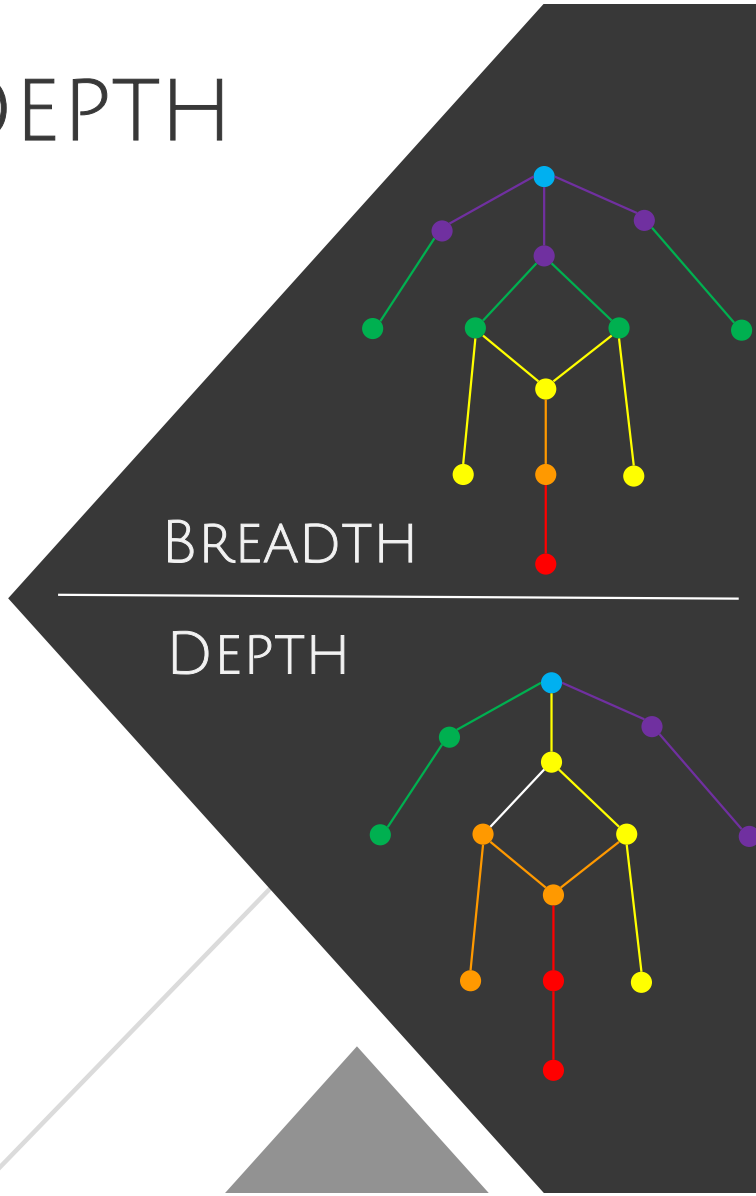
Solving a maze = finding a path in a graph!

But how do we do find this?

Breadth and Depth First Search

BFS: explore nodes uniformly from the starting vertex.

DFS: follows a path as far as possible, then backtracks to the previous branch and takes a different path. Also called “backtracking” or the “breadcrumbs” approach.



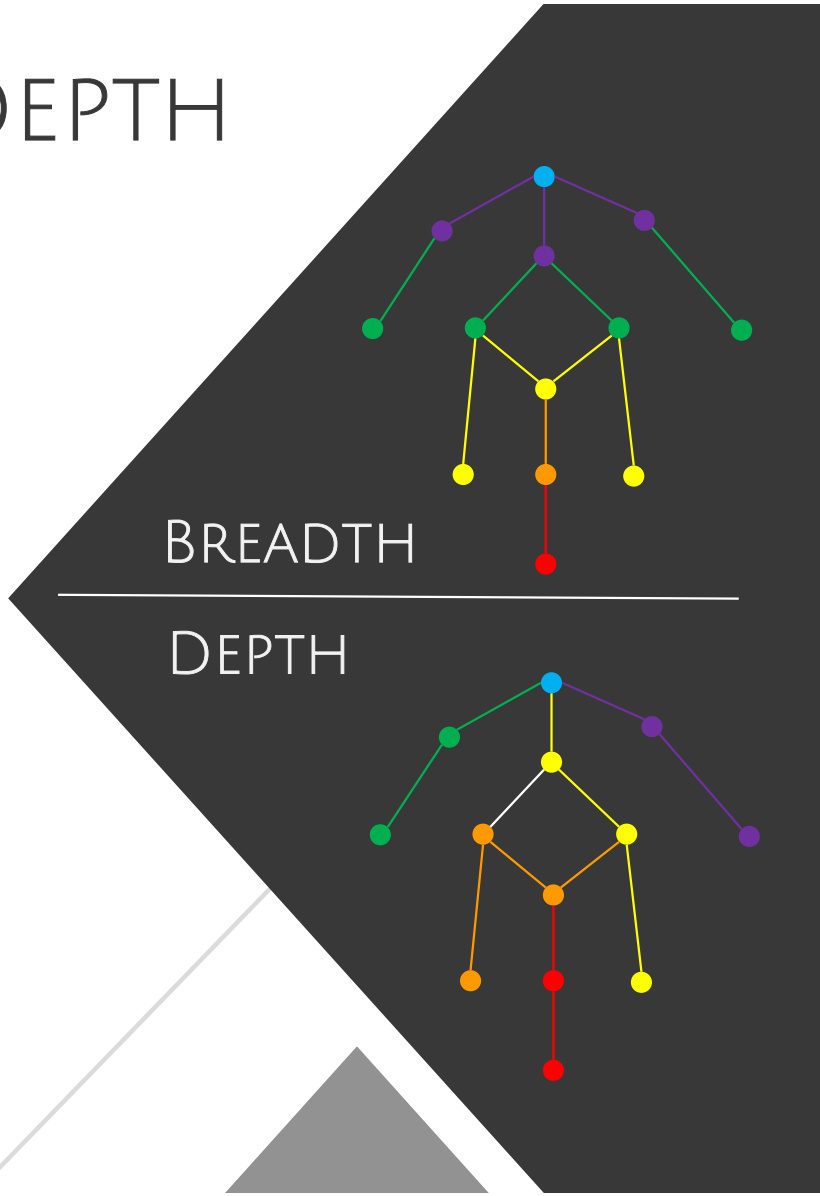
Breadth and Depth First Search

BFS: explore nodes uniformly from the starting vertex.

DFS: follows a path as far as possible, then backtracks to the previous branch and takes a different path. Also called “backtracking” or the “breadcrumbs” approach.

BFS requires more memory, but can be more efficient, especially if you are relatively close to the target, as you only need to explore to the radius of the target.

Both run in O(|V|+|E|) time in the worst case.



