Java: How to check diagonal Connect Four win in 2D array

Ask Question

I have a Connect Four "board" which is a 6*7 2D char array populated with either spaces, X or O. The win condition is met when there are either four Xs or four Os in a row vertically, horizontally or diagonally. I've managed to get the win conditions checked successfully for vertical and horizontal, where the winning char is returned by some methods as below:

```
private char CheckVerticalWinner(char[][] currentBoard) {
    // check vertical (move down one row, same column)
    char vWinner = ' ';
    for (int col=0; col<7; col++) {
        int vCount = 0;
        for (int row=0; row<5; row++) {
            if (currentBoard[row][col]!=' ' &&
                currentBoard[row][col] == currentBoard[row+1][col]) {
                vCount++;
                System.out.println("VERT "+vCount); //test
            } else {
                vCount = 1;
            }

            if (vCount>=4) {
                vWinner = currentBoard[row][col];
            }
        }
    }
    return vWinner;
}


private char CheckHorizontalWinner(char[][] currentBoard) {
    // check horizontal (move across one column, same row)
    char hWinner = ' ';
    for (int row=0; row<6; row++) {
        int hCount = 0;
        for (int col=0; col<6; col++) {
            if (currentBoard[row][col]!=' ' &&
                currentBoard[row][col] == currentBoard[row][col+1]) {
                hCount++;
                System.out.println("HORIZ "+hCount); //test
            } else {
                hCount = 1;
            }

            if (hCount>= 4) {
                hWinner = currentBoard[row][col];
            }
        }
    }
    return hWinner;
}
```

I'm just stuck on how to check for diagonal wins, without throwing an ArrayIndexOutOfBoundsException. I know I need to iterate through the 2D array twice, once for forward diagonals and once for backward diagonals **that are 4 squares long or more**, like in the below diagram:

Diagonals to be checked diagram

Basically, how would I fill in this method to return a winning char?

```
private char CheckDiagonalWinner(char[][] currentBoard) {

    // some iteration here

    return dWinner;
}
```

Any help would be much appreciated!

java arrays multidimensional-array

edited Aug 21 '16 at 16:50

asked Aug 21 '16 at 8:02

P. Tan
18 2 7

Isn't it a simple matter of preventing yourself from exceeding the range of the board? For an N x M board, you only need to consider the sub-matrix of `(0,0)` to `(N-4, M-4)` for right diagonals, for example, and "scanning" from top-left to bottom right. – cricket_007 Aug 21 '16 at 8:06

As a recommendation, I'd suggest you make a method that accepts a starting row and column of the grid, then "scans" in a particular direction for at least 4 values. It can return a boolean for when there are 4 values in a row. – cricket_007 Aug 21 '16 at 8:10

1 And hint: try to come up with more meaningful abstractions: using a two-dim array of chars is a very low-level abstraction for your purpose. Instead, you could use a enum to represent the cells in your board. In other words: don't go for the "immediate" idea that a matrix must be a two-dim array. Spent some time thinking if a different form of organizing your cells might make other things much easier. – GhostCat Aug 21 '16 at 8:11

2 Answers

The simplest algorithm probably is:

```
for every direction
    for every coordinate
        check whether the next 3 elemer
```

in code:

```
final int maxx = 7;
final int maxy = 6;

char winner(char[][] board) {
    int[][] directions = {{1,0}, {1,-1}
    for (int[] d : directions) {
        int dx = d[0];
        int dy = d[1];
        for (int x = 0; x < maxx; x++)
            for (int y = 0; y < maxy; y
                int lastx = x + 3*dx;
                int lasty = y + 3*dy;
                if (0 <= lastx && lastx
                    char w = board[x][y
                    if (w != ' ' && w =
                                && w =
                                && w =
                        return w;
                    }
                }
            }
        }
    }
    return ' '; // no winner
}
```

answered Aug 21 '16 at 10:48

meriton

51.9k 13 79 143

Thanks for the straightforward solution. I've just tried it out however, and it throws a bunch of ArrayIndexOutOfBoundsExceptions as is. I see where you went with it though, it's just a bit confusing to figure out which loops to modify to fix this... – P. Tan Aug 21 '16 at 14:06

Doesn't throw exceptions for me. Are we agreeing on the dimensions of `board` ? (Is it `board[7][6]` as I assumed, or a `board[6][7]`)? – meriton Aug 21 '16 at 15:10

Ah, just realised the typo in my question. It's actually `board[6][7]` in my code. Thanks for pointing that out, and for the solution! – P. Tan Aug 21 '16 at 16:53

First of all, you don't need to check the whole board at any given time, which I think is being done in your code. To optimise your code and make it simpler both for you and the computer, you can do the following:

BEFORE checking any direction on the board, first populate the board with user input. What that means is that every time you get new input, first thing you do is use that input and update your matrix. That way we always have updated matrix and we can do what we will do next:

Get x and y of the input from user. That means is that if user selects row 2, you populate matrix on let's say row 2 and column 4. What do we do with these coordinates? We use THEM and send them to the methods that check the matrix for 4 in a row. This enables us not to check whole matrix, but only the populated matrix.

In the vertical check, you can now check only from those coordinates down, for example:

```
boolean winner = false;

count = 1;

if (x > 3)

for (i = 0; i < 3; i++) {

    if (A[x][y-i] == 'X')

        count ++;

    if (count == 4) {

        winner = true;

        return;

    }
```

For horizontal check, do the same, but horizontally. You need two loops, for checking left and right.

For diagonal check, now you only need to check diagonal containing set coordinates. For example, if user inputted row 4 and column 5, for the left diagonal, we need to check (5,4) , (6,3) and (7,2).

Using coordinates to check for winner is much simpler, it allows us to only check the populated part of matrix.

Sorry for messy answer, written this from phone, I can update and add some more examples later if you want.

answered Aug 21 '16 at 9:37

leonz

619 4 20

What if the last element inserted is in the middle of the 4? (If the two elements to the left and the element to the right are equal, we have a winner, too ...) – meriton Aug 21 '16 at 10:52

You can simply do count++ on both left and right for loop and if you reach terminating character, just break the loop. This code needs a little bit of work, I didn't give fully working example. But the OP's code has complexity of O(n^2+n^2+4n) if I'm not mistaken, where n is 7×6. – leonz Aug 21 '16 at 11:53