- ## 529. Minesweeper

Hint:

1. DFS
2. 通过update附近格子进行dfs
3. 通过子函数计算附近mine数量

```
class Solution {
    public char[][] updateBoard(char[][] board, int[] click) {
        if(board[click[0]][click[1]] == 'M'){
            board[click[0]][click[1]] = 'X';
            return board;
        }
        else{
            int minenum = count(board, click[0], click[1]);
            if(minenum == 0){
                update(board, click[0], click[1]);
            } else{
                board[click[0]][click[1]] = (char)('0' + minenum);
            }
        }
        return board;
    }

    private int count(char[][] board, int row, int col){
        int num = 0;
        for(int i = row - 1; i <= row + 1; i++){
            for(int j = col - 1; j <= col + 1; j++){
                if(i < 0 || j < 0 || i >= board.length || j >= board[0].length){
                    continue;
                }
                if(board[i][j] == 'M'){
                    num++;
                }
            }
        }
        return num;
    }

    private void update(char[][] board, int row, int col){
        if(row < 0 || col < 0 || row >= board.length || col >= board[0].length){
            return;
        }
        int minenum = count(board, row, col);
        if(board[row][col] == 'E'){
            if(minenum == 0){
                board[row][col] = 'B';
                for(int i = row - 1; i <= row + 1; i++){
                    for(int j = col - 1; j <= col + 1; j++){
                        update(board, i, j);
                    }
                }
            }
            else{
                board[row][col] = (char)('0' + minenum);
            }
        }
    }
}
```

- ## 929. Unique Email Addresses
- ## 286. Walls and Gates

Hint:

1.经典bfs

先修改一个点，加入queue，再继续poll修改

```
class Solution {
    public void wallsAndGates(int[][] rooms) {
        int[][] direction = new int[][]{{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
        Queue<int[]> queue = new LinkedList<>();
        for(int row = 0; row < rooms.length; row++){
            for(int col = 0; col < rooms[0].length; col++){
                if(rooms[row][col] == 0){
                    queue.add(new int[]{row, col});
                }
            }
        }
        while(!queue.isEmpty()){
            int size = queue.size();
            int[] temp = queue.poll();
            for(int[] dir: direction){
                int row = temp[0] + dir[0];
                int col = temp[1] + dir[1];
                if(row < 0 || col < 0 || row >= rooms.length || col >= rooms[0].length){
                    continue;
                }
                if(rooms[row][col] != Integer.MAX_VALUE){
                    continue;
                }
                rooms[row][col] = rooms[temp[0]][temp[1]] + 1;
                queue.add(new int[]{row, col});

            }
        }

    }
}
```