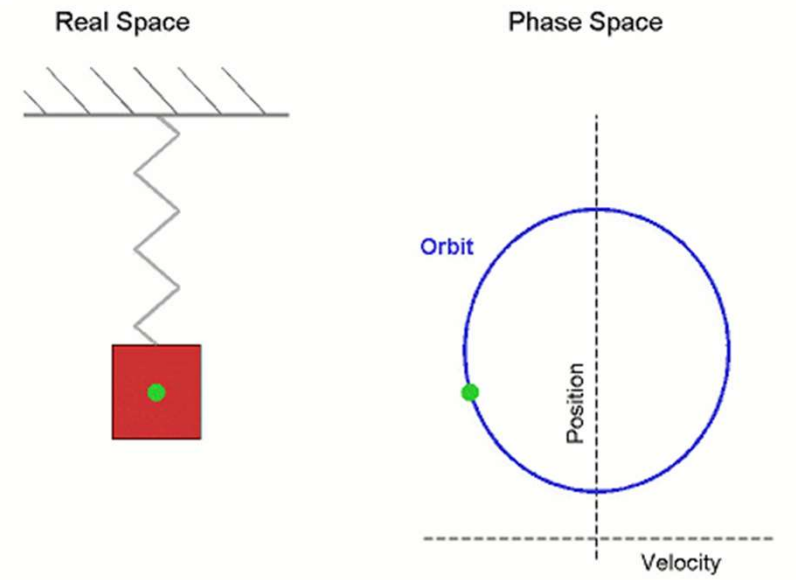
WHAT ABOUT PUZZLES?

The State Space

The key idea is to think of each "state," or configuration, of the puzzle as a point in some abstract space, with connections when a valid move changes between two states.

This is like the phase space used in differential equations.

In both cases, translating into this space provides and overview which makes solutions easy to see.

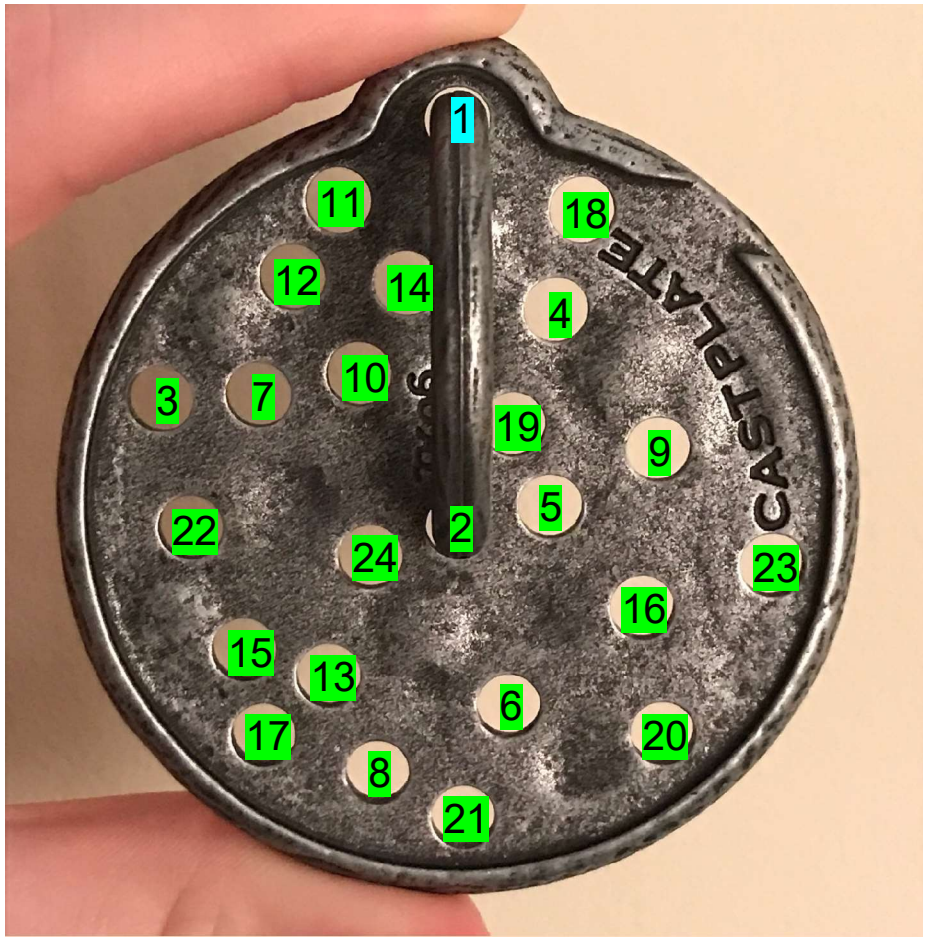


Gif source: Wikipedia Phase Space

Hanayama Cast Plate

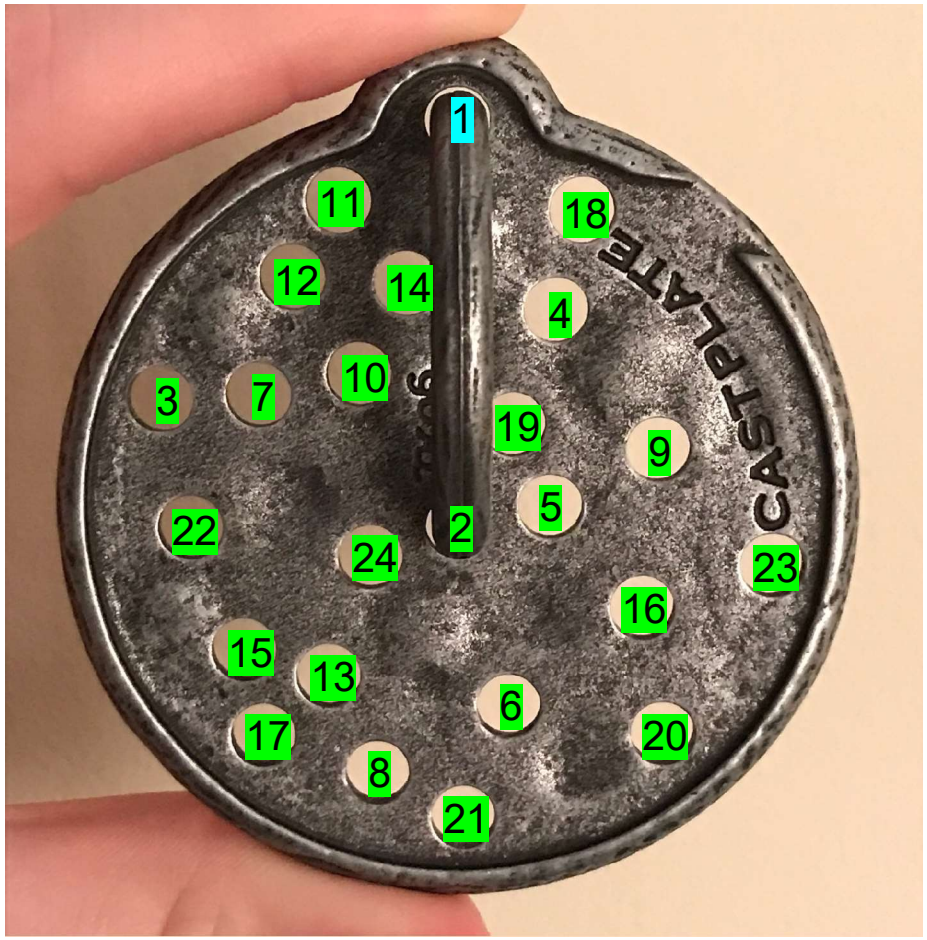
Feel free to try the puzzle I've passed around, and see if you can use the state space to navigate it!

