Valid Tic-Tac-Toe State

A Tic-Tac-Toe board is given as a string array `board`. Return True if and only if it is possible to reach this board position during the course of a valid tic-tac-toe game.

The `board` is a 3 x 3 array, and consists of characters `" "`, `"X"`, and `"O"`. The " " character represents an empty square.

Here are the rules of Tic-Tac-Toe:

- Players take turns placing characters into empty squares (" ").
- The first player always places "X" characters, while the second player always places "O" characters.
- "X" and "O" characters are always placed into empty squares, never filled ones.
- The game ends when there are 3 of the same (non-empty) character filling any row, column, or diagonal.
- The game also ends if all squares are non-empty.
- No more moves can be played if the game is over.

```
Example 1:
Input: board = ["O  ", "   ", "   "]
Output: false
Explanation: The first player always plays "X".

Example 2:
Input: board = ["XOX", " X ", "   "]
Output: false
Explanation: Players take turns making moves.

Example 3:
Input: board = ["XXX", "   ", "OOO"]
Output: false

Example 4:
Input: board = ["XOX", "O O", "XOX"]
Output: true
```

Note:

- `board` is a length-3 array of strings, where each string `board[i]` has length 3.
- Each `board[i][j]` is a character in the set `{" ", "X", "O"}`.

Solution 1

To find the validity of a given board, we could first think about the cases where the board is invalid

1. Since X starts first, x_count >= o_count. So if o_count > x_count, we can return False
2. Since the players take turns, we could also return False if x_count-o_count>1

After the corner cases, this is the algorithm used:

1. If player O has a winning condition, also check the following:
   a) If player X also has a winning condition, return False
   b) If x_count != o_count , return False (Since player O always plays second, it has to meet this condition always)
2. If player X has a winning condition, check the following:
   a) If x_count != o_count + 1, return False (Since player X plays the first move, if player X wins, the player X's count would be 1 more than player O)

```
class Solution(object):
    def check_win_positions(self, board, player):
 """
 Check if the given player has a win position.
 Return True if there is a win position. Else return False.
 """
        #Check the rows
        for i in range(len(board)):
            if board[i][0] == board[i][1] == board[i][2] == player:
                return True

        #Check the columns
        for i in range(len(board)):
            if board[0][i] == board[1][i] == board[2][i] == player:
                return True

        #Check the diagonals
        if board[0][0] == board[1][1] == board[2][2]  == player or \
               board[0][2] == board[1][1] == board[2][0] == player:
            return True

        return False

    def validTicTacToe(self, board):
        """
        :type board: List[str]
        :rtype: bool
        """

        x_count, o_count = 0, 0
        for i in range(len(board)):
            for j in range(len(board[0])):
                if board[i][j] == "X":
                    x_count += 1
                elif  board[i][j] == "O":
                    o_count += 1

        if o_count > x_count or x_count-o_count>1:
            return False

        if self.check_win_positions(board, 'O'):
            if self.check_win_positions(board, 'X'):
                return False
            return o_count == x_count

        if self.check_win_positions(board, 'X') and x_count!=o_count+1:
            return False

        return True
```

written by shriram2112 original link here

Solution 2

```
bool validTicTacToe(char** board, int n) {
    int x_row_count[3] = {0}, x_col_count[3] = {0};
    int o_row_count[3] = {0}, o_col_count[3] = {0};
    int diag1_x = 0, diag1_o = 0, diag2_x = 0, diag2_o = 0;
    int i, j, x = 0, o = 0;
    bool x_wins = false, o_wins = false;

    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            if(board[i][j] == 'X'){
                x_row_count[i]++;
                x_col_count[j]++;
                if(i == j) diag1_x++;
                if(i+j == 2) diag2_x++;
                x++;
            } else if (board[i][j] == 'O'){
                o_row_count[i]++;
                o_col_count[j]++;
                if(i == j) diag1_o++;
                if(i+j == 2) diag2_o++;
                o++;
            }

            if((x_row_count[i] == 3) || (x_col_count[j] == 3) ||
               (diag1_x == 3) || (diag2_x == 3)) x_wins = true;

            if((o_row_count[i] == 3) || (o_col_count[j] == 3) ||
               (diag1_o == 3) || (diag2_o == 3)) o_wins = true;
        }
    }

    // x should be equal to o or 1 greater than o as x always starts the game
    if(!((x == o) || (x == o+1))) return false;

    //if x wins, count of x should be greater than o
    if(x_wins && (x <= o)) return false;

    //if o wins, count of x should be equal to o
    if(o_wins && (x!= o)) return false;

    return true;
}
```

written by brickell original link here

Solution 3

When `turns` is 1, X moved. When `turns` is 0, O moved.
`rows` stores the number of X or O in each row. `cols` stores the number of X or O in each column. `diag` stores the number of X or O in diagonal. `antidiag` stores the number of X or O in antidiagonal. When any of the value gets to 3, it means X wins. When any of the value gets to -3, it means O wins.
When X wins, O cannot move anymore, so `turns` must be 1. When O wins, X cannot move anymore, so `turns` must be 0. Finally, when we return, `turns` must be either 0 or 1, and X and O cannot win at same time.

```java
class Solution {
    public boolean validTicTacToe(String[] board) {
        int turns = 0;
        boolean xwin = false;
        boolean owin = false;
        int[] rows = new int[3];
        int[] cols = new int[3];
        int diag = 0;
        int antidiag = 0;

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i].charAt(j) == 'X') {
                    turns++;
                    rows[i]++;
                    cols[j]++;
                    if (i == j) {
                        diag++;
                    }
                    if (i + j == 2) {
                        antidiag++;
                    }
                } else if (board[i].charAt(j) == 'O') {
                    turns--;
                    rows[i]--;
                    cols[j]--;
                    if (i == j) {
                        diag--;
                    }
                    if (i + j == 2) {
                        antidiag--;
                    }
                }
            }
        }

        xwin = rows[0] == 3 || rows[1] == 3 || rows[2] == 3 ||
               cols[0] == 3 || cols[1] == 3 || cols[2] == 3 ||
               diag == 3 || antidiag == 3;
        owin = rows[0] == -3 || rows[1] == -3 || rows[2] == -3 ||
               cols[0] == -3 || cols[1] == -3 || cols[2] == -3 ||
               diag == -3 || antidiag == -3;

        if (xwin && turns == 0 || owin && turns == 1) {
            return false;
        }
        return (turns == 0 || turns == 1) && (!xwin || !owin);
    }
}
```

written by myCafeBabe original link here