Glimmer & Gloom: Turning off hexagonal lights

By Toby Bowers

Introduction

Flight Rising is a web-based video game where players collect, customise and breed colourful dragons. This essay, unfortunately, isn't about dragons. Within Flight Rising, players can earn virtual currency by playing minigames. One minigame - "Glimmer & Gloom" (henceforth G&G) - sticks out.

Players consider G&G the easiest minigame because you can solve a level in seconds using a simple strategy. No guides I could find, though, explained *why* the strategy works. I hope to do that.

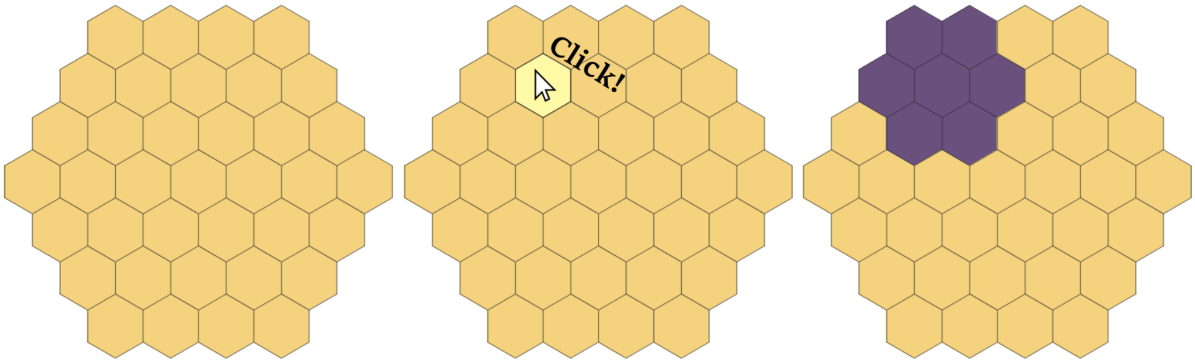
How do you play G&G?

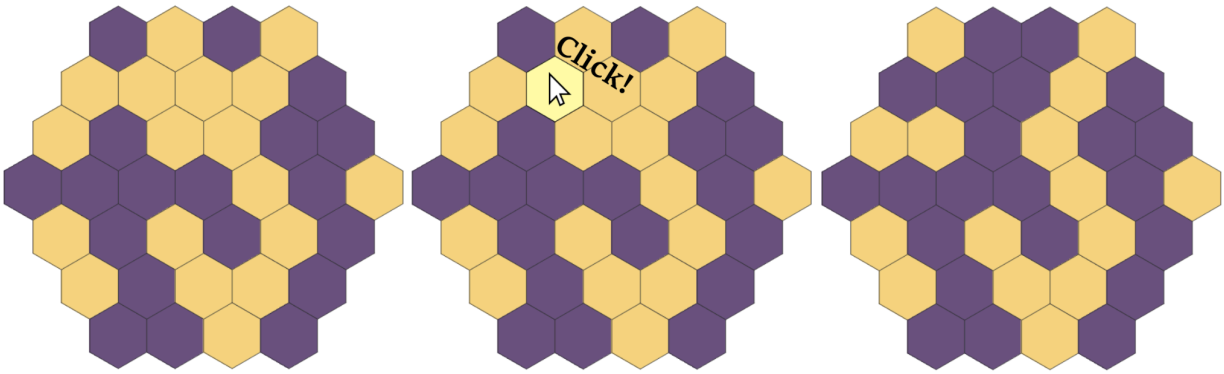
G&G is a single-player hexagonal-tile-based puzzle game where each tile has one of two states: light or shadow.

G&G has several modes, each with different board shapes. I will focus on Hard mode, as this is the mode most players recommend. Hard mode uses a board composed of 37 small pointy-top hexagons (tiles) arranged in rings around the center, so they form a larger flat-top hexagon.

(Screenshot of the official game)

You can swap the state of any tile by clicking it, but you will also swap the states of all tiles adjacent (up to 6 tiles). Every time you click a tile, this counts as a move.





When you start a game, the board appears all jumbled up. This jumbled up effect is created by starting with a board of all the same colour and “clicking” random tiles, then showing the result to the player.[1]

The aim of the game is to make all the tiles the same - either all light or all shadow. Every game can be won with either (though one solution may take less moves than the other). For the purposes of this essay, I’ll be treating all shadow tiles as my win condition.

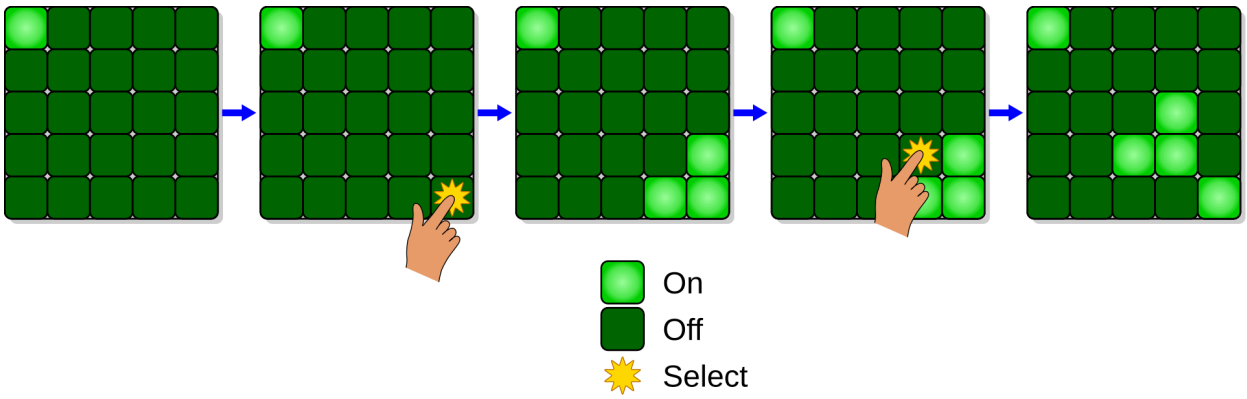
You can take as much time and use as many moves as you like.

Unfortunately, playing G&G requires a Flight Rising account. If you’re desperate to play along, I have coded a rudimentary version of the game. But, playing the game isn’t necessary to enjoy the explanation.

This reminds me of something…

Have you heard of the game Lights Out? If not, Lights Out was a game first released in 1995 by Tiger Electronics, with many spin-offs and variations thereafter. You are given a 5x5 square grid with some lights turned on. Whenever you click a tile, you toggle its state as well as the states of all neighbouring tiles. The goal of the game is to turn all the lights off.

[1] You might want to consider why the developers chose to get their random configurations this way, as opposed to just randomising the state of each tile. A more straight-forward question would be: are all configurations of this board solvable? We’ll answer this later.



([Image source])

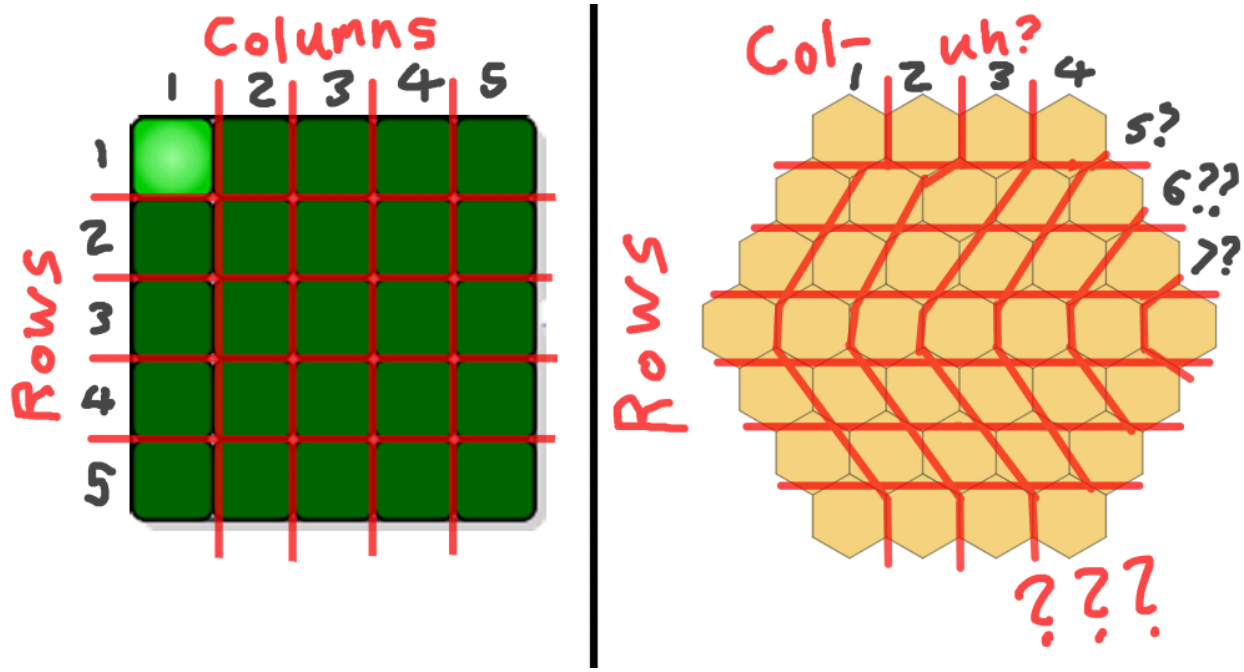
Sounds familiar, right? G&G is just the hexagonal version of Lights Out!

Mathematicians heavily studied Lights Out and I'll be repeating their work here, modified for G&G.