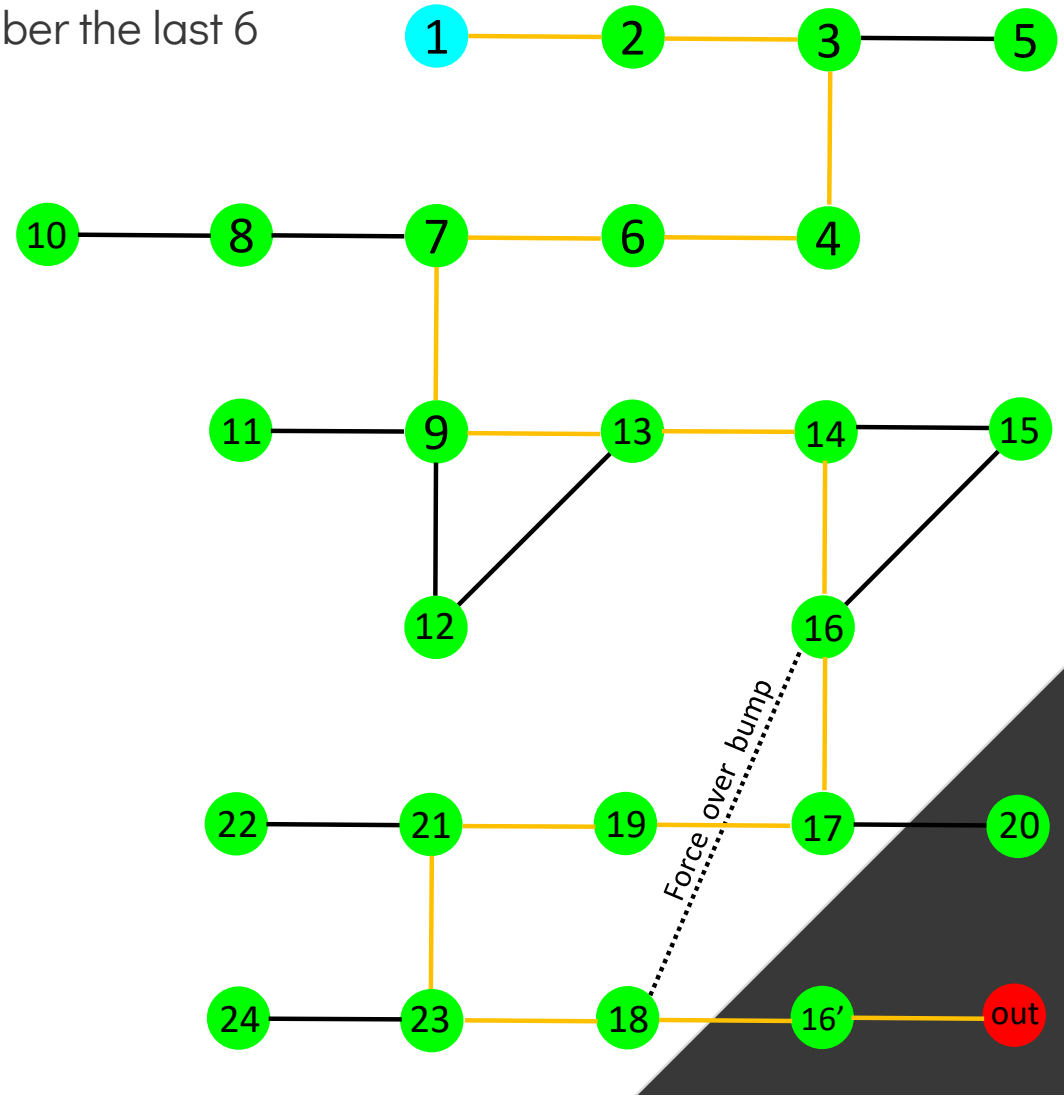
Solution



Solution

Pretty easy, only two loops. You just need to remember the last 6 states to solve it.

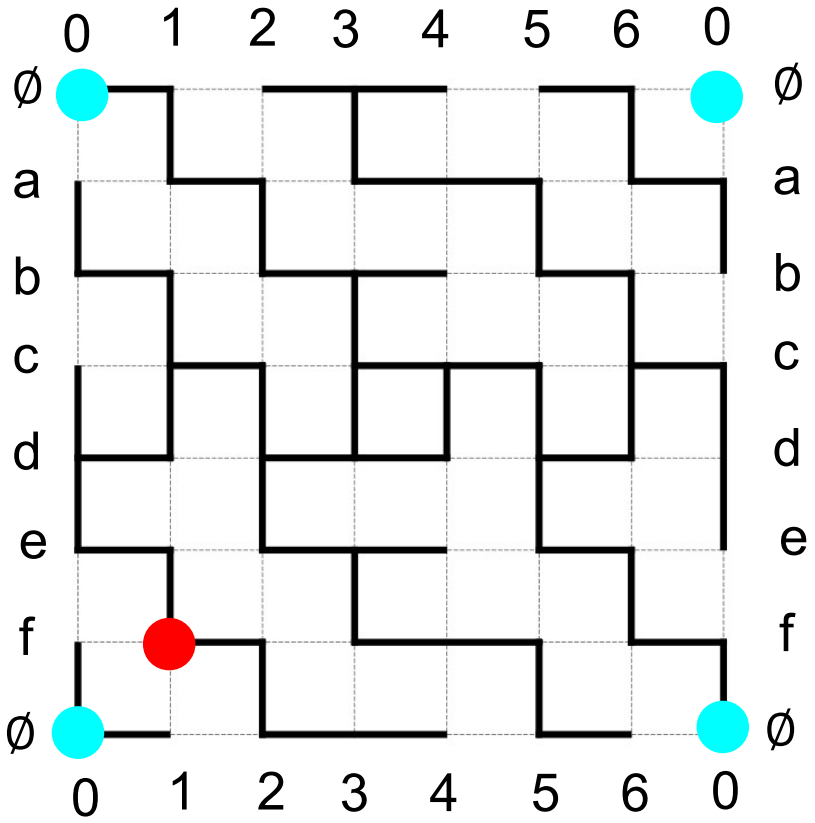


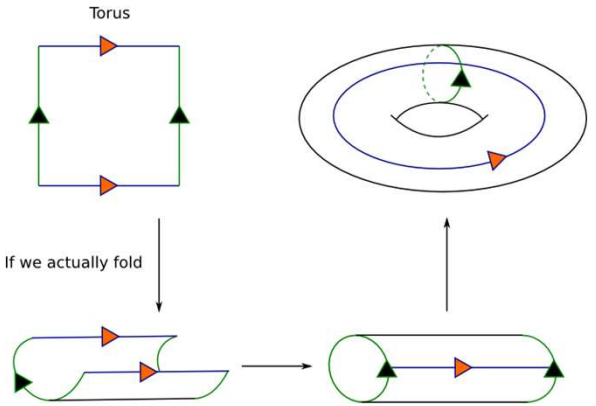


Hanayama Disk

Feel free to try the puzzle I've passed around, and see if you can use the state space to navigate it!

