**Justin\_Array\_0289.**  **Game of Life**

**Concept:**

先把所有 neighbors 都列出來 (點的8個鄰居)

把鄰居都搜一遍，符合 (r < rows and r >= 0) and (c < cols and c >= 0) and (copy\_board[r][c] == 1) 則 把 live\_neighbors + 1

r < rows、c < cols ==> 角點跟邊點沒有到 8 個鄰居，角點(3個)，邊點(5個)

最後看 live\_neighbors 有沒有 = 3

**Code:**

class Solution:

def gameOfLife(self, board: List[List[int]]) -> None:

"""

Do not return anything, modify board in-place instead.

"""

# Neighbors array to find 8 neighboring cells for a given cell

neighbors = [(1,0), (1,-1), (0,-1), (-1,-1), (-1,0), (-1,1), (0,1), (1,1)]

rows = len(board)

cols = len(board[0])

# Create a copy of the original board

copy\_board = [[board[row][col] for col in range(cols)] for row in range(rows)]

# Iterate through board cell by cell.

for row in range(rows):

for col in range(cols):

# For each cell count the number of live neighbors.

live\_neighbors = 0

for i in neighbors:

r = (row + i[0])

c = (col + i[1])

# Check the validity of the neighboring cell and if it was originally a live cell.

# The evaluation is done against the copy, since that is never updated.

if (r < rows and r >= 0) and (c < cols and c >= 0) and (copy\_board[r][c] == 1):

live\_neighbors += 1

# Rule 1 or Rule 3

if copy\_board[row][col] == 1 and (live\_neighbors < 2 or live\_neighbors > 3):

board[row][col] = 0

# Rule 4

if copy\_board[row][col] == 0 and live\_neighbors == 3:

board[row][col] = 1