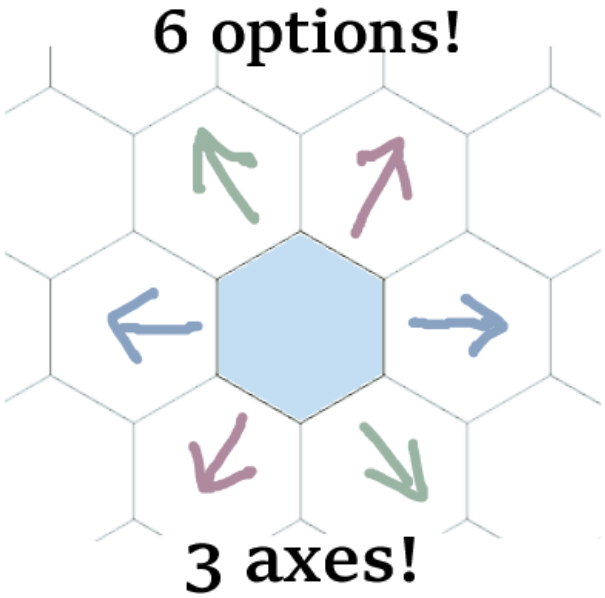
Glimmer & Gloom & Representation

Tile coordinates

We need a convenient way to refer to any given tile. With a square grid, this is quite straight forward: columns and rows. When we try applying this to hexagons though, it doesn't work as well…

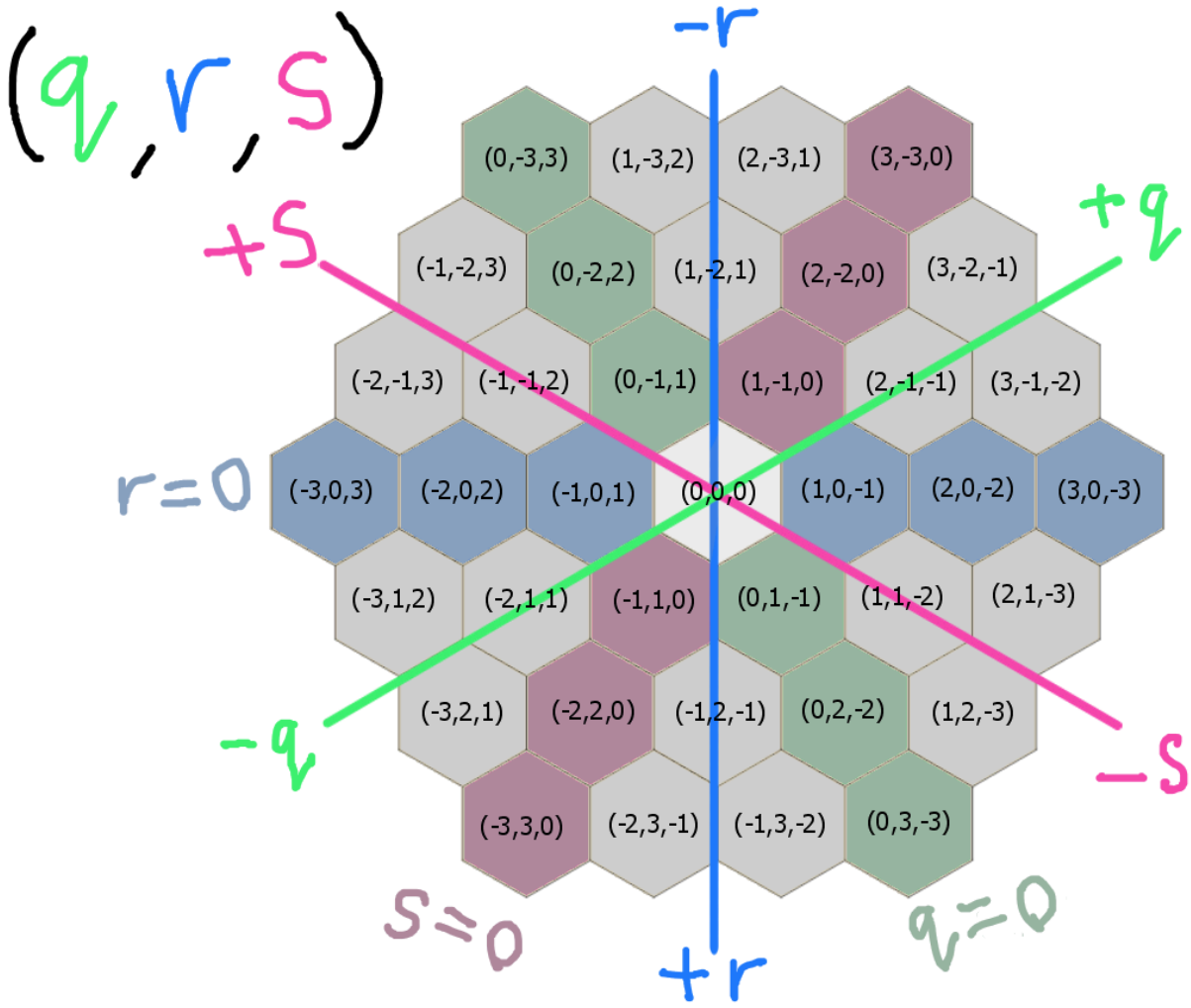


Rather than trying to force 2 primary axes, why don't we try 3 primary axes? Think about it: for a square grid, we use two axes because we can move up/down ("|") and left/right ("-"). On a hex lattice, we can move Northeast/Southeast ("/"), Northwest/Southwest ("\") and West/East ("-").



Let's imagine these tiles on the 3D plane of q + r + s = 0. Each tile is given a coordinate (q,r,s) with the center tile being (0,0,0). In any direction we move, we are moving

forwards (+1) in one axis, backwards (-1) in one axis and stationary (+0) in one axis. This system allows us to easily see which tiles are neighbours and which tiles are in a line. It also makes the symmetry of the board obvious.

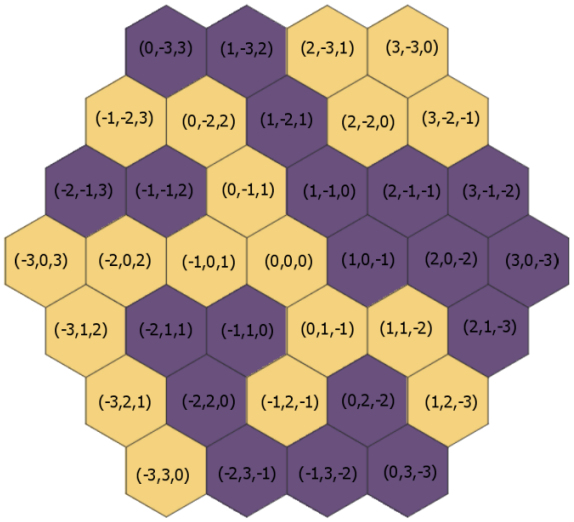


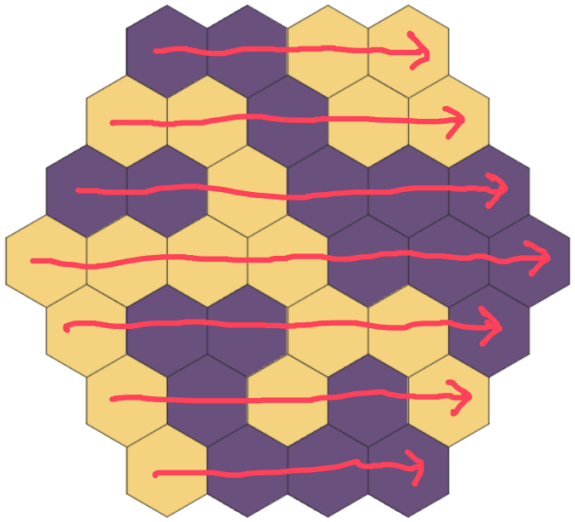
(If this confuses you, don't worry. I'll always label tiles clearly so you won't need to memorise this system. However, familiarising yourself with it will make some explanations more clear.)

Vector representation

We can represent the state of all the tiles on the board as a column vector. This is a list of 37 values, 0 if a tile is shadow, 1 if a tile is light. We will denote the state of a given tile $a_{q,r,s}$ where q,r,s denote the tile's coordinate (see previous section). It doesn't matter what order we write the tiles in as long as we're consistent, I'll write them left-to-right, row-by-row.

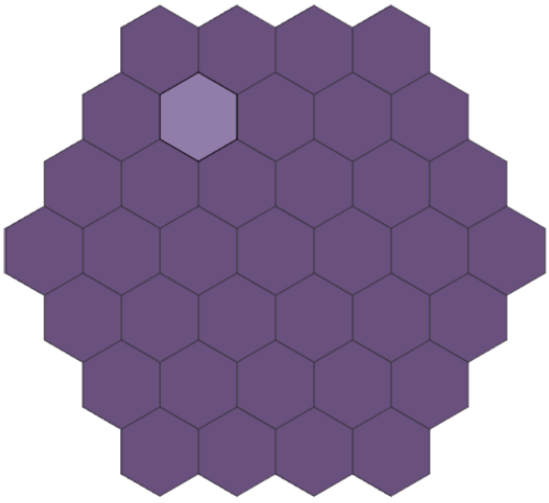
An example board is shown below.

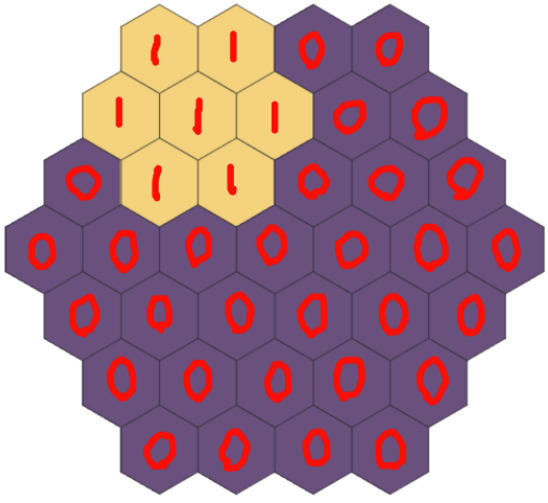




A = ($a_{0,-3,3}$, $a_{1,-3,2}$, $a_{2,-3,1}$, $a_{3,-3,0}$, $a_{-1,-2,3}$, …)
A = (0, 0, 1, 1, 1, …)

We can also write the effect of a tile press as a column vector too. We'll use 0 for the tiles that don't change, 1 for the tiles that do. We'll denote the tile we press as $X_{q,r,s}$. As an example, we'll press tile (0,-2,2).