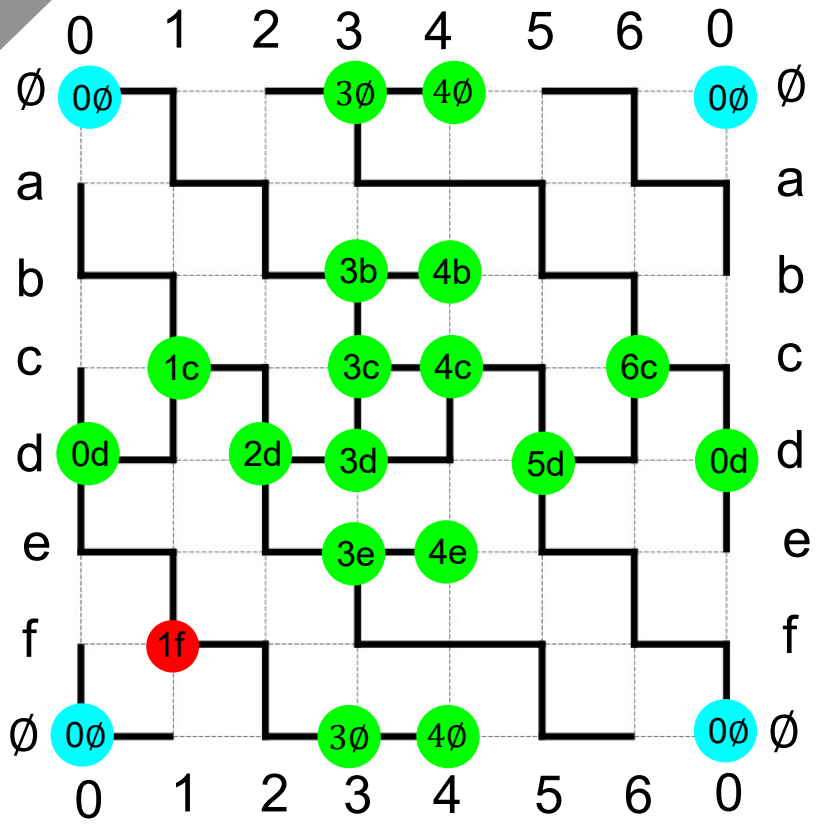
States are how much each disk is rotated, so $(\mathbb{Z}/7\mathbb{Z})^2$ this is a torus!!!

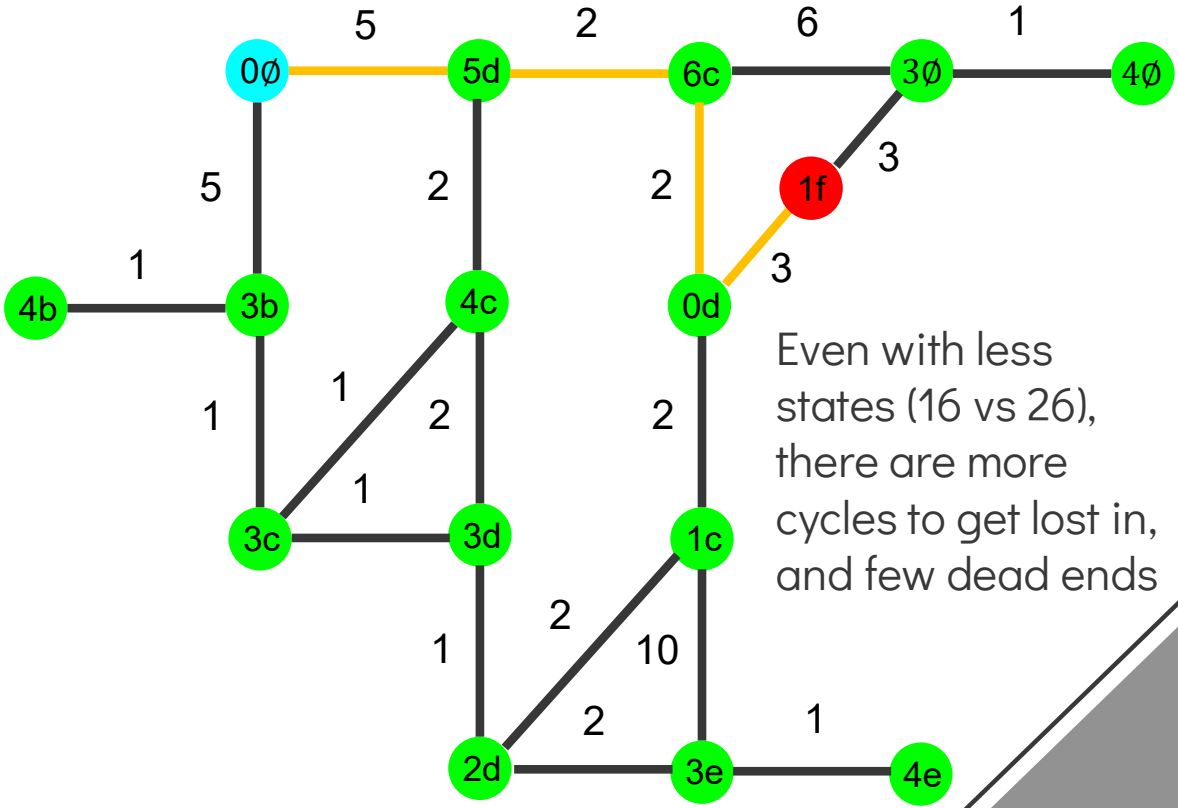




https://topologygeometry.blogspot.com/2010/06/notes-from-062310.html

Identifying the branches / dead ends, we can rewrite to simplify.
It's actually planar!





Even with less states (16 vs 26), there are more cycles to get lost in, and few dead ends

AND MORE...

Hanayama Box

Hanayama O'Gear

Hanayama Keyhole

Some puzzles from my office, everyone is welcome to come and make a configuration space for these!

A different Kind of Puzzle

For some obscure reason, you need to transport a fox, hen, and bag of grain across a river.

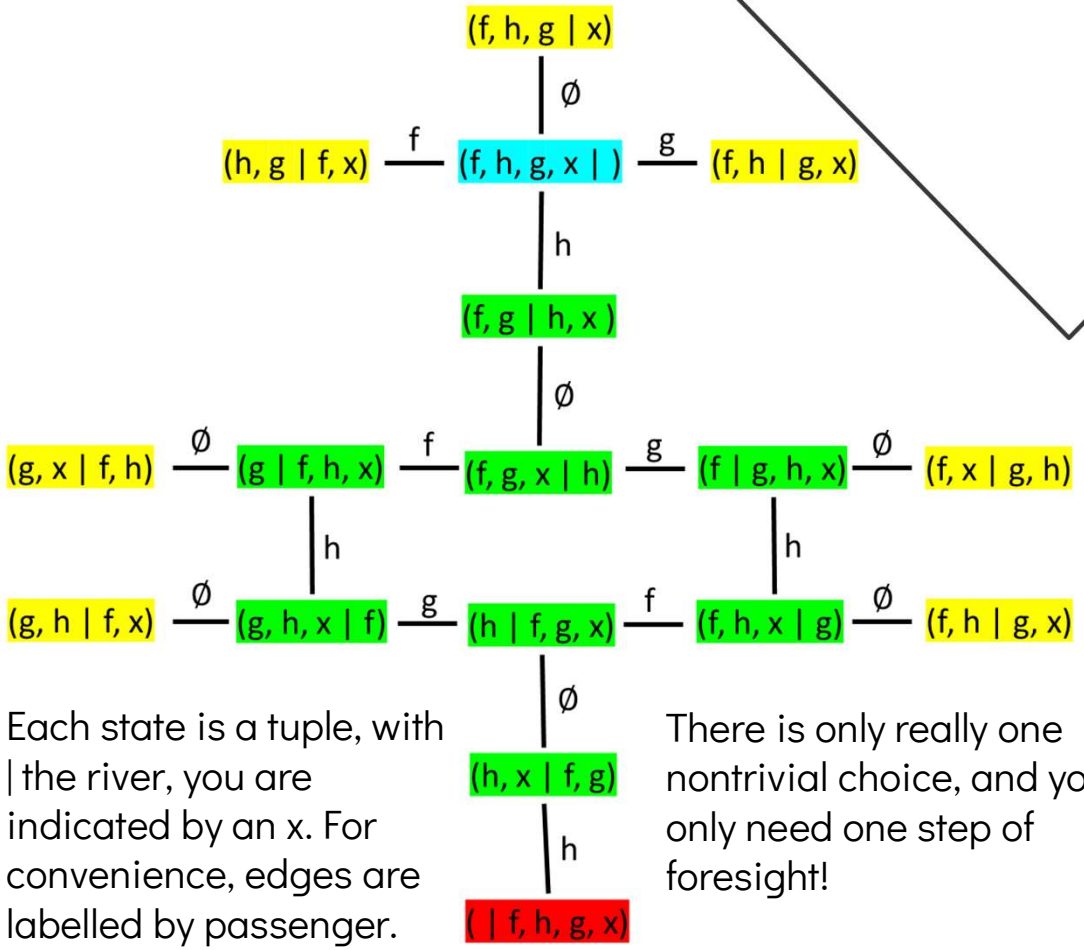
Your boat can only take one passenger at a time, and you can't leave the fox alone with the hen or hen alone with the grain, or else the former will eat the latter. How do you get all three across the river?

