- ## 130. Surrounded Regions

Hint:
1.Like Number of islands 从周边开始搜索，将为O的坐标加入queue，且将O改为S
2.将queue内坐标进行bfs， 碰到O就加入queue且改为S
3.将剩余的O改为X
4.将S改回O

```
class Solution {
    public void solve(char[][] board) {
        if(board == null || board.length == 0 || board[0].length == 0){
            return;
        }


        Queue<int []> queue = new LinkedList<>();
        for(int row = 0; row < board.length; row++){
            for(int col = 0; col < board[0].length; col++){
                if(board[row][col] == 'O'){
                    if(row == 0 || row == board.length - 1 || col == 0 || col == board[0].length - 1){
                        queue.add(new int[]{row, col});
                        board[row][col] = 'S';
                    }
                }
            }
        }
        while(!queue.isEmpty()){
            int [] temp = queue.poll();
            if(inbound(board, temp[0] + 1, temp[1]) && board[temp[0] + 1][temp[1]] == 'O'){
                board[temp[0] + 1][temp[1]] = 'S';
                queue.add(new int[]{temp[0] + 1, temp[1]});
            }
            if(inbound(board, temp[0] - 1, temp[1]) && board[temp[0] - 1][temp[1]] == 'O'){
                board[temp[0] - 1][temp[1]] = 'S';
                queue.add(new int[]{temp[0] - 1, temp[1]});
            }
            if(inbound(board, temp[0], temp[1] + 1) && board[temp[0]][temp[1] + 1] == 'O'){
                board[temp[0]][temp[1] + 1] = 'S';
                queue.add(new int[]{temp[0], temp[1] + 1});
            }
            if(inbound(board, temp[0], temp[1] - 1) && board[temp[0]][temp[1] - 1] == 'O'){
                board[temp[0]][temp[1] - 1] = 'S';
                queue.add(new int[]{temp[0], temp[1] - 1});
            }
        }

        for(int row = 0; row < board.length; row++){
            for(int col = 0; col < board[0].length; col++){
                if(board[row][col] == 'O'){
                    board[row][col] = 'X';
                }
            }
        }
        for(int row = 0; row < board.length; row++){
            for(int col = 0; col < board[0].length; col++){
                if(board[row][col] == 'S'){
                    board[row][col] = 'O';
                }
            }
        }
    }
    private boolean inbound(char[][] board, int row, int col){
        if(row < 0 || row >= board.length || col < 0 || col >= board[0].length){
            return false;
        }
        return true;
    }
}
```

- ## 323. Number of Connected Components in an Undirected Graph

Hint: UnionSet
1.没有规定大数在前，还是小数在前，因此需要对每一元素findhead后再添加
2.使用hashset记录集合的head，每个元素的添加要先findhead

```
class Solution {
    public int countComponents(int n, int[][] edges) {
        int[] ans = new int[n];
        for(int i = 0; i < n; i++){
            ans[i] = i;
        }
        for(int i = 0; i < edges.length; i++){
            int first = edges[i][0];
            int second = edges[i][1];
            ans[findHead(ans, second)] = findHead(ans, first);
        }
        Set<Integer> set = new HashSet<>();
        for(int i = 0; i < n; i++){
            set.add(findHead(ans, ans[i]));
        }
        return set.size();



    }


    private int findHead(int[] ans, int pos){
        if(ans[pos] == pos){
            return pos;
        }
        else{
            return findHead(ans, ans[pos]);
        }
    }
}
```