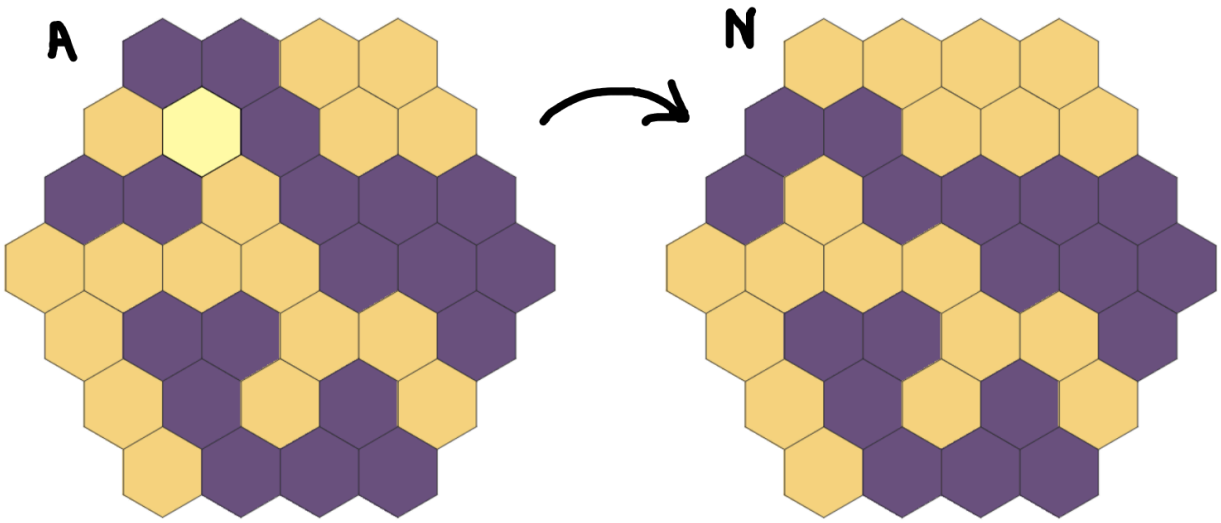
$X_{0,-2,2}$ = (1, 1, 0, 0, 1, …)

We can then find the configuration of the board after the button press by adding the two vectors together modulo 2. With the classic integers, 1+1=2, but this doesn't work

for us as tiles should only have the state 0 or 1. Modulo 2 means no integer is allowed to go above 2, so if we are ≥ 2, we subtract 2 until we reach 1 or 0. This is similar to the XOR function.

1 XOR 1 = 0	(Light tile toggled, turns to shadow)
1 XOR 0 = 1	(Light tile not toggled, remains light)
0 XOR 1 = 1	(Shadow tile toggled, turns to light)
0 XOR 0 = 0	(Shadow tile not toggled, remains shadow)

In short, if you add the vector describing the current board and a vector describing a particular button press, you'll get the vector describing the board after that button press.

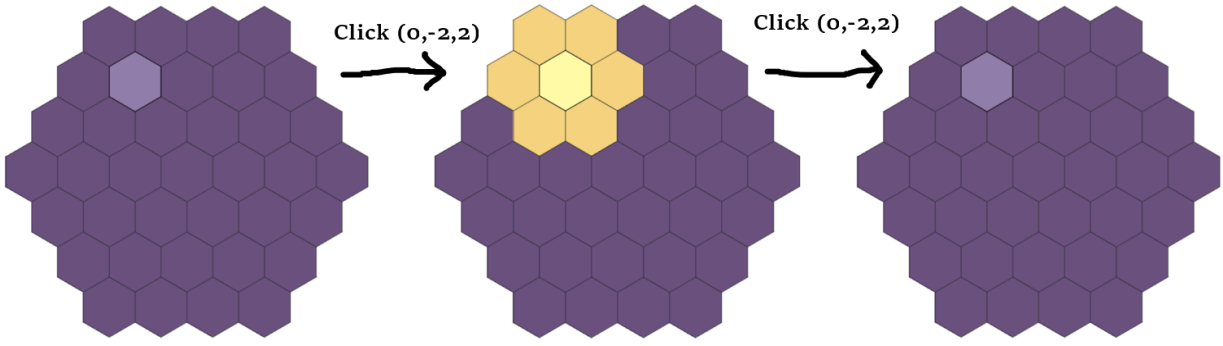


$$A + X_{0,-2,2} = N$$

Knowing this, we can deduce some key facts.

Vector addition is commutative: it can be done in any order. Since pressing a tile is the same as adding a vector, the order you press tiles in doesn't matter.

Clicking a tile twice is the same as not clicking it at all. This is quite easy to discover on your own.



This happens because we're using integers modulo 2. An even number of presses is congruent to 0 (not pressing it at all). An odd number of presses is congruent to 1 (only pressing once). If you want to reduce moves-played in a game, you will press each tile a maximum of one time.

All-ones problem

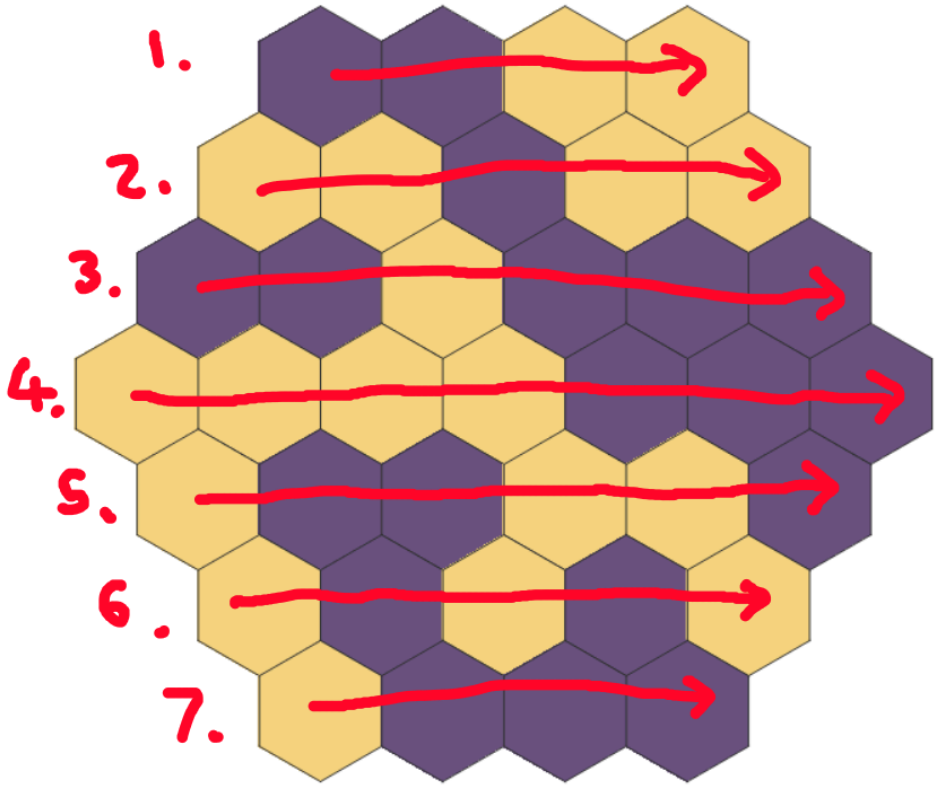
The all-ones problem is a classic when it comes to Lights Out variants. It asks the question: if you start with a configuration of all 1s, can you get to a configuration of all 0s?

We actually already know this. Remember: for any given starting configuration, we can win with light *or* shadow. We can see the starting board as a kind of intermediate of the all ones problem, half-way solved. From that intermediate, the tile presses to win with light and the tile presses to win with shadow could be combined, producing the tile presses to go from light to shadow.