Each state is a tuple, with | the river, you are indicated by an x. For convenience, edges are labelled by passenger.

A different Kind of Puzzle

For some obscure reason, you need to transport a fox, hen, and bag of grain across a river.

Your boat can only take one passenger at a time, and you can't leave the fox alone with the hen or hen alone with the grain, or else the former will eat the latter. How do you get all three across the river?



Each state is a tuple, with | the river, you are indicated by an x. For convenience, edges are labelled by passenger.

There is only really one nontrivial choice, and you only need one step of foresight!

And Another

Edge Matching Puzzles

We can make a state space by placing tiles one at a time.

The key idea is to order all edges from a vertex so we can iterate through the possibilities.

And Another

Edge Matching Puzzles

We can make a state space by placing tiles one at a time.

There are $9!\,4^9 \sim 9.5 \times 10^9$ configurations, and only 12 solutions!

One run of DFS found all solutions all after only checking 85,261 states, about a millionth of the total!

The key idea is to order all edges from a vertex so we can iterate through the possibilities.

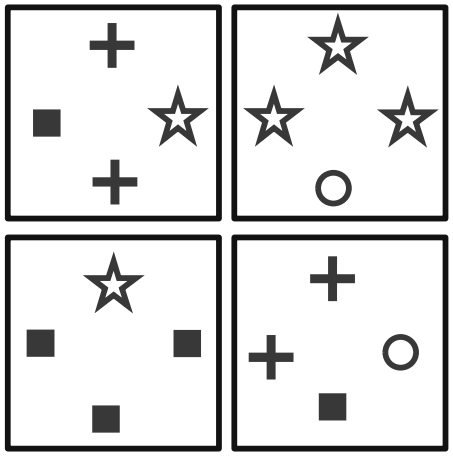
In general, solving one of these is NP-Complete

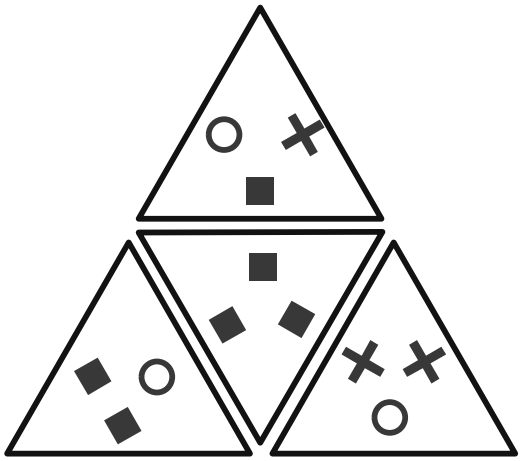
Edges with multiple matches increase the number of solutions, but also make more dead ends

Question: what portion of repeated edges maximizes the difficulty to solve via DFS?

Try it Yourself!

Use DFS to solve the below two puzzles, using the provided cutouts.



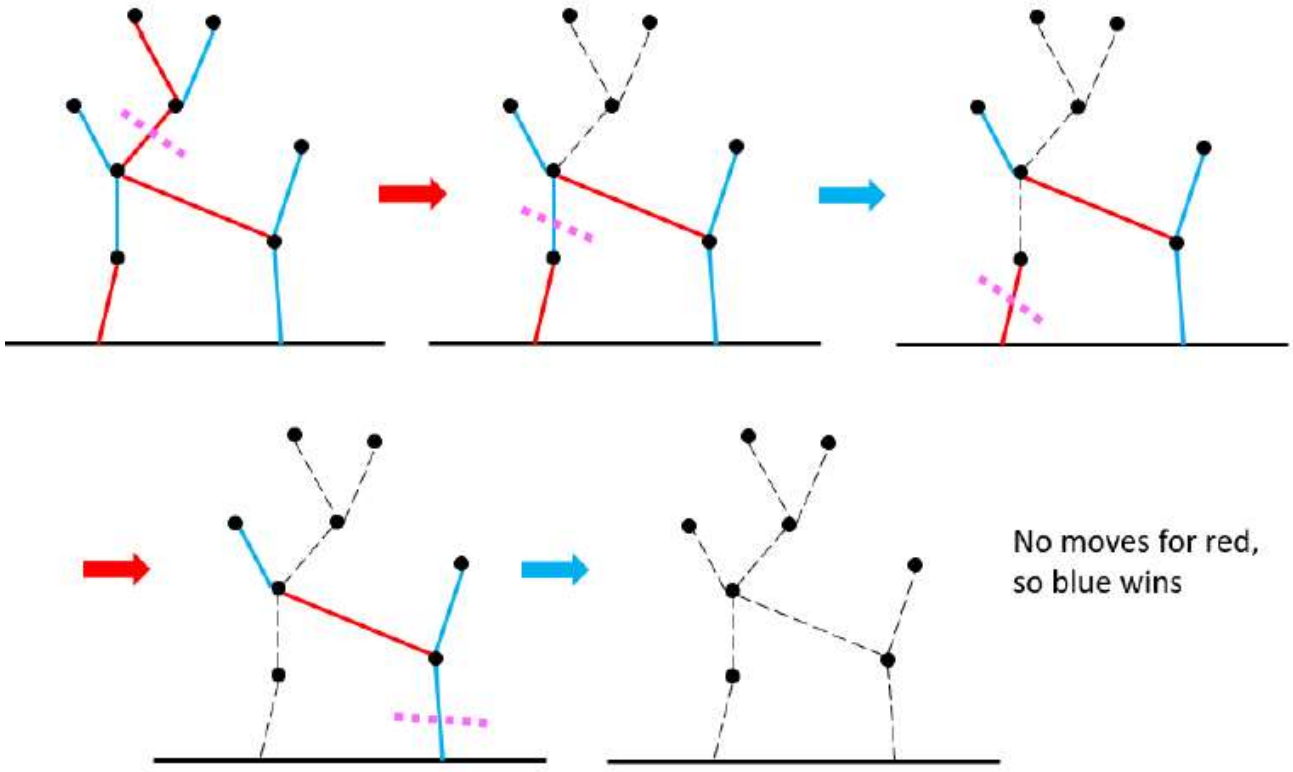


Another Amazing Application

Games and The Sprague-grundy Theorem

Games

Hackenbush: Two players, Red and Blue, alternate removing a segment of their own color until there are none left to choose. Further, all segments that become disconnected from the ground are also removed, as if they were ballons.

