529. Minesweeper

Add to List

Description SubmissionsSolutions

- Total Accepted: 3778
- Total Submissions: 7310
- Difficulty: Medium
- Contributors:fallcreek

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

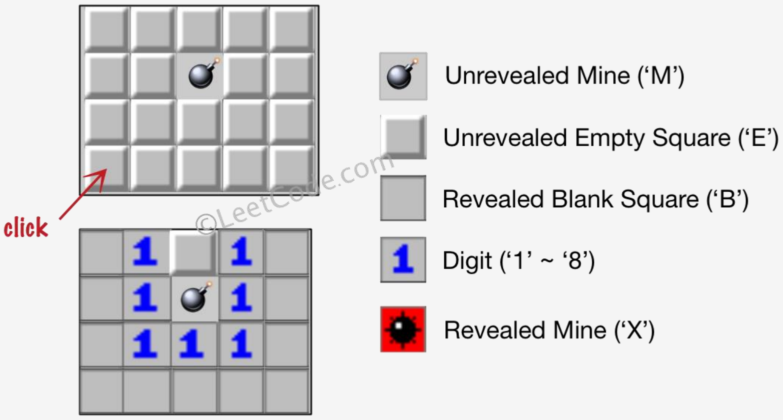
You are given a 2D char matrix representing the game board. **'M'** represents an **unrevealed** mine, **'E'** represents an **unrevealed** empty square, **'B'** represents a **revealed** blank square that has no adjacent (above, below, left, right, and all 4 diagonals) mines, **digit** ('1' to '8')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

1. If a mine ('M') is revealed, then the game is over - change it to **'X'**.
2. 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.
3. 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.
4. Return the board when no more squares will be revealed.

Example 1:

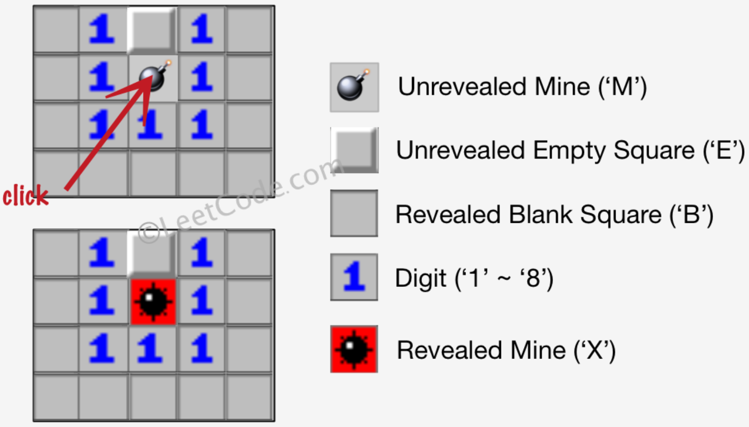
```
Input:



[['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:
```



Example 2:

```
Input:


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

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

 ['B', '1', '1', '1', 'B'],

 ['B', 'B', 'B', 'B', 'B']]
```

```
Click : [1,2]

Output:

[['B', '1', 'E', '1', 'B'],
 ['B', '1', 'X', '1', 'B'],
 ['B', '1', '1', '1', 'B'],
 ['B', 'B', 'B', 'B', 'B']]

Explanation:
```



Note:

1. The range of the input matrix's height and width is [1,50].
2. The click position will only be an unrevealed square ('M' or 'E'), which also means the input board contains at least one clickable square.
3. The input board won't be a stage when game is over (some mines have been revealed).
4. For simplicity, not mentioned rules should be ignored in this problem. For example, you **don't** need to reveal all the unrevealed mines when the game is over, consider any cases that you will win the game or flag any squares.

Subscribe to see which companies asked this question.

```
//C++
class Solution {
public:

    bool inboard(vector<vector<char>>& board, int x, int y)
    {
        return(x>=0 && x<board.size() && y>=0 && y<board[0].size());
    }

    void reveal(vector<vector<char>>& board,int x, int y)
    {
        if(!inboard(board,x,y)) return;
        if(board[x][y]=='E')
        {
            //search 8 neighbors
            int cnt = 0;
            if(inboard(board,x-1,y-1) && board[x-1][y-1] == 'M') cnt++;
            if(inboard(board,x-1,y  ) && board[x-1][y  ] == 'M') cnt++;
            if(inboard(board,x-1,y+1) && board[x-1][y+1] == 'M') cnt++;
            if(inboard(board,x  ,y-1) && board[x  ][y-1] == 'M') cnt++;
            if(inboard(board,x  ,y+1) && board[x  ][y+1] == 'M') cnt++;
            if(inboard(board,x+1,y-1) && board[x+1][y-1] == 'M') cnt++;
            if(inboard(board,x+1,y  ) && board[x+1][y  ] == 'M') cnt++;
            if(inboard(board,x+1,y+1) && board[x+1][y+1] == 'M') cnt++;
            //set board[x][y] cnt
            if (cnt>0)
            board[x][y] = cnt + '0';
            else{
                board[x][y] = 'B';
                reveal(board,x-1,y-1);
                reveal(board,x-1,y  );
                reveal(board,x-1,y+1);
                reveal(board,x  ,y-1);
                reveal(board,x  ,y+1);
                reveal(board,x+1,y-1);
                reveal(board,x+1,y  );
                reveal(board,x+1,y+1);
            }
        }
    }
    vector<vector<char>> updateBoard(vector<vector<char>>& board, 
vector<int>& click) {
        if(board[click[0]][click[1]] == 'M'){
            board[click[0]][click[1]] = 'X';
            return board;
        }
        reveal(board,click[0],click[1]);
        return board;
    }
};

// python
def updateBoard(self, A, click):
    click = tuple(click)
    R, C = len(A), len(A[0])

    def neighbors(r, c):
        for dr in xrange(-1, 2):
            for dc in xrange(-1, 2):
                if (dr or dc) and 0 <= r + dr < R and 0 <= c + dc < C:
                    yield r + dr, c + dc

    stack = [click]
    seen = {click}
    while stack:
        r, c = stack.pop()
        if A[r][c] == 'M':
            A[r][c] = 'X'
        else:
            mines_adj = sum( A[nr][nc] in 'MX' for nr, nc in neighbors(r, c) )
            if mines_adj:
                A[r][c] = str(mines_adj)
            else:
                A[r][c] = 'B'
                for nei in neighbors(r, c):
                    if A[nei[0]][nei[1]] in 'ME' and nei not in seen:
                        stack.append(nei)
                        seen.add(nei)
    return A
```