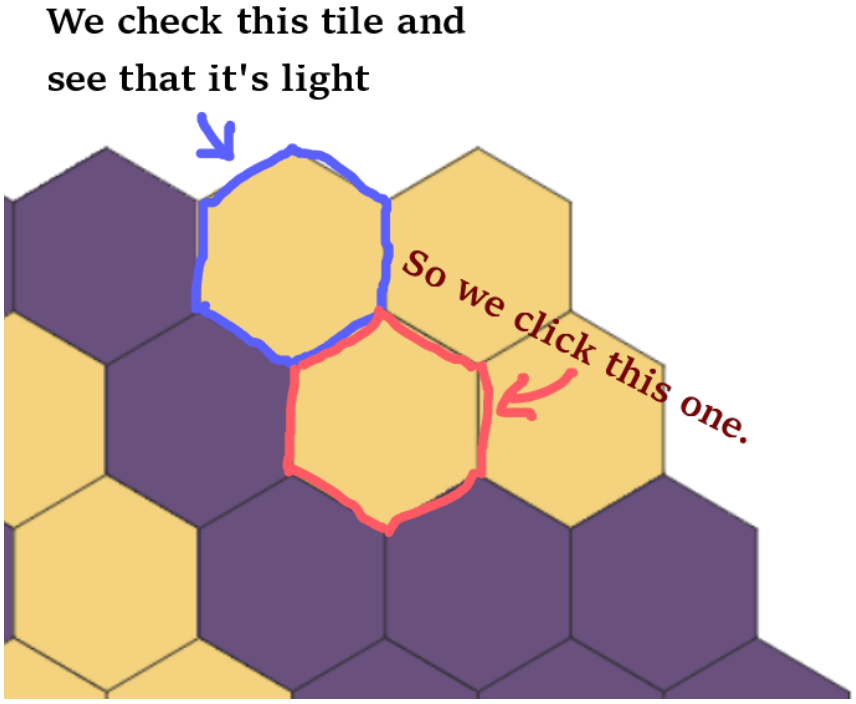
How to actually win

Stage 1

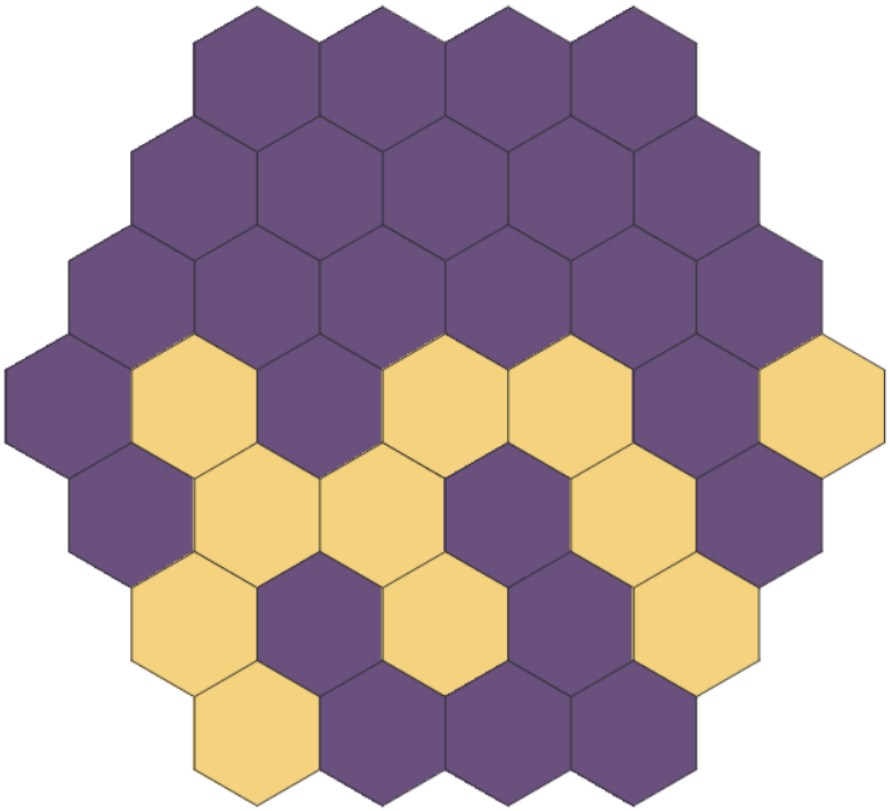
Our strategy begins by looking at the top row, at the left-most tile. We're checking if it's a light tile. If not, then we move to the neighbour on its right. We work left-to-right and at the end of the row, we jump to the row below, starting at the left-most tile again.



If the tile is light, then we click the tile at its bottom right (i.e. the neighbouring tile in the +r, -s direction). Do not worry about the state of the bottom right tile, it doesn't matter.

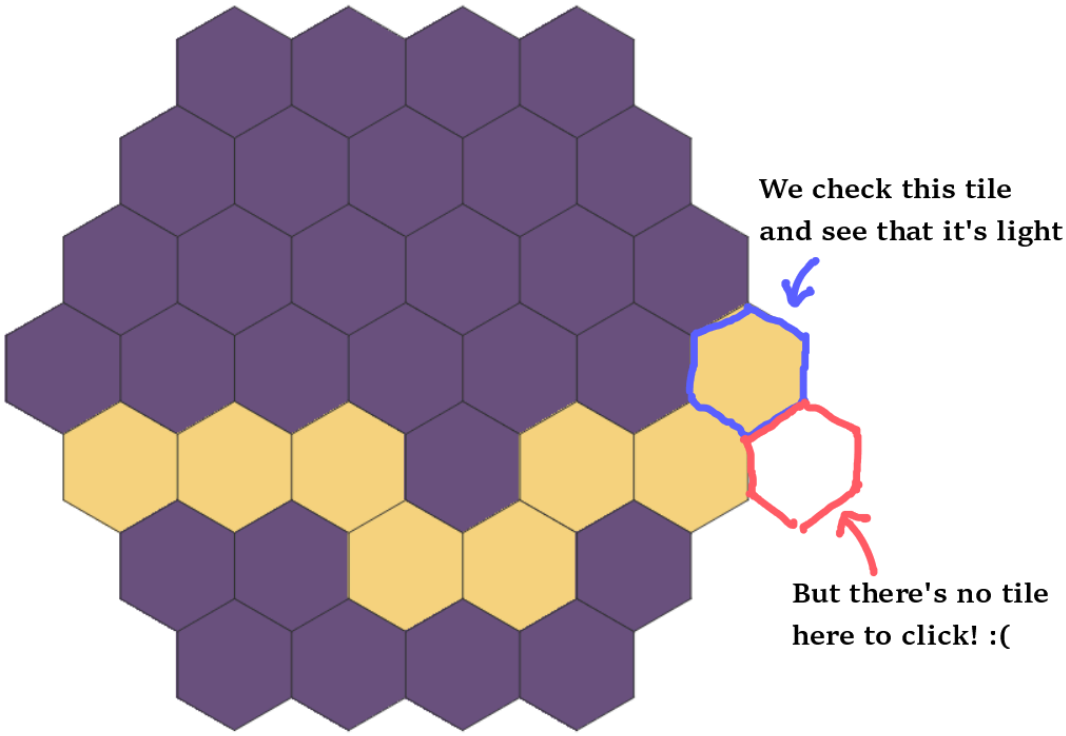


Continue on from where the tile we checked was. After a few rows, you'll notice we're gradually clearing the top of the board.

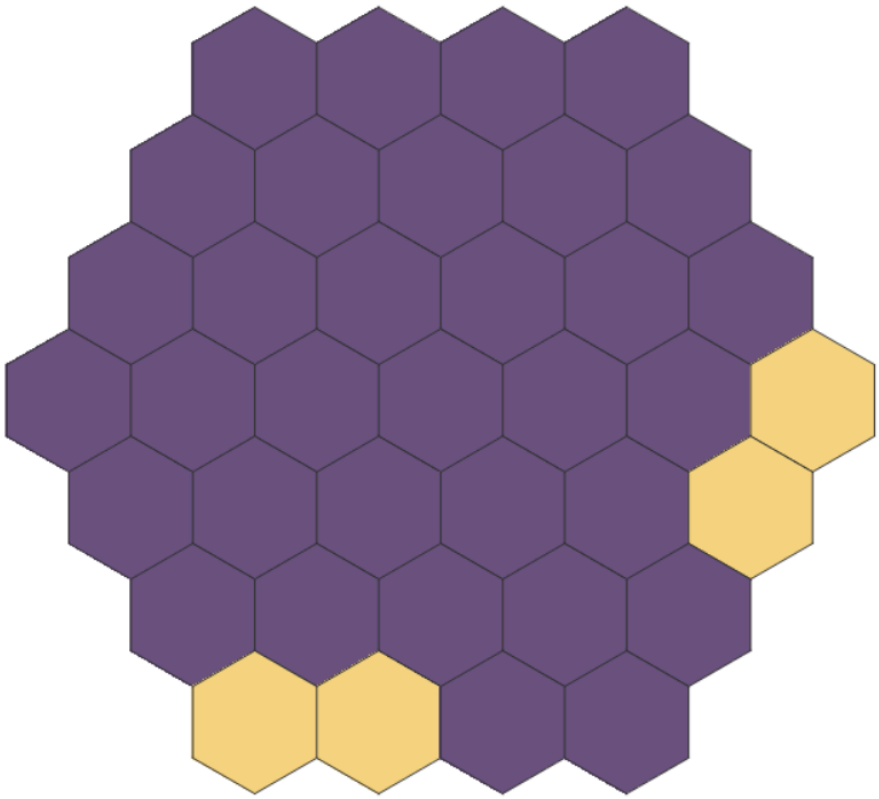


We continue going and the board completely solves. Yay!

But this doesn't always happen. For some initial configurations, we encounter a light tile on the lower-right edge (s=-3), which doesn't have a bottom right tile available.



When this happens, ignore it. Pretend the tile isn’t light and continue on until you reach the bottom row. You’ll end up with something like this.

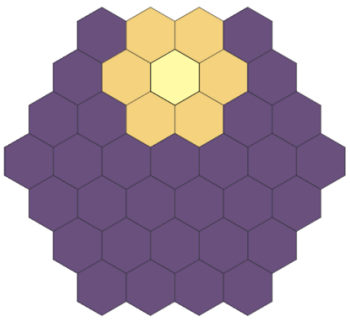


How are we supposed to remove these bits stuck on the edge?

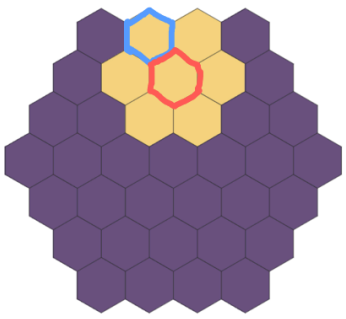
Stage 2

Let's take a closer look at what we were actually doing. If we take a centre tile (i.e. tile where none of q,r,s equals 3 or -3) on an otherwise blank board, press it, then apply our strategy, we'll press the same tile again, undoing the action.