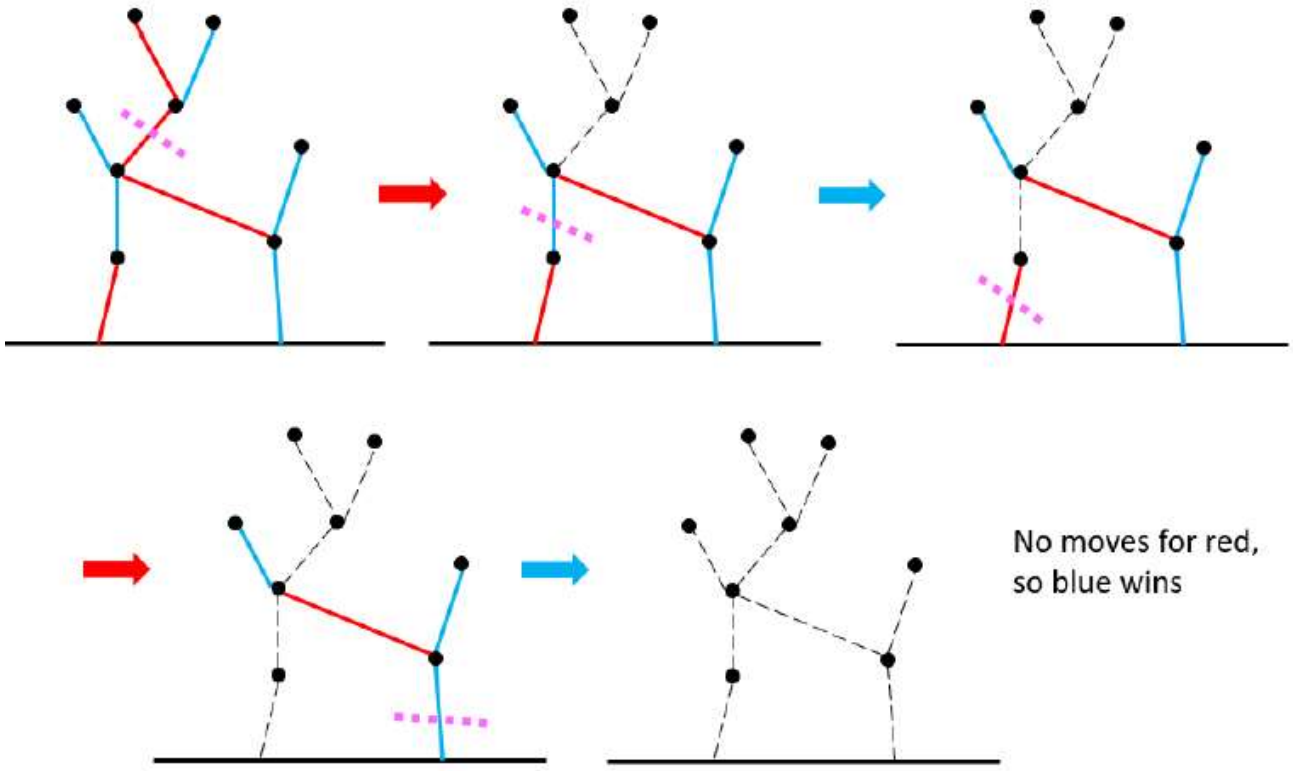


Games

Hackenbush: Two players, Red and Blue, alternate removing a segment of their own color until there are none left to choose. Further, all segments that become disconnected from the ground are also removed, as if they were ballons.

Just like puzzles, we can associate each state of a game to a vertex, with edges connecting states when an action is made.

The key idea is to associate a number to each state to tell us which action to choose. For this game, these turn out to be the "Surreal" numbers.



Minex

The Labeling

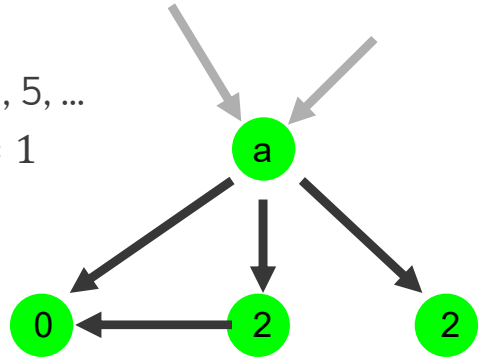
We are no longer just finding a path to the goal, as the other player can divert us.

Instead, we must decide for each node which edge to choose.

We start by labeling the terminal nodes (aka leaves) with 0, then we label each subsequent node the "minimum excluded," or minex, natural number of the labels of the nodes it points to.

Excluded values: 1, 3, 5, ...

Minimum is $a = 1$



Minex

The Labeling

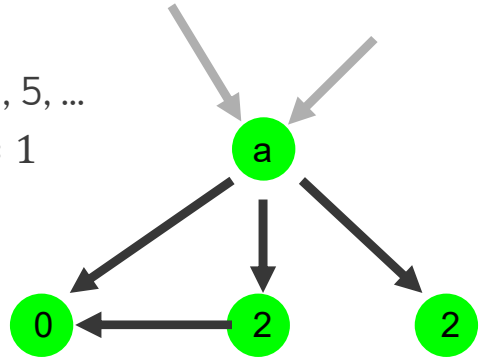
We are no longer just finding a path to the goal, as the other player can divert us.

Instead, we must decide for each node which edge to choose.

We start by labeling the terminal nodes (aka leaves) with 0, then we label each subsequent node the "minimum excluded," or minex, natural number of the labels of the nodes it points to.

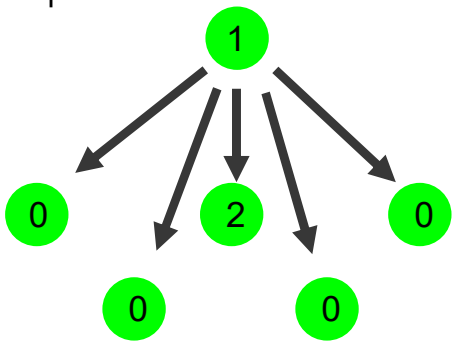
Excluded values: 1, 3, 5, ...

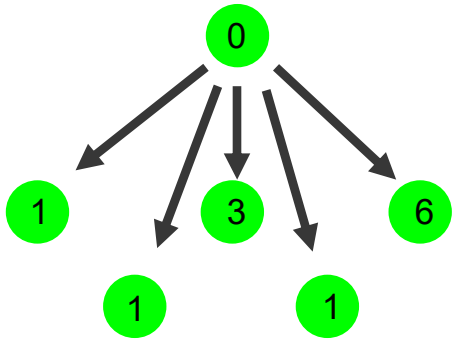
Minimum is $a = 1$



The strategy

Because of this rule, every state labeled zero only points to nonzero states, and every nonzero state points to at least one zero state.

