Flow free as a CSP:

# Possibility 1:

Variables: each color red, blue, yellow, ect.

Domain (each Variable): Any integer grid coordinate (x,y) in the input maze

Constraints:

* Cannot cover existing color
* Must be connected/continuous (can only add color in adjacent square)
* Max 2 adjacent squares colored by same color
* (somehow check board full)

Concern: how to check if board is full under conditions when treating colors as variables

Logically this seems to make more sence

# Possibility 2

Variables: each integer grid coordinate (x,y) in the input maze

Domain (each Variable): each possible color: red, blue, yellow, ect.

Constraints:

* Cannot cover existing color (cannot have more than one or replace)
* Must be connected/continuous (red touching red on 1-2 sides exactly)
* Each space filled

Concern: how to ensure that colors stay connected when treating spaces as variables

This is more like how map coloring was formed

Smart method maybe:

Create hash tables of paths for each color. Path is the “variable” ex: red can have two paths. Now for path one of read check if path one of blue “fits” satisfies the conditions if not check red path 1 - blue path 2. Like a combination lock trying all conditions until the puzzle fits, the lock opens, the flows are free. When a conflict is found make decision on how to iterate instead of going R1,B1,Y1 to R1,B1,Y2 if r1 and b1 conflict jump to r1,b2