289. Game of Life

Medium 6.1K 517

Companies

According to Wikipedia's article: "The **Game of Life**, also known simply as **Life**, is a cellular automaton devised by the British mathematician John Horton Conway in 1970."

The board is made up of an `m x n` grid of cells, where each cell has an initial state: **live** (represented by a `1`) or **dead** (represented by a `0`). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules (taken from the above Wikipedia article):

1. Any live cell with fewer than two live neighbors dies as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by over-population.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The next state is created by applying the above rules simultaneously to every cell in the

Code

```
class Solution {
    func gameOfLife(_ board: inout [[Int]]) {
        let b = board
        let m = board.count-1
        var n = board[0].count - 1
        for i in (0..<board.count) {
            for j in (0..<board[0].count) {
                var count = 0
                if(b[i][j] == 1) {
                    count = count + checkCell(b,i-1,j,m,n)
                    count = count + checkCell(b,i+1,j,m,n)
                    count = count + checkCell(b,i,j-1,m,n)
                    count = count + checkCell(b,i,j+1,m,n)
```

```
                    count = count + checkCell(b,i-1,j-1,m,n)
                    count = count + checkCell(b,i-1,j+1,m,n)
                    count = count + checkCell(b,i+1,j+1,m,n)
                    count = count + checkCell(b,i+1,j-1,m,n)

                    if count < 2 || count > 3{
                        board[i][j] = 0
                    }
                } else {
                     count = count + checkCell(b,i-1,j,m,n)
                    count = count + checkCell(b,i+1,j,m,n)
                    count = count + checkCell(b,i,j-1,m,n)
                    count = count + checkCell(b,i,j+1,m,n)
                    count = count + checkCell(b,i-1,j-1,m,n)
                    count = count + checkCell(b,i-1,j+1,m,n)
                    count = count + checkCell(b,i+1,j+1,m,n)
                    count = count + checkCell(b,i+1,j-1,m,n)

                    if count == 3 {
                        board[i][j] = 1
                    }
                }


            }
        }
    }


    func checkCell(_ board:[[Int]], _ i:Int, _ j:Int,_ m:Int,_
n:Int ) -> Int {
        if(i<0 || i>m || j<0 || j>n) {
            return 0
        }
        return board[i][j]
    }
}
```