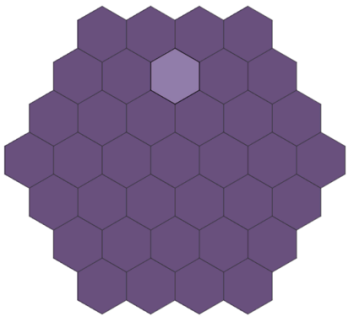
Click (1,-2,1)



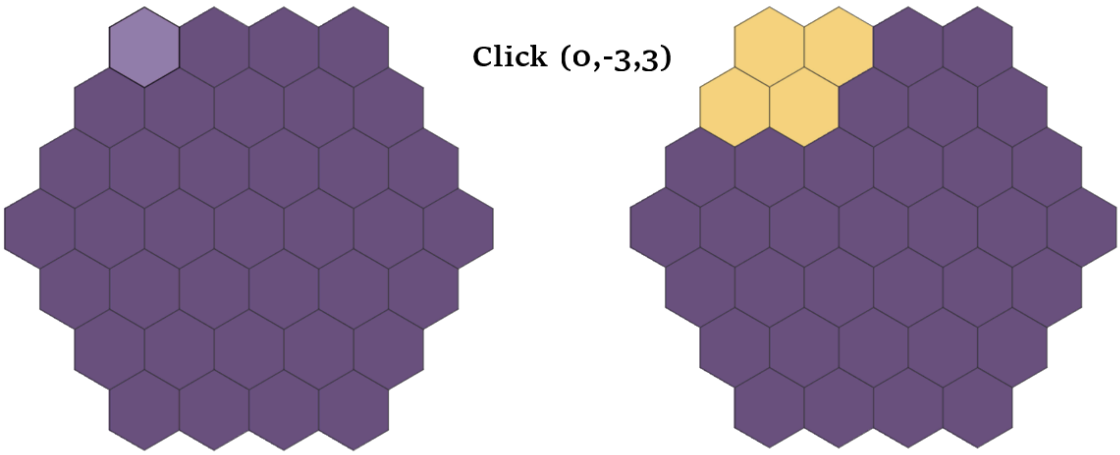
Apply our method

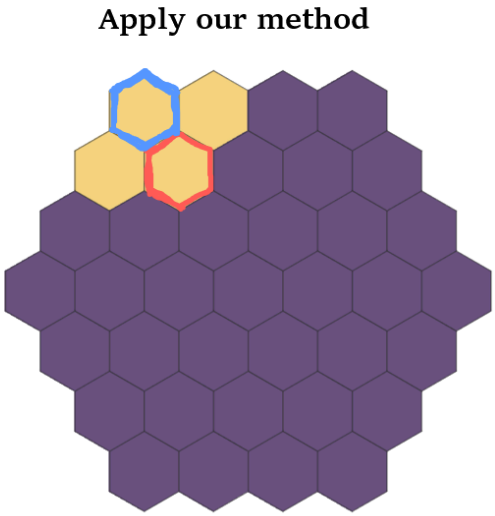


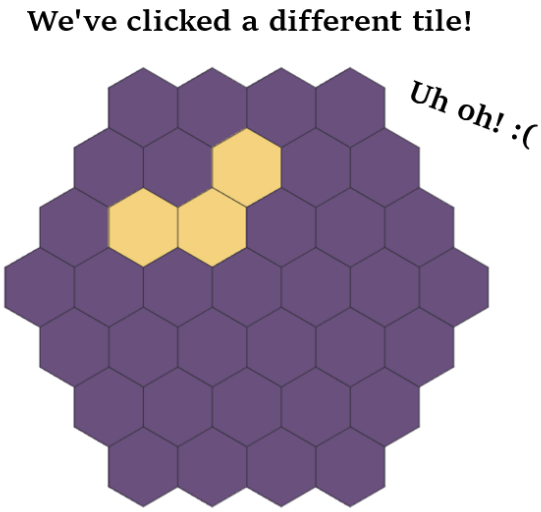
Click (1,-2,1) again
We've clicked in the same spot twice - "undoing" the action



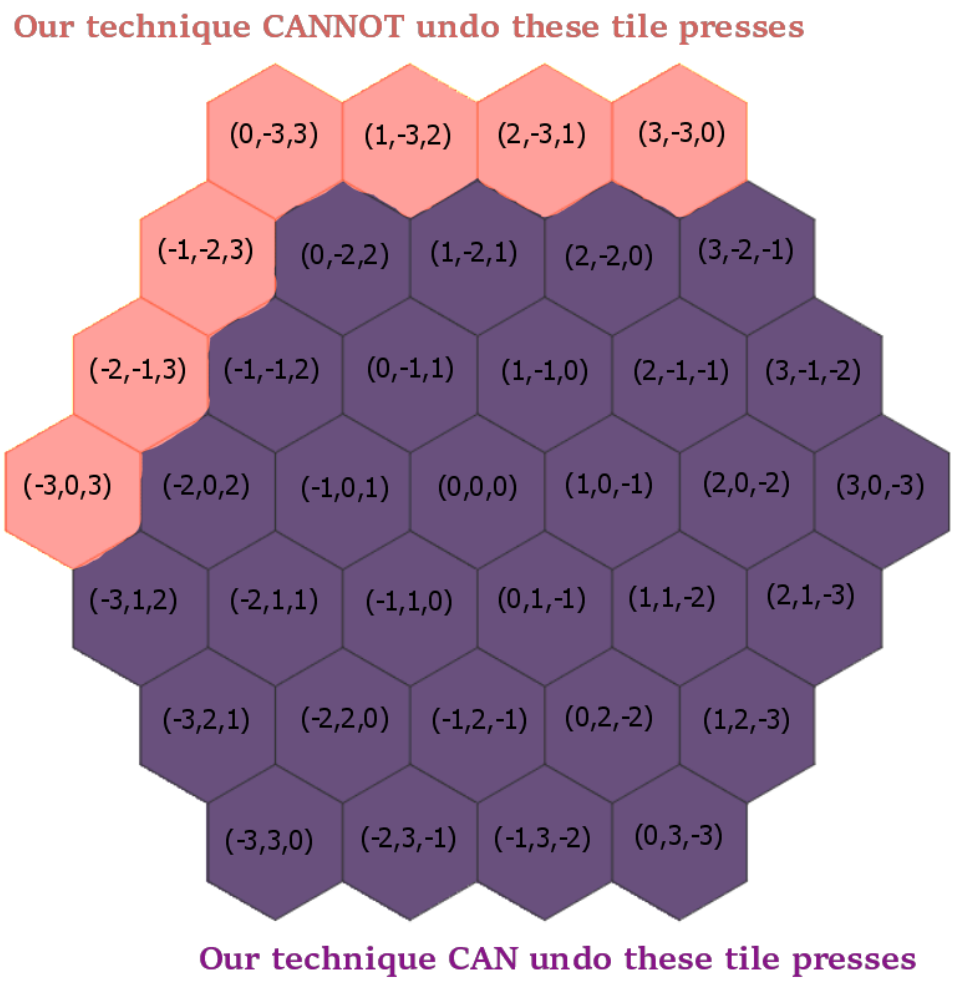
In fact, this is true for most tiles (testing of which is left up to the reader). The only time this isn't true is for tiles on the top edge and upper-left edge (i.e. tiles with r=-3 OR s=3). An example is shown below.



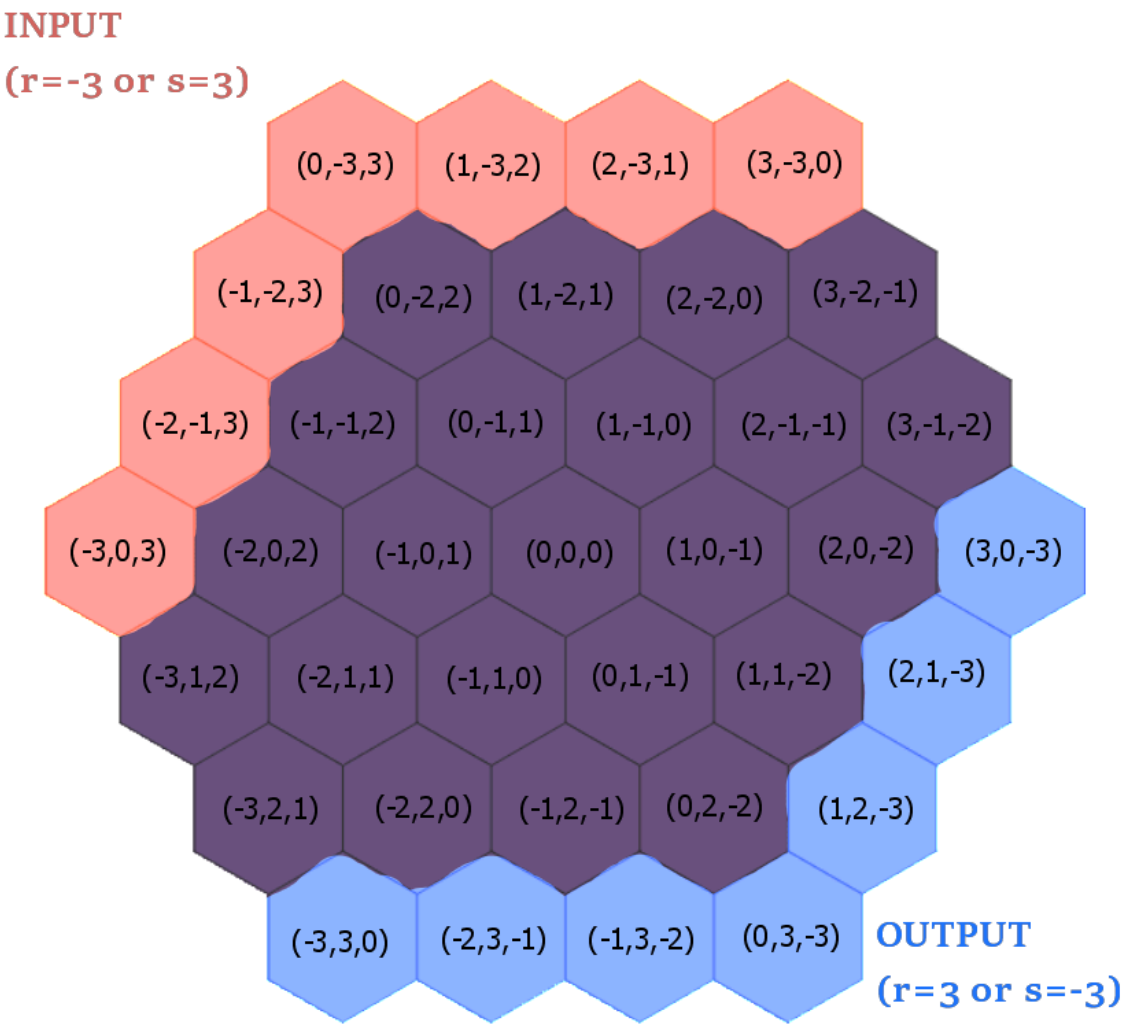




Remember how G&G starting boards are generated: we take a monotone board, press a random assortment of tiles and then give that to the player. We, the player, are "undoing" all the tile presses except for the upper-left edge and top edge.



