130. Surrounded Regions

Given a 2D board containing `'X'` and `'O'` (the **letter** O), capture all regions surrounded by `'X'`.

A region is captured by flipping all `'O'`s into `'X'`s in that surrounded region.

For example,

```
X X X X
X O O X
X X O X
X O X X
```

After running your function, the board should be:

```
X X X X
X X X X
X X X X
X O X X
```

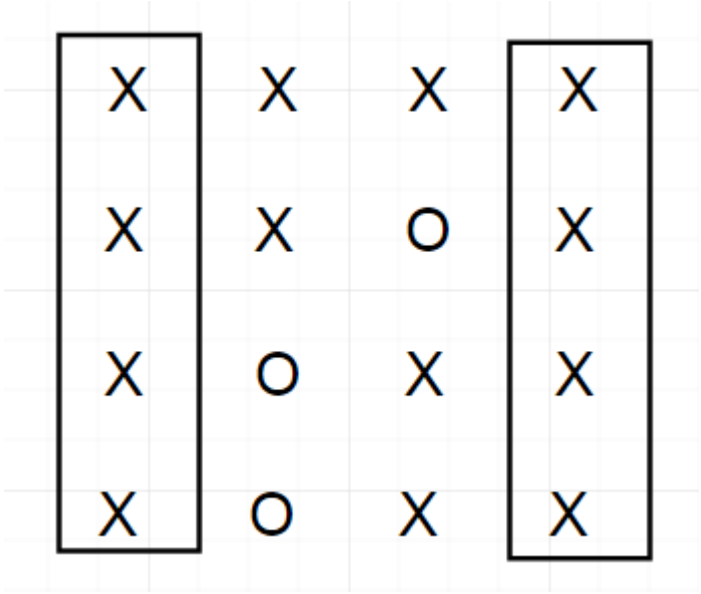
- First, check the four border of the matrix. If there is a element is 'O', alter it and all its neighbor 'O' elements to '1'.
- Then ,alter all the 'O' to 'X'
- At last,alter all the '1' to 'O'

For example:

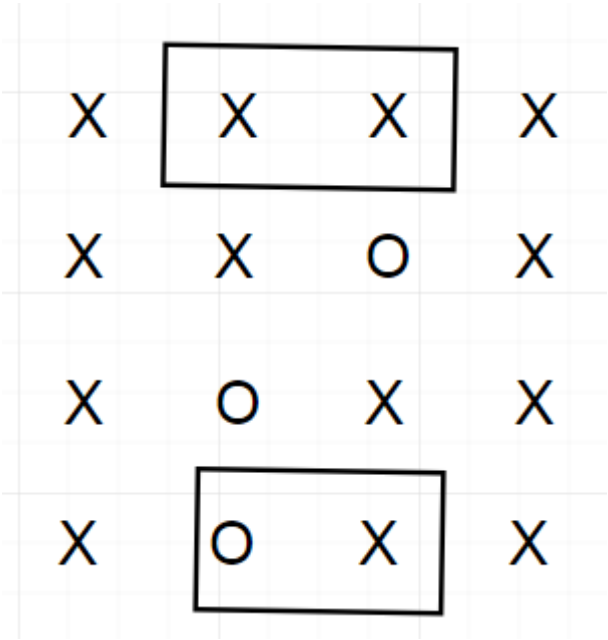
```
        X X X X           X X X X           X X X X
        X X O X  ->       X X O X    ->     X X X X
        X O X X           X 1 X X           X O X X
        X O X X           X 1 X X           X O X X
```

遍历四周的过程如下：

先行，



后遍历列，




```
// iter_swap example
#include <iostream>     // std::cout
#include <algorithm>    // std::iter_swap
#include <vector>       // std::vector
using namespace std;
class Solution {
public:
    void solve(vector<vector<char>>& board) {
        int i,j;
        int row=board.size();
        if(!row)
            return;
        int col=board[0].size();
```

```
        for(i=0;i<row;i++){
            check(board,i,0,row,col);
            if(col>1)
                check(board,i,col-1,row,col);    // 第 i 行的 第一列和最后一列
        }
        for(j=1;j+1<col;j++){                    // 除了第一列和最后一列，遍历 0  行的所有列 和 最后一行的所有列
            check(board,0,j,row,col);
            if(row>1)
                check(board,row-1,j,row,col);
        }
        for(i=0;i<row;i++)
            for(j=0;j<col;j++)
                if(board[i][j]=='O')
                    board[i][j]='X';
        for(i=0;i<row;i++)
            for(j=0;j<col;j++)
                if(board[i][j]=='1')
                    board[i][j]='O';
    }
    void check(vector<vector<char> >&vec,int i,int j,int row,int col){
        if(vec[i][j]=='O'){
            vec[i][j]='1';
            if(i>1)             /// 往上找
                check(vec,i-1,j,row,col);
            if(j>1)             /// 往左找
                check(vec,i,j-1,row,col);
            if(i+1<row)         /// 往下找
                check(vec,i+1,j,row,col);
            if(j+1<col)         /// 往右找
                check(vec,i,j+1,row,col);
        }
    }
};

int main () {

   std::vector<int> res{1,2,3};
    std::vector<std::vector<char>> in {
            {'X','X','X','X'},
            {'X','O','O','X'},
            {'X','X','O','X'},
            {'X','O','X','X'},
    };

    Solution().solve(in);

    return 0;
}
```