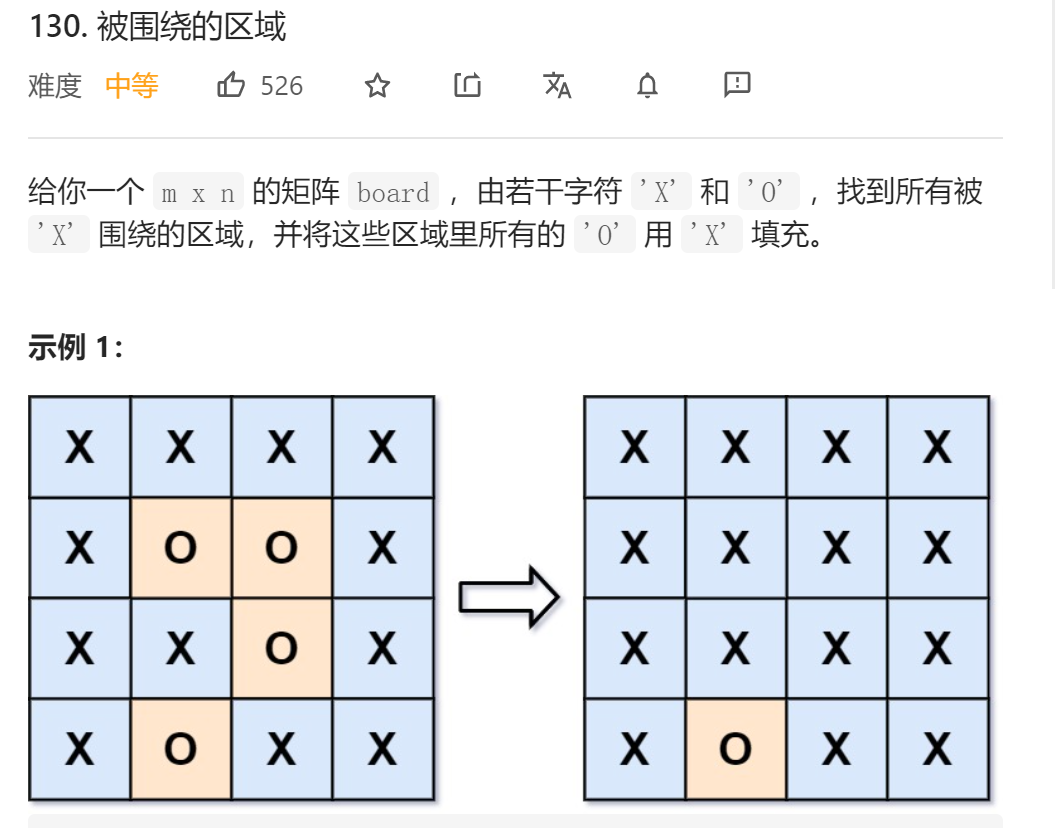
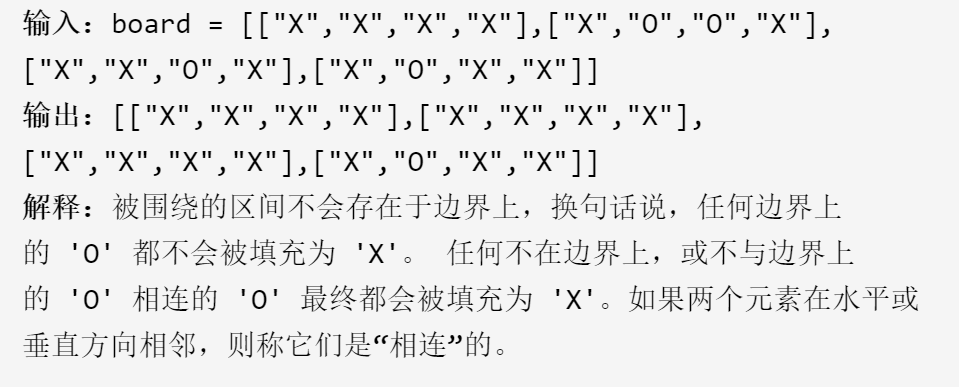
# 【逆向思维】【深搜+递归】

## 题目描述：





## 深搜递归+递归之间的相互影响

1. **[["O","O","O"],["O","O","O"],["O","O","O"]]**
2. **[["O","O","O","O","X","X"],["O","O","O","O","O","O"],["O","X","O","X","O","O"],["O","X","O","O","X","O"],["O","X","O","X","O","O"],["O","X","O","O","O","O"]]**
3. 对于上面两个用例；前一次递归的操作会影响到后一次递归的操作；又由于影响到了后一次递归，导致在回溯的时候会出现问题

```  
class Solution {  
 private:  
 int m,n;  
 bool fill(vector<vector<char>>&board,int x,int y){  
 if(x<0||x>=m||y<0||y>=n)return false;  
  
 //if(board[x][y]=='X')return true;  
 if(board[x][y]=='O'){  
 board[x][y]='X';  
 bool up=fill(board,x+1,y);  
 bool down=fill(board,x-1,y);  
 bool left=fill(board,x,y-1);  
 bool right=fill(board,x,y+1);  
 if(!up||!down||!left||!right){  
 board[x][y]='Y';  
 return false;  
 }  
 }  
 return true;  
 }  
public:  
 void solve(vector<vector<char>>& board) {  
 m=board.size();  
 n=board[0].size();  
 for(int i=0;i<m;i++){  
 for(int j=0;j<n;j++){  
 if(board[i][j]=='O'){  
 fill(board,i,j);  
 }  
 }  
 }  
 for(int i=0;i<m;i++){  
 for(int j=0;j<n;j++){  
 if(board[i][j]=='Y'){  
 board[i][j]='O';  
 }  
 }  
 }  
 }  
};  
```

## 【逆向思维】：利用递归深搜找出所有不符合条件的

class Solution {  
public:  
 int n, m;  
  
 void dfs(vector<vector<char>>& board, int x, int y) {  
 if (x < 0 || x >= n || y < 0 || y >= m || board[x][y] != 'O') {  
 return;  
 }  
 board[x][y] = 'A';  
 dfs(board, x + 1, y);  
 dfs(board, x - 1, y);  
 dfs(board, x, y + 1);  
 dfs(board, x, y - 1);  
 }  
  
 void solve(vector<vector<char>>& board) {  
 n = board.size();  
 if (n == 0) {  
 return;  
 }  
 m = board[0].size();  
 for (int i = 0; i < n; i++) {  
 dfs(board, i, 0);  
 dfs(board, i, m - 1);  
 }  
 for (int i = 1; i < m - 1; i++) {  
 dfs(board, 0, i);  
 dfs(board, n - 1, i);  
 }  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 if (board[i][j] == 'A') {  
 board[i][j] = 'O';  
 } else if (board[i][j] == 'O') {  
 board[i][j] = 'X';  
 }  
 }  
 }  
 }  
};