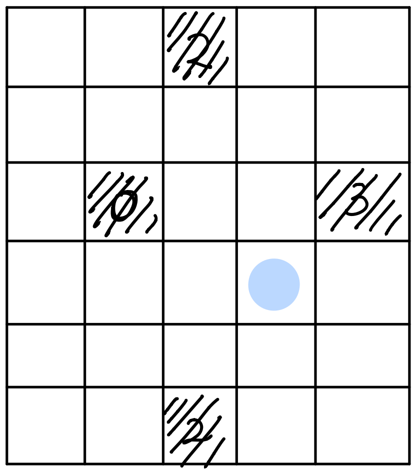
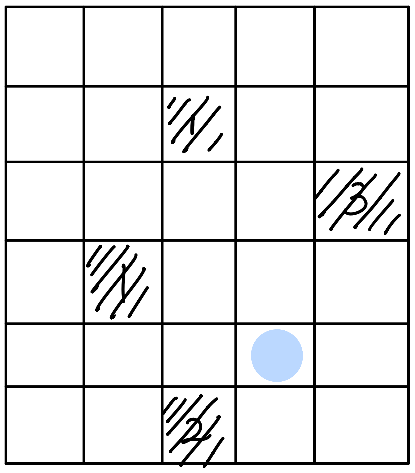
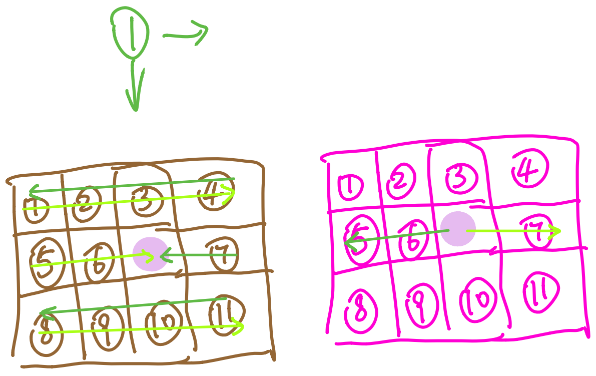




● wormholes

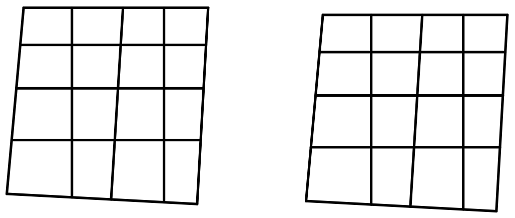


Bulb / None / wormhole / ▨

Puzzle Creating Logic:

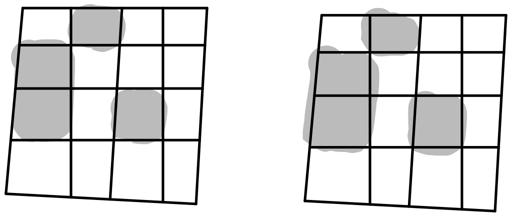
① Initialization:

given size (eg: 4x4) → 2 empty grids.



② Add Walls (Black Square)

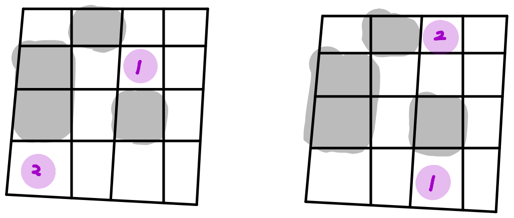
given percentage of black squares (eg: 25%)



③ Add wormholes in empty cells

is selected by user

1/2 ?



④ Add light bulbs

add in ① untill no ① is { empty / not lit by othe bulbs.

* priority : { ① empty cells adjacent to walls → <easy to use # in black squares to restrict bulbs' location> ↓ higher probability to be unique solution

② Other cells.