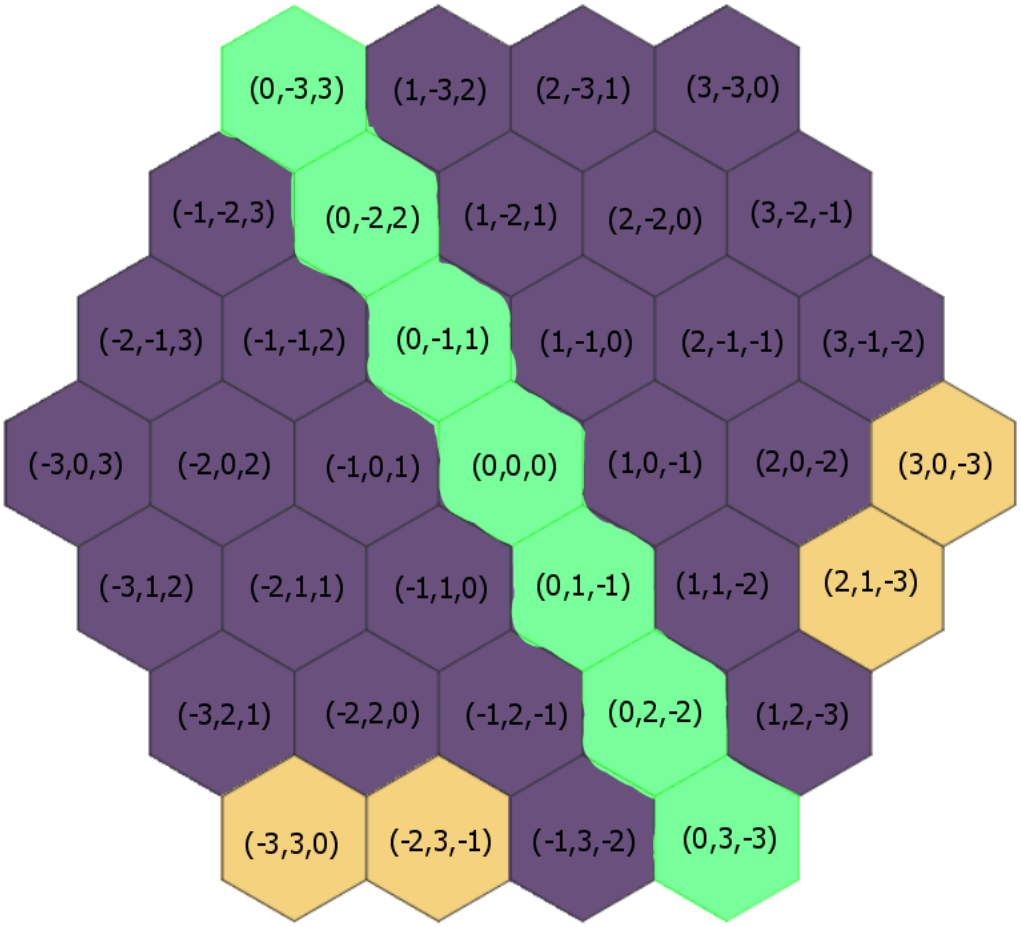
What this all means is that the pattern of the left-over light tiles (which we'll call the output) has been determined solely by the tiles which were pressed in the upper-left edge and top edge (which we'll call the input) before we started the game.



Now, all we need to do is figure out which input tiles have been pressed based on the pattern of the output. Once we know this, we can press them, repeat stage 1 and win the game.

Did you notice our output from earlier had reflectional symmetry through q=0? In fact, for every possible input, the output is symmetrical.[2] You can check this manually.

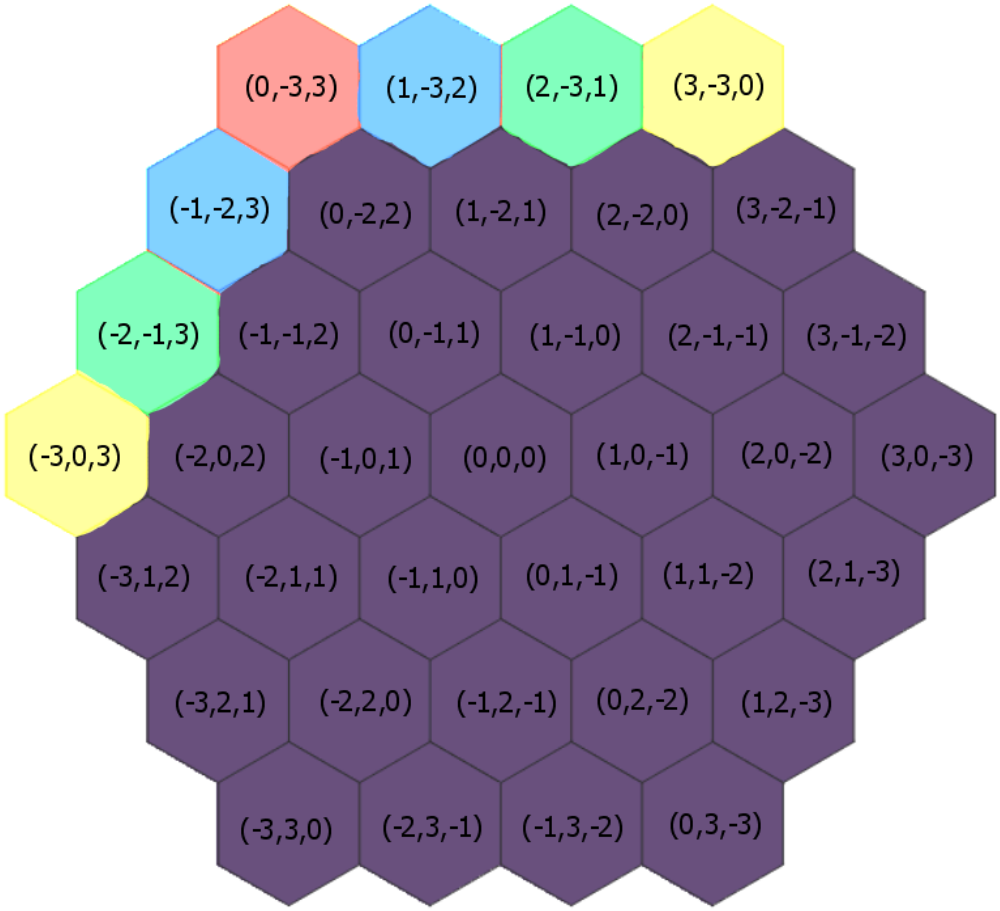
[2] Here's where we get our very easy proof that not every G&G hard mode board is solvable: if you have an asymmetric output, it's an unsolvable board.



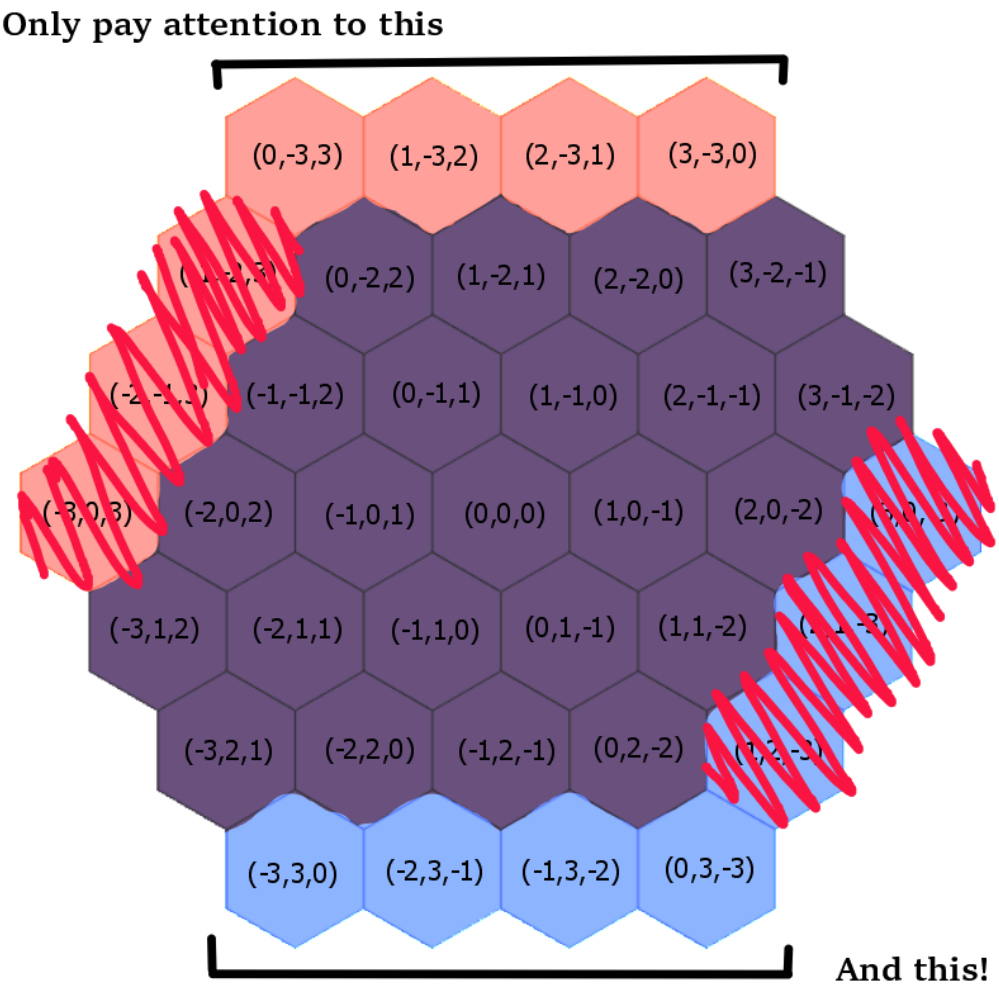
With some testing, you will also discover that this symmetry applies to the inputs themselves. For example, pressing (1,-3,2) produces the same output as pressing (-1,-2,3).[3]

