Game of Life

According to the 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."

Given a *board* with *m* by *n* cells, each cell has an initial state *live* (1) or *dead* (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.

Write a function to compute the next state (after one update) of the board given its current state. The next state is created by applying the above rules simultaneously to every cell in the current state, where births and deaths occur simultaneously.

Example:

```
Input:
[
  [0,1,0],
  [0,0,1],
  [1,1,1],
  [0,0,0]
]
Output:
[
  [0,0,0],
  [1,0,1],
  [0,1,1],
  [0,1,0]
]
```

Follow up:

1. Could you solve it in-place? Remember that the board needs to be updated at the same time: You cannot update some cells first and then use their updated values to update other cells.
2. In this question, we represent the board using a 2D array. In principle, the board is infinite, which would cause problems when the active area encroaches the border of the array. How would you address these problems?

Solution:

```
class Solution {
    public void gameOfLife(int[][] board) {
        int rows = board.length;
        int cols = board[0].length;
        int [][] copyOfBoard = new int[rows][cols];
        for(int i = 0; i< rows; i++){
            for(int j = 0; j< cols; j++){
                copyOfBoard[i][j] = board[i][j];
            }
        }

        int[] a = {-1,-1,-1,0,0,1,1,1};
        int[] b = {-1,0,1,-1,1,-1,0,1};

        for(int i=0; i< rows; i++){
            for(int j = 0; j<cols; j++){
                int liveNoOfNeighbors = 0;

                for(int x = 0; x < 8; x ++){
                    if(i+a[x] >= 0 && i+a[x] < rows && j +b[x] >= 0 && j+b[x]< cols) {
                        liveNoOfNeighbors +=  copyOfBoard[i+a[x]][j +b[x]];
                    }
                }

                //Rules
                if(liveNoOfNeighbors <2 || liveNoOfNeighbors > 3){
                    board[i][j] = 0;
                }else if (liveNoOfNeighbors == 3) {
                    board[i][j] = 1;
                }
            }
        }


    }
```

```
}
```