According to ↑, randomly add light

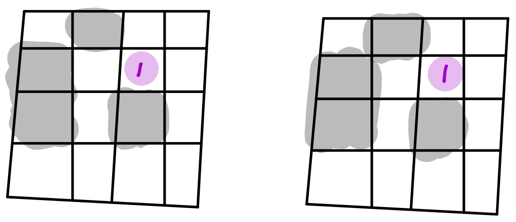
(empty cell = no { wall
light bulb
worm hole)

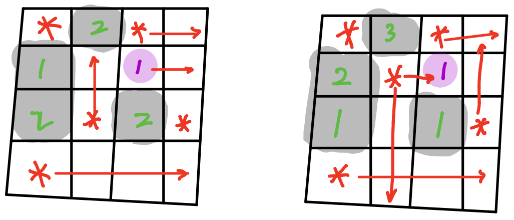
⑤ Add numbers on walls:

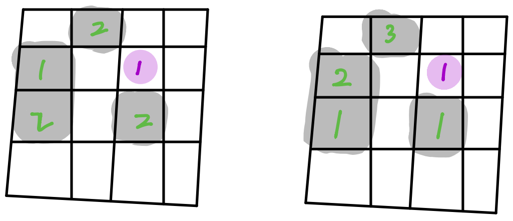
based on Difficulty {
- easy : Record # on all walls
- hard : Remain some walls without #

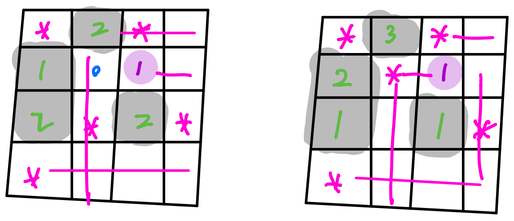
⑥ Verify whether it is unique solution

if
- yes : Print puzzle & solution
- no : redo ② ③ ④ ⑤



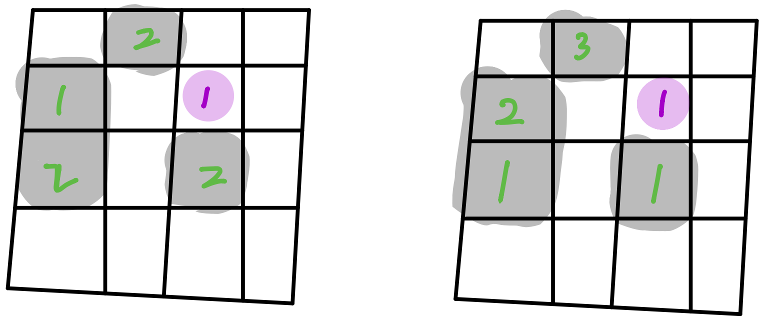






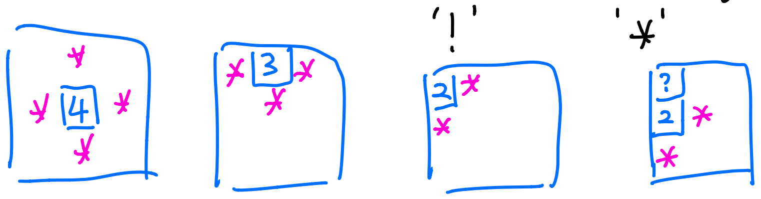
Unique solution example

Solve Logic:

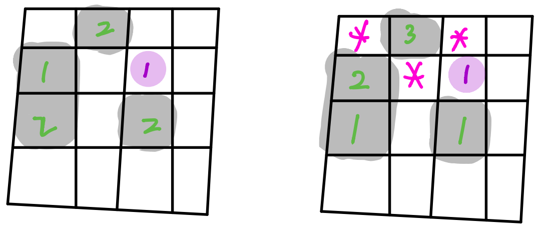


① if exist 0 / Full add No bulb sign / bulb sign
'!' '*'

* Full:



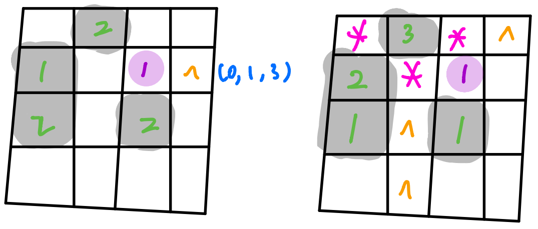
(# empty cell around = number on the wall)



Sometime: ~~Can't~~ use this logic for whole solving process
∵ in hard mode
Just deduction by numbers & not lit by other is not enough
Sometimes we need to guess and try.

② once add *,

Cal the light routes
lit_cells_list.append(~)
Update grid:
use '^': empty but lit by other bulbs



lit_cells_list =

[(0,1,3), (1,0,3), (1,2,1), (1,3,1)]

③ do ①② until :

① no empty cells on the grid → stop (only solution)

② still have empty cells

but can't deduct (100% confident) any more

↓

dfs.

↙ in dfs :

more than one solution

↓

X only solution

↓

re-create

↘ stop

(only solution)

④.