[3] Unfortunately, I don't know why this board is so symmetrical. I proved the symmetry by exhaustion (you're welcome to check). Other sized hexagonal boards (eg 2 rings, 4 rings) don't have this property.

Same colour input tiles will produce the same output

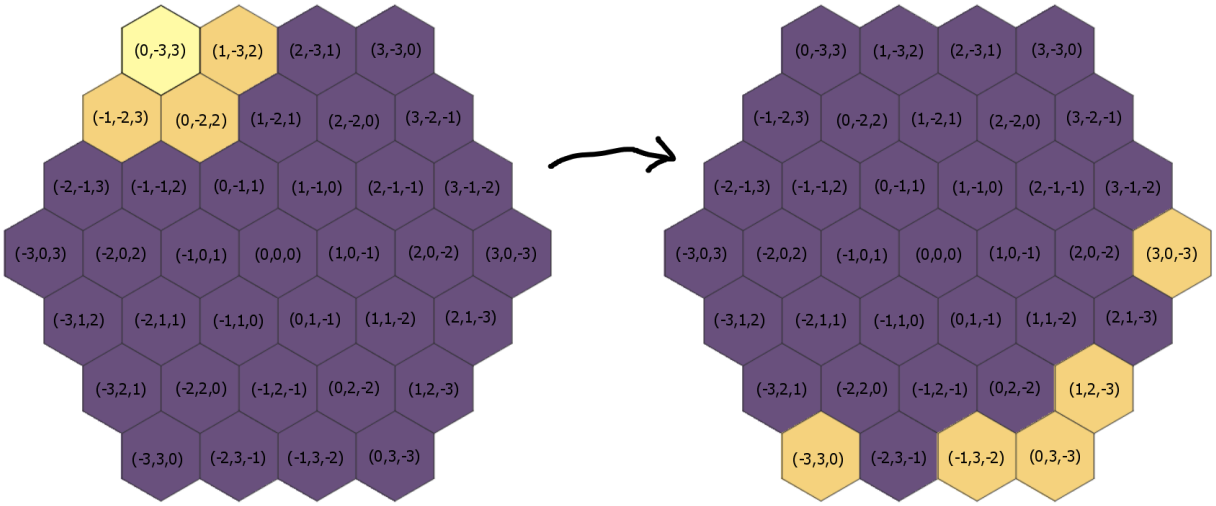


Reflected input tiles produce the same output, so they will cancel each other out, similar to pressing a tile twice. Thus, we only need to worry about pressing tiles on the top row (r=-3). Since the bottom row is symmetrical, we only need half of it too.



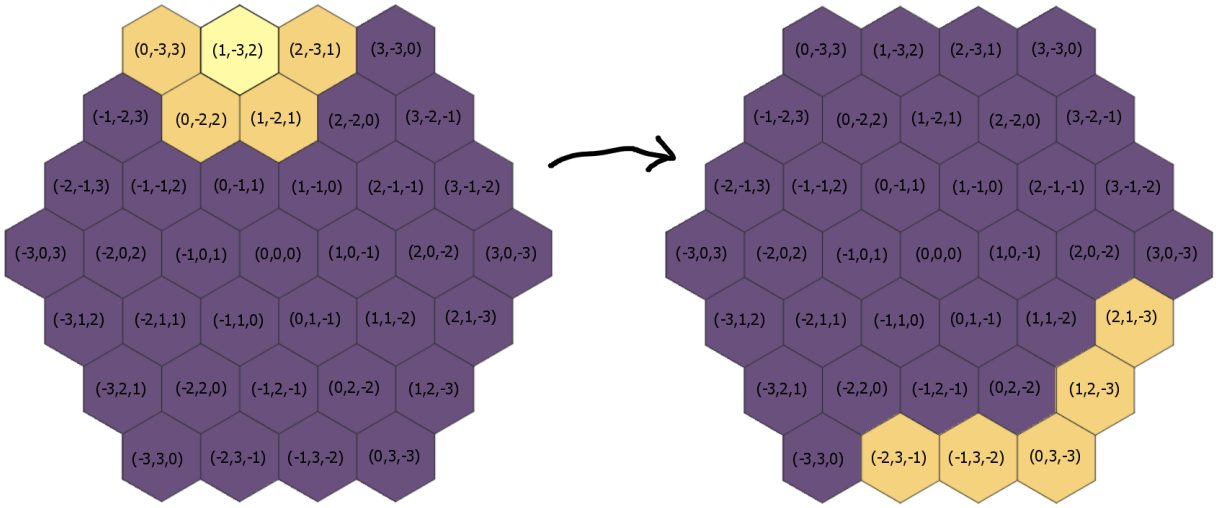
So, we only have 4 tiles to choose from. But 4 tiles (that can be pressed or not pressed) is still 2^4 possibilities. What's the right one?

Let's test what each input tile press does in isolation.



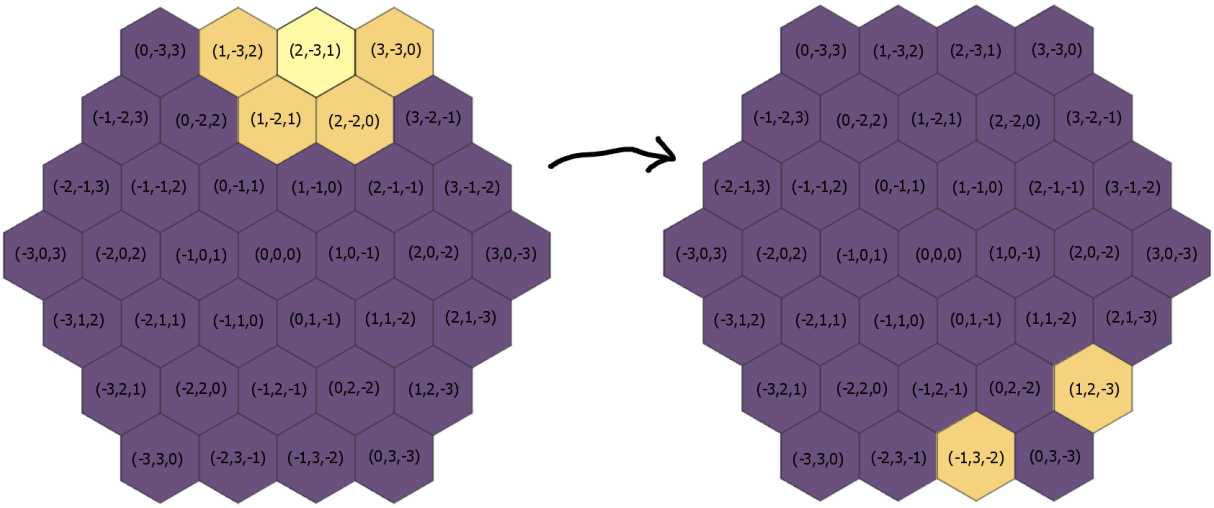
Input: press (0,-3,3)

Output: (-3,3,0), (-1,3,-2), (0,3,-3)



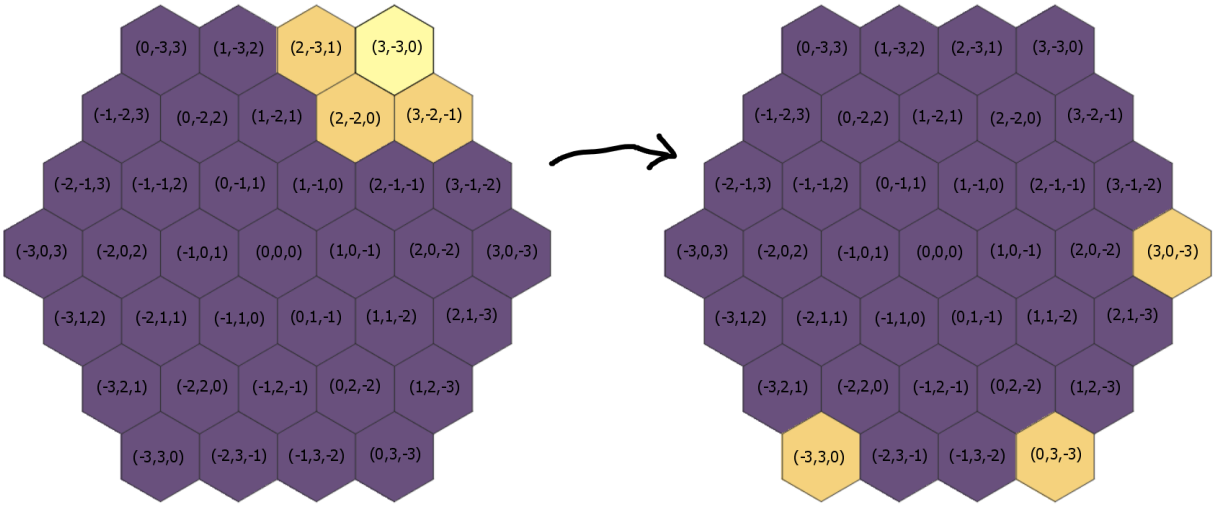
Input: press (1,-3,2)