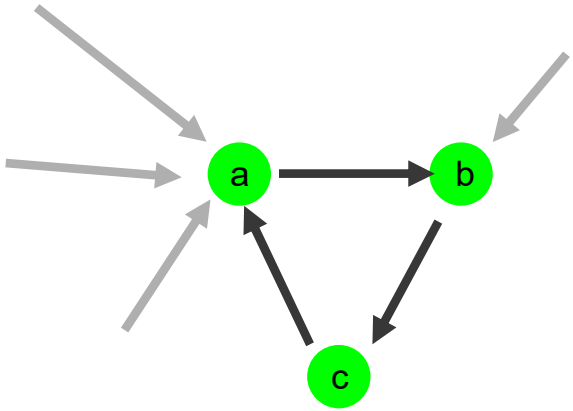




To play optimally, you just need to send the other player to a zero state. This repeats until the other player is stuck at a leaf, with no moves they lose. Thus, nonzero states are winning, and zero states are losing.

Amazingly, every finite impartial game can be labeled in this way! This result is known as the "**Sprague-Grundy Theorem**."

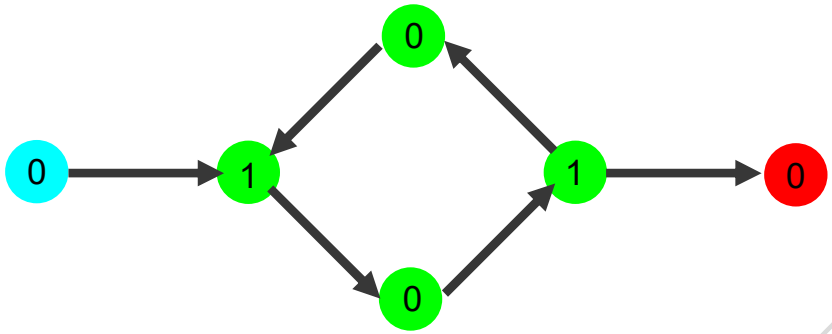
Why Cycles can be a problem...



$$c = 0 \rightarrow b = 1 \rightarrow a = 0$$

$$c \neq 0 \rightarrow b = 0 \rightarrow a = 1 \rightarrow c = 0$$

But don't need to be.



The Magic of Minex

Really, you only need labels Z and NZ, for zero and nonzero, to do this.

The true magic of minex is that it plays nice with the "sum" of two games.

An action in a sum of games consists of making a move in one game while the others stay the same.

The Magic of Minex

Really, you only need labels Z and NZ, for zero and nonzero, to do this.

The true magic of minex is that it plays nice with the "**sum**" of two games.

An action in a sum of games consists of making a move in one game while the others stay the same.

It turns out the label of a state in the sum of games is the **digital sum**, also known as the XOR of the binary representation, of the labels of the constituent states! This means if a game has a natural recursive structure, it is easy to find the label of any state.

If you know the label of any state, you can apply this strategy to win every time!

PROOF

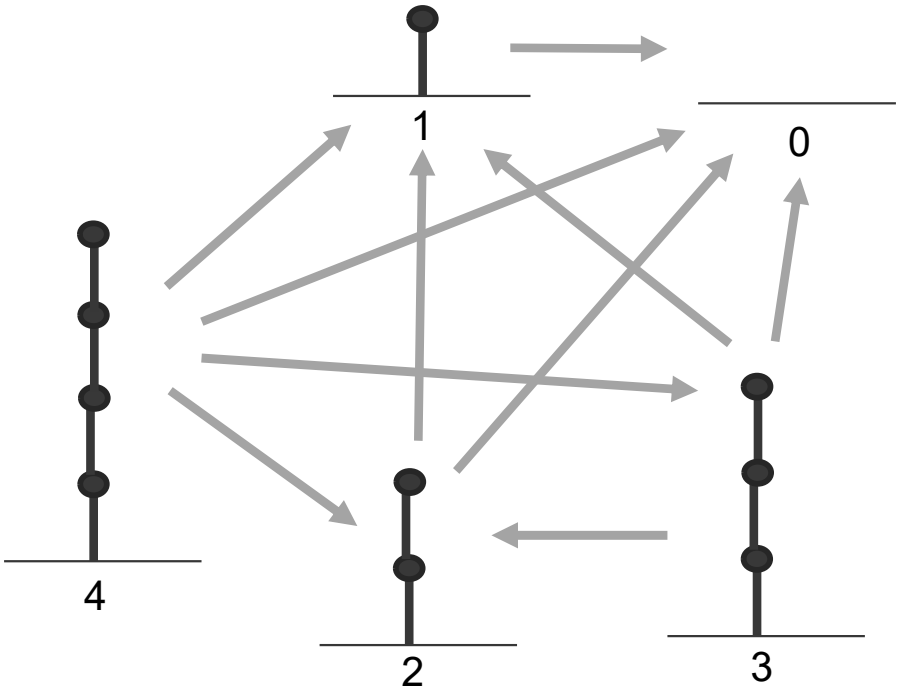
Theorem 2. *The label of a state (v_1, v_2) in the game $G_1 + G_2$ is the nim sum (XOR) or the labels of each state in its own game.*

Proof. First we establish some nice properties of the XOR operation. This operation is associative and commutative (kind of obvious, but should be stated). Next, $0 \oplus x = x$, because all 1's will stay 1's, as they differ from the 0's in 0, and all the 0's will stay 0's, as they match the 0's in 0. Lastly, $x \oplus x = 0$ so we have cancellation, $x \oplus (x \oplus y) = y$.

Let a and b be the labels of v_1 and v_2. By definition, the value of (v_1, v_2) is the mex of the set of labels of states it leads to, so we need to show $a \oplus b$ is excluded, and any value $c < a \oplus b$ is included in this set. To see these, it helps to characterize these labels. Since we play a sum of games by moving one, this means all reachable states look like (u, v_2) or (v_1, w) where v_1 can reach u and v_2 can reach w. By induction(ish), these will have labels $x \oplus b$ or $a \oplus y$ where $x \neq a$ and $y \neq b$, since mex. All $x < a$ and $y < b$ are included, since the mex is minimal, and some greater values may also be included.

