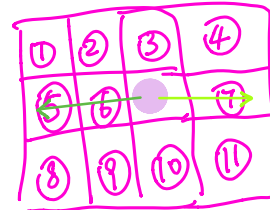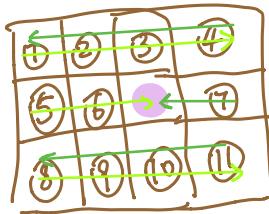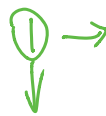Wormholes
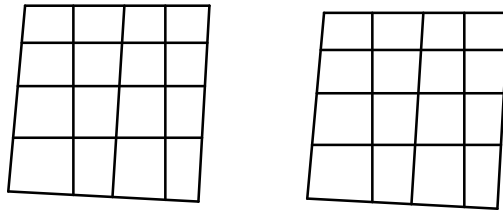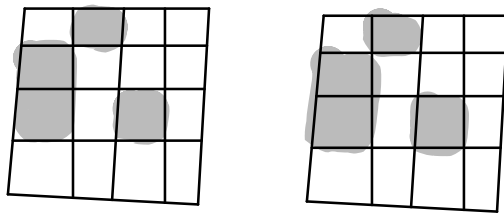
Bulb / None / Wormhole / ▨

Puzzle Creating Logic:

① Initialization:

given size (eg: 4x4) → 2 empty grids.

② Add Walls (Black Square)
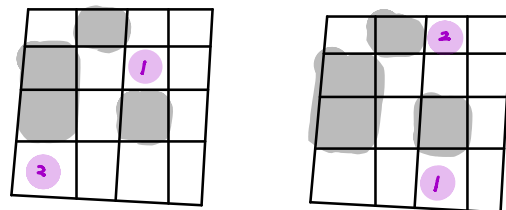given percentage of black squares (eg: 25%)

③ Add wormholes in empty cells
is selected by user
1/2 ?

④ Add light bulbs
  # priority: { ① empty cells adjacent to walls
              { ② other cells.

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

<easy to use # in black squares
to restrict bulbs' location>

↓

higher probability to be
unique solution

According to ↙, randomly add light

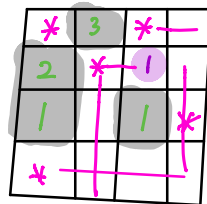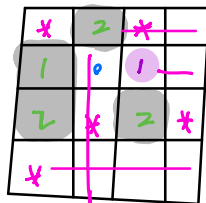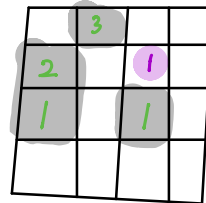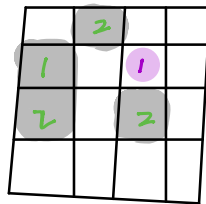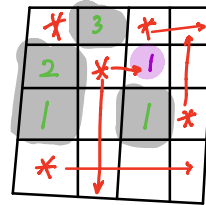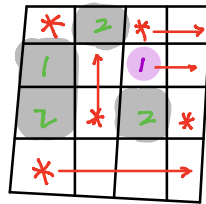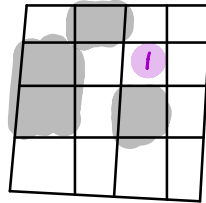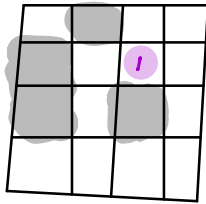(empty cell = no { wall
                    light bulb)
                    worm hole

⑤ Add numbers on walls:
    based on Difficulty { easy : Record # on all walls
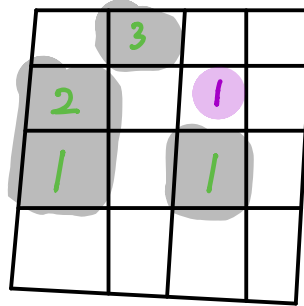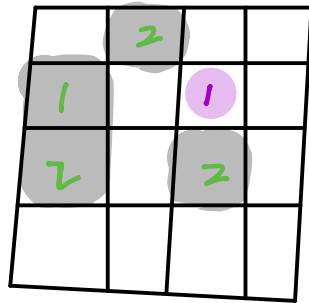   hard : Remain some walls without #

⑥ Verify whether it is unique solution
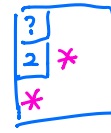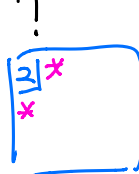   if { yes : Print puzzle & solution
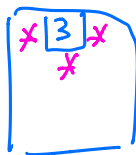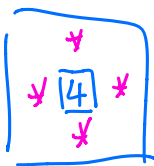   no : redo ②③④⑤

Unique solution example

Solve Logic:

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

↓ Full:

no { wall
    light bulb
    worm hole
  &
  not lit by others

(# empty cell around
  = number on the wall)
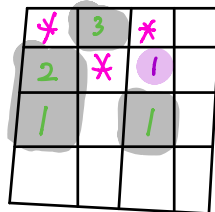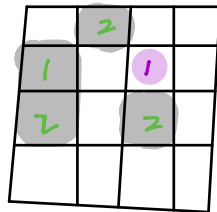
② once add *,

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

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.

lit_cells_list =

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

(grid diagrams with values, and label (0,1,3))

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.

④.

    in dfs :   stop

    more than one solution   (only solution)

         ↓

    X only solution

         ↓

    re-create