Output: (-2,3,1), (-1,3,-2), (0,3,-3)



Input: press (2,-3,1)

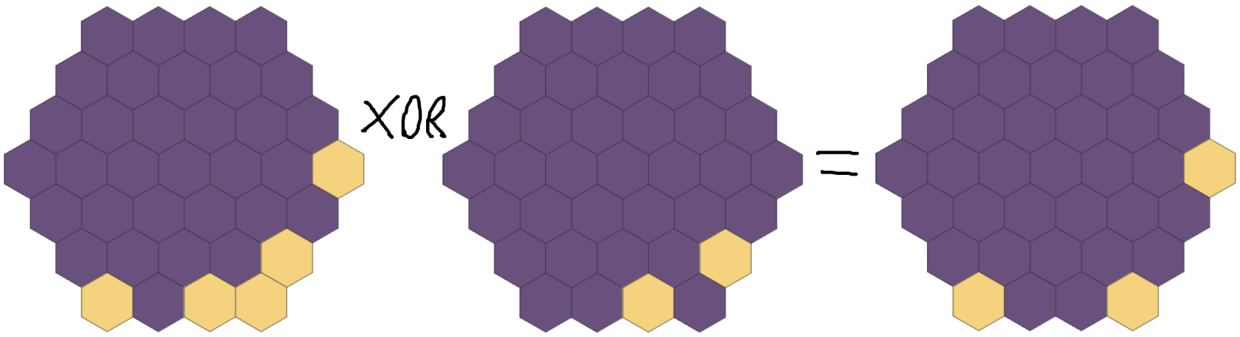
Output: (-1,3,-2)



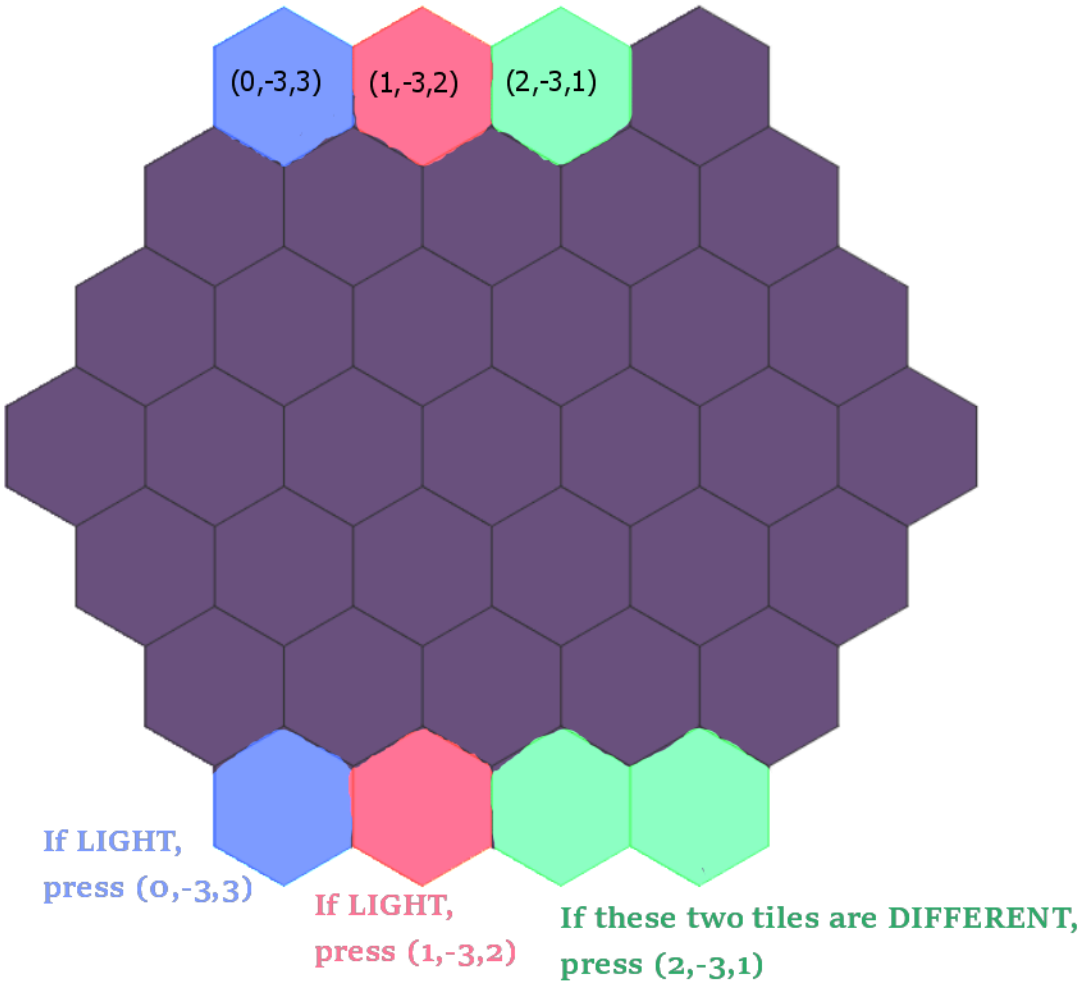
Input: press (3,-3,0)

Output: (-3,3,0), (0,3,-3)

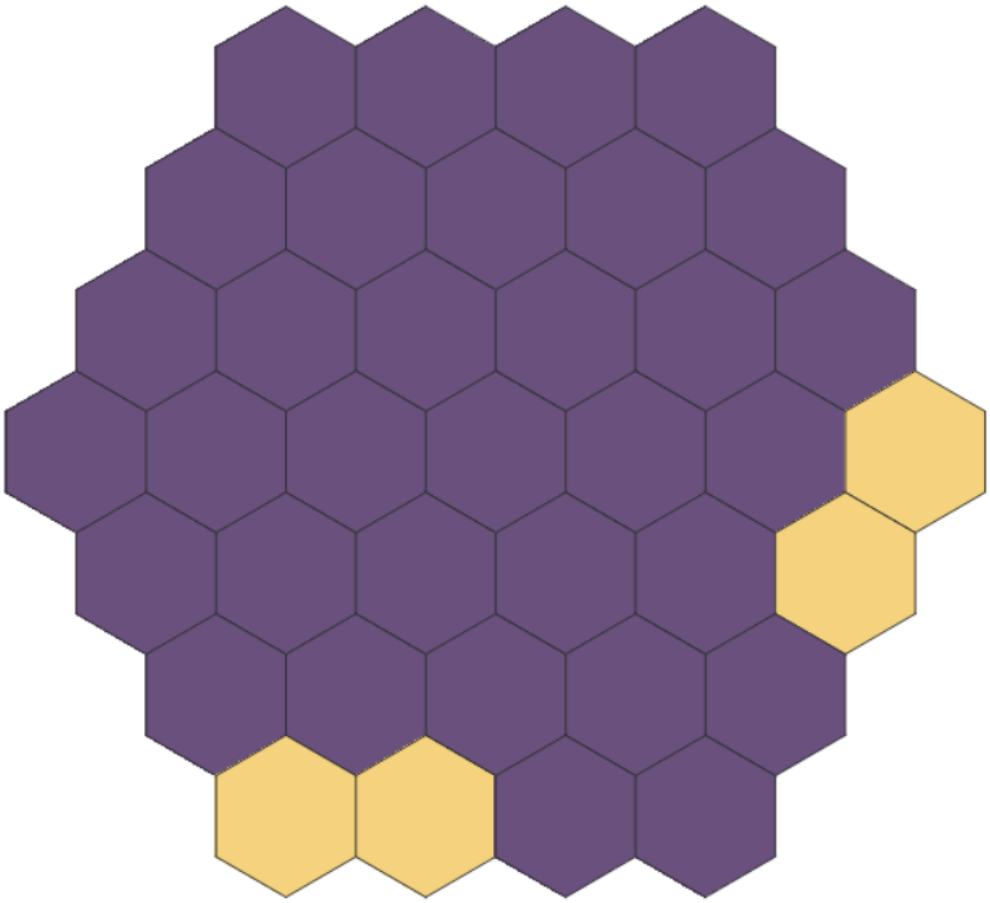
To consider the effect of multiple tile presses, we can just XOR the outputs together. For example, it turns out pressing inputs (0,-3,3) and (2,-3,1) results in the same output as pressing only (3,-3,0).



Once you know what the different outputs are and how they interact with each other, it's easy to deduce what the input must be. This is how I remember:



So, going back to our example from earlier, for this board, we'd press (0,-3,3) and (1,-3,2).



Stage 3

We repeat the steps we used in stage 1. The board should solve completely and we win the game!

References

- Flight Rising's Glimmer & Gloom
- Flight Rising wiki Glimmer & Gloom
- G&G hard mode guide
- Hexagonal grid notation
- Lights Out Wikipedia
- Lights Out maths
- Turning Lights out with Linear Algebra (DOI: 10.1080/0025570X.1998.11996658)

- Hexagonal tile game code