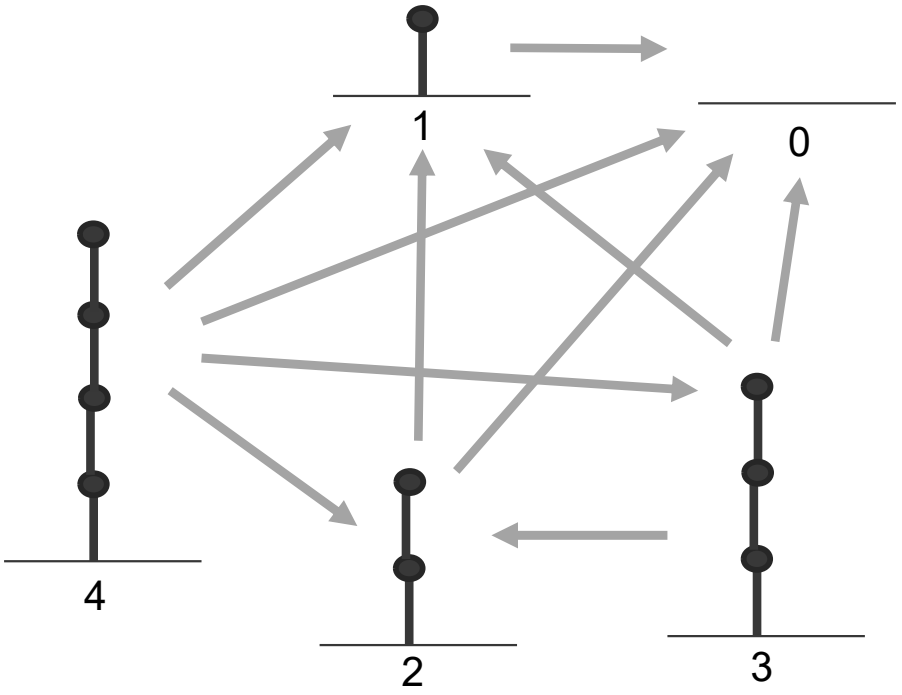
To see the first note $a \oplus b$ can only be obtained when $x = a$ or $b = y$, by cancellation, which can't occur since they are mex's. To see the second, define $d = a \oplus b \oplus c$. Note that XOR-ing d with a, b, and c must reduce at least one, since the leading 1 in d must appear in at least one of the three (if all were zero in this place, d wouldn't be, also $d \neq 0$ or else $c = c \oplus d = a \oplus b$). But $d \oplus c = a \oplus b > c$, so the reduction must be $a > d \oplus a$ or $b > d \oplus b$. In the former, this means $x = d \oplus a$ works, as then $x \oplus b = (d \oplus a) \oplus b = (b \oplus c) \oplus b = c$ by cancellation, and we know all lower values are achieved, by minimality of mex. The same can be done in the other case with $y = d \oplus b$. □

Hackenbush (NIM)



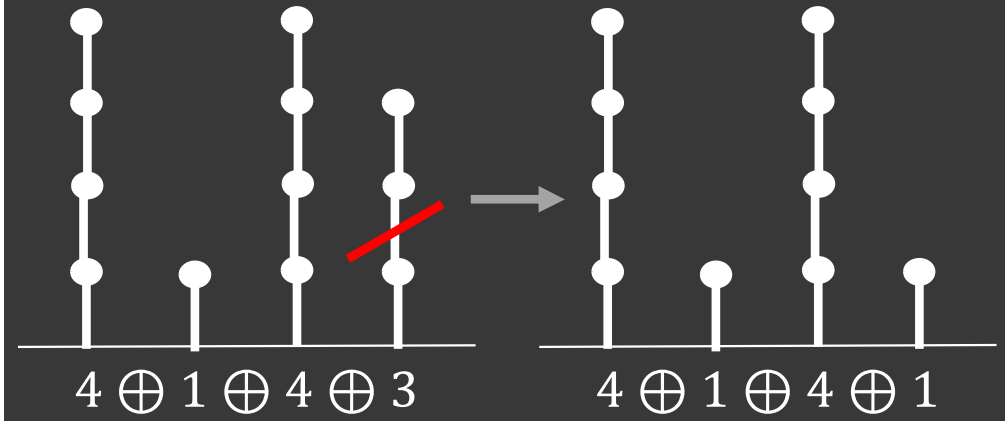
A stalk of length n has value n.

HACKENBUSH (NIM)



A stalk of length n has value n.

Proof by induction: it is connected to all stalks of length k where k < n, which have value k, so the minex value is n.



$4 \oplus 1 \oplus 4 \oplus 3$ $\quad$ $4 \oplus 1 \oplus 4 \oplus 1$

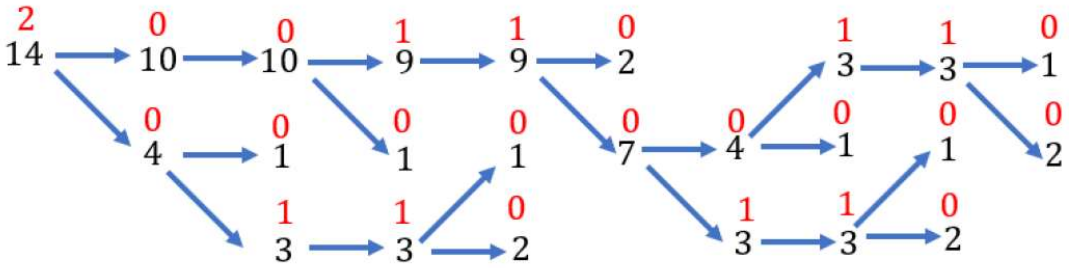
1	0	1	0	0
0	0	0	1	1
0	1	1	1	1

→ 0

The winning move is to make the sum zero. One trick to do that is to balance the piles, using the fact that $x \oplus x = 0$ for all x

AND MORE...

Grundy's Game – start with a pile of n coins. At each turn, a player can split any pile into two piles of unequal sizes.



n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	0	0	0	1	0	2	1	0	2	1	0	2	1	3	2	1	3

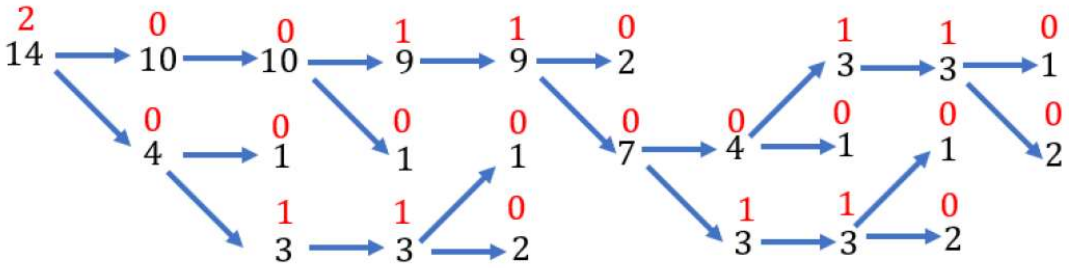
Last to 21 game: Starting from 1, players alternate adding either 1, 2, or 3 to the previous sum until one reaches or exceeds 21, and loses.

The value of a state is the remainder modulo four (seen by going backwards from 21).

n	1	…	14	15	16	17	18	19	20
	1	…	2	1	0	3	2	1	0

AND MORE…

Grundy's Game – start with a pile of n coins. At each turn, a player can split any pile into two piles of unequal sizes.



n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	0	0	0	1	0	2	1	0	2	1	0	2	1	3	2	1	3

Last to 21 game: Starting from 1, players alternate adding either 1, 2, or 3 to the previous sum until one reaches or exceeds 21, and loses.

The value of a state is the remainder modulo four (seen by going backwards from 21).

n	1	…	14	15	16	17	18	19	20
	1	…	2	1	0	3	2	1	0

AlphaGo

The idea of representing a game by a graph of its states and assigning values to these is at the core of many Artificial Intelligence algorithms for games.

Stockfish, the leading chess program

Conclusion

- We can use an abstract space to encode valid states, and their connections, of a puzzle or game to make it easier to find a solution.
- Solutions are paths in the state space, which can be found by BFS or DFS.
- This approach applies to a wide variety of puzzles, and many games.

I think of puzzles can serve as a metaphor for life.

It's important to understand the space you're in and where you are in it. Without this, you can get stuck in a cycle, or go down a dead end. As we navigate our lives, we're always trying to choose the actions that we think are most likely to help us reach our goals.

THANKS!