LeetCode 529

Description

Let's play the minesweeper game (Wikipedia, online game)!

You are given a 2D char matrix representing the game board. 'M' represents an unrevealed mine, 'E' represents an unrevealed empty square, 'B' represents a revealed blank square that has no adjacent (above, below, left, right, and all 4 diagonals) mines, digit ('1' to '8') represents how many mines are adjacent to this revealed square, and finally 'X' represents a revealed mine.

Now given the next click position (row and column indices) among all the unrevealed squares ('M' or 'E'), return the board after revealing this position according to the following rules:

If a mine ('M') is revealed, then the game is over - change it to 'X'.
If an empty square ('E') with no adjacent mines is revealed, then change it to revealed blank ('B') and all of its adjacent unrevealed squares should be revealed recursively.
If an empty square ('E') with at least one adjacent mine is revealed, then change it to a digit ('1' to '8') representing the number of adjacent mines.
Return the board when no more squares will be revealed.
Example 1:
Input:

[['E', 'E', 'E', 'E', 'E'],
['E', 'E', 'M', 'E', 'E'],
['E', 'E', 'E', 'E', 'E'],
['E', 'E', 'E', 'E', 'E']]

Click : [3,0]

Output:

[['B', '1', 'E', '1', 'B'],
['B', '1', 'M', '1', 'B'],
['B', '1', '1', '1', 'B'],
['B', 'B', 'B', 'B', 'B']]

Explanation:

Solution

```java
public class Solution {
    public char[][] updateBoard(char[][] board, int[] click) {
        int m = board.length, n = board[0].length;
        int row = click[0], col = click[1];

        if (board[row][col] == 'M') { // Mine
            board[row][col] = 'X';
        }
        else { // Empty
            // Get number of mines first.
            int count = 0;
            for (int i = -1; i < 2; i++) {
                for (int j = -1; j < 2; j++) {
                    if (i == 0 && j == 0) continue;
                    int r = row + i, c = col + j;
                    if (r < 0 || r >= m || c < 0 || c < 0 || c >= n) continue;
                    if (board[r][c] == 'M' || board[r][c] == 'X') count++;
                }
            }

            if (count > 0) { // If it is not a 'B', stop further DFS.
                board[row][col] = (char)(count + '0');
            }
            else { // Continue DFS to adjacent cells.
                board[row][col] = 'B';
                for (int i = -1; i < 2; i++) {
                    for (int j = -1; j < 2; j++) {
                        if (i == 0 && j == 0) continue;
                        int r = row + i, c = col + j;
                        if (r < 0 || r >= m || c < 0 || c < 0 || c >= n) continue;
                        if (board[r][c] == 'E') updateBoard(board, new int[] {r, c}
                    }
                }
            }
        }

        return board;
    }
}
```